# Manuel MySQL 4.1.x − Version Française

http://www.nexen.net

Copyright (c) 2002−2003 MySQL AB

# Sommaire

**1 Informations générales....1**
1.1 A propos du manuel....1
1.1.1 Conventions utilisées dans ce manuel....2
1.2 Qu'est ce que MySQL?....3
1.2.1 Histoire de MySQL....4
1.2.2 Les fonctionnalités principales de MySQL....5
1.2.3 Jusqu'à quel point MySQL est il stable ?....7
1.2.4 Quelles tailles de tables supporte MySQL ?....8
1.2.5 Compatibilité an 2000....9
1.3 Qui est MySQL AB ?....10
1.3.1 Les services et le modèle d'affaire de MySQL AB....10
1.3.1.1 Support....11
1.3.1.2 Formation et certification....11
1.3.1.3 Conseil....11
1.3.1.4 Licences commerciales....12
1.3.1.5 Partenariats....12
1.3.1.6 Publicité....12
1.3.2 Contacts....13
1.4 Support MySQL et licences....14
1.4.1 Support proposé par MySQL AB....14
1.4.2 Copyrights et licences utilisées par MySQL....14
1.4.3 Licences MySQL....15
1.4.3.1 Utiliser MySQL avec la licence commerciale....15
1.4.3.2 Utiliser MySQL sous licence GPL libre....16
1.4.4 Logos MySQL AB et marque déposée....17
1.4.4.1 Le logo original de MySQL....17
1.4.4.2 Logos MySQL qui peuvent être utilisés dans autorisation préalable....17
1.4.4.3 Quand avez vous besoin d'autorisation pour utiliser le logo MySQL?....17
1.4.4.4 Logos de partenariat MySQL AB....18
1.4.4.5 Utiliser le nom MySQL sur des documents imprimés ou des présentations....18
1.4.4.6 Utilisation du nom MySQL dans un nom de société ou de produit....18
1.5 MySQL 4.0 In A Nutshell....18
1.5.1 Fonctionnalités disponibles depuis MySQL 4.0....19
1.5.2 MySQL Server intégré (embedded)....20
1.6 MySQL 4.1 In A Nutshell....20
1.6.1 Fonctionnalités disponibles depuis MySQL 4.1....20
1.6.2 Phases de publication....21
1.6.3 Utilisation immédiate en production....22
1.6.4 MySQL 5.0, The Next Development Release....22
1.7 Sources d'informations MySQL....22
1.7.1 Listes de diffusion MySQL....22
1.7.1.1 Les listes de diffusions de MySQL....22
1.7.1.2 Poser des questions ou rapporter un bogue....24
1.7.1.3 Comment rapporter un bogue ou un problème....25
1.7.1.4 Conseils pour répondre sur la liste de diffusion....28
1.7.2 Support de la communauté MySQL sur IRC (Internet Relay Chat)....29
1.8 Quels standards respecte MySQL?....29
1.8.1 Quels standards suit MySQL ?....30
1.8.2 Exécuter MySQL en mode ANSI....30
1.8.3 Extensions de MySQL à la norme ANSI SQL92....30
1.8.4 Différences de MySQL avec ANSI SQL92....32

# Sommaire

**1 Informations générales**
1.8.4.1 Sous sélections (Sub SELECT s)....33
1.8.4.2 SELECT INTO TABLE....33
1.8.4.3 Transactions et opérations atomiques....34
1.8.4.4 Procédures stockées et triggers....36
1.8.4.5 Les clés étrangères....36
1.8.4.6 Les vues....37
1.8.4.7 '−−' comme début de commentaire....38
1.8.5 Erreurs connues, et limitations de MySQL....38
1.8.5.1 Erreurs connues en 3.23 et corrigées ultérieurement....38
1.8.5.2 Bugs connus / limitations de MySQL....39
1.9 Les évolutions de MySQL (la liste des tâches)....42
1.9.1 New Features Planned For 4.1....42
1.9.2 New Features Planned For 5.0....42
1.9.3 Nouvelles fonctionnalités prévues pour 5.1....43
1.9.4 Ce qui doit être fait dans un futur proche....44
1.9.5 Ce qui est prévu à moyen terme....46
1.9.6 Ce qui n'est pas prévu....47
1.10 Comparatif de MySQL avec les autres serveurs SQL....47
1.10.1 MySQL face à mSQL....47
1.10.1.1 Comment convertir des outils mSQL pour MySQL....49
1.10.1.2 Différences entre les protocoles de communication de mSQL et de MySQL....50
1.10.1.3 Comparatif des syntaxes SQL de mSQL 2.0 et MySQL....50
1.10.2 Comparatif de MySQL avec PostgreSQL....53
1.10.2.1 Stratégies de développement de MySQL et PostgreSQL....53
1.10.2.2 Comparaison des fonctionnalités de MySQL et PostgreSQL....54
1.10.2.3 Performances comparées de MySQL et PostgreSQL....57

**2 Installation de MySQL....61**
2.1 Installation standard rapide de MySQL....61
2.1.1 Installer MySQL sous Linux....61
2.1.2 Installer MySQL sous Windows....62
2.1.2.1 Installation des binaires....63
2.1.2.2 Préparation de l'environnement MySQL de Windows....63
2.1.2.3 Démarrer le serveur pour la première fois....64
2.1.3 Installer MySQL sur Mac OS X....65
2.1.4 Installer MySQL sur NetWare....66
2.1.4.1 Installer les exécutables MySQL pour NetWare....66
2.2 Notes générales à propos de l'installation....67
2.2.1 Comment obtenir MySQL ?....67
2.2.2 Vérifier l'intégrité des paquets avec MD5 ou GnuPG....67
2.2.3 Systèmes d'exploitation supportés par MySQL....69
2.2.4 Quelle version de MySQL utiliser ?....71
2.2.5 Dispositions d'installation....73
2.2.6 Quand et comment sont publiées les nouvelles versions....74
2.2.7 Politique de publication : aucun bug connu dans les versions....74
2.2.8 Binaires compilés par MySQL AB....75
2.2.9 Installer MySQL à partir d'une distribution binaire....77
2.3 Installer MySQL à partir des sources....79
2.3.1 Vue d'ensemble de l'installation rapide....80
2.3.2 Appliquer des patches....82

# Sommaire

**2 Installation de MySQL**
2.3.3 Options habituelles de configure....82
2.3.4 Installer à partir de l'arbre source de développement....84
2.3.5 Problèmes de compilation?....85
2.3.6 Notes relatives aux MIT−pthreads....87
2.3.7 La distribution source Windows....88
2.4 Configuration après l'installation, et tests....89
2.4.1 Problèmes d'exécution de mysql_install_db....92
2.4.2 Problèmes de démarrage du serveur MySQL....93
2.4.3 Lancer et arrêter MySQL automatiquement....95
2.5 Changer de version de MySQL....96
2.5.1 Passer de la version 4.0 à la version 4.1....96
2.5.2 Préparer le passage de version 4.0 en 4.1....97
2.5.3 Passer de la version 3.23 à la version 4.0....97
2.5.4 Passer de la version 3.22 à la version 3.23....100
2.5.5 Passer de la version 3.21 à la version 3.22....101
2.5.6 Passer de la version 3.20 à la version 3.21....101
2.5.7 Migrer depuis une autre architecture....102
2.6 Notes spécifiques aux systèmes d'exploitation....103
2.6.1 Notes relatives à Linux (toutes versions)....103
2.6.1.1 Notes relatives à Linux pour les distributions binaires....106
2.6.1.2 Notes relatives à Linux x86....108
2.6.1.3 Notes relatives à Linux SPARC....108
2.6.1.4 Notes relatives à Linux Alpha....109
2.6.1.5 Note relative à Linux PowerPC....109
2.6.1.6 Notes relatives à Linux MIPS....109
2.6.1.7 Notes relatives à Linux IA64....110
2.6.2 Notes relatives à Windows....110
2.6.2.1 Démarrer MySQL sous Windows 95, 98 ou Me....110
2.6.2.2 Démarrer MySQL sur Windows NT, 2000 ou XP....111
2.6.2.3 Faire fonctionner MySQL sous Windows....112
2.6.2.4 Connexion à un serveur MySQL distants, sous Windows avec SSH....113
2.6.2.5 Paratger les données entre plusieurs disque sous Windows....113
2.6.2.6 Compiler les clients MySQL sous Windows....114
2.6.2.7 MySQL pour Windows face à MySQL pour Unix....114
2.6.3 Remarques pour Solaris....116
2.6.3.1 Notes relatives à Solaris 2.7/2.8....118
2.6.3.2 Remarques pour Solaris x86....119
2.6.4 Notes relatives à BSD....120
2.6.4.1 Notes relatives à FreeBSD....120
2.6.4.2 Notes concernant NetBSD....121
2.6.4.3 Notes relatives à OpenBSD 2.5....121
2.6.4.4 Notes relatives à OpenBSD 2.8....121
2.6.4.5 Notes relatives aux versions 2.x de BSD/OS....121
2.6.4.6 Notes relatives aux versions 3.x de BSD/OS....122
2.6.4.7 Notes relatives aux versions 4.x de BSD/OS....122
2.6.5 Notes relatives à Mac OS X....123
2.6.5.1 Mac OS X 10.x....123
2.6.5.2 Mac OS X Server 1.2 (Rhapsody)....123
2.6.6 Notes sur les autres Unix....123
2.6.6.1 Notes relatives à HP−UX pour les distributions binaires....123
2.6.6.2 Notes relatives à la version 10.20 de HP−UX....124

# Sommaire

**2 Installation de MySQL**
2.6.6.3 HP−UX Version 11.x Notes....124
2.6.6.4 Notes relatives à IBM−AIX....125
2.6.6.5 Notes relatives à SunOS 4....127
2.6.6.6 Notes pour Alpha−DEC−UNIX (Tru64)....127
2.6.6.7 Notes pour Alpha−DEC−OSF/1....128
2.6.6.8 Notes relatives à SGI Irix....129
2.6.6.9 Notes pour Caldera (SCO)....130
2.6.6.10 Notes relatives à la version 7.0 fr Caldera (SCO) Unixware....131
2.6.7 Notes relatives à OS/2....132
2.6.8 Notes relatives à BeOS....132
2.6.9 Notes relatives à Novell NetWare....133
2.7 Commentaires sur l'installation de Perl....133
2.7.1 Installer Perl sur Unix....133
2.7.2 Installer ActiveState Perl sur Windows....134
2.7.3 Installer la distribution Perl de MySQL sous Windows....134
2.7.4 Problèmes lors de l'utilisation des interfaces Perl DBI et DBD....134

**3 Tutoriels d'introduction....137**
3.1 Connexion et déconnexion au serveur....137
3.2 Entrer des requêtes....138
3.3 Création et utilisation d'une base de données....140
3.3.1 Créer et sélectionner une base de données....141
3.3.2 Création d'une table....141
3.3.3 Charger des données dans une table....143
3.3.4 Récupérer des informations à partir d'une table....144
3.3.4.1 Sélectionner toutes les données....144
3.3.4.2 Sélectionner des lignes particuliéres....145
3.3.4.3 Sélectionner des colonnes particulières....145
3.3.4.4 Trier les enregistrements....146
3.3.4.5 Calcul sur les Dates....147
3.3.4.6 Travailler avec la valeur NULL....149
3.3.4.7 Recherche de modèles....149
3.3.4.8 Compter les lignes....151
3.3.4.9 Utiliser plus d'une table....153
3.4 Obtenir des informations à propos des bases de données et des tables....154
3.5 Exemples de requêtes usuelles....155
3.5.1 La valeur maximale d'une colonne....156
3.5.2 La ligne contenant le maximum d'une certaine colonne....156
3.5.3 Maximum d'une colonne par groupe....156
3.5.4 La ligne contenant la plus grande valeur d'un certain champ par rapport à un groupe....156
3.5.5 Utiliser les variables utilisateur....157
3.5.6 Utiliser les clefs étrangères....158
3.5.7 Recherche sur deux clefs....158
3.5.8 Calcul du nombre de visites par jour....159
3.5.9 Utiliser AUTO_INCREMENT....159
3.6 Utilisation de mysql en mode batch....160
3.7 Requêtes du projet Twin....161
3.7.1 Trouver tous les jumeaux répondant aux critères....162
3.7.2 Afficher une table avec l'état des paires de jumeaux....163
3.8 Utilisation de MySQL avec Apache....164

# Sommaire

**4 Administration du serveur....165**
4.1 Configuration de MySQL....165
4.1.1 Options de ligne de commande de mysqld....165
4.1.2 Fichier d'options my.cnf....169
4.1.3 Installer plusieurs serveurs sur la même machine....171
4.1.4 Faire fonctionner plusieurs serveurs MySQL sur la même machine....172
4.2 Règles de sécurité et droits d'accès au serveur MySQL....174
4.2.1 Instructions générales de sécurité....174
4.2.2 Comment protéger MySQL contre les pirates....176
4.2.3 Options de démarrage qui concernent la sécurité....178
4.2.4 Problèmes de sécurité avec LOAD DATA LOCAL....178
4.2.5 Rôle du système de privilèges....179
4.2.6 Comment fonctionne le système de droits....179
4.2.7 Droits fournis par MySQL....182
4.2.8 Se connecter au serveur MySQL....184
4.2.9 Contrôle d'accès, étape 1 : Vérification de la connexion....184
4.2.10 Contrôle d'accès, étape 2 : Vérification de la requête....187
4.2.11 Causes des erreurs Access denied....189
4.3 Gestion des comptes utilisateurs de MySQL....192
4.3.1 Syntaxe de GRANT et REVOKE....193
4.3.2 Nom d'utilisateurs MySQL et mots de passe....196
4.3.3 Quand les modifications de privilèges prennent−ils effets ?....197
4.3.4 Création des premiers droits MySQL....198
4.3.5 Ajouter de nouveaux utilisateurs à MySQL....199
4.3.6 Limiter les ressources utilisateurs....201
4.3.7 Configurer les mots de passe....202
4.3.8 Garder vos mots de passe en lieu sûr....203
4.3.9 Utilisation des connexions sécurisées....204
4.3.9.1 Introduction aux connexions sécurisées....204
4.3.9.2 Pré requis aux connexions sécurisées....204
4.3.9.3 Configurer les certificats SSL pour MySQL....205
4.3.9.4 Options de GRANT....207
4.4 Prévention des désastres et restauration....208
4.4.1 Sauvegardes de base de données....208
4.4.2 Syntaxe de BACKUP TABLE....209
4.4.3 Syntaxe de RESTORE TABLE....209
4.4.4 Syntaxe de CHECK TABLE....210
4.4.5 Syntaxe de REPAIR TABLE....211
4.4.6 Utilisation de myisamchk pour la maintenance des tables et leur recouvrement....212
4.4.6.1 Syntaxe de l'utilitaire myisamchk....213
4.4.6.2 Options générales de myisamchk....214
4.4.6.3 Options de vérifications pour myisamchk....215
4.4.6.4 Options de réparation de myisamchk....215
4.4.6.5 Autres options de myisamchk....216
4.4.6.6 Utilisation de la mémoire par myisamchk....217
4.4.6.7 Utiliser myisamchk pour restaurer une table....217
4.4.6.8 Comment vérifier la cohérence d'une table....218
4.4.6.9 Comment réparer des tables....219
4.4.6.10 Optimisation de tables....221
4.4.7 Mettre en place un régime d'entretien de MySQL....221
4.4.8 Obtenir des informations sur une table....222

# Sommaire

**4 Administration du serveur**
4.5 Référence de langage d'administration de la base de données....226
4.5.1 Syntaxe de OPTIMIZE TABLE....226
4.5.2 Syntaxe de ANALYZE TABLE....226
4.5.3 Syntaxe de FLUSH....227
4.5.4 Syntaxe de la commande RESET....227
4.5.5 Syntaxe de KILL....228
4.5.6 Syntaxe de SHOW....228
4.5.6.1 Obtenir des informations sur les bases, tables, colonnes et index....229
4.5.6.2 SHOW TABLE STATUS....230
4.5.6.3 Syntaxe de SHOW STATUS....230
4.5.6.4 Syntaxe de SHOW VARIABLES....232
4.5.6.5 Syntaxe de SHOW LOGS....239
4.5.6.6 Syntaxe de SHOW PROCESSLIST....239
4.5.6.7 SHOW GRANTS....241
4.5.6.8 Syntaxe de SHOW CREATE TABLE....241
4.5.6.9 SHOW WARNINGS | ERRORS....241
4.5.6.10 SHOW TABLE TYPES....241
4.5.6.11 SHOW PRIVILEGES....242
4.6 Localisation de MySQL et utilisation internationale....242
4.6.1 Le jeu de caractères utilisé pour les données et le stockage....242
4.6.1.1 Jeu de caractères allemand....243
4.6.2 Langue des messages d'erreurs....244
4.6.3 Ajouter un nouveau jeu de caractères....244
4.6.4 Le tableau de définition des caractères....245
4.6.5 Support d'assemblage des chaînes....246
4.6.6 Support des caractères multi−octets....246
4.6.7 Problèmes avec les jeux de caractères....246
4.7 Scripts serveur MySQL et utilitaires....247
4.7.1 Présentation des scripts serveurs et des utilitaires....247
4.7.2 safe_mysqld , le script père de mysqld....247
4.7.3 mysqld_multi , un programme pour gérer plusieurs serveurs MySQL....249
4.7.4 myisampack , le générateur de tables MySQL compressées en lecture seule...251
4.7.5 mysqld−max , la version étendue du serveur mysqld....256
4.8 MySQL Scripts clients et utilitaires....257
4.8.1 Présentation des scripts serveurs et utilitaires....257
4.8.2 mysql , l'outil en ligne de commande....258
4.8.3 mysqladmin , administrer un serveur MySQL....263
4.8.4 Utiliser mysqlcheck pour l'entretien et la réparation....264
4.8.5 mysqldump , exporter les structures de tables et les données....266
4.8.6 mysqlhotcopy , copier les bases et tables MySQL....270
4.8.7 mysqlimport , importer des données depuis des fichiers texte....271
4.8.8 Afficher les bases, tables et colonnes....272
4.8.9 mysql_config lit les options de compilations du client MySQL....273
4.8.10 perror , expliquer les codes d'erreurs....274
4.8.11 Comment exécuter des commandes SQL depuis un fichier texte....274
4.9 Les fichiers de log de MySQL....274
4.9.1 Le log d'erreurs....275
4.9.2 Le log général de requêtes....275
4.9.3 Le log de modification....275
4.9.4 Le log binaire....276
4.9.5 Le log des requêtes lentes....278

# Sommaire

**4 Administration du serveur**
4.9.6 Entretien des fichiers de log....278
4.10 Réplication de MySQL....279
4.10.1 Introduction à la réplication....279
4.10.2 Présentation de l'implémentation de la réplication....279
4.10.3 Comment mettre en place la réplication....280
4.10.4 Fonctionnalités de la réplication et problèmes connus....282
4.10.5 Options de réplication dans le fichier my.cnf....284
4.10.6 Commandes SQL liées à la réplication....287
4.10.7 FAQ de la réplication....288
4.10.8 Correction de problèmes courants....292

**5 Optimisation de MySQL....294**
5.1 Vue d'ensemble de l'optimisation....294
5.1.1 Limitations et inconvénients des choix conceptuels de MySQL....294
5.1.2 Portabilité....295
5.1.3 Pour quoi avons nous utilisé MySQL ?....296
5.1.4 La suite de tests MySQL....297
5.1.5 Utiliser vos propres tests de performance....298
5.2 Optimisation des SELECT s et autres requêtes....299
5.2.1 Syntaxe de EXPLAIN (Obtenir des informations sur les SELECT )....299
5.2.2 Mesurer les performances d'une requête....304
5.2.3 Vitesse des requêtes SELECT....304
5.2.4 Comment MySQL optimise les clauses WHERE....305
5.2.5 Comment MySQL optimise la clause DISTINCT....306
5.2.6 Comment MySQL optimise LEFT JOIN et RIGHT JOIN....306
5.2.7 Comment MySQL optimise les clauses ORDER BY....307
5.2.8 Comment MySQL optimise la clause LIMIT....308
5.2.9 Vitesse des requêtes INSERT....309
5.2.10 Vitesses des commandes UPDATE....310
5.2.11 Rapidité des requêtes DELETE....311
5.2.12 Autres conseils d'optimisation....311
5.3 Verrouillage de tables....313
5.3.1 Comment MySQL verrouille les tables....313
5.3.2 Problème de verrouillage de tables....314
5.4 Optimisation de la structure de la base de données....315
5.4.1 Conception....315
5.4.2 Rendre vos tables aussi compactes que possible....316
5.4.3 Comment MySQL utilise les index....317
5.4.4 Index de colonnes....318
5.4.5 Index sur plusieurs colonnes....319
5.4.6 Pourquoi tant de tables ouvertes ?....320
5.4.7 Quand MySQL ouvre et ferme les tables....320
5.4.8 Inconvénients de la création d'un grand nombre de tables dans la même base de données....321
5.5 Optimisation du serveur MySQL....321
5.5.1 Réglage du système, au moment de la compilation, et paramètres du démarrage....321
5.5.2 Réglage des paramètres du serveur....322
5.5.3 Influences de la compilation et des liaisons sur la vitesse de MySQL....324
5.5.4 Comment MySQL gère la mémoire....325
5.5.5 Comment MySQL utilise le DNS....326

# Sommaire

**5 Optimisation de MySQL**
5.5.6 Syntaxe de SET....327
5.6 Problèmes avec les disques....330
5.6.1 Utiliser des liens symboliques....330
5.6.1.1 Utiliser les liens symboliques pour les bases....331
5.6.1.2 Utiliser des liens symboliques avec les tables....331

**6 Référence du langage MySQL....333**
6.1 Structure du langage....333
6.1.1 Literals: Comment écrire les chaînes et les nombres....333
6.1.1.1 Chaînes....333
6.1.1.2 Nombres....335
6.1.1.3 Valeurs hexadécimales....335
6.1.1.4 Valeurs NULL....335
6.1.2 Noms de bases, tables, index, colonnes et alias....335
6.1.3 Sensibilité à la casse pour les noms....336
6.1.4 Variables utilisateur....337
6.1.5 Variables système....338
6.1.6 Syntaxe des commentaires....340
6.1.7 Est−ce que MySQL est sensible aux mots réservés ?....341
6.2 Types de colonnes....341
6.2.1 Types numériques....345
6.2.2 Les types date et heure....347
6.2.2.1 An 2000 et les types date....348
6.2.2.2 Les types DATETIME , DATE , et TIMESTAMP....348
6.2.2.3 Le type TIME....351
6.2.2.4 Le type YEAR....352
6.2.3 Les types chaînes....353
6.2.3.1 Les types CHAR et VARCHAR....353
6.2.3.2 Les types BLOB et TEXT....354
6.2.3.3 Le type ENUM....354
6.2.3.4 Le type SET....356
6.2.4 Choisir le bon type de colonne....357
6.2.5 Utilisation des types de données issues d'autres SGBDR....357
6.2.6 Capacités des colonnes....357
6.3 Fonctions à utiliser dans les clauses SELECT et WHERE....358
6.3.1 Opérateurs et fonctions tout−types....359
6.3.1.1 Parenthèses....359
6.3.1.2 Opérateurs de comparaison....359
6.3.1.3 Opérateurs logiques....362
6.3.1.4 Les fonctions de contrôle....363
6.3.2 Fonctions de chaînes de caractères....364
6.3.2.1 Opérateurs de comparaison pour les chaînes de caractères....370
6.3.2.2 Sensibilité à la casse....371
6.3.3 Fonctions numériques....372
6.3.3.1 Opérations arithmétiques....372
6.3.3.2 Fonctions mathématiques....372
6.3.4 Fonctions de dates et d'heures....376
6.3.5 Fonctions de transtypage....382
6.3.6 Autres fonctions....383
6.3.6.1 Fonctions sur les bits....383
6.3.6.2 Fonctions diverses....384

# Sommaire

**6 Référence du langage MySQL**
6.3.7 Fonctions avec la clause GROUP BY....389
6.4 Manipulation de données : SELECT , INSERT , UPDATE , DELETE....391
6.4.1 Syntaxe de SELECT....391
6.4.1.1 Syntaxe de JOIN....394
6.4.1.2 Syntaxe de UNION....395
6.4.2 Syntaxe de HANDLER....396
6.4.3 Syntaxe de INSERT....397
6.4.3.1 Syntaxe de INSERT ... SELECT....398
6.4.4 Syntaxe de INSERT DELAYED....399
6.4.5 Syntaxe de UPDATE....400
6.4.6 Syntaxe de DELETE....401
6.4.7 Syntaxe de TRUNCATE....402
6.4.8 Syntaxe de REPLACE....403
6.4.9 Syntaxe de LOAD DATA INFILE....403
6.4.10 Syntaxe de DO....408
6.5 Définition de données : CREATE , DROP , ALTER....408
6.5.1 Syntaxe de CREATE DATABASE....409
6.5.2 Syntaxe de DROP DATABASE....409
6.5.3 Syntaxe de CREATE TABLE....409
6.5.3.1 Modification automatique du type de colonnes....415
6.5.4 Syntaxe de ALTER TABLE....415
6.5.5 Syntaxe de RENAME TABLE....418
6.5.6 Syntaxe de DROP TABLE....419
6.5.7 Syntaxe de CREATE INDEX....419
6.5.8 Syntaxe de DROP INDEX....420
6.6 Commandes de bases de l'utilisateur de MySQL....420
6.6.1 Syntaxe de USE....420
6.6.2 Syntaxe de DESCRIBE (obtenir des informations sur les colonnes)....420
6.7 Commandes relatives aux verrous et aux transactions....421
6.7.1 Syntaxe de BEGIN/COMMIT/ROLLBACK....421
6.7.2 Syntaxe de LOCK TABLES/UNLOCK TABLES....421
6.7.3 Syntaxe de SET TRANSACTION....423
6.8 Recherche en Texte−entier (Full−text) dans MySQL....424
6.8.1 Restrictions avec full−text....427
6.8.2 Paramétrage précis de la recherche Full−text de MySQL....427
6.8.3 A faire dans la recherche Full−text....427
6.9 Cache de requêtes MySQL....428
6.9.1 Comment fonctionne le cache de requêtes....428
6.9.2 Configuration du cache de requêtes....429
6.9.3 Options relatives au cache de requêtes dans un SELECT....430
6.9.4 Status du cache de requêtes et maintenance....430

**7 Types de tables MySQL....431**
7.1 Tables MyISAM....431
7.1.1 Espace requis pour les clefs....434
7.1.2 Formats de table MyISAM....434
7.1.2.1 Caractéristiques des tables statiques (taille fixée)....434
7.1.2.2 Caractéristiques des tables à format de ligne dynamiques....435
7.1.2.3 Caractéristiques des tables compressées....436
7.1.3 Problèmes avec les tables MyISAM....436
7.1.3.1 Tables MyISAM corrompues....436

# Sommaire

**7 Types de tables MySQL**
7.1.3.2 Clients is using or hasn't closed the table properly....437
7.2 Tables assemblées MERGE....438
7.2.1 Problèmes avec les tables MERGE....440
7.3 Tables ISAM....440
7.4 Tables HEAP....441
7.5 Tables InnoDB....442
7.5.1 Présentation des tables InnoDB....442
7.5.2 Options de démarrage InnoDB....443
7.5.3 Créer des bases InnoDB....446
7.5.3.1 Si quelque chose se passe mal à la création de la base de données....447
7.5.4 Créer des tables InnoDB....447
7.5.5 Ajouter et retirer des données et des logs InnoDB....450
7.5.6 Sauver et restaurer une base InnoDB....450
7.5.6.1 Points de contrôle....451
7.5.7 Transférer une base de données InnoDB vers une autre machine....452
7.5.8 Modèle transactionnel de InnoDB....452
7.5.8.1 Lecture cohérente....453
7.5.8.2 Verrous de lecture....454
7.5.8.3 Verrou de clé suivante : éviter le problème des lignes fantômes....454
7.5.8.4 Les verrous posés par différentes requêtes SQL avec InnoDB....455
7.5.8.5 Détection des blocages et annulation....456
7.5.8.6 Un exemple de lecture cohérente avec InnoDB....456
7.5.8.7 Comment gérer les blocages de verrous?....457
7.5.8.8 Conseils sur l'amélioration des performances InnoDB....457
7.5.8.9 Le moniteur InnoDB....459
7.5.9 Implémentation du multi−versionnage....460
7.5.10 Structures de tables et d'index....461
7.5.10.1 Structure physique d'un index....461
7.5.10.2 Bufferisation des insertions....462
7.5.10.3 Index hash adaptatifs....462
7.5.10.4 Structure physique d'une ligne....462
7.5.11 Gestion de l'espace fichiers et des entrées/sorties disque....463
7.5.11.1 Accès disques....463
7.5.11.2 Gestion de l'espace fichier....464
7.5.11.3 Défragmentation des tables....465
7.5.12 Gestion des erreurs....465
7.5.13 Restrictions sur les tables InnoDB....465
7.5.14 Historique de l'évolution InnoDB....466
7.5.15 Informations de contact InnoDB....474
7.6 Tables BDB ou BerkeleyDB....474
7.6.1 Vue d'ensemble des tables BDB....474
7.6.2 Installation de BDB....475
7.6.3 Options de démarrage BDB....475
7.6.4 Caractéristiques des tables BDB....476
7.6.5 Ce que nous devons corriger dans BDB dans un futur proche :....477
7.6.6 Systèmes d'exploitation supportés par BDB....477
7.6.7 Restrictions avec les tables BDB....477
7.6.8 Erreurs pouvant survenir lors de l'utilisation des tables BDB....478

# Sommaire

**8 Les interfaces pour MySQL....479**
8.1 API PHP pour MySQL....479
8.1.1 Problèmes fréquents avec MySQL et PHP....479
8.2 API Perl pour MySQL....479
8.2.1 DBI avec DBD::mysql....479
8.2.2 L'interface DBI....480
8.2.3 Plus d'informations relatives à DBI / DBD....484
8.3 Support ODBC avec MySQL....485
8.3.1 Comment installer MyODBC....485
8.3.2 Comment remplir les différents champs dans le programme d'administrateur ODBC....486
8.3.3 Paramètres de connexion de MyODBC....486
8.3.4 Comment reporter les problèmes avec ODBC....487
8.3.5 Programmes qui fonctionnent avec MyODBC....488
8.3.6 Comment obtenir la valeur d'une colonne AUTO_INCREMENT avec ODBC....491
8.3.7 Rapporter des problèmes avec MyODBC....492
8.4 MySQL C API....492
8.4.1 Types de données de l'API C....493
8.4.2 Vue d'ensemble des fonctions de l'API C....495
8.4.3 Description des fonctions de l'API C....498
8.4.3.1 mysql_affected_rows()....498
8.4.3.2 mysql_change_user()....499
8.4.3.3 mysql_character_set_name()....500
8.4.3.4 mysql_close()....500
8.4.3.5 mysql_connect()....500
8.4.3.6 mysql_create_db()....501
8.4.3.7 mysql_data_seek()....501
8.4.3.8 mysql_debug()....502
8.4.3.9 mysql_drop_db()....502
8.4.3.10 mysql_dump_debug_info()....502
8.4.3.11 mysql_eof()....503
8.4.3.12 mysql_errno()....504
8.4.3.13 mysql_error()....504
8.4.3.14 mysql_escape_string()....505
8.4.3.15 mysql_fetch_field()....505
8.4.3.16 mysql_fetch_fields()....506
8.4.3.17 mysql_fetch_field_direct()....506
8.4.3.18 mysql_fetch_lengths()....507
8.4.3.19 mysql_fetch_row()....507
8.4.3.20 mysql_field_count()....508
8.4.3.21 mysql_field_seek()....509
8.4.3.22 mysql_field_tell()....509
8.4.3.23 mysql_free_result()....509
8.4.3.24 mysql_get_client_info()....510
8.4.3.25 mysql_get_server_version()....510
8.4.3.26 mysql_get_host_info()....510
8.4.3.27 mysql_get_proto_info()....511
8.4.3.28 mysql_get_server_info()....511
8.4.3.29 mysql_info()....511
8.4.3.30 mysql_init()....512
8.4.3.31 mysql_insert_id()....512
8.4.3.32 mysql_kill()....512

# Sommaire

**8 Les interfaces pour MySQL**
8.4.3.33 mysql_list_dbs()....513
8.4.3.34 mysql_list_fields()....513
8.4.3.35 mysql_list_processes()....513
8.4.3.36 mysql_list_tables()....514
8.4.3.37 mysql_num_fields()....514
8.4.3.38 mysql_num_rows()....515
8.4.3.39 mysql_options()....516
8.4.3.40 mysql_ping()....517
8.4.3.41 mysql_query()....517
8.4.3.42 mysql_real_connect()....518
8.4.3.43 mysql_real_escape_string()....519
8.4.3.44 mysql_real_query()....519
8.4.3.45 mysql_reload()....520
8.4.3.46 mysql_row_seek()....520
8.4.3.47 mysql_row_tell()....521
8.4.3.48 mysql_select_db()....521
8.4.3.49 mysql_shutdown()....521
8.4.3.50 mysql_stat()....522
8.4.3.51 mysql_store_result()....522
8.4.3.52 mysql_thread_id()....523
8.4.3.53 mysql_use_result()....523
8.4.3.54 mysql_commit()....524
8.4.3.55 mysql_rollback()....524
8.4.3.56 mysql_autocommit()....524
8.4.3.57 mysql_more_results()....525
8.4.3.58 mysql_next_result()....525
8.4.4 Fonctions C de commandes préparées....525
8.4.5 C API Prepared Statements DataTypes....526
8.4.6 Présentation des API C pour les commandes préparées....527
8.4.7 Descriptions des fonctions C pour les requêtes préparées....528
8.4.7.1 mysql_prepare()....528
8.4.7.2 mysql_param_count()....528
8.4.7.3 mysql_prepare_result()....529
8.4.7.4 mysql_bind_param()....529
8.4.7.5 mysql_execute()....530
8.4.7.6 mysql_stmt_affected_rows()....532
8.4.7.7 mysql_bind_result()....532
8.4.7.8 mysql_stmt_store_result()....533
8.4.7.9 mysql_fetch()....534
8.4.7.10 mysql_send_long_data()....536
8.4.7.11 mysql_stmt_close()....537
8.4.7.12 mysql_stmt_errno()....537
8.4.7.13 mysql_stmt_error()....538
8.4.8 C API pour gérer les requêtes multiples....538
8.4.9 C API de gestions des DATE, TIME et TIMESTAMP....539
8.4.10 Description des fonctions threadées de C....539
8.4.10.1 my_init()....539
8.4.10.2 mysql_thread_init()....540
8.4.10.3 mysql_thread_end()....540
8.4.10.4 mysql_thread_safe()....540
8.4.11 Description des fonctions C du serveur embarqué....540

# Sommaire

**8 Les interfaces pour MySQL**
8.4.11.1 mysql_server_init() ....541
8.4.11.2 mysql_server_end() ....541
8.4.12 Questions courantes sur la librairie C ....541
8.4.12.1 Pourquoi est ce qu'après mysql_query() ait indiqué un résultat positif, mysql_store_result() retourne parfois NULL ? ....541
8.4.12.2 Quels résultats puis−je obtenir d'une requête? ....542
8.4.12.3 Comment obtenir l'identifiant unique du dernier enregistrement inséré ? ....542
8.4.12.4 Problèmes lors de la liaison avec l'API C ....542
8.4.13 Compiler les clients ....543
8.4.14 Comment faire un client MySQL threadé ....543
8.4.15 libmysqld, la librairie du serveur embarqué MySQL ....544
8.4.15.1 Vue d'ensemble de la librairie du serveur embarqué MySQL ....544
8.4.15.2 Compiler des programmes avec libmysqld ....544
8.4.15.3 Restrictions lors de l'utilisation du serveur embarqué MySQL ....544
8.4.15.4 Utilisation de fichiers d'options avec le serveur embarqué ....545
8.4.15.5 Choses à faire pour le serveur embarqué (TODO) ....545
8.4.15.6 Un exemple simple de serveur embarqué ....545
8.4.15.7 Licence du serveur embarqué ....547
8.5 Interfaces MySQL pour C++ ....547
8.5.1 Borland C++ ....547
8.6 Connectivité Java/MySQL (JDBC) ....547
8.7 Interface Python pour MySQL ....548
8.8 Interface Tcl pour MySQL ....548
8.9 Couche MySQL pour Eiffel ....548

**9 Etendre MySQL ....549**
9.1 Rouages de MySQL ....549
9.1.1 Threads MySQL ....549
9.1.2 Suite de test de MySQL ....549
9.1.2.1 Exécuter la suite de tests MySQL ....549
9.1.2.2 Améliorer la suite de tests MySQL ....550
9.1.2.3 Rapporter des bugs dans la suite de tests MySQL ....550
9.2 Ajouter des fonctions à MySQL ....551
9.2.1 Syntaxe de CREATE FUNCTION/DROP FUNCTION ....551
9.2.2 Ajouter une nouvelle fonction définie par l'utilisateur (UDF) ....551
9.2.2.1 Fonctions utiliateur : appeler des fonctions simples ....552
9.2.2.2 Appeler des fonctions utilisateurs pour les groupements ....553
9.2.2.3 Traitement des arguments ....554
9.2.2.4 Valeurs de retour et gestion d'erreurs ....554
9.2.2.5 Compiler et installer des fonctions utilisateurs ....555
9.2.3 Ajouter de nouvelles fonctions natives ....556
9.3 Ajouter une nouvelle procédure à MySQL ....556
9.3.1 La procédure Analyse ....557
9.3.2 Ecrire une procédure ....557

**10 Problèmes et erreurs communes ....558**
10.1 Comment déterminer ce qui pose problème ....558
10.2 Erreurs communes rencontrées avec MySQL ....558
10.2.1 Erreur Access denied ....559
10.2.2 Erreur MySQL server has gone away ....559

# Sommaire

**10 Problèmes et erreurs communes**
10.2.3 Erreur Can't connect to [local] MySQL server....559
10.2.4 Host '...' is blocked Error....560
10.2.5 Erreur Too many connections....561
10.2.6 Erreur Some non−transactional changed tables couldn't be rolled back....561
10.2.7 Erreur Out of memory....561
10.2.8 Erreur Packet too large....561
10.2.9 Erreurs de communication / Connexion annulée....562
10.2.10 Erreur The table is full....563
10.2.11 Erreur Can't create/write to file....563
10.2.12 Erreur du client Commands out of sync....563
10.2.13 Erreur Ignoring user....563
10.2.14 Erreur Table 'xxx' doesn't exist....564
10.2.15 Erreur Can't initialize character set xxx....564
10.2.16 Fichier non trouvé....564
10.3 Notes relatives à l'installation....565
10.3.1 Problèmes lors de la liaison avec la librairie du client MySQL....565
10.3.2 Comment exécuter MySQL comme un utilisateur normal....566
10.3.3 Problèmes avec les permissions sur fichiers....566
10.4 Notes relatives à l'administration....566
10.4.1 Que faire si MySQL crashe constamment ?....567
10.4.2 Comment réinitialiser un mot de passe Root oublié....568
10.4.3 Comment MySQL gère un disque plein....568
10.4.4 Où MySQL stocke les fichiers temporaires ?....568
10.4.5 Comment protéger ou changer le fichier socket /tmp/mysql.sock....569
10.4.6 Problèmes de fuseaux horaires....569
10.5 Problèmes relatifs aux requêtes....569
10.5.1 Sensibilité à la casse dans les recherches....570
10.5.2 Problèmes avec l'utilisation des colonnes DATE....570
10.5.3 Problèmes avec les valeurs NULL....571
10.5.4 Problèmes avec les alias....571
10.5.5 Effacer des lignes de tables reliées....572
10.5.6 Résoudre les problèmes des lignes non retournées....572
10.5.7 Problèmes de comparaisons avec nombres à virgule flottante....572
10.6 Questions relatives aux définitions de tables....574
10.6.1 Problèmes avec ALTER TABLE ....574
10.6.2 Comment changer l'ordre des colonnes dans une table....574
10.6.3 Problèmes avec les tables temporaires....574

**11 Contributions....576**

**12 Crédits....578**
12.1 Développeurs chez MySQL AB....578
12.2 Contributeurs à MySQL....579
12.3 Supporters de MySQL....581

**13 Historique des changements MySQL....583**
13.1 Changes in release 5.0.0 (Development)....583
13.2 Changes in release 4.1.x (Alpha)....583
13.2.1 Changes in release 4.1.0....583
13.3 Changements dans la publication 4.0.x (Beta)....584
13.3.1 Changes in release 4.0.13 (not released yet)....584

# Sommaire

**13 Historique des changements MySQL**
13.3.2 Changes in release 4.0.12 (15 Mar 2003: Production)....585
13.3.3 Changes in release 4.0.11 (20 Feb 2003)....585
13.3.4 Changes in release 4.0.10 (29 Jan 2003)....586
13.3.5 Changes in release 4.0.9 (09 Jan 2003)....586
13.3.6 Changes in release 4.0.8 (07 Jan 2003)....587
13.3.7 Changes in release 4.0.7 (20 Dec 2002)....587
13.3.8 Changes in release 4.0.6 (14 Dec 2002: Gamma)....587
13.3.9 Changes in release 4.0.5 (13 Nov 2002)....588
13.3.10 Changes in release 4.0.4 (29 Sep 2002)....588
13.3.11 Changes in release 4.0.3 (26 Aug 2002: Beta)....589
13.3.12 Changes in release 4.0.2 (01 Jul 2002)....590
13.3.13 Changes in release 4.0.1 (23 Dec 2001)....591
13.3.14 Changes in release 4.0.0 (Oct 2001: Alpha)....591
13.4 Changes in release 3.23.x (Recent; still supported)....592
13.4.1 Changes in release 3.23.57 (not released yet)....592
13.4.2 Changes in release 3.23.56 (13 Mar 2003)....592
13.4.3 Changes in release 3.23.55 (23 Jan 2003)....593
13.4.4 Changes in release 3.23.54 (05 Dec 2002)....593
13.4.5 Changes in release 3.23.53 (09 Oct 2002)....593
13.4.6 Changes in release 3.23.52 (14 Aug 2002)....594
13.4.7 Changes in release 3.23.51 (31 May 2002)....594
13.4.8 Changes in release 3.23.50 (21 Apr 2002)....594
13.4.9 Changes in release 3.23.49....595
13.4.10 Changes in release 3.23.48 (07 Feb 2002)....595
13.4.11 Changes in release 3.23.47 (27 Dec 2001)....595
13.4.12 Changes in release 3.23.46 (29 Nov 2001)....595
13.4.13 Changes in release 3.23.45 (22 Nov 2001)....595
13.4.14 Changes in release 3.23.44 (31 Oct 2001)....596
13.4.15 Changes in release 3.23.43 (04 Oct 2001)....596
13.4.16 Changes in release 3.23.42 (08 Sep 2001)....596
13.4.17 Changes in release 3.23.41 (11 Aug 2001)....596
13.4.18 Changes in release 3.23.40....597
13.4.19 Changes in release 3.23.39 (12 Jun 2001)....597
13.4.20 Changes in release 3.23.38 (09 May 2001)....597
13.4.21 Changes in release 3.23.37 (17 Apr 2001)....597
13.4.22 Changes in release 3.23.36 (27 Mar 2001)....598
13.4.23 Changes in release 3.23.35 (15 Mar 2001)....598
13.4.24 Changes in release 3.23.34a....598
13.4.25 Changes in release 3.23.34 (10 Mar 2001)....598
13.4.26 Changes in release 3.23.33 (09 Feb 2001)....599
13.4.27 Changes in release 3.23.32 (22 Jan 2001: Stable)....599
13.4.28 Changes in release 3.23.31 (17 Jan 2001)....599
13.4.29 Changes in release 3.23.30 (04 Jan 2001)....600
13.4.30 Changes in release 3.23.29 (16 Dec 2000)....600
13.4.31 Changes in release 3.23.28 (22 Nov 2000: Gamma)....600
13.4.32 Changes in release 3.23.27 (24 Oct 2000)....601
13.4.33 Changes in release 3.23.26 (18 Oct 2000)....601
13.4.34 Changes in release 3.23.25 (29 Sep 2000)....602
13.4.35 Changes in release 3.23.24 (08 Sep 2000)....602
13.4.36 Changes in release 3.23.23 (01 Sep 2000)....602
13.4.37 Changes in release 3.23.22 (31 Jul 2000)....603

# Sommaire

**13 Historique des changements MySQL**
13.4.38 Changes in release 3.23.21 ... 603
13.4.39 Changes in release 3.23.20 ... 603
13.4.40 Changes in release 3.23.19 ... 603
13.4.41 Changes in release 3.23.18 ... 603
13.4.42 Changes in release 3.23.17 ... 604
13.4.43 Changes in release 3.23.16 ... 604
13.4.44 Changes in release 3.23.15 (May 2000: Beta) ... 604
13.4.45 Changes in release 3.23.14 ... 604
13.4.46 Changes in release 3.23.13 ... 605
13.4.47 Changes in release 3.23.12 (07 Mar 2000) ... 605
13.4.48 Changes in release 3.23.11 ... 605
13.4.49 Changes in release 3.23.10 ... 605
13.4.50 Changes in release 3.23.9 ... 605
13.4.51 Changes in release 3.23.8 (02 Jan 2000) ... 606
13.4.52 Changes in release 3.23.7 (10 Dec 1999) ... 606
13.4.53 Changes in release 3.23.6 ... 606
13.4.54 Changes in release 3.23.5 (20 Oct 1999) ... 607
13.4.55 Changes in release 3.23.4 (28 Sep 1999) ... 607
13.4.56 Changes in release 3.23.3 ... 607
13.4.57 Changes in release 3.23.2 (09 Aug 1999) ... 607
13.4.58 Changes in release 3.23.1 ... 608
13.4.59 Changes in release 3.23.0 (05 Aug 1999: Alpha) ... 608
13.5 Changes in release 3.22.x (Older; still supported) ... 608
13.5.1 Changes in release 3.22.35 ... 609
13.5.2 Changes in release 3.22.34 ... 609
13.5.3 Changes in release 3.22.33 ... 609
13.5.4 Changes in release 3.22.32 (14 Feb 2000) ... 609
13.5.5 Changes in release 3.22.31 ... 609
13.5.6 Changes in release 3.22.30 ... 609
13.5.7 Changes in release 3.22.29 (02 Jan 2000) ... 609
13.5.8 Changes in release 3.22.28 (20 Oct 1999) ... 609
13.5.9 Changes in release 3.22.27 ... 609
13.5.10 Changes in release 3.22.26 (16 Sep 1999) ... 610
13.5.11 Changes in release 3.22.25 ... 610
13.5.12 Changes in release 3.22.24 (05 Jul 1999) ... 610
13.5.13 Changes in release 3.22.23 (08 Jun 1999) ... 610
13.5.14 Changes in release 3.22.22 (30 Apr 1999) ... 610
13.5.15 Changes in release 3.22.21 ... 610
13.5.16 Changes in release 3.22.20 (18 Mar 1999) ... 610
13.5.17 Changes in release 3.22.19 (Mar 1999: Stable) ... 610
13.5.18 Changes in release 3.22.18 ... 611
13.5.19 Changes in release 3.22.17 ... 611
13.5.20 Changes in release 3.22.16 (Feb 1999: Gamma) ... 611
13.5.21 Changes in release 3.22.15 ... 611
13.5.22 Changes in release 3.22.14 ... 611
13.5.23 Changes in release 3.22.13 ... 611
13.5.24 Changes in release 3.22.12 ... 611
13.5.25 Changes in release 3.22.11 ... 612
13.5.26 Changes in release 3.22.10 ... 612
13.5.27 Changes in release 3.22.9 ... 612
13.5.28 Changes in release 3.22.8 ... 612

# Sommaire

**13 Historique des changements MySQL**
13.5.29 Changes in release 3.22.7 (Sep 1998: Beta) ..... 613
13.5.30 Changes in release 3.22.6 ..... 613
13.5.31 Changes in release 3.22.5 ..... 613
13.5.32 Changes in release 3.22.4 ..... 613
13.5.33 Changes in release 3.22.3 ..... 614
13.5.34 Changes in release 3.22.2 ..... 614
13.5.35 Changes in release 3.22.1 (Jun 1998: Alpha) ..... 614
13.5.36 Changes in release 3.22.0 ..... 614
13.6 Changes in release 3.21.x ..... 615
13.6.1 Changes in release 3.21.33 ..... 615
13.6.2 Changes in release 3.21.32 ..... 615
13.6.3 Changes in release 3.21.31 ..... 615
13.6.4 Changes in release 3.21.30 ..... 615
13.6.5 Changes in release 3.21.29 ..... 616
13.6.6 Changes in release 3.21.28 ..... 616
13.6.7 Changes in release 3.21.27 ..... 616
13.6.8 Changes in release 3.21.26 ..... 616
13.6.9 Changes in release 3.21.25 ..... 616
13.6.10 Changes in release 3.21.24 ..... 616
13.6.11 Changes in release 3.21.23 ..... 617
13.6.12 Changes in release 3.21.22 ..... 617
13.6.13 Changes in release 3.21.21a ..... 617
13.6.14 Changes in release 3.21.21 ..... 617
13.6.15 Changes in release 3.21.20 ..... 617
13.6.16 Changes in release 3.21.19 ..... 617
13.6.17 Changes in release 3.21.18 ..... 618
13.6.18 Changes in release 3.21.17 ..... 618
13.6.19 Changes in release 3.21.16 ..... 618
13.6.20 Changes in release 3.21.15 ..... 618
13.6.21 Changes in release 3.21.14b ..... 618
13.6.22 Changes in release 3.21.14a ..... 618
13.6.23 Changes in release 3.21.13 ..... 619
13.6.24 Changes in release 3.21.12 ..... 619
13.6.25 Changes in release 3.21.11 ..... 619
13.6.26 Changes in release 3.21.10 ..... 619
13.6.27 Changes in release 3.21.9 ..... 620
13.6.28 Changes in release 3.21.8 ..... 620
13.6.29 Changes in release 3.21.7 ..... 620
13.6.30 Changes in release 3.21.6 ..... 620
13.6.31 Changes in release 3.21.5 ..... 620
13.6.32 Changes in release 3.21.4 ..... 620
13.6.33 Changes in release 3.21.3 ..... 620
13.6.34 Changes in release 3.21.2 ..... 621
13.6.35 Changes in release 3.21.0 ..... 621
13.7 Changes in release 3.20.x ..... 621
13.7.1 Changes in release 3.20.18 ..... 621
13.7.2 Changes in release 3.20.17 ..... 622
13.7.3 Changes in release 3.20.16 ..... 622
13.7.4 Changes in release 3.20.15 ..... 622
13.7.5 Changes in release 3.20.14 ..... 622
13.7.6 Changes in release 3.20.13 ..... 622

# Sommaire

**13 Historique des changements MySQL**
13.7.7 Changes in release 3.20.11 ....623
13.7.8 Changes in release 3.20.10 ....623
13.7.9 Changes in release 3.20.9 ....623
13.7.10 Changes in release 3.20.8 ....623
13.7.11 Changes in release 3.20.7 ....623
13.7.12 Changes in release 3.20.6 ....623
13.7.13 Changes in release 3.20.3 ....624
13.7.14 Changes in release 3.20.0 ....624
13.8 Changes in release 3.19.x ....624
13.8.1 Changes in release 3.19.5 ....625
13.8.2 Changes in release 3.19.4 ....625
13.8.3 Changes in release 3.19.3 ....625

**14 Port vers d'autres systèmes ....626**
14.1 Déboguer un serveur MySQL ....626
14.1.1 Compiler MYSQL pour le débogage ....627
14.1.2 Créer un fichier de traçage ....628
14.1.3 Déboguer mysqld sous gdb ....628
14.1.4 Utilisation d'un traçage de pile mémoire ....629
14.1.5 Utilisation des fichiers de log pour trouver d'où viennent les erreurs de mysqld ....629
14.1.6 Faire une batterie de tests lorsque vous faites face à un problème de table corrompue ....630
14.2 Débogage un client MySQL ....630
14.3 Le package DBUG ....631
14.4 Méthodes de verrouillage ....631
14.5 Commentaires à propos des threads RTS ....633
14.6 Différences entre les différents packages de threads ....634

**15 Variables d'environnement ....635**

**16 Expressions régulières MySQL ....636**

**17 Licence Publique Générale GNU ....638**

**18 Licence Publique Générale GNU Limitée ....642**

**19 Index des commandes, types et fonctions SQL ....647**

**20 Index conceptuel ....650**

# 1 Informations générales

Le logiciel MySQL (TM) est un serveur de base de données SQL très rapide, multi−threadé, multi−utilisateur et robuste. Le serveur MySQL est destiné aux missions stratégiques et aux systèmes de production à forte charge, ainsi qu'à l'intégration dans des logiciels déployés à grande échelle. MySQL est une marque déposée de MySQL AB .

Le logiciel MySQL dispose de deux licenses . Les utilisateurs peuvent choisir entre utiliser MySQL comme un logiciel Open Source / Logiciel libre , sous les termes de la licence GNU General Public License ( http://www.gnu.org/licenses/ ) ou bien, ils peuvent acheter une licence commerciale auprès de MySQL AB . MySQL Support and Licensing .

Le site web de MySQL ( http://www.mysql.com/ ) fournit les dernières informations sur le serveur MySQL .

La liste suivante décrit les sections particulières de ce manuel :

- Pour plus d'informations sur la société derrière le serveur de base de données MySQL , voyez Qu'est ce que MySQL AB ? .
- Pour une présentation des capacités de serveur de base de données MySQL , voyez Principales fonctionnalités de MySQL .
- Pour les instructions d'installation, voyez Installation de MySQL .
- Pour des conseils sur le port du serveur de base de données MySQL sur de nouvelles architectures ou systèmes d'exploitation, voyez Port .
- Pour des informations sur la mise à jour vers la version 3.23, voyez Upgrading From Version 3.23 to 4.0 .
- Pour des informations sur la mise à jour vers la version 3.22, voyez Upgrading From Version 3.22 to 3.23 .
- Pour une introduction au serveur de base de données MySQL , voyez Tutoriel MySQL .
- Pour des exemples de SQL et des tests de performances, voyez le dossier de tests ( sql−bench de la distribution.
- Pour connaître l'historique des fonctionnalités et bogues, voyez Historique des versions de MySQL .
- Pour une liste des bogues connus et des limitations, voyez Erreurs connues et manques de MySQL .
- Pour les plans de développement, voyez MySQL and The Future (The TODO) .
- Pour une liste de tous les contributeurs à ce projet, voyez Contributeurs .

**Important** : Les rapports d'erreurs (aussi appelés bogues), ainsi que les questions et commentaires, doivent être envoyés à la liste de diffusion mysql@lists.mysql.com . Comment rapporter des bogues ou problèmes .

Le script mysqlbug doit être utilisé pour générer le rapport de bogues. Pour les distributions sources, le script mysqlbug est accessible dans le dossier scripts . Pour les distributions binaires, mysqlbug est installé dans le dossier bin ( /usr/bin pour le package RPM du serveur MySQL ).

Si vous avez trouvé un problème de sécurité critique dans le code du serveur MySQL , vous devez envoyez un email à security@mysql.com .

## 1.1 A propos du manuel

Ceci est le manuel de référence de MySQL ; il documente MySQL jusqu'à la version 4.0.5a. Les évolutions fonctionnelles sont toujours indiquées avec une référence à la version d'évolution, de manière à ce

que ce manuel soit toujours valable, même si vous utilisez une ancienne version de MySQL . Etant un manuel de référence, il ne fournit aucune description générale sur le langage SQL ou les concepts de bases de données relationnelles.

Comme le logiciel de base de données MySQL est en développement constant, ce manuel es mis à jour fréquemment. La version la plus récente est disponibles à http://www.mysql.com/documentation/ en différents formats, incluant HTML, PDF et Windows HLP.

L'original du document est un fichier au format Texinfo. La version HTML est produite automatiquement avec une version modifiée de texi2html . La version en texte plein et version Info sont produites par makeinfo . La version PostScript est produite avec texi2dvi et dvips . La version PDF est produite avec pdftex .

Si vous avez du mal à trouver des informations dans ce manuel, vous pouvez essayer notre version avec moteur de recherche, sur notre site web : http://www.mysql.com/doc/ .

Si vous avez des suggestions concernant des ajouts ou des corrections à ce manuel, vous pouvez les envoyez à l'équipe de documentation à docs@mysql.com .

Ce manuel a été écrit initialement par David Axmark et Michael (Monty) Widenius. Il est actuellement entretenu par Michael (Monty) Widenius, Arjen Lentz et Paul DuBois. Pour les autres contributeurs, voyez les Contributeurs .

La traduction de ce manuel a été faite sous la direction de Damien Seguy, et organisée par Mehdi Achour. David Manusset, Guillaume Plessis, Patrick Haond, Sylvain Maugiron et Yannick Torres ont contribué largement à cette traduction et son entretien.

Le copyright (2002) de ce manuel est la propriété de la société suédoise MySQL AB . Copyrights et licenses utilisées par MySQL .

## 1.1.1 Conventions utilisées dans ce manuel

Ce manuel utilise certaines conventions typographiques :

`constant`
: La police à largeur fixe est utilisée pour les noms de commandes et les options, les requêtes SQL, les noms de bases de données, de tables et de colonnes, le code C et Perl, les variables d'environnement. Par exemple, ``Pour voir comment mysqladmin fonctionne, exécutez−le avec l'option --help ."

`filename`
: La police à largeur fixe avec des guillemets d'encadrement indique des noms de fichiers et de chemins de dossiers. Par exemple : ``La distribution est installée dans le dossier /usr/local/ ."

`'c'`
: La police à largeur fixe avec des guillemets d'encadrement est aussi utilisée pour indiquer des séquences de caractères. Par exemple : ``Pour spécifier un caractère joker, utilisez le caractère '%' ."

***italic***
: Les polices en italique sont utilisées pour attirer l'attention, **comme ceci** .

***boldface***
: Le gras est utilisé pour les entêtes de tables, et aussi pour **attirer fortement votre attention** .

Lorsque les commandes qui sont affichées sont destinées à être exécutées par un programme particulier, le nom du programme est indiqué dans l'invite de la commande. Par exemple, shell>

indique une commande que vous exécutez depuis votre console shell, et `mysql>` indique une commande que vous exécutez depuis le client `mysql` :

```
shell> tapez une commande shell ici
mysql> tapez une requête SQL ici
```

Les commandes shell sont affichées en utilisant la syntaxe du Bourne. Si vous utilisez le style `csh` −shell, vous aurez peut être a adapter légèrement les commandes. Par exemple, la syntaxe de modification d'une variable et d'exécution d'une commande ressemble à ceci en Bourne shell :

```
shell> NOMVAR=valeur une_commande
```

En `csh` , vous auriez a exécuter la commande suivante :

```
shell> setenv NOMVAR valeur
shell> une_commande
```

Souvent, les noms de bases de données, tables ou colonnes doivent être remplacés dans les commandes. Pour indiquer qu'une telle substitution est nécessaire, ce manuel utilise les noms de `nom_de_base` , `nom_de_table` et `nom_colonne` . Par exemple, vous pourriez avoir une requête comme ceci :

```
mysql> SELECT nom_colonne FROM nom_de_base.nom_de_table;
```

Cela signifie que si vous devez saisir une requête semblable, vous devriez utiliser votre propre nom de colonne, table et base de données, ce qui pourrait se traduire par ceci :

```
mysql> SELECT author_name FROM biblio_db.author_list;
```

Les mot réservés SQL ne sont pas sensibles à la casse, et peuvent être écrits en majuscules ou minuscules. Ce manuel utilise les majuscules.

Dans les illustrations de syntaxe, les crochets ( `'['` et `']'` ) sont utilisés pour indiquer des clauses ou mots optionnels. Par exemple, dans la requête suivante, `IF EXISTS` est optionnel :

```
DROP TABLE [IF EXISTS] nom_de_table
```

Lorsqu'un élément de syntaxe est constitué d'un certain nombre d'alternatives, les alternatives sont séparées par des barres verticales ( `'|'` ). Lorsqu'un membre d'un tel jeu de possibilités **peut** être choisi, les alternatives sont listées entre crochets ( `'['` et `']'` ):

```
TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)
```

Lorsqu'un élément d'un jeu de possibilités **doit** être choisi, les alternatives sont placées entre accolades ( `'{'` et `''` }):

```
&#123;DESCRIBE | DESC&#125; nom_de_table &#123;nom_colonne | wild&#125;
```

# 1.2 Qu'est ce que MySQL?

`MySQL` , le plus populaire des serveurs de bases de données SQL `Open Source` , est développé, distribué et supporté par `MySQL AB` . `MySQL AB` est une société commerciale, fondée par les développeurs de MySQL, qui développent leur activité en fournissant des services autour de `MySQL` . Qu'est ce que MySQL AB ?

Le site web de `MySQL` ( http://www.mysql.com/ ) fournit les toutes dernières actualités sur le logiciel `MySQL` et sur la société `MySQL AB` .

`MySQL` *est un système de gestion de bases de données.*

Une base de données est un ensemble organisé de données. Cela peut aller d'une simple liste de courses au supermarché à une galerie de photos, ou encore les grands systèmes d'informations des multi−nationales. Pour ajouter, lire et traiter des données dans une base de données, vous avez besoin d'un système de gestion de bases de données tel que le serveur `MySQL` . Comme les ordinateurs sont très bons à manipuler de grandes quantités de données, le système de gestion de bases de données joue un rôle central en informatique, aussi bien en tant qu'application à part

entière, qu'intégré dans d'autres logiciels.

*MySQL est un serveur de bases de données relationnel.*

Un serveur de bases de données stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table. Cela améliore la rapidité et la souplesse de l'ensemble. Les tables sont reliées par des relations définies, qui rendent possible la combinaison de données entre plusieurs tables durant une requête. Le SQL dans `` MySQL '' signifie `` Structured Query Language '' : le langage standard pour les traitements de bases de données.

*MySQL est Open Source .*

Open Source (Standard Ouvert) signifie qu'il est possible à chacun d'utiliser et de modifier le logiciel. Tout le monde peut télécharger MySQL sur Internet, et l'utiliser sans payer aucun droit. Toute personne en ayant la volonté peut étudier et modifier le code source pour l'adapter à ses besoins propres. Le logiciel MySQL utilise la licence GPL ( GNU General Public License ), http://www.gnu.org/licenses/ , pour définir ce que vous pouvez et ne pouvez pas faire avec ce logiciel, dans différentes situations. Si vous ne vous sentez pas confortable avec la licence GPL ou bien que vous devez intégrer MySQL dans une application commerciale, vous pouvez acheter une licence commercial auprès de MySQL AB. Licences MySQL .

*Pourquoi utiliser le serveur de bases de données MySQL ?*

Le serveur de bases de données MySQL est très rapide, fiable et facile à utiliser. Si c'est ce que vous recherchez, vous devriez faire un essai. Le serveur de bases de données MySQL dispose aussi de fonctionnalités pratiques, développées en coopération avec nos utilisateurs. Vous pouvez trouver une comparaison des performances du serveur MySQL avec d'autres systèmes de bases de données dans nos pages de tests de performances. La suite de tests comparatifs de MySQL .Le serveur MySQL a été développé à l'origine pour gérer de grandes bases de données plus rapidement que les solutions existantes, et a été utilisé avec succès dans des environnements de production très contraints et très exigeants, depuis plusieurs années. Bien que toujours en développement, le Le serveur MySQL offre des fonctions nombreuses et puissantes. Ses possibilités de connexions, sa rapidité et sa sécurité font du serveur MySQL une serveur hautement adapté à Internet.

*Les caractéristiques techniques du serveur MySQL .*

Pour des détails techniques avancés, voyez Référence du langage MySQL . Le logiciel de bases de données MySQL est un système client/serveur, constitué d'un serveur SQL multi−threadé qui supporte différents systèmes de stockage, plusieurs logiciels clients et librairies, outils d'administration, ainsi que de nombreuses interfaces de programmation (des API ).Nous fournissons aussi le serveur MySQL sous la forme d'une librairie multi−threadé que vous pouvez inclure dans vos applications, pour obtenir un produit plus compact, plus rapide et plus facile à gérer.

*Il existe un grand nombre de contributions à MySQL .*

Il est très probable que vous pourrez trouver votre éditeur préféré ou que votre environnement de programmation supporte déjà le serveur de base de données MySQL .

La prononciation officielle de MySQL est ``My Ess Que Ell'' (en anglais), ce qui donne ``Maille Esse Cu Elle'' en phonétique française. Evitez d'utiliser la prononciation ``my sequel'', mais nous ne nous formaliserons pas que vous utilisiez ``my sequel'' ou une autre prononciation adaptée.

## 1.2.1 Histoire de MySQL

Nous avons débuté avec l'intention d'utiliser mSQL pour se connecter à nos tables en utilisant nos propres routines bas niveau ISAM. Cependant, après quelques tests, nous sommes arrivés à la conclusion que mSQL n'était pas assez rapide et flexible pour nos besoins. Cela nous a conduit à créer une nouvelle interface SQL pour notre base de données, mais en gardant la même API que mSQL .

Cette API a été choisie pour la facilité de port des programmes de tiers.

Les liens avec le nom MySQL ne sont pas parfaitement établis. Notre dossier de base et un grand nombre de librairies et outils étaient préfixés par ``my'' depuis plus de 10 ans. Mais la fille de Monty, plus jeune que lui, était aussi appelée My. Lequel des deux a conduit au nom de MySQL est toujours un mystère, même pour nous.

Le nom du dauphin MySQL (notre logo) est Sakila , qui a été choisi par les fondateurs de MySQL AB à partir d'une grande liste de noms suggérés par les utilisateurs dans le concours "Name the Dolphin" ("Nommez le dauphin"). Le nom a été suggéré par Ambrose Twebaze, un développeur de softwares open source de Swaziland, Afrique. D'après Ambrose, le nom Sakila puise ses origines du SiSwati, la langue locale du Swaziland. Sakila est aussi le nom d'une ville dans Arusha, Tanzanie, près du pays d'origine d'Ambrose, Uganda.

## 1.2.2 Les fonctionnalités principales de MySQL

La liste suivante décrit les caractéristiques principales du logiciel de bases de données MySQL . MySQL 4.0 In A Nutshell .

*Interne et portabilité*

Ecrit en C et C++. Testé sur un large éventail de compilateurs différents.
Fonctionne sur de nombreuses plate−formes. Operating Systems Supported by MySQL .
Utilise GNU Automake, Autoconf et Libtool pour une meilleure portabilité.
Dispose d'API pour C, C++, Eiffel, Java, Perl, PHP, Python, Ruby et Tcl.
Outils du client MySQL et API .
Complètement multi−threadé, grâce aux threads du noyau. Cela signifie que vous pouvez l'utiliser facilement sur un serveur avec plusieurs processeurs.
Tables B−tree très rapide, avec compression d'index.
Système l'allocation mémoire très rapide, exploitant les threads.
Jointures très rapides, exploitant un système de jointures multiples en une seule passe optimisé.
Tables en mémoire, pour réaliser des tables temporaires.
Les fonctions SQL sont implémentées grâce à une librairie de classes optimisées, qui sont aussi rapides que possible! Généralement, il n'y a aucune allocation mémoire une fois que la requête a été initialisée.
Le code de MySQL est vérifié avec Purify (un utilitaire de détection des fuites mémoires commercial) ainsi qu'avec Valgrind, un outil GPL ( http://developer.kde.org/~sewardj/ ).

*Column Types*

Nombreux types de colonnes : entiers signés ou non, de 1, 2, 3, 4, et 8 octets, FLOAT , DOUBLE , CHAR , VARCHAR , TEXT , BLOB , DATE , TIME , DATETIME , TIMESTAMP , YEAR , SET et ENUM . Types de colonnes .
Enregistrements de taille fixe ou variable.
Toutes les colonnes ont des valeurs par défaut. Vous pouvez utiliser la commande INSERT pour insérer un sous ensemble de colonnes : les colonnes qui ne sont pas explicitement cités prennent alors leur valeur par défaut.

*Commandes et fonctions*

Support complet des opérateurs et fonctions dans la commande SELECT et la clause WHERE . Par exemple :

```
mysql> SELECT CONCAT(first_name, " ", last_name)
   -> FROM tbl_name
   -> WHERE income/dependents > 10000 AND age > 30;
```

Support complet des clauses SQL GROUP BY et ORDER BY . Support des fonctions de

groupages ( COUNT() , COUNT(DISTINCT ...) , AVG() , STD() , SUM() , MAX() et MIN() ).
Support des clauses LEFT OUTER JOIN et RIGHT OUTER JOIN avec les syntaxes ANSI SQL et ODBC.
Les alias de tables et colonnes sont compatibles avec le standard SQL92.
DELETE , INSERT , REPLACE et UPDATE retourne le nombre de lignes affectées. Il est possible d'obtenir le nombre de lignes trouvées en modifiant une option lors de la connexion au serveur.
La commande spécifique à MySQL SHOW est utilisée pour obtenir des informations sur les bases, tables et index. La commande EXPLAIN sert à optimiser les requêtes.
Les noms de fonctions ne sont jamais en conflit avec les noms de tables ou colonnes. Par exemple, ABS est un nom de colonne valide. La seule restriction est que, lors d'un appel de fonction, aucun espace n'est toléré entre le nom de la fonction et la parenthèse ouvrante '(' suivante. Est−ce que MySQL est sensible aux mots réservés ? .
Vous pouvez utiliser simultanément des tables de différentes bases (depuis la version 3.22).

*Securité*

Un système de droits et de mots de passe très souple et sécuritaire, qui vérifie aussi les hôtes se connectant. Les mots de passe sont bien protégés, car tout les échanges de mot de passe sont chiffrés, même lors des connexions.

*Charges supportées et limites*

Gère les très grandes bases de données. Nous utilisons le serveur MySQL avec des bases qui contiennent 50 millions de lignes et nous connaissons des utilisateurs qui utilisent le serveur MySQL avec plus de 60 000 tables and et 5 000 000 000 (milliards) de lignes.
Jusqu'à 32 index sont permis par table. Chaque index est constitué de 1 à 16 colonnes ou parties de colonnes. La taille maximale d'un index est de 500 octets (ce qui peut être configuré à la compilation du serveur MySQL . Un index peut utiliser un préfixe issu d'un champs CHAR ou VARCHAR .

*Connectivité*

Les clients peuvent se connecter au serveur MySQL en utilisant les sockets TCP/IP, les sockets Unix (Unix) ou les Named Pipes (NT).
Support de ODBC (Open−DataBase−Connectivity) pour Win32 (avec les sources). Toutes les fonctions ODBC 2.5 et de nombreuses autres. Par exemple, vous pouvez utiliser MS Access pour vous connecter au serveur MySQL . Support ODBC pour MySQL .

*Traductions*

Le serveur fournit des messages d'erreurs au client dans de nombreuses langues, y compris le français. Messages d'erreurs non−anglophones .
Support complet de plusieurs jeux de caractères, comprenant ISO−8859−1 (Latin1), german, big5, ujis, and more. Par exemple, les caractères nordiques 'Â', 'ä' et 'ö' sont autorisés dans les noms de tables et colonnes.
Toutes les données sont sauvées dans le jeu de caractères choisi. Les comparaisons normales de chaînes sont insensibles à la casse.
Le tri est fait en fonction du jeu de caractères choisi (par défaut, le jeu suédois). Il est possible de le changer lorsque le serveur MySQL est démarré. Pour voir un exemple très avancé de tri, voyez le code de tri pour le Tchèque.
Le serveur MySQL supporte de nombreux jeux de caractères qui peuvent être spécifié à la compilation et durant l'exécution.

*Clients et utilitaires*

Inclut myisamchk , un utilitaire rapide pour vérifier les tables, les optimiser et les réparer. Toutes les fonctionnalités de myisamchk sont aussi disponibles via

l'interface SQL. Administration de la base .
Tous les programmes MySQL peuvent être appelés avec l'option --help ou -? pour obtenir de l'aide en ligne.

## 1.2.3 Jusqu'à quel point MySQL est il stable ?

Cette section répond aux questions `` **Jusqu'à quel point MySQL est il stable ?** '' et `` **Puis−je faire confiance à MySQL pour mon projet ?** '' Nous allons tenter d'apporter des réponses claires à ces questions importantes qui concernent tous les utilisateurs potentiels. Les informations de cette section sont fournies par les listes de diffusions, qui sont très actives et promptes à identifier les problèmes et les rapporter.

Le code original date du début des années 80 et fournit une base de code stable, tout en assurant une compatibilité ascendante avec le format ISAM. A TcX, le prédécesseur de MySQL AB , le code de MySQL a fonctionné sur des projets depuis la mi 1996, sans aucun problème. Lorsque le Serveur MySQL a été livré à un public plus large, nous avons réalisé qu'il contenait du code ``jamais testé'' qui a été rapidement identifié par les utilisateurs, qui effectuait des requêtes différentes des nôtres. Chaque nouvelle version avait moins de problèmes de portabilité, même si chaque nouvelle version avait de nombreuses nouvelles fonctionnalités.

Chaque version du Serveur MySQL était parfaitement fonctionnelle. Les seuls problèmes étaient rencontrés par les utilisateurs de code de ces ``zone d'ombres''. Naturellement, les nouveaux utilisateurs ne connaissent pas ces zones : cette section tente de les présenter, dans la mesure de nos connaissances. Les descriptions correspondent surtout aux versions 3.23 du Serveur MySQL . Tous les bogues connus et rapportés ont été corrigés dans la dernière version, à l'exception de ceux qui sont listés dans la section Bugs, qui sont des problèmes de conception. Erreurs connues et manques de MySQL .

La conception du serveur MySQL est faite en plusieurs couches, avec des modules indépendants. Certains des modules les plus récents sont listés ici, avec leur niveau de test :

*Réplication −− Gamma*
: De grands serveurs en grappe utilisant la réplication sont en production, avec de bons résultats. L'amélioration de la réplication continue avec MySQL 4.x.

*Tables InnoDB −− Stable (en 3.23 depuis 3.23.49)*
: Le gestionnaire transactionnel de tables InnoDB a été déclaré stable en MySQL version 3.23, à partir de la version 3.23.49. InnoDB est utilisé dans de grands systèmes complexes, avec forte charge.

*Tables BDB −− Gamma*
: Le code de Berkeley DB est très stable, mais nous sommes encore en train d'améliorer l'interface du gestionnaire transactionnel de table BDB du serveur MySQL . Cela demande encore du temps pour qu'il soit aussi bien testé que les autres types de tables.

FULLTEXT *−− Beta*
: La recherche en texte plein fonctionne mais n'est pas encore largement adoptée. Des améliorations importantes sont prévues pour MySQL 4.0.

MyODBC 2.50 *(utilise ODBC SDK 2.5) −− Gamma*
: En utilisation croissante. Certains problèmes sont apparus avec des applications tierces, et indépendemment du pilote ODBC ou du serveur utilisé.

*Tables à restauration automatique MyISAM −− Gamma*
: Ce statut ne concerne que le nouveau code du gestionnaire de tables MyISAM qui vérifie si la table a été correctement fermée lors de l'ouverture, et qui exécute automatiquement la vérification et réparation éventuelles de la table.

*Insertions de masse −− Alpha*

Nouvelle fonctionnalité des tables MyISAM pour MySQL 4.0 qui permet des insertions plus rapides.

*Verrouillage −− Gamma*

Cette fonctionnalité est très dépendante du système. Sur certains systèmes, il y a de gros problèmes lors de l'utilisation du verrouillage système (avec la fonction fcntl() ). Dans ces cas, il faut utiliser mysqld avec l'option --skip-external-locking . Les problèmes sont connus sur certaines distributions Linux, et sur SunOS lorsqu'il est utilisé avec des disques en mode NFS.

MySQL AB fournit un support de première qualité pour les clients payant, mais les listes de diffusions de MySQL sont généralement rapides à donner des réponses aux questions les plus communes. Les bogues sont généralement corrigés aussitôt avec un patch. Pour les bogues sérieux, il y a presque toujours une nouvelle version.

## 1.2.4 Quelles tailles de tables supporte MySQL ?

MySQL version 3.22 a une limite de 4Go par table. Avec le nouveau format de table MyISAM , disponible avec MySQL version 3.23, la taille maximale des tables a été poussée à 8 millions de teraoctets (2 ^ 63 octets).

Notez, toutefois, que les systèmes d'exploitation ont leur propres limites. Voici quelques exemples :

| **Système d'exploitation** | **Limite** |
|---|---|
| Linux-Intel 32 bit | 2Go, 4Go ou plus, suivant la version de Linux |
| Linux-Alpha | 8To (?) |
| Solaris 2.5.1 | 2Go (4Go possibles avec un patch) |
| Solaris 2.6 | 4Go (peut être modifié avec une option) |
| Solaris 2.7 Intel | 4Go |
| Solaris 2.7 UltraSPARC | 512Go |

En Linux 2.2, vous pouvez avoir des tables plus grandes que 2Go en utilisant le patch LFS pour les systèmes de fichiers ext2. En Linux 2.4, le patche existe aussi pour ReiserFS.

Cela signifie que les tables MySQL sont généralement limitées par le système d'exploitation.

Par défaut, les tables MySQL peuvent atteindre une taille de 4Go. Vous pouvez vérifier la taille des tables avec la commande SHOW TABLE STATUS ou la commande en ligne myisamchk -dv nom_de_table . Syntaxe de SHOW .

Si vous avez besoin de tables plus grandes que 4Go (et que votre système d'exploitation le supporte, modifiez les paramètres AVG_ROW_LENGTH et MAX_ROWS lorsque vous créez votre table. Syntaxe de CREATE TABLE . Vous pouvez aussi les modifier ultérieurement avec ALTER TABLE . Syntaxe de ALTER TABLE .

Si vos tables sont accessibles uniquement en lecture, vous pouvez aussi utiliser l'utilitaire myisampack pour rassembler et compresser plusieurs tables en une seule. myisampack compresse généralement la table de près de 50%, ce qui vous augmente d'autant la taille maximale de la table. myisampack , le générateur de tables MySQL compressées en lecture seule .

Vous pouvez aussi contourner les limites du système d'exploitation avec les tables MyISAM , en utilisant l'option RAID . Syntaxe de CREATE TABLE .

Une autre solution est d'utiliser la librairie MERGE , qui permet de gérer plusieurs tables comme une seule. Tables MERGE .

## 1.2.5 Compatibilité an 2000

Le serveur MySQL lui même n'a aucun problème de compatibilité avec l'an 2000 (Y2K) :

- Le serveur MySQL utilise les fonctions de date Unix, et n'a aucun problème avec les dates jusqu'en 2069 ; toutes les années écrites en deux chiffres sont supposées faire partie de l'intervalle allant de 1970 à 2069 , ce qui signifie que si vous stockez la date 01 dans une colonne de type year , le serveur MySQL la traitera comme 2001 .
- Toutes les fonctions de dates de MySQL sont stockées dans un fichier sql/time.cc , et sont codées très soigneusement pour être compatibles avec l'an 2000.
- En MySQL version 3.22 et plus récent, le type de colonne YEAR peut stocker les valeurs 0 et de 1901 à 2155 sur un seul octet, tout en affichant 2 ou 4 chiffres.

Vous pouvez rencontrer des problèmes avec les applications qui utilisent le serveur MySQL sans être compatible avec l'an 2000. Par exemple, les vieilles applications utilisent des valeurs d'années sur deux chiffres (ce qui est ambigu), plutôt qu'avec 4 chiffres. Ce problème peut être complété par des applications qui utilisent des valeurs telles que 00 ou 99 comme indicateur de données ``manquante''. Malheureusement, ces problèmes peuvent se révéler difficiles à corriger car différentes applications peuvent être écrites par différents programmeurs, et chacun utilise un jeu différent de conventions et de fonctions de gestion des dates.

Voici une illustration simple qui montre que le serveur MySQL n'a aucun problème avec les dates jusqu'en 2030 :

```
 mysql> DROP TABLE IF EXISTS y2k;
Query OK, 0 rows affected (0.01 sec)

 mysql> CREATE TABLE y2k (date DATE,
    ->                    date_time DATETIME,
    ->                    time_stamp TIMESTAMP);
Query OK, 0 rows affected (0.00 sec)

 mysql> INSERT INTO y2k VALUES
    -> ("1998-12-31","1998-12-31 23:59:59",19981231235959),
    -> ("1999-01-01","1999-01-01 00:00:00",19990101000000),
    -> ("1999-09-09","1999-09-09 23:59:59",19990909235959),
    -> ("2000-01-01","2000-01-01 00:00:00",20000101000000),
    -> ("2000-02-28","2000-02-28 00:00:00",20000228000000),
    -> ("2000-02-29","2000-02-29 00:00:00",20000229000000),
    -> ("2000-03-01","2000-03-01 00:00:00",20000301000000),
    -> ("2000-12-31","2000-12-31 23:59:59",20001231235959),
    -> ("2001-01-01","2001-01-01 00:00:00",20010101000000),
    -> ("2004-12-31","2004-12-31 23:59:59",20041231235959),
    -> ("2005-01-01","2005-01-01 00:00:00",20050101000000),
    -> ("2030-01-01","2030-01-01 00:00:00",20300101000000),
    -> ("2050-01-01","2050-01-01 00:00:00",20500101000000);
Query OK, 13 rows affected (0.01 sec)
Records: 13  Duplicates: 0  Warnings: 0

 mysql> SELECT * FROM y2k;
+------------+---------------------+----------------+
| date       | date_time           | time_stamp     |
+------------+---------------------+----------------+
| 1998-12-31 | 1998-12-31 23:59:59 | 19981231235959 |
| 1999-01-01 | 1999-01-01 00:00:00 | 19990101000000 |
| 1999-09-09 | 1999-09-09 23:59:59 | 19990909235959 |
| 2000-01-01 | 2000-01-01 00:00:00 | 20000101000000 |
| 2000-02-28 | 2000-02-28 00:00:00 | 20000228000000 |
| 2000-02-29 | 2000-02-29 00:00:00 | 20000229000000 |
| 2000-03-01 | 2000-03-01 00:00:00 | 20000301000000 |
| 2000-12-31 | 2000-12-31 23:59:59 | 20001231235959 |
| 2001-01-01 | 2001-01-01 00:00:00 | 20010101000000 |
| 2004-12-31 | 2004-12-31 23:59:59 | 20041231235959 |
| 2005-01-01 | 2005-01-01 00:00:00 | 20050101000000 |
| 2030-01-01 | 2030-01-01 00:00:00 | 20300101000000 |
| 2050-01-01 | 2050-01-01 00:00:00 | 00000000000000 |
+------------+---------------------+----------------+
13 rows in set (0.00 sec)
```

Cet exemple montre que les types DATE et DATETIME ne poseront aucun problème avec les dates futures (ils gèrent les dates jusqu'en 9999).

Le type TIMESTAMP , qui est utilisé pour stocker la date courante, est valide jusqu'en 2030−01−01 . TIMESTAMP va de 1970 en 2030 sur les machines 32 bits (valeur signée). Sur les machines 64 bits, il gère les dates jusqu'en 2106 (valeur non signée).

Même si le serveur MySQL est compatible an 2000, il est de votre responsabilité de fournir des données non ambiguës. Voyez Y2K issues and date types pour les règles du serveur MySQL pour traiter les dates ambiguës (les données contenant des années exprimées sur deux chiffres).

# 1.3 Qui est MySQL AB ?

MySQL AB est l'entreprise des fondateurs de MySQL et les principaux développeurs. A l'origine, MySQL AB a été établie en Suède, par David Axmark, Allan Larsson et Michael Monty Widenius.

Tous les développeurs du serveur MySQL sont employés par l'entreprise. Nous sommes une organisation virtuelle, avec des employés répartis dans une douzaine de pays à travers le monde. Nous communiquons intensivement entre nous sur l'Internet tous les jours, et avec nos utilisateurs, fans et partenaires.

Nous nous consacrons au développement du logiciel MySQL et à la diffusion de notre base de données auprès des nouveaux utilisateurs. MySQL AB est propriétaire des droits du code source de MySQL , du logo MySQL et de la marque de commerce, ainsi que du manuel. Qu'est ce que MySQL ? .

Les valeurs fondamentales de MySQL témoignent de notre implication auprès de MySQL et des Logiciels libres .

Nous souhaitons que la base de données MySQL soit :

- Le meilleur serveur de bases de données dans le monde et le plus répandu.
- Le rendre accessible à tous.
- Facile à utiliser.
- En amélioration constante, tout en restant rapide et sécuritaire.
- Plaisant à utiliser et améliorer.
- Sans aucun bogue.

MySQL AB et les collaborateurs de MySQL AB :

- Font la promotion des logiciels libres et supportent la communauté des Logiciels libres .
- Sont de bons citoyens.
- Préfèrent les partenaires qui partagent nos valeurs et notre état d'esprit.
- Répondent aux emails et fournissent de l'aide.
- Sont une entreprise virtuelle, qui travaille en réseau avec les autres.
- Militent contre les brevets logiciels.

Le site web de MySQL ( http://www.mysql.com/ ) fournit les dernières informations à propos de MySQL et MySQL AB .

## 1.3.1 Les services et le modèle d'affaire de MySQL AB

Une des questions les plus fréquentes que nous rencontrons est : `` **Comment arrivez−vous à vivre en développant un produit gratuit ?** '' Voici comment.

MySQL AB fait des affaires en vendant du support, des services, des licences commerciales et en percevant des royalties. Nous utilisons ces fonds pour le développement des produits et des affaires de MySQL .

La compagnie est profitable depuis sa conception. En octobre 2001, nous avons accepté un financement de la part d'un groupe d'investisseurs scandivanes important et de quelques business angels. Cet investissement est utilisé pour consolider notre modèle d'affaires et assurer les bases de notre croissance à long terme.

#### 1.3.1.1 Support

MySQL AB est dirigé par ses propriétaires, qui sont les fondateurs et les principaux développeurs de la base de données MySQL . Les développeurs se consacrent au support des utilisateurs et des autres utilisateurs, afin de rester au courant de leurs besoins et leurs problèmes. Tout notre support est donné par des développeurs qualifiés. Les questions vraiment épineuses sont étudiées par Michael Monty Widenius, auteur principal du code du serveur MySQL . Support offert par MySQL AB .Pour plus d'informations et pour commander un support de différents niveaux, voyez http://www.mysql.com/support/ ou contactez notre équipe de vente à sales@mysql.com .

#### 1.3.1.2 Formation et certification

MySQL AB organise des formations MySQL à travers le monde entier. Nous offrons des cours inter et intra entreprise, adaptés aux besoins spécifiques de chaque société. La formation MySQL est aussi disponible auprès de nos partenaires, les centres de formation certifiés MySQL .Nos documents de formation utilisent les mêmes exemples de bases de données que notre documentation et nos applications d'exemple, et ils sont toujours mis à jour pour prendre en compte les dernières versions de MySQL . Nos formateurs sont épaulés par notre équipe de développement pour garantir la qualité de la formation et le développement continu des documents de cours. Cela vous assure aussi qu'il n'y aura pas de questions laissées ouvertes durant les cours.Suivre nos formations vous permettra d'atteindre tous vos buts avec votre application MySQL . Vous allez aussi :

- Gagner du temps.
- Améliorer les performances de vos applications.
- Réduire ou éliminer les besoins en matériels supplémentaires, ce qui réduira vos coûts.
- Améliorer la sécurité.
- Améliorer la satisfaction de vos clients et collaborateurs.
- Vous préparer à la certification MySQL .

Si vous êtes intéressé par nos formations, en tant que participant potentiel, ou comme partenaire de formation, visitez la section de formation à http://www.mysql.com/training/ ou contactez−nous à : training@mysql.com .Le programme de certification MySQL est publié dans le second semestre 2002. Pour plus de détails, voyez http://www.mysql.com/certification/ .

#### 1.3.1.3 Conseil

MySQL AB et ses partenaires accrédités offrent des services de conseil aux utilisateurs du serveur MySQL et à ceux qui intègrent le serveur MySQL dans leurs logiciels, à travers le monde.Nos consultants peuvent vous aider à concevoir et paramétrer vos bases, construire des requêtes efficaces, optimiser votre plate−forme, résoudre les problèmes de migration, installer la réplication, bâtir des applications transactionnelles robustes et bien plus encore. Nous aidons aussi les clients à intégrer le serveur MySQL dans leurs produits et applications, pour un déploiement d'envergure.

Nos consultants travaillent en collaboration étroite avec notre équipe de développement pour assurer la qualité technique de nos services professionnels. Les missions de conseil peuvent aller de sessions de démarrage de deux jours à des projets de plusieurs semaines ou mois. Notre expertise couvre non seulement le serveur MySQL , mais s'étend aussi aux langages de programmation tels que PHP, Perl et d'autres encore.

Si vous êtes interessé par nos services de conseil ou si vous souhaitez devenir un partenaire conseil, visitez la section conseil de notre site web à http://www.mysql.com/consulting/ ou contactez notre équipe de conseil à consulting@mysql.com .

### 1.3.1.4 Licences commerciales

La base de données MySQL est publiée sous la licence GNU General Public License ( GPL ). Cela signifie que le logiciel MySQL peut être utilisé gratuitement, en acceptant les termes de la licence GPL . Si vous ne voulez pas être lié par les termes de la licence GPL (comme le fait que votre application aussi doit être GPL ), vous pouvez acheter une licence du même produit auprès de MySQL AB . Voyez http://www.mysql.com/support/arrangements/price.html . Comme MySQL AB est propriétaire du copyright du code source de MySQL , nous pouvons utiliser une double licence , qui fait que le même produit est disponible sous la licence GPL et sous une licence commerciale. Cela ne change en rien l'implication de MySQL AB dans le mouvement des logiciels libres . Pour plus de détails sur quand une licence commerciale est nécessaire, voyez Licences MySQL .

Nous vendons aussi des licences commerciales aux logiciels Open Source GPL qui ajoutent à la valeur du serveur serveur MySQL . Un bon exemple est le gestionnaire de table transactionnel InnoDB qui offre le support ACID , le verrouillage de ligne, la restauration après crash, le multi versionnage, le support des clés étrangères, etc. Tables InnoDB .

### 1.3.1.5 Partenariats

MySQL AB a un programme de partenariat mondial qui couvre la formation, le conseil et le support, les publications, la revente et la distribution des produits MySQL . Les partenaires MySQL AB gagnent en visibilité grâce au site http://www.mysql.com/ et le droit d'utiliser certaines versions spéciales des marques de commerce MySQL pour identifier leurs produits et promouvoir leur entreprise.

Si vous êtes intéressé à devenir un partenaire MySQL AB , envoyez un email à partner@mysql.com .

Le mot MySQL et le logo MySQL avec le dauphin sont des marques commerciales de MySQL AB . Logos MySQL AB et marques déposées . Ces marques représentent un investissement capital que les fondateurs de MySQL ont placé depuis plusieurs années.

### 1.3.1.6 Publicité

Le site web MySQL ( http://www.mysql.com/ ) est très populaire auprès des développeurs et des utilisateurs. En Octobre 2001, nous avons eu 10 millions de pages vues. Nos visiteurs représentent un groupe effectuant des décisions d'achat et des recommandations pour le matériel et les logiciels. Douze pour cents de nos visiteurs émettent des décisions d'achat, et seulement 9 % n'en prennent pas du tout. Plus de 65 % d'entre eux ont effectué un ou plusieurs achats sur internet ces six derniers mois, et 70 % prévoient d'en faire un dans le mois à venir.

## 1.3.2 Contacts

Le site web de MySQL ( http://www.mysql.com/ ) fournit les dernières informations à propos de MySQL et MySQL AB .

Pour les contacts presse et les questions qui ne sont pas couvertes par les annonces officielles ( http://www.mysql.com/news/ ), envoyez un email à press@mysql.com .

Si vous avez un contrat de support valide avec MySQL AB , vous obtiendrez des réponses rapides et précises de notre équipe technique sur le logiciel MySQL . Pour plus d'informations, voyez Support offert par MySQL AB . Sur notre site web, voyez http://www.mysql.com/support/ , ou envoyez un email à sales@mysql.com .

Pour des informations sur les formations MySQL , visitez la section formation à http://www.mysql.com/training/ . Si vous avez un accès restreint à Internet, contactez l'équipe de formation de MySQL AB à l'adresse training@mysql.com . Training and Certification .

Pour des informations sur le programme de certification MySQL , voyez la section http://www.mysql.com/certification/ . Training and Certification .

Si vous êtes interessé par du conseil, visitez la section conseil à http://www.mysql.com/consulting/ . Si vous avez un accès restreint à Internet, contactez l'équipe de conseil de MySQL AB à consulting@mysql.com . Consulting .

Les licences commerciales peuvent être commandées en ligne à http://order.mysql.com/ . Vous trouverez aussi des informations sur les commandes par fax à MySQL AB . Plus d'informations sur les licences sont disponibles à http://www.mysql.com/products/pricing.html . Si vous avez des questions qui concernent les licences ou que vous souhaitez un devis pour un achat de nombreuses licences, remplissez le formulaire de contact disponible sur le site web ( http://www.mysql.com/ ) ou envoyez un email à licensing@mysql.com (pour les questions de licence) ou à sales@mysql.com (pour les devis). Licences MySQL .

Si vous représentez une entreprise qui est interessée par un partenariat avec MySQL AB , envoyez un email à partner@mysql.com . Partnering .

Pour plus d'informations sur la politique de marque de commerce de MySQL , reportez−vous à http://www.mysql.com/company/trademark.html ou envoyez un email à trademark@mysql.com . Logos MySQL AB et marques déposées .

Si vous êtes interessé par un des emplois à MySQL AB présentés dans notre section job ( http://www.mysql.com/company/jobs/ ), envoyez un email à jobs@mysql.com . N'envoyez pas votre CV en pièce jointe, mais collez−le dans le corps du texte à la fin de votre message.

Pour des discussions générales entre utilisateurs, dirigez vos questions sur la liste de diffusion appropriée. Liste de diffusion MySQL .

Les rapports d'erreurs (aussi appelés bogues), ainsi que les questions et suggestions doivent être envoyés à la liste de diffusion mysql@lists.mysql.com . Si vous avez trouvé un trou de sécurité critique dans le serveur MySQL , envoyez votre courriel à security@mysql.com . Comment rapporter des bogues ou problèmes .

Si vous avez un résultat de test que nous pouvons publier, contactez−nous à benchmarks@mysql.com .

Si vous avez des suggestions concernant les améliorations ou les corrections de ce manuel, envoyez−les à l'équipe du manuel docs@mysql.com . Pour les remarques spécifiques à la version française, vous pouvez aussi contacter directement Damien Seguy à damien.seguy@nexen.net .

Pour les questions ou commentaires à propos du fonctionnement ou du contenu du site web MySQL ( http://www.mysql.com/ ), envoyez un email à webmaster@mysql.com .

MySQL AB a une politique de protection des données privées qui est présentée à http://www.mysql.com/company/privacy.html . Pour toutes les questions concernant cette politique, envoyez un email à privacy@mysql.com .Pour toutes les autres questions, envoyez un email à info@mysql.com .

# 1.4 Support MySQL et licences

Cette section décrit le support MySQL et les accords de licence.

## 1.4.1 Support proposé par MySQL AB

Le support technique de MySQL AB est la réponse individualisée à vos problèmes particuliers, en direct, de l'équipe d'ingénieurs qui programme la base de données MySQL .

Nous tâchons d'avoir un support technique exhaustif et global. Presque tous les problèmes qui impliquent MySQL sont importants pour nous, s'ils sont importants pour vous. Typiquement, les clients qui recherchent de l'aide sur les différentes commandes, qui souhaitent résoudre des problèmes de performance, réparer des systèmes corrompus, comprendre les impacts des systèmes d'exploitation ou des réseaux sur les performances de MySQL , mettre en place des bonnes pratiques pour la sauvegarde et l'entretien, utiliser les API s, etc. Notre support couvre uniquement le serveur MySQL et nos propres utilitaires, pas les produits tiers d'entreprises qui accèdent au serveur MySQL , même si nous pouvons parfois aider.

Les informations détaillées sur les différents niveaux de support sont disponibles à http://www.mysql.com/support/ , et les contrats de support peuvent aussi être commandés en ligne. Si vous avez un accès restreint à Internet, vous pouvez contacter notre équipe commerciale à sales@mysql.com .

Le support technique est identique à l'assurance vie. Vous pouvez vivre très heureux sans pendant plusieurs années, mais lorsque vous rencontrerez une catastrophe, il sera trop tard pour l'acheter. Si vous utilisez MySQL pour des applications importantes et que vous rencontrez des problèmes, cela peut vous prendre très longtemps pour comprendre par vous− même. Vous pourriez alors avoir besoin de communiquer avec les techniciens les plus expérimentés de MySQL , ceux qui sont employés par MySQL AB .

## 1.4.2 Copyrights et licences utilisées par MySQL

MySQL AB est propriétaire du copyright du code source de MySQL , des logos MySQL , de la marque de commerce et de ce manuel. Qu'est ce que MySQL AB ? . Plusieurs licences distinctes sont disponibles pour la diffusion de MySQL :

- Toutes les sources spécifiques de MySQL pour le serveur, le client mysqlclient et la librairie, ainsi que la librairie GNU readline sont couverts par la licence GNU General Public License . GNU General

Public License . Le texte de cette licence est aussi disponible dans le fichier intitulé COPYING dans les distributions.

- La librairie GNU getopt est sous la licence GNU Lesser General Public License . GNU Lesser General Public License .
- Certaines parties du code source (la librairie regexp ) sont placées sous un copyright de type Berkeley.
- Les anciennes versions de MySQL (3.22 et plus récentes) sont sujettes à des licences plus strictes ( http://www.mysql.com/support/arrangements/mypl.html ). Voyez la documentation spécifique de chaque version pour plus de détails.
- La manuel n'est actuellement **PAS** distribué sous la licence GPL . L'utilisation du manuel est sujette à ces conditions :
  - Les conversions en d'autres formats sont autorisées, mais le contenu ne doit pas être modifié ou édité de quelque manière que ce soit.
  - Vous pouvez créer une version imprimée pour votre utilisation personnelle.
  - Pour toutes les autres utilisations, telles que la vente de versions imprimées, ou l'utilisation (en partie ou en totalité) du manuel dans d'autres publications, un accord doit être conclu au préalable avec MySQL AB .

  Communiquez par e−mail avec docs@mysql.com pour plus d'informations, ou si vous êtes interessé par la traduction du manuel.

Pour plus de détails sur comment les licences MySQL fonctionnent, voyez Licences MySQL . Voyez aussi Logos MySQL AB et marques déposées .

## 1.4.3 Licences MySQL

Le logiciel MySQL est publié sous la licence GNU General Public License ( GPL ), qui est probablement mieux connue sous le nom de Open Source . Les termes exacts de la licence GPL sont disponibles sur le site de http://www.gnu.org/licenses/ . Voyez aussi http://www.gnu.org/licenses/gpl−faq.html et http://www.gnu.org/philosophy/enforcing−gpl.html .

Comme le logiciel MySQL est publié sous la licence GPL , il est souvent utilisé gratuitement, mais pour certains usages vous souhaiterez peut−être acheter une licence commerciale auprès de MySQL AB à http://order.mysql.com/ . Voyez http://www.mysql.com/support/arrangements.html pour plus d'informations.

Les anciennes versions de MySQL (3.22 et plus anciennes) sont sujettes à une licence plus stricte ( http://www.mysql.com/support/arrangements/mypl.html ). Voyez la documentation spécifique de chaque version pour information.

Notez bien que l'utilisation du logiciel MySQL sous une licence commerciale. GPL , ou toute autre ancienne licence MySQL ne vous donne pas automatiquement le droit d'utiliser les marques commerciales de MySQL AB . Logos MySQL AB et marques déposées .

### 1.4.3.1 Utiliser MySQL avec la licence commerciale

La licence GPL est contagieuse, dans le sens où lorsqu'un programme est lié à la licence GPL , toutes les sources de toutes les parties du produit final doivent aussi être publiée sous la licence GPL . Sinon, vous violez la licence, et annulez vos droits d'utiliser le programme GPL .

Vous avez besoin d'une licence commerciale pour :

- Lorsque vous reliez un programme à un logiciel GPL issu de MySQL et que vous ne voulez pas

que le produit final soit publié sous la licence GPL , peut être parce que vous souhaitez publier un produit commercial ou conserver du code non− GPL pour d'autres raisons. Lorsque vous achetez une licence commerciale vous n'utilisez plus MySQL sous la licence GPL , même si c'est le même code.

- Lorsque vous distribuez une application non− GPL qui fonctionne **uniquement** avec MySQL , et que vous distribuez cette application avec MySQL . Ce type de solution est considéré comme un lien, même si c'est fait via le réseau.
- Lorsque vous distribuez des copies de MySQL sans fournir le code source original, comme requis par la licence GPL .
- Lorsque vous voulez supporter le développement de MySQL même si vous n'avez pas besoin formellement de la licence commerciale. Acheter du support auprès de MySQL AB est une autre bonne solution pour contribuer au développement de MySQL , avec des avantages directs pour vous. Support offert par MySQL AB .

Si vous avez besoin d'une licence, vous en aurez besoin d'une pour chaque installation de MySQL . Cela est valable quelque soit le nombre de processeurs de la machine, et il n'y a pas de limite artificielle de nombre de connexion simultanées.Pour les licences commerciales, voyez notre site web http://www.mysql.com/support/arrangements/price.html . Pour les contrats de support, voyez http://www.mysql.com/support/ . Si vous avez des besoin spéciaux, ou que vous avez un accès restreint à internet, contactez notre équipe de vente à sales@mysql.com .

### 1.4.3.2 Utiliser MySQL sous licence GPL libre

Vous pouvez utiliser le logiciel MySQL sous la licence gratuite GPL si vous acceptez les termes et conditions de la licence GPL . Pour une liste exhaustive des questions courantes à propos de la licence GPL , voyez la FAQ générale de la Free Software Foundation à http://www.gnu.org/licenses/gpl−faq.html . Quelques cas courants :

- Lorsque vous distribuez votre application et le code source MySQL , ensemble avec la licence GPL .
- Lorsque vous distribuez le code source MySQL avec d'autres programmes qui ne sont pas liés ou dépendant de MySQL pour leur fonctionnalités, même si vous vendez la distribution commercialement. Ceci s'appelle une agrégation simple en termes GPL .
- Si vous ne distribuez **aucune** partie de MySQL , vous pouvez l'utiliser gratuitement.
- Lorsque vous êtes un fournisseur de services Internet (ISP), qui offre un hébergement web avec le MySQL . Cependant, nous vous encourageons à faire affaires avec un ISP qui dispose du support MySQL, car cela vous donne l'assurance que si votre hébergeur a des problèmes avec l'installation de MySQL , votre hébergeur aura les ressources pour résoudre votre problème. Notez que si votre ISP n'a pas de licence commerciale pour le le MySQL , il devrait au moins fournir l'accès en lecture aux sources aux clients pour qu'ils vérifient que l'installation de MySQL est correctement patchée.
- Lorsque vous utilisez le serveur de bases de données MySQL en conjonction avec un serveur web, vous n'avez pas besoin de licence commerciale (aussi longtemps que ce n'est pas un produit que vous distribuez). Ceci est même vrai si vous avec un serveur web qui utilise le serveur MySQL , car vous ne distribuez pas de partie de MySQL . Cependant, dans ce cas, nous vous demandons d'acheter du support MySQL car MySQL aide votre entreprise

Si vous utilisez le serveur de bases de données MySQL sans avoir besoin de licence commerciale, nous vous encourageons à acheter le support auprès de MySQL AB malgré tout. De cette façon, vous contribuez au développement de MySQL et en obtenez des avantages immédiats vous−même. Support offert par MySQL AB .

Si vous utilisez MySQL dans un contexte commercial tel que vous en tirez profit, nous vous demandons

de participer au développement de MySQL en achetant du support. Nous pensons que comme MySQL aide votre entreprise, il est raisonnable de vous demander d'aider à votre tour MySQL AB . (Sinon, lorsque vous nous posez des questions de support, non seulement vous utilisez gratuitement un système dans lequel nous avons investi beaucoup de temps, mais en plus, vous nous demandez du support gratuit, en plus !)

## 1.4.4 Logos MySQL AB et marque déposée

Beaucoup d'utilisateurs de MySQL souhaitent afficher le logo du dauphin MySQL AB sur leur sites web, leur livres ou leurs produits. Nous encourageons ces actes, tant qu'on part du principe que le mot MySQL et le logo du dauphin MySQL sont des marques déposées par MySQL AB et ne doivent être utilisés que dans les conditions décrites à la page suivante : http://www.mysql.com/company/trademark.html .

### 1.4.4.1 Le logo original de MySQL

Le dauphin MySQL a été conçu par l'agence de publicité finlandaise Priority, en 2001. Le dauphin a été choisi en tant que symbole représentatif de MySQL , car c'est un animal intelligent, rapide et élancé, qui parcourt sans effort l'océan des données. Nous aimons aussi les dauphins.

Le logo original de MySQL ne peut être utilisé que par les représentants de MySQL AB et par ceux qui ont un accord signé leur permettant de le faire.

### 1.4.4.2 Logos MySQL qui peuvent être utilisés dans autorisation préalable

Nous avons conçu un jeu de logos qui peuvent être utilisés **sous conditions** , et qui peuvent être téléchargés depuis notre site web à http://www.mysql.com/press/logos.html et utilisés sur les sites web tiers sans autorisation écrite de MySQL AB . L'utilisation de ces logos n'est pas inconditionnelle, mais, comme leur nom l'implique, sujette à notre politique de marque de commerce qui est aussi disponible sur notre site web. Il est recommandé de lire ce document avant d'utiliser les logos sur votre site web. En bref, les pré−requis sont :

- Utilisez le logo que vous souhaitez tel que présenté sur le site web http://www.mysql.com/ . Vous pouvez l'étendre ou le réduire suivant vos besoins, mais ne modifier ni les couleurs, ni le design, ni le graphisme en aucune façon.
- Faîtes en sorte qu'il soit évident que vous, et non pas MySQL AB , êtes le créateur de ce site, qui arbore les couleurs de MySQL .
- N'utilisez pas le logo d'une façon qui porte préjudice à MySQL AB ou aux valeurs de MySQL AB . Nous nous réservons le droit de retirer notre autorisation d'utiliser le logo de MySQL AB .
- Si vous utilisez le logo sur un site web, rendez−le cliquable, avec un lien sur le site http://www.mysql.com/ .
- Si vous utilisez le serveur web MySQL sous licence GPL dans votre application, votre application doit porter la mention Open Source et être capable de se connecter à un serveur MySQL .

Contactez−nous à trademark@mysql.com pour voir avec nous tous les arrangements spéciaux qui vous conviendraient.

### 1.4.4.3 Quand avez vous besoin d'autorisation pour utiliser le logo MySQL?

Dans les cas suivants vous devez obtenir une permission écrite de MySQL AB avant d'utiliser les logos MySQL :

- Lorsque vous affichez un des logos MySQL AB n'importe où, en dehors de votre site web.
- Lorsque vous affichez l'un des logos MySQL AB hormis les logos à **utilisation conditionnée** mentionnés auparavant sur le site web, ou ailleurs

En dehors des raisons commerciales et légales, nous nous devons de suivre l'utilisation du logo MySQL sur les produits, livres, etc. Nous demanderons une compensation pour l'affichage des logos MySQL AB sur les produits, car nous pensons qu'il est raisonnable qu'une partie des revenus ainsi générés servent à poursuivre le développement de la base de données MySQL .

#### 1.4.4.4 Logos de partenariat MySQL AB

Le logo partenariat MySQL ne doit être utilisé que par les compagnies et les personnes ayant un partenariat écrit avec MySQL AB Le partenariat inclut une certification en tant que consultant ou professeur MySQL .

Merci de visiter Partnering .

#### 1.4.4.5 Utiliser le nom MySQL sur des documents imprimés ou des présentations

MySQL AB apprécie les références à la base MySQL . Il faut toutefois garder à l'esprit que MySQL est une marque commerciale, propriété de MySQL AB . A cause de cela, il faut ajouter au logo le symbole de marque de commerce ( TM ) lors de la première ou de la plus visible utilisation du mot MySQL dans le texte et, là où approprié, établir clairement que MySQL est une marque commerciale de MySQL AB . Reportez−vous à notre politique de marque commerciale à http://www.mysql.com/company/trademark.html pour plus de détails.

#### 1.4.4.6 Utilisation du nom MySQL dans un nom de société ou de produit

L'utilisation du mot MySQL dans le nom d'une compagnie, d'un produit ou d'un nom de domaine Internet est interdite sans une permission écrite de MySQL AB .

## 1.5 MySQL 4.0 In A Nutshell

Promise depuis longtemps par MySQL AB et attendue avec impatience par nos utilisateur, le serveur MySQL 4.0 est disponible en version de production.

MySQL 4.0 est disponible au téléchargement depuis http://www.mysql.com/ et nos mirroirs. MySQL 4.0 a été testée par un grand nombre d'utilisateurs et il est en production sur de très grands sites.

Les fonctionnalités principales de MySQL serveur 4.0 sont destinées à nos utilisateurs professionnels et communautaire : elles améliorent le capacités de MySQL pour gérer les missions critiques et les systèmes fortement chargés. D'autres fonctionnalités sont destinées aux utilisateurs de solutions intégrées.

MySQL version 4.0.12 a été déclarée stable pour utilisation en production, en mars 2003. Cela signifie que désormais, seules les corrections de bugs pour la série 4.0 seront faites. Les corrections de bugs critiques pour les séries 3.23 seront aussi proposées. Upgrading From Version 3.23 to 4.0 .

Les nouvelles fonctionnalités sont ajoutées en MySQL 4.1, qui est désormais disponibles sur notre serveur de versions BitKeeper. MySQL 4.1 In A Nutshell .

## 1.5.1 Fonctionnalités disponibles depuis MySQL 4.0

*Amélioration des performances*

MySQL 4.0 dispose d'un cache de requêtes qui peut vous accélérer gradement vos applications qui utilisent souvent les mêmes requêtes. Cache de requêtes MySQL .

La version 4.0 accélère la vitesse du serveur MySQL dans de nombreux domaines, notamment les `INSERT` de masse, la recherche sur les index compressés, la création d'index `FULLTEXT` ainsi que les comptes `COUNT(DISTINCT)` .

*MySQL Server embarqué*

La nouvelle librairie Embedded Server (au lieu de client/server) peut être facilement utilisée pour créer des applications indépendantes ou intégrées. Embedded MySQL Server .

*Le moteur `InnoDB` en standard*

Le moteur de tables `InnoDB` est désormais livré en standard avec le serveur MySQL, apportant le support complet des **transactions** ACID, les **clés étrangères** avec modifications et effacement en cascade, ainsi que le **verrouillage de ligne** . Tables `InnoDB` .

*Nouvelels fonctionnalités*

Les nouvelles possibilités de recherche en `FULLTEXT` de MySQL Server 4.0 permettent l'utilisation d'index `FULLTEXT` sur de grandes quantités de texte, avec des logiques binaires ou en langage naturel. Les utilisateurs peuvent paramétrer la taille minimum des mots, et définir leur propre liste de mots interdits, dans n'importe quel langue. Cela ouvre la possibilité de nombreuses applications avec MySQL Server. Recherche full−text avec MySQL .

*Standards compliance, portability and migration*

Simplification de la migration depuis d'autres bases de données vers MySQL Server, et notamment `TRUNCATE TABLE` (comme sous Oracle) et `IDENTITY` comme synonyme pour les clés automatiquement incrémentées (comme sous Sybase).

De nombreux utilisateurs seront heureux de savoir que le serveur MySQL supporte aussi les requêtes `UNION` , une fonctionnalité SQL attendue avec impatience.

MySQL peut s'exécuter nativement sur les plates−formes NetWare 6.0. Notes relatives à Novell NetWare .

*Internationalisation*

Nos utilisateurs allemands, autrichiens et suisses remarqueront que nous avons un nouveau jeu de caractères, `latin1_de` , qui corrige les problèmes de **tri des valeurs allemandes** , en plaçant les umlauts allemands dans le même ordre que dans l'annuaire d'Allemagne.

*Amélioration de l'ergonomie*

Durant la mise en place de fonctionnalités pour de nouveaux utilisateurs, nous n'avons pas oublié notre communauté de loyaux utilisateurs.

Une fonctionnalité pratique pour les administrateurs de base de données est que la plupart des paramètres de démarrage de `mysqld` peuvent être modifiées sans redémarrer le serveur. Syntaxe de `SET` .

Les commandes `DELETE` et `UPDATE` peuvent désormais fonctionner sur plusieurs tables.

En ajoutant le support des `liens symboliques` à `MyISAM` au niveau des tables (et non plus au niveau des bases, comme auparavant), et en autorisant les liens

symboliques sur Windows, nous espérons que nous avons pris au sérieux vos demandes d'amélioration.
Des fonctions comme `SQL_CALC_FOUND_ROWS` et `FOUND_ROWS()` rendent possible le comptages de lignes sans utiliser la clause `LIMIT` .

La section sur les nouveautés du manuel rassemble toutes les nouveautés. Changements dans la publication 4.0.x (Production) .

### 1.5.2 MySQL Server intégré (embedded)

`libmysqld` rend le serveur MySQL disponible pour toute une gamme d'applications très vaste. En utilisant la librairie du serveur MySQL intégré, vous pouvez utiliser MySQL dans différentes applications et appareillages, où l'utilisateur final n'aura même pas idée de sa présence. Le serveur MySQL intégré est idéal pour équiper les bornes internet, les kiosques publics, les packages matériel/ logiciels clé en main, les serveurs MySQL haute performances, et les bases de données autonomes sur CDrom.

De nombreux utilisateurs de `libmysqld` profiteront de la **double licence** . Pour ceux qui ne souhaitent pas être liés par la licence GPL, la librairie est aussi disponible avec une licence commerciale. La librairie MySQL intégrée utilise la même interface que la librairie cliente classique, ce qui la rend pratique à utiliser. libmysqld, the Embedded MySQL Server Library .

## 1.6 MySQL 4.1 In A Nutshell

MySQL Server 4.0 a posé les fondations pour de nouvelles fonctionnalités telles que les **sous−requêtes imbriquées** et l' **Unicode** qui sont dors et déjà implémentée en version 4.1, ainsi que les **procédures stockées** SQL−99, qui seront disponibles pour la version 5.0. Ils représentent les fonctionnalités les plus demandées par de nombreux clients.

Avec ces améliorations, les critiques du serveur de base de données MySQL devront être plus imaginatifs que jamais pour identifier des manques dans le serveur MySQL. Déjà connu depuis longtemps pour sa stabilité, sa rapidité et sa facilité d'emploi, le serveur MySQL va désormais satisfaire la liste de tous les voeux des clients les plus exigents.

### 1.6.1 Fonctionnalités disponibles depuis MySQL 4.1

Les fonctionnalités ci−dessous sont implémentées en MySQL 4.1. Quelques autres fonctionnalités sont prévues pour MySQL 4.1, mais très peu. Ce qui est prévu pour la version 4.1 .

Les plus récentes fonctionnalités en cours de réalisation, comme par exemple les procédures stockées, seront disponibles en MySQL 5.0. Ce qui est prévu pour la version 5.0 .

*Support for subqueries and derived tables*

Attendues depuis longtemps, les sous−requêtes sont disponibles :

```
SELECT * FROM t1 WHERE t1.a=(SELECT t2.b FROM t2);

SELECT * FROM t1 WHERE (1,2,3) IN (SELECT a,b,c FROM t2);
```

En plus des sous−requêtes, les tables dérivées (vues anonymes), sont aussi disponibles. Une table dérivée est une sous−requête placées dans la clause `FROM` d'une commande `SELECT` :

```
SELECT t1.a FROM t1, (SELECT * FROM t2) t3 WHERE t1.a=t3.a;
```

*Speed enhancements*

Protocole binaire plus rapide, avec préparation des commandes et paramétrisation. C API Prepared Statements .
Indexation `BTREE` pour les tables `HEAP` , ce qui améliore significativement le temp de réponse pour les recherches non exactes.

*New functionality*

`CREATE TABLE table LIKE table` vous permet de créer une nouvelle table avec la même structure que la table d'origine, dans la même commande.
Support pour OpenGIS (Données géographiques).

*Standards compliance, portability and migration*

Le nouveau protocole client−serveur apporte la possibilité de faire passer plusieurs alertes au client, plutôt qu'une seule. Cela améliore grandement la gestion des erreurs lors des manipulations de masse. `SHOW WARNINGS` affiche les erreurs de la dernière commande. `SHOW WARNINGS | ERRORS` .

*Internationalisation*

Pour supporter notre base d'utilisateurs en pleine croisssance, et leur configurations locales, MySQL exploite désormais l'Unicode (UTF8).
Les jeux de caractères peuvent désormais être définis par colonnes, tables et bases. Cela permet d'améliorer la souplesse dans la conception des applications, en particuliers pour les sites multi−langues.

*Usability enhancements*

En réponse à la demande populaire, nous avons ajouté une commande `HELP command` coté serveur, qui peut être utilisée en ligne de commande du client `mysql` et d'autres clients, pour obtenir de l'aide sur les commandes SQL. Avec ces informations sur le serveur, elles seront parfaitement adaptées à la version et configuration du serveur.
Avec le nouveau protocole client/serveur, les requêtes multiples sont désormais activées. Cela vous premet d'émettre plusieurs requêtes en une seule ocmmande, puis de lire tous les résultats en une seule fois. C API Handling multiple query executions .
Nous avons implémenté une syntaxe pratique `INSERT ... ON DUPLICATE KEY UPDATE ...` .
Elle vous permet de modifier une ligne avec `UPDATE` , si l'insertion `INSERT` avait généré un double dans la colonne `PRIMARY` ou `UNIQUE` . Syntaxe des `INSERT` .
Le nouveau protocole client/serveur supporte différents jeux de résultats.

La section sur les nouveautés du manuel rassemble toutes les nouveautés. Changements dans la publication 4.1.x (Alpha) .

## 1.6.2 Phases de publication

De nouvelles fonctionnalités sont ajoutées à MySQL 4.1, qui est déjà disponible dans notre système BitKeeper. Installer à partir des sources de l'arbre de développement .

Le jeu de fonctionnalités destiné à la version 4.1 est globalement définitif. Les nouveautés iront essentiellement à la nouvelle version de développement 5.0. MySQL 4.1 va suivre son développement normal avec une phase **Alpha** (durant laquelle de nouvelles fonctionnalités peuvent être ajoutées ou modifiées), puis **Beta** (où les fonctionnalités sont fixées, et seules des corrections sont faîtes), puis **Gamma** (qui indique qu'une version de production verra le jour dans quelques semaines), avant que la version 4.1 ne deviennent la nouvelle version officielle.

### 1.6.3 Utilisation immédiate en production

Il est recommandé aux utilisateurs de ne pas passer leurs systèmes en production sous le serveur MySQL 4.x, jusqu'à ce qu'elle soit publiée en phase béta (telle que la 4.0.3 beta). Toutefois, même la version initiale a passé avec succès notre batterie de tests, sans aucune erreur sur aucune plate−forme que nous avons utilisé. Etant donné le grand nombre de fonctionalités supplémentaires, nous recommandons le serveur MySQL, même en version alpha, pour les phases de développement. L'agenda de publication du serveur MySQL 4.x est tel qu'il atteindra un état stable avant les applications qui sont aujourd'hui en phase de développement.

### 1.6.4 MySQL 5.0, The Next Development Release

Les nouveaux développements de MySQL sont désormais concentrés sur la version 5.0. Les procédures stockées et d'autres fonctionnalités seront en vedette. Ce qui est prévu pour la version 5.0 .

## 1.7 Sources d'informations MySQL

### 1.7.1 Listes de diffusion MySQL

Cette section vous présente les listes de diffusions MySQL, et donne des conseils quand à leur utilisation. En vous inscrivant à une des listes de diffusion, vous recevrez les messages que les autres auront envoyé, et vous pourrez envoyer vos propres questions et réponses.

#### 1.7.1.1 Les listes de diffusions de MySQL

Pour vous inscrire à la liste de diffusion principale de MySQL, envoyez un courrier électronique à mysql−subscribe@lists.mysql.com .

Pour vous désinscrire à la liste de diffusion principale de MySQL, envoyez un courrier électronique à mysql−unsubscribe@lists.mysql.com .

Seule l'adresse d'où vous envoyez votre message est importante. La ligne de sujet et le corps sont ignorés.

Si votre adresse de réponse n'est pas valide, vous pouvez spécifier votre adresse explicitement, en ajoutant un tiret après le mot subscribe ou unsubscribe, suivi de votre adresse dans laquelle vous aurez remplacé le caractère `'@'` par `'='` . Par exemple, pour inscrire l'adresse `your_name@host.domain` , envoyez un message à `mysql−subscribe−your_name=host.domain@lists.mysql.com` .

Les mails envoyés à mysql−subscribe@lists.mysql.com et mysql−unsubscribe@lists.mysql.com sont géré automatiquement par le robot ezmlm. Des informations sur ezmlm sont disponibles à http://www.ezmlm.org/ .

Pour poster un message sur la liste elle−même, envoyez votre message à `mysql@lists.mysql.com` . Toutefois, **n'envoyez pas** de mail d'inscription ou de désinscription à mysql@lists.mysql.com car tous les mails envoyés à cette adresse sont distribués automatiquement à des milliers d'utilisateurs.

Votre site local peut avoir beaucoup d'inscrits à la liste mysql@lists.mysql.com . Si c'est le cas, vous pouvez avoir une liste de diffusion locale, de façon à ce que les messages envoyés par `lists.mysql.com` à votre site local soit propagés par votre serveur local. Dans ce cas, contactez votre administrateur local pour être ajouté ou retiré de la liste.

Si vous voulez que le trafic de cette liste soit envoyé à une autre boîte aux lettres de votre client mail, installez un filtre basé sur les entêtes du message. Vous pouvez utiliser notamment les entêtes `List-ID:` et `Delivered-To:` pour identifier les messages de la liste. Les listes de diffusion MySQL suivantes existent :

*announce−subscribe@lists.mysql.com announce*
: Ceci est la liste de diffusion d'annonces des versions de MySQL et des programmes compagnons. C'est une liste à faible volume, et tout utilisateur doit y être inscrit.

*mysql−subscribe@lists.mysql.com mysql*
: La liste de diffusion principale pour les discussions générales sur MySQL. Notez que certains sujets sont à diriger sur les listes spécialisées. Si vous postez sur la mauvaise liste, vous pourriez ne pas avoir avoir de réponse.

*mysql−digest−subscribe@lists.mysql.com mysql−digest*
: La liste `mysql` en format journalier. Cela signifie que vous recevrez tous les messages de la journée en un seul gros email.

*bugs−subscribe@lists.mysql.com bugs*
: Sur cette liste, vous ne devriez envoyez que des bogues complets, reproductibles ainsi que le rapport qui va avec, en utilisant le script `mysqlbug` (si vous utilisez Windows, il faut aussi inclure la description du système d'exploitation et la version de MySQL MySQL). De préférence, vous devriez tester le problème avec la dernière version stable ou de développement du serveur MySQL avant de l'envoyer. Tout le monde doit être capable de reproduire le bogue simplement avec la ligne de commande `mysql test < script` avec le cas de test inclus. Tous les bogues doivent être posté sur la liste, et seront corrigés ou documentés dans la prochaine version de MySQL!. Si les modifications sont trop petites, nous publierons aussi un patch qui résout le problème.

*bugs−digest−subscribe@lists.mysql.com bugs−digest*
: La liste `bugs` en format journalier.

*internals−subscribe@lists.mysql.com internals*
: Une liste pour ceux qui travaillent sur le code MySQL. Sur cette liste, vous pouvez discuter du développement de MySQL et envoyer des patchs.

*internals−digest−subscribe@lists.mysql.com internals−digest*
: La liste `internals` en format journalier.

*java−subscribe@lists.mysql.com java*
: Une liste pour ceux qui utilisent MySQL et java. Elle concerne majoritairement les pilotes JDBC.

*java−digest−subscribe@lists.mysql.com java−digest*
: La liste `java` en format journalier.

*win32−subscribe@lists.mysql.com win32*
: Une liste pour ceux qui utilisent MySQL sur les systèmes d'exploitation de Microsoft, tels que Windows 9x/Me/NT/2000/XP.

*win32−digest−subscribe@lists.mysql.com win32−digest*
: La liste `win32` en format journalier.

*myodbc−subscribe@lists.mysql.com myodbc*
: Une liste pour tout ce qui concerne la connexion à MySQL avec le pilote ODBC.

*myodbc−digest−subscribe@lists.mysql.com myodbc−digest*
: La liste `myodbc` en format journalier.

*mysqlcc−subscribe@lists.mysql.com mysqlcc*
: Une liste pour tout ce qui concerne le client graphique `MySQL Control Center` .

*mysqlcc−digest−subscribe@lists.mysql.com mysqlcc−digest*
La liste `mysqlcc` en format journalier.
*plusplus−subscribe@lists.mysql.com plusplus*
Une liste pour tout ce qui concerne la programmation avec les API C++ de MySQL.
*plusplus−digest−subscribe@lists.mysql.com plusplus−digest*
La liste `plusplus` en format journalier.
*msql−mysql−modules−subscribe@lists.mysql.com msql−mysql−modules*
Une liste pour tout ce qui concerne Perl et le support du module msql−mysql−modules.
*msql−mysql−modules−digest−subscribe@lists.mysql.com msql−mysql−modules−digest*
La liste `msql−mysql−modules` en format journalier.

Vous pouvez vous inscrire ou vous désinscrire de toutes les listes en même temps de la même façon que nous l'avons décrit au début. Dans votre message d'inscription, utilisez simplement le nom de liste approprié. Par exemple, pour vous inscrire à la liste `myodbc` , envoyez un message à myodbc−subscribe@lists.mysql.com ou myodbc−unsubscribe@lists.mysql.com .

Si vous ne pouvez pas obtenir d'informations sur la liste de diffusion, une de vos options est de prendre un contrat de support auprès de MySQL AB, qui vous donnera un contact direct avec les développeurs MySQL. Support offert par MySQL AB .

Le tableau suivant présente diverses autres listes de diffusions consacrée à MySQL, dans d'autres langues que l'anglais. Notez que ces ressources ne sont pas gérées par MySQL AB, ce qui fait que nous ne pouvons pas garantir leur qualité.

*mysql−france−subscribe@yahoogroups.com Une liste de diffusion française*
*list@tinc.net Une liste de diffusion coréenne*
Envoyez un message à `subscribe mysql your@e−mail.address` .
*mysql−de−request@lists.4t2.com Une liste de diffusion allemande*
Envoyez un message à `subscribe mysql−de your@e−mail.address` . Vous aurez plus d'informations sur cette liste à http://www.4t2.com/mysql/ .
*mysql−br−request@listas.linkway.com.br Une liste de diffusion portugaise*
Envoyez un message à `subscribe mysql−br your@e−mail.address` .
*mysql−alta@elistas.net Une liste de diffusion espagnole*
Envoyez un message à `subscribe mysql your@e−mail.address` .

#### 1.7.1.2 Poser des questions ou rapporter un bogue

Avant de soumettre un rapport de bogue ou une question, commencez par ces étapes simples :

- Etudiez le manuel MySQL et faites y une recherche à : http://www.mysql.com/doc/

  Nous nous efforçons de mettre à jour le manuel fréquemment, en y ajoutant les solutions aux nouveaux problèmes. L'historique de modification ( http://www.mysql.com/doc/en/News.html ) est particulièrement pratique car il est possible qu'une nouvelle version de MySQL propose déjà la solution à votre problème.
- Cherchez dans la base de données des bogues sur http://bugs.mysql.com/ pour voir si le bogue a déjà été reporté/résolu.
- Recherchez dans les archives des listes de diffusion de MySQL : http://lists.mysql.com/
- Vous pouvez aussi utiliser l'URL http://www.mysql.com/search/ pour rechercher dans toutes les pages web (y compris le manuel) qui sont situés à http://www.mysql.com/ .

Si vous n'arrivez pas à trouver une réponse à votre question dans le manuel ou dans les archives,

vérifiez auprès de votre expert MySQL local. Si vous ne trouvez toujours pas la réponse, vous pouvez lire la section suivante et envoyer un mail bien préparé à mysql@lists.mysql.com .

### 1.7.1.3 Comment rapporter un bogue ou un problème

Ecrire un bon rapport de bogue requiert de la patience, et le faire dès le début épargnera votre temps et le notre. Un bon rapport de bogue qui contient un cas de test complet améliorera vos chances de voir le bogue corrigé à la prochaine version. Cette section vous aidera à écrire correctement un rapport de bogue, de manière à ce que vous ne gaspillez pas votre temps à faire des textes qui ne nous aideront que peu ou pas.

Nous vous recommandons d'utiliser le script `mysqlbug` pour générer un rapport de bogue (ou rapporter un problème), dans la mesure du possible. `mysqlbug` est situé dans le dossier `scripts` de la distribution, ou, pour les distributions binaire, dans le dossier `bin` du dossier d'installation de MySQL. Si vous êtes dans l'incapacité d'utiliser `mysqlbug` , vous devez tout de même inclure toutes les informations nécessaires listées dans cette section.

Le script `mysqlbug` vous aide à générer un rapport en déterminant automatiquement les informations suivantes, mais si quelque chose d'important lui échappe, ajoutez le dans votre message! Lisez cette section avec attention, et assurez vous que toutes les informations décrites ici sont présentes dans votre message.

Pour rapporter un bogue ou un problème, il faut le soumettre à la liste mysql@lists.mysql.com . Si vous pouvez rédiger un cas de test qui démontre clairement le bogue, il faut le poster sur la liste bugs@lists.mysql.com . Notez que sur cette liste, vous ne devez poster qu'un rapport complet de bogue regénérable, avec le script `mysqlbug` . Si vous travaillez sous Windows, vous devez ajouter une description de votre système d'exploitation et de votre version de MySQL. De préférence, il vaut mieux utiliser la dernière version de MySQL, version stable ou de développement, avant d'envoyer un message. Tout le monde doit être capable de reproduire le bogue en utilisant simplement la ligne de commande `` `mysql test < script` '', avec le cas de bogue fourni, ou bien d'exécuter un script shell ou Perl qui est inclut dans le rapport de bogue. Tous les bogues postés sur la liste `bogues` seront corrigés ou documentés dans la prochaine version de MySQL. Si seul de petits changements sont nécessaires, nous publierons aussi un patch.

Si vous avez découvert un problème de sécurité sensible dans MySQL, il faut envoyer un email à security@mysql.com .

Sachez qu'il est toujours possible de répondre à un message qui contient trop d'informations, alors qu'il est impossible de répondre à um message qui contient trop peu d'informations. Souvent, il est facile d'omettre des faits parce que vous pensez connaître la cause du problème et supposez que ces détails ne sont pas importants. Un bon principe à suivre est : si vous avez un doute à propos de quelque chose, faites nous en part. Il est bien plus rapide et bien moins frustrant d'écrire quelques lignes de plus dans un rapport plutôt que d'être obligé de demander une nouvelle fois et d'attendre une réponse parce que vous avez oublié une partie des informations la première fois.

L'erreur la plus commune est de ne pas indiquer le numéro de la version de MySQL qui est utilisé, ou de ne pas indiquer le système d'exploitation que vous utilisez (y compris le numéro de version de ce système d'exploitation). Ce sont des informations de première importance, et dans 99% d4es cas, le rapport de bogue est inutilisable sans ces informations. Souvent, nous recevons des questions telles que ``Pourquoi est ce que cela ne fonctionne pas pour moi?''. Puis nous nous aperçevons que la fonctionnalités en question n'est même pas programmée dans la version de MySQL utilisée, ou que le bogue décrit est déjà corrigé dans une nouvelle version de MySQL. Parfois aussi, les erreurs sont dépendantes des plate−formes. Dans ce cas, il est presque

impossible de les corriger sans savoir quel système d'exploitation et quelle version exacte est utilisée.

Pensez aussi à fournir des informations concernant votre compilateur, si c'est pertinent. Souvent, les développeurs trouvent des bogues dans les compilateurs, et pensent que c'est liés à MySQL. La plupart des compilateurs sont en constant développement, et s'améliorent de version en version. Pour déterminer si votre problèe dépend de votre compilateur, nous avons besoin de savoir quel compilateur est utilisé. Notez que les problèmes de compilations sont des bogues, et doivent être traités avec un rapport de bogues.

Il est particulièrement utile de fournir une bonne description du bogue dans le rapport de bogue. Cela peut être un exemple de ce que vous avez fait qui a conduit au problème, ou une description précise. Les meilleurs rapports sont ceux qui incluent un exemple complet permettant de reproduire le bogue. Faire une batterie de tests lorsque vous faites face à un problème de table corrompue .

Si un programme produit un message d'erreur, il est très important d'inclure ce message dans votre rapport. Il est préférable que le message soit le message exact, car il est alors possible de le retrouver en utilisant les archives : même la casse doit être respectée. N'essayez jamais de vous rappeler d'un message d'erreur, mais faites plutôt un copier/coller du message complet dans votre rapport.

Si vous avez un problème avec MyODBC, essayez de générer un fichier de trace MyODBC. Reporter les problèmes rencontrés avec MyODBC .

Pensez aussi que de nombreux personnes qui liront votre rapport utilisent un formatage de 80 colonnes. Lorsque vous générez votre rapport et vos exemples avec l'outil de ligne de commande, utilisez une largeur de 80 colonnes. Utilisez l'option `--vertical` (ou la fin de commande `\G` ) pour les affichages qui excèdent une telle largeur (par exemple, avec la commande `EXPLAIN SELECT` ; voyez l'exemple un peu plus tard dans cette section.

Voici un pense−bête des informations à fournir dans votre rapport :

- Le numéro de version de la distribution de MySQL que vous utilisez (par exemple MySQL Version 3.22.22). Vous pouvez connaitre cette version en excutant la commande `mysqladmin version` . `mysqladmin` est situé dans le dossier `bin` de votre distribution MySQL.
- Le fabricant et le modèle de votre serveur.
- Le système d'exploitation et la version que vous utilisez. Pour la plupart des systèmes d'exploitation, vous pouvez obtenir cette information en utilisant la commande Unix command `uname -a` .
- Parfois, la quantité de mémoire (physique et virtuelle) est important. Si vous hésitez, ajoutez la.
- Si vous utilisez une version de MySQL sous forme de source, le nom et le numéro de version du compilateur sont nécessaires. Si vous utilisez une version exécutable, le nom de la distribution est important.
- Si le problème intervient lors de la compilation, incluez le message d'erreur exact et les quelques lignes de contexte autour du code en question dans le fichier où il est situé.
- Si `mysqld` s'est arrété, il est recommandé d'inclure la requête qui a mené à cet arrêt de `mysqld` . Vous pouvez généralement la trouver en exécutant `mysqld` en ayant activé les logs. Utilisation des fichiers de log pour trouver d'où viennent les erreurs de mysqld .
- Si une table ou une base sont liés au problème ajoutez le résultat de la commande `mysqldump --no-data db_name tbl_name1 tbl_name2 ...` . C'est très simple à faire, et c'est un moyen efficace d'obtenir un descriptif de table, qui nous permettra de recrééer une situation comparable à la votre.
- Pour les problèmes liés à la vitesse, ou des problèmes liés à la commande `SELECT` , pensez à inclure le résultat de la commande `EXPLAIN SELECT ...` , et au moins le nombre de ligne que la

commande `SELECT` doit produire. Vous devriez aussi inclure le résultat de la commande `SHOW CREATE TABLE table_name` pour chaque table impliquée. Plus vous nous fournirez d'informations, plus nous aurons de chance de vous aider efficacement. Par exemple, voici un excellent rapport de bogue (posté avec le script `mysqlbug` , et effectivement rédigé en ANGLAIS) :Exemple réalisé avec `mysql` en ligne de commande (notez l'utilisation de la fin de commande `\G` , pour les résultats qui pourraient dépasser les 80 colonnes de large) :

```
 mysql> SHOW VARIABLES;
mysql> SHOW COLUMNS FROM ...\G
       <output from SHOW COLUMNS>
mysql> EXPLAIN SELECT ...\G
       <output from EXPLAIN>
mysql> FLUSH STATUS;
mysql> SELECT ...;
       <A short version of the output from SELECT,
       including the time taken to run the query>
mysql> SHOW STATUS;
       <output from SHOW STATUS>
```

- Si un bogue ou un problème survient lors de l'exécution de `mysqld` , essayez de fournir un script qui reproduit l'anomalie. Ce script doit inclure tous les fichiers sources nécessaires. Plus votre script reproduira fidèlement votre situation, le mieux ce sera. Si vous pouvez réaliser un cas de test postez le sur la liste bugs@lists.mysql.com pour un traitement en priorité!Si vous ne pouvez pas fournir de script, fournissez tout au moins le résultat de la commande `mysqladmin variables extended-status processlist` dans votre mail pour fournir des informations sur les performances de votre système.
- Si vous ne pouvez pas reproduire votre situation en quelques lignes, ou si une table de test est trop grosse à être envoyée par mail (plus de 10 lignes), exportez vos tables sous forme de fichier avec la commande `mysqldump` et créez un fichier `README` qui décrit votre problème.Créez une archive compressée de votre fichier en utilisant `tar` et `gzip` ou `zip` , et placez le via `ftp` sur le site de ftp://support.mysql.com/pub/mysql/secret/ . Puis, envoyez une courte description de votre problème à bugs@lists.mysql.com .
- Si vous pensez que le serveur MySQL fournit des résultats étranges pour une requête, incluez non seulement le résultat, mais aussi votre propre explication sur ce que le résultat devrait être, et un diagnostic de la situation.
- Lorsque vous donnez un exemple du problème, il est mieux d'utiliser des noms de variables et de tables qui existent dans votre situation, plutôt que d'inventer de nouveaux noms. Le problème peut être lié au noms des variables ou tables que vous utilisez! Ces cas sont rares, mais il vaut mieux éviter les ambiguités. Après tout, il est plus facile pour vous de fournir un exemple qui utilise votre situation réelle, et c'est bien mieux pour nous aussi. Si vous avez des données que vous ne souhaitez pas divulguer, vous pouvez utiliser le site `ftp` pour les transférer dans le dossier secret ftp://support.mysql.com/pub/mysql/secret/ . Si les données sont vraiment ultra secrètes et que vous ne souhaitez même pas nous les montrer, alors utilisez d'autres noms et données pour votre rapport, mais considérez cela comme un dernier recours.
- Incluez toutes les options utilisées, si possible. Par exemple, indiquez les options que vous utilisez lors du démarrage de `mysqld` , et celle que vous utilisez avec les programmes comme `mysqld` et `mysql` , et le script `configure` , qui sont souvent primordiaux et pertinents. Ce n'est jamais une mauvaise idée que de les inclure. Si vous utilisez des modules, comme Perl ou PHP, incluez aussi les versions de ces logiciels.
- Si votre question porte sur le système de droits, incluez le résultat de l'utilitaire `mysqlaccess` , celui de `mysqladmin reload` et tous les messages d'erreurs que vous obtenez lors de la connexion. Lorsque vous testez votre système de droits, il faut commencer par utiliser la commande `mysqladmin reload version` et de vous connecter avec le proramme qui vous pose problème. `mysqlaccess` est situé dans le dossier `bin` de votre installation de MySQL.
- Si vous avez un patch pour un bogue, c'est une excellente chose. Mais ne supposez pas que nous n'avons besoin que du patch, ou même que nous allons l'utiliser, si vous ne fournissez pas les informations nécessaires pour le tester. Nous pourrions trouver des problèmes générés par votre patch, ou bien nous pourrions ne pas le comprendre du tout. Si tel est le cas, nous ne l'utiliserons pas.Si nous ne pouvons pas vérifier exactement ce pourquoi est fait le patch, nous ne l'utiliserons pas. Les cas de tests seront utiles ici. Montrez

nous que votre patch va générer toutes les situations qui pourraient arriver. Si nous trouvons un cas limite dans lequel votre patche ne fonctionne pas, même si il est rare, il risque d'être inutile.

- Les diagnostics sur la nature du bogue, la raison de son déclenchement ou les effets de bords sont généralement faux. Même l'équipe MySQL ne peut diagnostiquer sans commencer par utiliser un débogueur pour déterminer la cause véritable.
- Indiquez dans votre mail que vous avez vérifié le manuel de référence et les archives de courrier, de façon à avoir épuiser les solutions que d'autres avant vous auraient pu trouver.
- Si vous obtenez un message `parse error` , vérifiez votre syntaxe avec attention. Si vous ne pouvez rien y trouvez à redire, il est très probable que votre version de MySQL ne supporte pas encore cette fonctionnalité que vous essayez d'utiliser. Si vous utilisez la version courante de mySQL et que le manuel http://www.mysql.com/doc/ ne couvre pas la syntaxe que vous utilisez, c'est que MySQL ne supporte pas votre syntaxe. Dans ce cas, vos seules options sont d'implémenter vous même la syntaxe ou d'envoyez un message à licensing@mysql.com pour proposer de l'implémenter.Si le manuel présente la syntaxe que vous utilisez, mais que vous avez une ancienne version du serveur MySQL, il est recommandé de vérifier l'historique d'évolution de MySQL pour savoir quand la syntaxe a été supportée. Dans ce cas, vous avez l'option de mettre à jour votre MySQL avec une version plus récente. Historique des versions de MySQL .
- Si vous avez un problème tel que vos données semblent corrompues, ou que vous recevez constamment des errors lors d'accès à une table, vous devriez commener par essayer de réparer votre table avec l'utilitaire de ligne de commande `myisamchk` ou les syntaxes SQL `CHECK TABLE` et `REPAIR TABLE` . Administration de la base .
- Si vous avez des tables qui se corrompent facilement, il vous faut essayer de trouver quand et pourquoi cela arrive. Dans ce cas, le fichier `mysql−data−directory/'hostname'.err` peut contenir des informations pertinentes qu'il est bon d'inclure dans votre rapport de bogues. Normalement, `mysqld` **ne doit jamais** corrompre une table si il a été interrompu au milieur d'une mise à jour. Si vous pouvez trouvez la cause de l'arrêt de `mysqld` , il est bien plus facile pour nous de fournir un correctif. Comment déterminer ce qui cause problème .
- Si possible, téléchargez et installez la version la plus récente du serveur MySQL, et vérifiez si cela résoud votre problème. Toutes les versions de MySQL sont testées à fond, et doivent fonctionner sans problème. Nous croyons à la compatibilité ascendante, et vous devriez pouvoir passer d'une version à l'autre facilement. Which MySQL Version to Use .

Si vous disposez de l'accès au support client, contactez aussi le support client à mysql−support@mysql.com , en plus de la liste de rapport de bogues, pour un traitement prioritaire. Pour des informations sur les rapports de bogues avec `MyODBC` , voyez Comment reporter les problèmes rencontrés avec MySQL ODBC .

Pour des solutions aux problèmes les plus courants, voyez Problèmes .

Lorsque des solutions vous sont envoyées individuellement et non pas à la liste, il est considéré comme bien vu de rassembler ces réponses et d'en envoyer un résumé sur le liste, de manière à ce que les autres en profitent aussi.

#### 1.7.1.4 Conseils pour répondre sur la liste de diffusion

Si vous pensez que votre réponse peut avoir un intérêt général, vous pouvez envisager de l'envoyer sur la liste de diffusion, plutôt que de faire une réponse personnelle aux demandeurs. Essayez de rendre votre réponse aussi générale que possible, pour que suffisamment d'autres personnes puissent en profiter. Lorsque vous envoyez une réponse sur la liste, assurez vous qu'elle ne représente pas un doublon d'une réponse précédente.

Essayez de résumer l'essentiel de la question dans votre réponse. Ne vous croyez pas obligé de citer tout le message original.

Attention : n'envoyez pas de message avec le mode HTML activé ! De nombreux utilisateurs ne lisent pas leurs emails avec un navigateur.

### 1.7.2 Support de la communauté MySQL sur IRC (Internet Relay Chat)

En plus des différentes listes de diffusion MySQL, vous pouvez rencontrer des utilisateurs expérimentés sur `IRC` ( `Internet Relay Chat` ). Voici les meilleurs canaux, à notre connaissance :

- **freenode** (see http://www.freenode.net/ pour les serveurs)
  - `#mysql` Principalement, des questions sur MySQL, mais les autres bases de données et le langage SQL sont aussi acceptés.
  - `#mysqlphp` Des questions à propos de MySQL et PHP, une combinaison populaire.
  - `#mysqlperl` Des question sur MySQL et Perl, une autre combinaison populaire.
- **EFnet** (voir http://www.efnet.org/ pour les serveurs)
  - `#mysql` Questions sur MySQL.

Si vous recherchez un client IRC pour vous connecter à un réseau IRC, voyez donc `X−Chat` ( http://www.xchat.org/ ). X−Chat est disponible sous Unix et sous Windows.

## 1.8 Quels standards respecte MySQL?

Cette section présente comment MySQL interprète les standards SQL ANSI. Le serveur MySQL dispose de nombreuses extensions au standard ANSI et vous trouverez ici comment les exploiter. Vous trouvrez aussi des informations sur les fonctionnalités manquantes de MySQL et comment y trouver des palliatifs.

Notre but n'est pas, sans une bonne raison, de restreindre les capacités de MySQL à un usage unique. Même si nous n'avons pas les ressources de développement à consacrer à toutes les opportunités, nous sommes toujours interessés et prêts à aider ceux qui utilisent MySQL dans de nouveaux domaines.

Un de nos objectifs avec ce produit est de tendre à la compatibilité ANSI 99, mais sans sacrifier la vitesse ou la robustesse. Nous ne reculons pas devant l'ajout de nouvelle fonctionnalités au langage SQL, ou le support de fonctionnalités hors SQL, qui améliorent le confort d'utilisation de MySQL. La nouvelle interface de gestionnaires `HANDLER` de MySQL 4.0 est un exemple de cette stratégie. Syntaxe des `HANDLER` .)

Nous continuons de supporter les bases transactionnelles et non transactionnelles pour combler les besoins des sites web ou des applications à fort besoin d'archivage, ainsi que les applications critiques à très haute disponibilité.

Le serveur MySQL a été conçu pour travailler avec des bases de taille moyenne (de 10 à 100 millions de lignes, ou des tables de 100 Mo) sur des systèmes de petite taille. Nous continuons d'améliorer MySQL pour qu'il fonctionne avec des bases gigantesques (tera octets), tout en conservant la possibilité de compiler une version réduite de MySQL pour qu'il fonctionne sur des appareils embarqués ou nomades. L'architecture compacte de MySQL rend possible le support de ces applications si différentes, sans aucun conflit dans les sources.

Nous n'étudions pas le support du temps réel ou des bases de données en grappe (même si vous pouvez dores et déjà réaliser de nombreuses applications avec les services de réplication).

Nous ne croyons pas au support natif du XML en base, mais nous allons faire en sorte d'ajouter le support XML que réclame nos clients du coté client. Nous pensons qu'il est préférable de conserver le serveur central aussi ``simple et efficace'' que possible, et développer les librairies qui gèrent la complexité du coté client. Cela fait partie de la stratégie que nous avons mentionné plus tôt, pour ne sacrifier ni la vitesse, ni la robustesse du serveur.

## 1.8.1 Quels standards suit MySQL ?

SQL92, niveau de base. ODBC niveau 0−3.51.

Nous nous dirigeons vers le support complet du standard ANSI SQL99, mais sans aucune concession sur la vitesse ou la qualité du code.

## 1.8.2 Exécuter MySQL en mode ANSI

Si vous démarrez `mysqld` avec l'option `--ansi` , les comportements suivants du serveurs MySQL changent :

- `||` devient l'opérateur de concaténation de chaîne, et non pas l'opérateur binaire `OR` .
- Vous pouvez ajouter des espaces entre le nom d'une fonction et la parenthèse ouvrante sur les arguments : `'('` . Cela impose le traitement des noms de fonctions comme des mots réservés.
- `'"'` devient le caractère de protection des identifiants (comme le caractère `'`'` de MySQL) et n'est plus un caractère de limite de chaîne.
- `REAL` est synonyme de `FLOAT` au lieu d'être synonyme de `DOUBLE` .
- Le niveau d'isolation par défaut des transactions est `SERIALIZABLE` . Syntaxe de `SET TRANSACTION` .

Ceci revient à utiliser les options suivantes : `--sql-mode=REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,SERIALIZE,ONLY_FULL_GROUP_BY` .

## 1.8.3 Extensions de MySQL à la norme ANSI SQL92

Le serveur MySQL inclut des extensions que vous ne trouverez probablement pas dans les autres bases de données. Soyez prévenus que si vous les utilisez, votre code ne sera probablement pas portable sur d'autres serveurs SQL. Dans certains cas, vous pouvez écrire du code qui inclut des spécificités de MySQL, mais qui restent portables, en les incluant dans des commentaires de la forme `/*! ... */` . Dans ce cas, le serveur MySQL va analyser la chaîne et exécuter le code à l'intérieur de ces commentaires comme une commande normale, mais d'autres serveurs ignoreront ces commentaires. Par exemple :

```
SELECT /*! STRAIGHT_JOIN */ col_name FROM table1,table2 WHERE ...
```

Si vous ajoutez le numéro de version après le point d'exclamation `'!'` , la syntaxe sera exécutée uniquement si la version du serveur MySQL est égale ou plus récente que le numéro de version utilisé.

```
CREATE /*!32302 TEMPORARY */ TABLE t (a int);
```

Cela signifie que si vous avez la version 3.23.02 ou plus récente, le serveur MySQL va utiliser le mot réservé `TEMPORARY` .Voici une liste des apports spécifiques de MySQL :

- Les types de colonnes `MEDIUMINT` , `SET` , `ENUM` et les types `BLOB` et `TEXT` .
- Les attributs de champs `AUTO_INCREMENT` , `BINARY` , `NULL` , `UNSIGNED` et `ZEROFILL` .
- Toutes les comparaisons de chaînes sont insensibles à la casse par défaut, et l'ordre de tri est déterminé par le jeu de caractères courant (ISO−8859−1 Latin1 par défaut). Si vous en souhaitez un autre, il faut déclarer les colonnes avec l'attribut `BINARY` ou utiliser l'opérateur `BINARY` pour forcer les comparaisons à prendre en compte la casse, en fonction du jeu de caractères utilisé sur l'hôte du serveur MySQL.
- le serveur MySQL fait correspondre à chaque base un dossier dans le système de fichiers, et à chaque table, il fait correspondre un fichier, placé dans le dossier de base.

  Ceci a quelques conséquences :

  - Les noms des bases de données et des tables sont sensibles à la casse sur les systèmes d'exploitation qui ont des systèmes de fichiers sensibles à la casse (comme la plupart des systèmes Unix). Sensibilité à la casse pour les noms .
  - Les bases de données, les tables, les index, les colonnes et les alias peuvent commencer par un chiffre (mais ne peuvent pas être constitués uniquement de chiffres).
  - Vous pouvez utiliser les commandes systèmes standard pour sauver, renommer, déplacer, effacer et copier des tables. Par exemple, pour renommer une table, il suffit de renommer les fichiers `.MYD` , `.MYI` et `.frm` et de leur donner un nouveau nom.
- Dans une requête SQL, vous pouvez accéder à des tables situées dans différentes bases de données, avec la syntaxe `db_name.tbl_name` . Certains serveurs SQL fournissent la même fonctionnalité, mais l'appellent un `User space` . Le serveur MySQL ne supporte par les espaces de nom de tables, comme dans : `create table ralph.my_table...IN my_tablespace` .
- `LIKE` est possible avec des colonnes numériques.
- Utilisez `INTO OUTFILE` et `STRAIGHT_JOIN` dans les requêtes `SELECT` . Syntaxe des `SELECT` .
- L'option `SQL_SMALL_RESULT` de la commande `SELECT` .
- La commande `EXPLAIN SELECT` pour avoir le détail des jointures de tables.
- L'utilisation de noms d'index, de préfixes d'index, et l'utilisation des mots−clés `INDEX` or `KEY` dans une commande de création de table `CREATE TABLE` . Syntaxe de `CREATE TABLE` .
- L'utilisation des clauses `TEMPORARY` et `IF NOT EXISTS` avec `CREATE TABLE` .
- L'utilisation de `COUNT(DISTINCT list)` où `list` contient plus d'un élément.
- L'utilisation de `CHANGE col_name` , `DROP col_name` ou `DROP INDEX` , `IGNORE` ou `RENAME` dans une commande `ALTER TABLE` . Syntaxe de `ALTER TABLE` .
- L'utilisation de `RENAME TABLE` . Syntaxe de `RENAME TABLE` .
- L'utilisation de multiple `ADD` , `ALTER` , `DROP` et `CHANGE` dans les clauses de la commande `ALTER TABLE` .
- L'utilisation de `DROP TABLE` avec les mots−clés `IF EXISTS` .
- Vous pouvez effacer plusieurs tables avec une seule commande `DROP TABLE` .
- La clause `LIMIT` de la commande `DELETE` .
- La clause `DELAYED` des commandes `INSERT` et `REPLACE` .
- La clause `LOW_PRIORITY` des commandes `INSERT` , `REPLACE` , `DELETE` et `UPDATE` .
- L'utilisation de la commande `LOAD DATA INFILE` . Dans de nombreuses situations, cette syntaxe est compatible avec la commande d'Oracle `LOAD DATA INFILE` . Syntaxe des `LOAD DATA INFILE` .
- Les commandes `ANALYZE TABLE` , `CHECK TABLE` , `OPTIMIZE TABLE` et `REPAIR TABLE` .
- La commande `SHOW` . Syntaxe de `SHOW` .
- Les chaînes de caractères peuvent être soit délimitées par `'"'` , soit par `'''` . Pas seulement par `'''` .
- L'utilisation du caractère de protection `'\'` .
- La commande `SET` . Syntaxe de `SET` .
- Vous n'êtes pas obligé de nommer toutes les colonnes que vous sélectionnez dans la

clause GROUP BY . Cela donne de meilleures performances pour certaines situations spécifiques, mais classiques. Fonctions à utiliser dans les définitions GROUP BY .

- Vous pouvez spécifier ASC ou DESC dans la clause GROUP BY .
- Pour aider les utilisateurs qui viennent d'autres environnements SQL, le serveur MySQL supporte des alias de nombreuses fonctions. Par exemple, toutes les fonctions de chaînes de caractères supportent simultanément les syntaxes ANSI SQL et ODBC.
- Le serveur MySQL comprend les opérateurs || et comme opérateurs logiques OR et AND, comme en langage C. Pour le serveur MySQL, les opérateurs || et OR sont synonymes, ainsi que et AND . En conséquence, MySQL ne supporte pas l'opérateur de concaténation de chaînes ANSI SQL || . Utilisez plutôt la fonction CONCAT() . Comme CONCAT() prend un nombre illimité d'arguments, il est facile de convertir des expressions utilisant || , pour qu'elles fonctionnent sur le serveur MySQL.
- CREATE DATABASE et DROP DATABASE . Syntaxe de CREATE DATABASE .
- L'opérateur % est synonyme de MOD() . C'est à dire que N % M est équivalent à MOD(N,M) . % est supporté pour les programmeurs C, et pour la compatibilité avec PostgreSQL.
- Les opérateurs = , <> , <= , < , >= , > , << , >> , <=> , AND , OR ou LIKE peuvent être utilisés pour les comparaisons de colonnes à gauche de la clause FROM dans les commandes SELECT . Par exemple :

```
mysql> SELECT col1=1 AND col2=2 FROM tbl_name;
```

- La fonction LAST_INSERT_ID() . mysql_insert_id() .
- Les opérateurs d'expressions régulières étendus REGEXP et NOT REGEXP .
- CONCAT() et CHAR() avec un argument ou plus de deux arguments. Avec le serveur MySQL, ces fonctions peuvent prendre n'importe quel nombre d'arguments.
- Les fonctions BIT_COUNT() , CASE , ELT() , FROM_DAYS() , FORMAT() , IF() , PASSWORD() , ENCRYPT() , MD5() , ENCODE() , DECODE() , PERIOD_ADD() , PERIOD_DIFF() , TO_DAYS() et WEEKDAY() .
- L'utilisation de la fonction TRIM() pour réduire les chaînes. L'ANSI SQL ne supporte que les suppressions de caractères uniques.
- Les fonctions de groupe de la clause GROUP BY STD() , BIT_OR() et BIT_AND() .
- L'utilisation de REPLACE à la place de DELETE + INSERT . Syntaxe des REPLACE .
- Les commandes FLUSH , RESET et DO .
- La possibilité de modifier les variables dans les commandes avec l'opérateur := :

```
 SELECT &#64;a:=SUM(total),&#64;b=COUNT(*),&#64;a/&#64;b AS avg FROM test_table;
SELECT &#64;t1:=(&#64;t2:=1)+&#64;t3:=4,&#64;t1,&#64;t2,&#64;t3;
```

## 1.8.4 Différences de MySQL avec ANSI SQL92

Nous tâchons de rendre le serveur MySQL compatible avec le standard ANSI SQL, et le standard ODBC SQL, mais dans certains cas, MySQL se comporte différemment.

- Pour les colonnes de type VARCHAR , les espaces terminaux sont supprimés lors du stockage de la valeur. Erreurs connues et manques de MySQL .
- Dans certains cas, les colonnes CHAR sont transformées automatiquement en colonnes VARCHAR . Silent column changes .
- Les droits d'un utilisateur sur une table ne sont pas supprimés si la table est détruite. Vous devez explicitement utiliser la commande REVOKE pour supprimer les droits d'un utilisateur sur une table. Syntaxe de GRANT et REVOKE .
- NULL AND FALSE donnera la valeur NULL et non pas FALSE . Nous pensons que dans cette situation, il n'est pas bon d'évaluer de nombreuses conditions supplémentaires.

Pour voir la liste des priorités de développement des nouvelles extensions de MySQL, consultez la liste de tâches sur http://www.mysql.com/doc/en/TODO.html . C'est la toute dernière liste de tâche de ce manuel. MySQL and The Future (The TODO) .

### 1.8.4.1 Sous sélections (Sub SELECT s)

Les sous−requêtes ont été implémentées en MySQL 4.1. Features Available From MySQL 4.1 .

Jusqu'à la version 4.0, le serveur MySQL ne prenait en charge que les requêtes imbriquées de la forme `INSERT ... SELECT ...` et `REPLACE ... SELECT ...` . Vous pouvez néanmoins utiliser la fonction `IN()` dans d'autres contextes. Les sous−requêtes sont implémentées dans l'arborescence de développement 4.1.

En attendant, vous pourrez dans la plupart des cas réécrire la requête sans sous−requêtes :

```
SELECT * FROM table1 WHERE id IN (SELECT id FROM table2);
```

Ceci peut être réécrit de la façon suivante :

```
SELECT table1.* FROM table1,table2 WHERE table1.id=table2.id;
```

Les requêtes :

```
 SELECT * FROM table1 WHERE id NOT IN (SELECT id FROM table2);
SELECT * FROM table1 WHERE NOT EXISTS (SELECT id FROM table2
                                       WHERE table1.id=table2.id);
```

peuvent être réécrites ainsi :

```
 SELECT table1.* FROM table1 LEFT JOIN table2 ON table1.id=table2.id
                                       WHERE table2.id IS NULL;
```

Pour des requêtes plus complexes, vous pouvez généralement créer des tables temporaires contenant les sous−requêtes. Cependant, dans certains cas cette option ne fonctionnera pas. Le cas le plus fréquent se produit avec les instructions `DELETE` , pour lesquelles le SQL standard ne prend pas en charge les jointures (sauf dans les sous Select). Dans une telle situation, en attendant que les sous−requêtes soient prises en charge par MySQL Server, vous disposez de deux options.

La première option consiste à utiliser un langage de programmation procédural (tel que Perl ou PHP) afin de soumettre une requête `SELECT` pour obtenir les clés primaires des enregistrements à supprimer. On utilise ensuite ces valeurs pour construire l'instruction `DELETE` ( `DELETE FROM ... WHERE ... IN (key1,key2, ...)` ).

La seconde option consiste à utiliser le SQL interactif pour construire automatiquement un ensemble d'instructions `DELETE` , grâce à l'extension MySQL `CONCAT()` (à la place de l'opérateur standard `||` ).

Par exemple :

```
SELECT CONCAT('DELETE FROM tab1 WHERE pkid = ', "'", tab1.pkid, "'", ';')
 FROM tab1, tab2
WHERE tab1.col1 = tab2.col2;
```

Vous pouvez placer cette requête dans un fichier script et rediriger son entrée vers l'interpréteur de ligne de commande `mysql` , tout en ouvrant un canal de communication (pipe) pour diriger sa sortie vers une seconde instance de l'interpréteur :

```
shell> mysql −−skip−column−names mydb < myscript.sql | mysql mydb
```

La version 4.0 de MySQL prend en charge les instructions `DELETE` sur plusieurs tables pouvant être utilisées afin de supprimer efficacement des lignes à partir d'informations issues d'une table, voire de plusieurs tables en même temps.

### 1.8.4.2 SELECT INTO TABLE

Le serveur MySQL ne supporte pas encore l'extension Oracle SQL : `SELECT ... INTO TABLE ...` . A la place, le serveur MySQL supporte la syntaxe ANSI SQL `INSERT INTO ... SELECT ...` , qui revient au même.

Syntaxe de `INSERT ... SELECT` .

```
INSERT INTO tblTemp2 (fldID) SELECT tblTemp1.fldOrder_ID
      FROM tblTemp1 WHERE tblTemp1.fldOrder_ID > 100;
```

Vous pouvez aussi utiliser `SELECT INTO OUTFILE...` ou `CREATE TABLE ... SELECT` .

### 1.8.4.3 Transactions et opérations atomiques

Le serveur MySQL support les transactions avec les gestionnaires de tables `InnoDB` et `BDB` . Types de tables MySQL . `InnoDB` dispose aussi de la compatibilité `ACID` .

Toutefois, les tables non transactionnelles de MySQL telles que `MyISAM` exploitent un autre concept pour assurer l'intégrité des données, appelé `` `opérations atomiques` ''. Les opérations atomiques disposent d'une bien meilleure protection des données pour des performances également accrues. Comme MySQL supporte les deux méthodes, l'utilisateur est capable de choisir celle qui correspond à ses besoins, suivant qu'il a besoin de vitesse ou de sécurité. Ce choix peut être fait table par table.

Comment exploiter les capacités de MySQL pour protéger l'intégrité des données, et comment ces fonctionnalités se comparent−elles avec les méthodes transactionnelles ?

- En mode transactionnel, si votre application a été écrite en dépendant de l'appel de `ROLLBACK` au lieu de `COMMIT` dans les situations critiques, les transactions sont plus plus pratiques. Les transactions s'assurent que les modifications non achevées ou les activités corrosives ne sont pas archivées dans la base. Le serveur a l'opportunité d'annuler automatiquement l'opération, et votre base de données est sauve.

  Le serveur MySQL, dans la plupart des cas, vous permet de résoudre les problèmes potentiels en incluant de simples vérifications avant les modifications, et en exécutant des scripts simples pour vérifier l'intégrité de vos bases de données, ainsi que les incohérences, et pour réparer automatiquement les problèmes, ou encore vous alerter si une erreur est identifiée. Notez qu'en utilisant simplement le log de MySQL, ou en utilisant un log supplémentaire, vous pouvez normalement réparer à la perfection toutes les tables, sans aucune perte de données.
- Souvent, les modifications de données transactionnelles fatales peuvent être réécrites de manière atomique. En général, tous les problèmes d'intégrité que les transactions résolvent peuvent être corrigés avec la commande `LOCK TABLES` ou des modifications atomiques, qui assurent que vous n'aurez jamais d'annulation automatique de la base, ce qui est un problème commun des bases transactionnelles.
- Même un système transactionnel peut perdre des données si le serveur s'arrête. La différence entre les systèmes repose alors dans ce petit laps de temps où ils peuvent perdre des données. Aucun système n'est sécurisé à 100 %, mais simplement ``suffisament sécurisé''. Même Oracle, réputé pour être la plus sûre des bases de données transactionnelles, est montré du doigt pour perdre des données dans ces situations.Pour être tranquille avec MySQL, que vous utilisiez les tables transactionnelles ou pas, vous n'avez besoin que de sauvegardes et de logs de modifications. Avec ces deux outils, vous pourrez vous protéger de toutes les situations que vous pourriez rencontrer avec d'autres bases de données transactionnelles. De toute manière, il est bon d'avoir des sauvegardes, indépendamment de la base que vous utilisez.

La méthode transactionnelle a ses avantages et ses inconvénients. De nombreux utilisateurs et développeurs d'applications dépendent de la facilité de pallier un problème lorsqu'une annulation semble nécessaire ou presque. Cependant, même si vous êtes néophyte des opérations

atomiques, ou plus familier avec les transactions, prenez en considération le gain de vitesse que les tables non transactionnelles offrent. Ces gains vont de 3 a 5 fois la vitesse des tables transactionnelles les plus rapides et les mieux optimisées.

Dans des situations où l'intégrité est de la plus grande importance, le serveur MySQL assure une intégrité du niveau des transactions, ou encore mieux avec les tables non transactionnelles. Si vous verrouillez les tables avec `LOCK TABLES` , toutes les modifications seront bloquées jusqu'à ce que la vérification d'intégrité soit faite (à comparer avec un verrou en écriture), les lectures et insertions sont toujours possibles. Les nouvelles lignes ne seront pas accessibles en lecture tant que le verrou n'aura pas été levé. Avec `INSERT DELAYED` , vous pouvez faire attendre les insertions dans une pile, jusqu'à ce que les verrous soit levés, sans que le client n'attende cette levée de verrou. Syntaxe des `INSERT DELAYED` .

``Atomique'', avec le sens que nous lui donnons, n'a rien de magique. Ce terme signifie simplement que vous pouvez être certain que lorsque vous modifiez des données dans une table, aucun autre utilisateur ne peut interférer avec votre opération, et qu'il n'y aura pas d'annulation automatique (ce qui pourrait arriver avec des tables transactionnelles si nous ne sommes pas trop soigneux). Le serveur MySQL garantit aussi qu'il n'y aura pas de lectures erronées.

Voici quelques techniques pour travailler avec des tables non transactionnelles :

- Les boucles qui requièrent les transactions peuvent normalement être implémentées avec la commande `LOCK TABLES` , et vous n'avez nul besoin de curseur lorsque vous modifiez des lignes à la volée.
- Pour éviter d'utiliser l'annulation `ROLLBACK` , vous pouvez adopter la stratégie suivante :
  - Utilisez la commande `LOCK TABLES ...` pour verrouiller toutes les tables que vous voulez utiliser.
  - Testez vos conditions.
  - Modifiez si tout est correct.
  - Utilisez `UNLOCK TABLES` pour libérer vos tables.

  Ceci est probablement une méthode bien plus rapide que ne le proposent les transactions, avec des annulations `ROLLBACK` possibles mais pas certaines. La seule situation que ce cas ne prend pas en compte est l'interruption du processus au milieu d'une mise à jour. Dans ce cas, tous les verrous seront levés, mais certaines modifications peuvent ne pas avoir été exécutées.
- Vous pouvez aussi utiliser des fonctions pour modifier des lignes en une seule opération. Vous pouvez créer une application très efficace en utilisant cette technique :
  - Modifiez les champs par rapport à leur valeur actuelle.
  - Modifiez uniquement les champs que vous avez réellement changé.

  Par exemple, lorsque nous modifions les données d'un client, nous ne modifions que les données du client qui ont changé et nous vérifions uniquement si les données modifiées ou les données qui en dépendent ont changé comparativement aux données originales. Les tests sur les données modifiées sont faits avec la clause `WHERE` dans la commande `UPDATE` . Si la ligne a été modifiée, nous indiquons au client : "Some of the data you have changed has been changed by another user". En français : "certaines données que vous voulez modifier ont été modifiées par un autre utilisateur". Puis nous affichons l'ancienne ligne et la nouvelle ligne, pour laisser l'utilisateur décider quelle version il veut utiliser.

  Cela nous conduit à un résultat proche du verrouillage de ligne, mais en fait, c'est bien mieux, car nous ne modifions que les colonnes qui en ont besoin, en utilisant des valeurs relatives. Cela signifie qu'une commande `UPDATE` typique ressemble à ceci :

```
UPDATE tablename SET pay_back=pay_back+'relative change';

UPDATE customer
 SET
```

```
  customer_date='current_date',
  address='new address',
  phone='new phone',
  dette=dette+'emprunt'
WHERE
  customer_id=id AND address='old address' AND phone='old phone';
```

Comme vous pouvez le voir, c'est très efficace, et fonctionne même si un autre client a modifié la valeur `pay_back` ou `dette` .

- Dans de nombreuses situations, les utilisateurs ont souhaité les commandes `ROLLBACK` et/ou `LOCK TABLES` afin de gérer des identifiant uniques pour certaines tables. Ils peuvent être gérés bien plus efficacement en utilisant une colonne de type `AUTO_INCREMENT` , en corrélation avec la fonction `LAST_INSERT_ID()` ou la fonction C `mysql_insert_id()` . `mysql_insert_id()` . Vous pouvez éviter le verrouillage de ligne. Certaines situations le requièrent vraiment, mais elles sont rares. Les tables `InnoDB` supportent le verrouillage de ligne. Avec les tables MyISAM, vous pouvez utiliser une colonne de type flag, et faire ceci :

```
UPDATE tbl_name SET row_flag=1 WHERE id=ID;
```

MySQL retournera 1 pour le nombre de lignes affectées si la ligne a été trouvée, car `row_flag` ne vaut pas déjà 1 dans la ligne originale.Vous pouvez comprendre la requête ci−dessus comme si le serveur MySQL avait utilisé la commande suivante :

```
UPDATE tbl_name SET row_flag=1 WHERE id=ID AND row_flag <> 1;
```

### 1.8.4.4 Procédures stockées et triggers

Une procédure stockée est une liste de commandes SQL qui peuvent être compilées et stockées sur le serveurs. Une fois que cela est fait, les clients n'ont pas besoin de soumettre à nouveau toute la commande, mais font simplement référence à la procédure stockée. Cela conduit à des performances bien meilleures, car les commandes n'ont pas à être analysées plusieurs fois, et que bien moins d'informations transitent sur le réseau. Vous pouvez aussi élevez le niveau conceptuel de votre application en mettant en place des librairies de fonctions sur le serveur.

Un trigger est une procédure stockée qui est activée lorsqu'un événement particulier survient. Par exemple, vous pouvez installer une procédure stockée qui est déclenchée dès qu'une ligne est effacée dans une table d'achat, pour que le client soit automatiquement effacé si tous ses achats sont effacés.

La future version du langage sera capable de supporter les procédures stockées. Notre objectif est d'impolémenter les procédures stockées pour la version 5.0 du serveur MySQL. Nous nous pencherons aussi sur les triggers.

### 1.8.4.5 Les clés étrangères

Notez que les clés en SQL ne sont pas utilisées pour joindre des tables, mais pour assurer l'intégrité référentielle (contraintes de clés étrangères). Si vous voulez obtenir des résultats issus de tables multiples avec une commande `SELECT` , vous allez joindre les tables comme ceci :

```
SELECT * FROM table1,table2 WHERE table1.id = table2.id;
```

Syntaxe de `JOIN` . Utiliser les clefs étrangères .

En MySQL version 3.23.44 et plus récentes, les tables `InnoDB` supportent les vérifications d'intégrité référentielles. Tables `InnoDB` . Pour les autres types de tables, le serveur mySQL accepte la syntaxe `FOREIGN KEY` dans la commande `CREATE TABLE` , mais ne la prend pas en compte.

La syntaxe `FOREIGN KEY` dans la clause `ON DELETE ...` est principalement utilisée pour des raisons de

documentation. Certaines applications ODBC peuvent utiliser cette clause pour produire des conditions `WHERE` automatiques, mais c'est généralement simple à éviter. `FOREIGN KEY` est parfois utilisé pour vérifier une contrainte, mais cette vérification n'est pas nécessaire en pratique, si les lignes sont insérées dans la table dans le bon ordre.

Avec le serveur MySQL, vous pouvez contourner le problème de l'absence de la clause `ON DELETE ...` en ajoutant la commande `DELETE` appropriée dans l'application, lorsque vous effacez une ligne dans une table qui a une clé étrangère. En pratique, cette technique est aussi rapide (et même parfois plus rapide), et bien plus portable que l'utilisation des clés étrangères.

Avec le serveur MySQL 4.0, vous pouvez utiliser les commandes d'effacement multi−tables pour effacer des lignes dans plusieurs tables en une seule commande. Syntaxe des `DELETE` .

Dans un futur proche, nous allons étendre l'implémentation de la clause `FOREIGN KEY` pour que cette information soit stockée dans le fichier de spécification des tables et puisse être lu par `mysqldump` et ODBC. Dans une étape ultérieure, nous allons implémenter les contraintes de clé étrangère pour les applications qui ne peuvent pas être codées sans.

Gardez bien en tête que les clés étrangères sont souvent méconnues, ce qui peut causer de graves problèmes. Même lorsqu'elles sont utilisées correctement, ce n'est pas une solution magique pour les problèmes d'intégrité référentielle, même si cela simplifie parfois les choses.

Voici des avantages aux contraintes de clés étrangères :

- En supposant que les relations soient proprement conçues, les clés étrangères rendent plus difficile pour un programmeur d'insérer des valeurs incohérentes dans la base.
- L'utilisation des modifications et effacement en cascade simplifie le code du client.
- Les règles de clés étrangères proprement conçues aident à la documentation des relations entre les tables.

Inconvénients :

- Les erreurs, qui sont faciles à faire durant la conception des relations, peuvent causer des problèmes graves : par exemple, des règles de contrainte circulaires, ou des combinaisons erronées d'effacement.
- Une application correctement écrite s'assure d'elle−même de l'intégrité référentielle des données avant d'exécuter une requête. De ce fait, les vérifications supplémentaires de la base sont inutiles et réduisent les performances de l'application.
- Il n'est pas rare pour un administrateur de bases de données de faire une topologie complexe des relations entre tables qui rendent très complexe, voire impossible de restaurer des tables.

#### 1.8.4.6 Les vues

Il est prévu d'implémenter les vues dans la version 5.0 du serveur MySQL.

Les vues sont la plupart du temps utiles pour donner accès aux utilisateurs à un ensemble de relations représentées par une table (en mode inaltérable). Beaucoup de bases de données SQL ne permettent pas de mettre à jour les lignes dans une vue, vous devez alors faire les mises à jour dans les tables séparées.

Vu que le serveur MySQL est la plupart du temps utilisé dans des applications où le développeur à un contrôle total sur l'utilisation de la base de données, la plupart de nos utilisateurs n'ont pas

considéré les vues comme très importantes (du moins, personne n'a été assez intéressé à ce sujet pour être prêt à financer l'implémentation des vues).

On n'a pas forcément besoin des vues pour restreindre l'accès aux colonnes, et ce, parce que MySQL a un système de droits très sophistiqué. Sécurité générale et accès à MySQL .

#### 1.8.4.7 '−−' comme début de commentaire

Certaines bases de données SQL utilisent `'--'` comme début de commentaire. Le serveur MySQL utilise `'#'` . Vous pouvez aussi utiliser la syntaxe du langage C `/* ceci est un commentaire */` . Syntaxe des commentaires .

MySQL version 3.23.3 et plus récent supporte les commentaires de type `'--'` , si ce commentaire est suivi d'un espace. Ceci est dû au fait que ce type de commentaire a causé beaucoup de problèmes avec les requêtes générées automatiquement, qui contiennent du code tel que celui−ci, où nous insérons automatiquement la valeur du paiement dans la table `!payment!` :

```
UPDATE tbl_name SET credit=credit-!payment!
```

Pensez à ce qui se passe lorsque la valeur de `payment` est négative. Comme `1--1` est valide en SQL, la conséquence est de commencer un commentaire `'--'` .

En utilisant notre implémentation des commentaires avec le serveur MySQL version 3.23.3 et plus récent, `1-- ceci est un commentaire` ne pose pas ce type de problème.

Une autre fonctionnalité supplémentaire est que le client en ligne de commande `mysql` supprime toutes les lignes qui commencent par `'--'` .

Les informations suivantes sont destinées aux utilisateurs de MySQL avec des versions antérieures à la version 3.23.3 :

Si vous avez un programme SQL dans un fichier texte qui contient des commentaires au format `'--'` , il est recommandé d'utiliser :

```
shell> replace " --" " #" < text-file-with-funny-comments.sql \
        | mysql database
```

au lieu du classique :

```
shell> mysql database < text-file-with-funny-comments.sql
```

Vous pouvez aussi éditer le fichier de commande ``lui−même'' pour remplacer les commentaires `'--'` par des commentaires `'#'` :

```
shell> replace " --" " #" -- text-file-with-funny-comments.sql
```

Puis, rétablissez−les avec :

```
shell> replace " #" " --" -- text-file-with-funny-comments.sql
```

### 1.8.5 Erreurs connues, et limitations de MySQL

#### 1.8.5.1 Erreurs connues en 3.23 et corrigées ultérieurement

Les erreurs suivantes sont connues mais restent non corrigées en MySQL 3.23, car pour les corriger, il nous faudrait modifier trop de code : cela risquerait d'introduire des bugs bien pire. Ces bugs sont considérés comme 'non nuisibles' ou 'supportables'.

- Il est possible de rencontrer un blocage en utilisant la commande `LOCK TABLE` sur de multiples tables, puis, avec la même connexion, faire un `DROP TABLE` sur l'une d'entre elle, alors qu'un

autre thread essai de verrouiller la table. Il est toutes fois possible d'utiliser la commande `KILL` sur l'un des threads en question, pour résoudre ce problème. Corrigé en 4.0.12.
- `SELECT MAX(key_column) FROM t1,t2,t3...` où l'une des tables est vide ne retourne aps `NULL` mais plutôt le valeur maximale de la colonne. Corrigé en 4.0.11.

### 1.8.5.2 Bugs connus / limitations de MySQL

Les problèmes suivants sont connus, et sont en tête de liste pour être corrigés :

- `ANALYZE TABLE` sur une table de type BDB, peut rendre la table inutilisable, dans certains cas, jusqu'au prochain redémarrage de `mysqld` . Lorsque cela survient, vous rencontrez les erreurs suivantes dans le fichier d'erreur MySQL :
```
001207 22:07:56  bdb:  log_flush: LSN past current end-of-log
```
- N'exécutez pas de commande `ALTER TABLE` sur une table `BDB` sur laquelle vous avez exécuté des transactions à plusieurs commandes, jusqu'à ce que ces transactions soient achevées : la transaction sera probablement ignorée.
- `ANALYZE TABLE` , `OPTIMIZE TABLE` et `REPAIR TABLE` peuvent causer des problèmes sur les tables avec lesquelles vous utilisez la commande `INSERT DELAYED` .
- Faire un `LOCK TABLE ...` et `FLUSH TABLES ...` ne vous garantit pas qu'il n'y a pas une transaction en court sur la table.
- Les tables BDB sont lentes à ouvrir. Si vous avez de nombreuses tables BDB dans une base, cela prendra du temps au client `mysql` pour accéder à la base si vous n'utilisez pas l'option `-A` , ou si vous utilisez la commande `rehash` . C'est particulièrement vrai si vous n'avez pas de cache de table important.

Les problèmes suivants sont connus et seront corrigés en leur temps :

- Lors de l'utilisation de la fonction `RPAD` , ou de toute autre fonction de chaîne qui peut ajouter des espaces à droite de la chaîne, dans une requête qui utilise une table temporaire pour la résolution, alors toutes les chaînes verront leurs espaces terminaux être supprimés. Voici un exemple d'une telle requête : `SELECT RPAD(t1.field1, 50, ' ') AS f2, RPAD(t2.field2, 50, ' ') AS f1 FROM table1 as t1 LEFT JOIN table2 AS t2 ON t1.record=t2.joinID ORDER BY t2.record;`

  Le résultat final de ceci est que vous ne pourres pas obtenir les espaces à gauche dans ces chaînes.

  Le comportement décrit ci-dessus existe dans toutes les versions de MySQL.

  La raison à cela est due au fait que les tables de type HEAP, qui sont utilisées en premier comme table temporaires, ne sont pas capables de gérer des colonnes de type VARCHAR.

  Ce comportement sera corrigé dans l'une des versions de la série des 4.1.
- A cause de la méthode de stockage des tables de définitions de fichiers, il n'est pas possible d'utiliser le caractère 255 ( `CHAR(255)` ) dans les noms des tables, colonnes ou énumérations. Il est prévu de corriger de problème dans les versions version 5.1, lorsque nous aurons établi un nouveau format de définition des fichiers.
- Lorsque vous utilisez la commande `SET CHARACTER SET` , il n'est pas possible d'utiliser les caractères traduits dans les noms de bases, de tables ou de colonnes.
- Il n'est pas possible d'utiliser `_` ou `%` avec la commande `ESCAPE` dans la clause `LIKE... ESCAPE` .
- Si vous avez une colonne de type `DECIMAL` avec un nombre stocké dans un autre format (+01.00, 1.00, 01.00), `GROUP BY` peut considérer ces valeurs comme différentes.
- Lorsque `DELETE FROM merge_table` est utilisé sans la clause `WHERE` , elle va simplement effacer le fichier de la table, et ne pas effacer les tables associées.

- Vous ne pouvez pas compiler le serveur dans un autre dossier lorsque vous utilisez les MIT−pthreads. Comme cela requiert une modification des MIT−pthreads, nous ne corrigerons pas ce problème. Remarques sur MIT−pthreads .
- Les valeurs de type `BLOB` ne peuvent pas être utilisées ``correctement'' dans les clauses `GROUP BY` ou `ORDER BY` ou `DISTINCT` . Seuls, les `max_sort_length` premiers octets (par défaut, 1024) seront utilisés pour les comparaisons de `BLOB` . Ceci peut être modifié avec l'option `−O max_sort_length` de `mysqld` . Un palliatif à ce problème est d'utiliser une sous partie de chaîne : `SELECT DISTINCT LEFT(blob,2048) FROM tbl_name` .
- Les calculs sont faits avec des `BIGINT` ou `DOUBLE` (les deux sont normalement de 64 bits). La précision dépend alors de la fonction utilisée. La règle générale est que les fonctions de bits utilisent la précision des `BIGINT` , `IF` et `ELT()` utilisent la précision des `BIGINT` ou `DOUBLE` , et les autres utilisent la précision des `DOUBLE` . Il faut donc éviter d'utiliser les entiers non signés de grande taille, surtout s'ils dépassent la taille de 63 bits (9223372036854775807) pour toute autre fonction que les champs de bits ! La version 4.0 gère bien mieux les `BIGINT` que la 3.23.
- Toutes les colonnes de type chaînes, hormis les `BLOB` et `TEXT` , voient automatiquement leurs caractères blancs finaux supprimés. Pour le type `CHAR` c'est correct, et c'est considéré comme une fonctionnalité par la norme ANSI SQL92. Le hic est que pour le serveur MySQL les colonnes `VARCHAR` sont traitées de la même façon.
- Vous ne pouvez avoir que des colonnes de taille 255 pour les `ENUM` et `SET` .
- Avec les fonctions d'agrégation `MIN()` , `MAX()` et compagnie, MySQL compare actuellement les colonnes de type `ENUM` et `SET` par leur valeur de chaîne, plutôt que par leur position relative dans l'ensemble.
- `safe_mysqld` redirige tous les messages de `mysqld` vers le log `mysqld` . Le problème est que si vous exécutez `mysqladmin refresh` pour fermer et ouvrir à nouveau l'historique, `stdout` et `stderr` sont toujours redirigés vers l'ancien log. Si vous utilisez `−−log` extensivement, vous devriez éditer `safe_mysqld` pour loger vers `'hostname'.err` au lieu de `'hostname'.log` , de façon à pouvoir facilement récupérer la place de l'ancien log, en effaçant les vieux, et en exécutant `mysqladmin refresh` .
- Dans la commande `UPDATE` , les colonnes sont modifiées de gauche à droite. Si vous faite référence à une colonne modifiée, vous obtiendrez sa valeur modifiée, plutôt que sa valeur originale. Par exemple :

  ```
  mysql> UPDATE tbl_name SET KEY=KEY+1,KEY=KEY+1;
  ```

  Cette commande va modifier la colonne `KEY` avec `2` au lieu de `1` .
- Vous ne pouvez pas utiliser les tables temporaires plus d'une fois dans la même requête. Par exemple, cette commande ne fonctionne pas :

  ```
  mysql> SELECT * FROM temporary_table, temporary_table AS t2;
  ```
- `RENAME` ne fonctionne pas avec les tables `TEMPORARY` , ou les tables utilisées dans un rassemblement ( `MERGE` ).
- L'optimiseur peut gérer la clause `DISTINCT` différemment si vous utilisez des colonnes cachées dans une jointure. Dans une jointure, les colonnes cachées sont comptées comme une partie du résultat (même si elles ne sont pas montrées), tandis que dans les requêtes normales, les colonnes cachées ne participent pas aux `DISTINCT` . Nous allons probablement modifier ceci dans le futur, pour ne jamais exploiter les colonnes cachées avec `DISTINCT` .Voici un exemple :

  ```
  SELECT DISTINCT mp3id FROM band_downloads
        WHERE userid = 9 ORDER BY id DESC;
  ```

  et

  ```
  SELECT DISTINCT band_downloads.mp3id
        FROM band_downloads,band_mp3
        WHERE band_downloads.userid = 9
        AND band_mp3.id = band_downloads.mp3id
        ORDER BY band_downloads.id DESC;
  ```

  Dans le second cas, MySQL 3.23.x pourrait vous donner deux lignes identiques dans le résultat (car les lignes cachées `id` diffèrent).

  Notez que cela n'arrive que pour les requêtes où vous n'avez pas de colonnes de la clause ORDER BY dans le résultat, ce que vous ne pourriez pas faire en ANSI SQL.
- Comme le serveur MySQL vous permet de travailler avec des tables qui ne supportent pas

les transactions, et donc, l'annulation `rollback` , certains comportements sont différents avec MySQL d'avec d'autres serveurs SQL. C'est nécessaire pour s'assurer que MySQL n'a jamais besoin d'annuler une commande SQL. Cela peut sembler un peu étrange au moment ou les colonnes doivent être vérifiées par l'application, mais cela vous fournit une accélération notable, à cause d'optimisations qui ne pourraient pas avoir lieu ailleurs.

Si vous donnez une valeur incorrecte à une colonne, MySQL va stocker le `meilleur code possible` dans la colonne, au lieu d'annuler la transaction :

- Si vous essayez de stocker une valeur qui est hors de l'intervalle de validité dans une colonne numérique, MySQL va stocker la plus petite ou la plus grande valeur qu'il connaisse dans cette colonne.
- Si vous essayez de stocker une chaîne qui ne commence pas par un chiffre dans une colonne numérique, MySQL va stocker 0.
- Si vous essayez de stocker la valeur `NULL` dans une colonne qui n'accepte pas la valeur `NULL` , le serveur MySQL va stocker 0 ou `''` (chaîne vide) à la place : ce comportement peut être modifié avec l'option de compilation −DDONT_USE_DEFAULT_FIELDS).
- MySQL vous autorise le stockage de dates erronées dans les colonnes de type `DATE` et `DATETIME` (comme 2000−02−31 ou 2000−02−00). L'idée est que ce n'est pas au serveur SQL de faire le travail de validation. Si MySQL peut stocker une date, et relire exactement cette date, alors MySQL va stocker cette date. Si la date est totalement fausse (hors de l'intervalle de validité du serveur), la valeur spéciale 0000−00−00 sera utilisée.
- Si vous utilisez une valeur non supportée avec une colonne de type `ENUM` , la valeur stockée sera la chaîne vide, de valeur numérique 0.
- Si vous utilisez une valeur invalide dans une colonne de type `SET` , la valeur sera ignorée.

- Si vous exécutez une `PROCEDURE` sur une requête qui retourne un résultat vide, dans certains cas, `PROCEDURE` ne transformera pas les colonnes.
- La création de table de type `MERGE` ne vérifie pas si les tables sous−jacentes sont de type compatible.
- Le serveur MySQL ne supporte pas encore les valeurs Server `NaN` , `-Inf` et `Inf` pour les doubles. Utiliser ces valeurs générera des problèmes lorsque vous essayerez d'exporter et d'importer des données. Comme solution temporaire, vous pouvez remplacer `NaN` par `NULL` (si possible) et `-Inf` et `Inf` par les valeurs maximales possibles des colonnes `double` .
- Les valeurs négatives passées à `LIMIT` sont traitées comme des valeurs positives.
- Si vous utilisez la commande `ALTER TABLE` pour ajouter un index de type `UNIQUE` à un table utilisée dans un rassemblement de tables `MERGE` , puis que vous utilisez `ALTER TABLE` pour ajouter un index normal à la table `MERGE` , l'ordre des clés sera différent pour les tables s'il y avait déjà une ancienne clé qui n'était pas unique. Ceci est dû au fait que `ALTER TABLE` place les clés `UNIQUE` avant les clés normales, pour être capable de détecter les clés doublons plus vite.

Les bogues suivants sont connus dans les anciennes versions de MySQL :

- Vous pouvez obtenir un thread gelé si vous utilisez la commande `DROP TABLE` sur une table qui fait partie des tables verrouillées par `LOCK TABLES` .
- Dans les cas suivants, vous pouvez obtenir un crash :
  - Le gestionnaire d'insertions retardées a déjà des insertions en attente pour une table.
  - `LOCK table` avec `WRITE` .
  - `FLUSH TABLES` .
- Pour les versions de MySQL avant la 3.23.2, une commande `UPDATE` qui modifiait une clé avec la clause `WHERE` sur la même clé, pouvait échouer car la même clé était utilisée pour

rechercher les lignes et la même ligne pouvait être trouvée plusieurs fois :

```
UPDATE tbl_name SET KEY=KEY+1 WHERE KEY > 100;
```

Un palliatif est :

```
mysql> UPDATE tbl_name SET KEY=KEY+1 WHERE KEY+0 > 100;
```

Cela fonctionnera, car MySQL ne va pas utiliser d'index sur une expression dans la clause `WHERE` .

- Avant la version 3.23 de MySQL, tous les types numériques étaient traités comme des champs à virgule fixe. Cela signifie que vous deviez spécifier le nombre de décimales que le champ devait avoir. Tous les résultats étaient retournés avec le nombre correct de décimales.

Pour les bogues spécifiques aux systèmes d'exploitation, voyez la section sur la compilation et le port.

# 1.9 Les évolutions de MySQL (la liste des tâches)

Cette section liste les fonctionnalités que nous prévoyons d'ajouter à MySQL.

La liste est grossièrement réalisée dans l'ordre de priorité. Si vous voulez modifier cet ordre, achetez une licence de support, et dites−nous ce que vous souhaitez voir en premier. MySQL Support and Licensing .

L'objectif est de supporter complètement la norme ANSI SQL99, tout en lui ajoutant de nombreuses améliorations. Le défi est de faire tout cela sans sacrifier la vitesse d'exécution, ni la qualité du code.

## 1.9.1 New Features Planned For 4.1

Les fonctionnalités suivantes ne sont pas encore en MySQL 4.1, mais il est prévu de les implémenter avant que MySQL 4.1 passe en version beta. Pour une liste des fonctionnalités déjà disponibles en MySQL 4.1, voyez Features Available From MySQL 4.1 .

- Support OpenSSL stable (MySQL 4.0 le supporte déjà de manière rudimentaire, mais ce n'est pas un support testé à 100%).
- Transtypages de jeux de caractères, et syntaxe pour gérer plusieurs jeux de caractères simultanés.
- Plus de tests sur les requêtes préparées et les jeux de caractères multiples.
- Options de groupe `ROLLUP` OLAP (Online Analytical Processing) pour les applications d'entrepôt de données.

Le développement des nouvelles fonctionnalités est passé en version 5.0.

## 1.9.2 New Features Planned For 5.0

Les fonctionnalités suivantes sont prévues pour la version 5.0. Notez que comme nous avons de nombreux développeurs qui travaillent sur différents projets, il peut aussi y avoir des ajouts de fonctionnalités. Il y a aussi des chances que ces fonctionnalités soient ajoutées en MySQL 4.1. Pour une liste des fonctionnalités déjà disponibles en MySQL 4.1, voyez Features Available From MySQL 4.1 .

*Procédures stockées*

Les procédures stockées sont actuellement développées avec notre version 5.0. Elles sont basées sur les spécifications SQL−99, qui a une syntaxe simple, proche (mais différente) de celle d'Oracle PL/SQL. Nous allons aussi ajouter un environnement SQL−99 pour permettre l'utilisation de langage externes et assurer (lorsque c'est possible) une compatibility avec PL/SQL et T−SQL.

*Nouvelle fonctionnalité*

Support élémentaire des curseurs.

*Compatibilité avec les standards, portabilité et migration.*

Ajout d'un véritable type `VARCHAR` (ce support est déjà disponible avec le format de table `MyISAM` ).

*Amélioration de la vitesse*

`SHOW COLUMNS FROM table_name` (utilisé par le client `mysql` pour permettre la recherche de nom de colonnes) ne devrait pas ouvrir la table, mais seulement le fichier de définition. Cela prendra moins de mémoire, et sera bien plus rapide.

Permettre à la commande `DELETE` sur les tables `MyISAM` l'utilisation des caches de lignes. Pour cela, nous devons modifier le thread de cache de ligne lorsque nous modifions le fichier `.MYD` .

Amélioration des tables en mémoire `HEAP` :

- Lignes de taille dynamique.
- Gestion plus rapide des lignes (moins de copies).

*Internationalisation*

Lorsque nous utilisons `SET CHARACTER SET` , nous devrions traduire toute la requête en un coup, et non par chaînes. Cela permettra aux utilisateurs d'exploiter des caractères traduits dans les noms de base, table et colonne.

*Amélioration de l'ergonomie*

Résolution du problème avec `RENAME TABLE` , utilisé sur une table active dans un `MERGE` , qui conduit à la corruption de la table.

## 1.9.3 Nouvelles fonctionnalités prévues pour 5.1

*Nouvelle fonctionnalité*

`FOREIGN KEY` supporte tous les types de tables.

Contraintes de niveau colonne.

Réplication protégé contre les crash.

Sauvegarde à chaud, avec peu d'impact sur les performances. La sauvegarde à chaud permettra de simplifier l'ajout d'esclaves de réplication sans avoir à éteindre le maître.

*Amélioration de la vitesse*

Nouveau format de définition des tables en format texte (fichiers `.frm` ) et un cache de table pour les définitions de tables. Cela permettra d'accélérer les requêtes de structure de tables, et de consolider le support des clés étrangères.

Optimisation du type `BIT` pour qu'il prenne 1 bit (actuellement, `BIT` prend 1 caractère).

*Amélioration de l'ergonomie*

Ajout Ajout d'option au protocole client/server pour obtenir des indications de progression pour les requêtes longues.

Implementation de `RENAME DATABASE` . Pour que cela soit sécuritaire pour tous les pilotes de stockage, cela doit fonctionner comme ceci :

- Crée une nouvelle base de données.
- Pour chaque table, faire un renommage d'une base à l'autre, comme nous le faisons avec `RENAME` .

Effacer l'ancienne base.

Nouvelle interface interne pour les fichirs. Cela rendra la gestion des fichiers bien plus générale, et simplifiera l'ajout d'extensions comme RAID. (l'implémentation actuelle est un hack).

## 1.9.4 Ce qui doit être fait dans un futur proche

*Nouvelle fonctionnalité*

Commande `CONNECT BY PRIOR ...` , inspirée d'Oracle, pour traiter les structures de type arbre (hiérarchical).

Tous les types manquants de ANSI92 et ODBC 3.0.

Ajout de `SUM(DISTINCT)` .

`INSERT SQL_CONCURRENT` et `mysqld --concurrent-insert` pour faire des insertions concurrents à la fin du fichier, même si il est verrouillé en lecture.

Permettre la modification de variables dans une commande `UPDATE` . Par exemple : `UPDATE TABLE foo SET @a=a+b,a=@a, b=@a+c` .

Changer le moment de modifications des variables, pour que l'on puisse les utiliser avec les commandes de groupement `GROUP BY` , comme ceci : `SELECT id, @a:=COUNT(*), SUM(sum_col)/@a FROM table_name GROUP BY id` .

Ajout de l'option `IMAGE` à la commande `LOAD DATA INFILE` pour ne pas modifier les champs `TIMESTAMP` et `AUTO_INCREMENT` .

Ajout de la syntaxe `LOAD DATE INFILE ... UPDATE` .

Pour les tables ayant des clés primaires, si les données contiennent une clé primaire, les données qui existent et correspondent à cette clé, sont modifiées. Cependant, les colonnes **omises** sont ignorées. Pour les tables ayant une clé primaire dont il manque une partie dans les données entrantes, ou qui n'ont pas de clé primaire, le flux est traité comme `LOAD DATA INFILE ... REPLACE INTO` .

Rendre la syntaxe `LOAD DATA INFILE` utilisable comme ceci :

```
 LOAD DATA INFILE 'file_name.txt' INTO TABLE tbl_name
     TEXT_FIELDS (text_field1, text_field2, text_field3)
     SET table_field1=CONCAT(text_field1, text_field2),
         table_field3=23
     IGNORE text_field3
```

Cela peut être utilisé pour ignorer des colonnes supplémentaires dans le fichier texte, ou pour modifier des colonnes en fonction de données dans les valeurs entrantes.

`LOAD DATA INFILE 'file_name' INTO TABLE 'table_name' ERRORS TO err_table_name` . Cela va faire que les erreurs et alertes ne seront pas enregistrées dans la table `err_table_name` .

Cette table a la strucutre suivante :

```
 line_number    - numéro de ligne dans le fichier texte
error_message  - message d'alerte ou d'erreur
et peut être
 data_line      - le numéro de ligne dans le fichier de données
```

De nouvelles fonctions pour travailler avec le type de colonne `SET` :

```
ADD_TO_SET(value,set)
REMOVE_FROM_SET(value,set)
```

Ajout de l'utilisation de `t1 JOIN t2 ON ...` et `t1 JOIN t2 USING ...` Actuellement, nous ne pouvons utiliser cette syntaxe qu'avec `LEFT JOIN` .

Si vous interrompez `mysql` au beau milieu d'une requête, vous pouvez ouvrir une autre connexion pour tuer la requête en cours. Ou bien, une tentative doit être faite pour détecter cela sur le serveur.

Ajouter d'une interface au moteur de sauvegarde pour que vous puissiez l'utiliser comme table système. Cela sera un peu lent si vous demandez toutes les informations de toutes les tables, mais très souple. `SHOW INFO FROM tbl_name` doit être implémenté pour connaitre les informations de bases des tables.

Permettre `SELECT a FROM crash_me LEFT JOIN crash_me2 USING (a)` ; dans ce cas, `a` est supposé provenir de la table `crash_me` .
Les options `DELETE` et `REPLACE` dans les commandes `UPDATE` (cela va effacer les lignes lors d'un doublon dans la requête).
Modification du format de `DATETIME` pour stocker les fractions de secondes.
Rendre possible l'utilisation de la librairie GNU regexp library au lieu de la librairie actuelle (la librairie GNU library devrait être plus rapide que l'ancienne).

*Compatibilité avec les standard, migration et portabilité.*

Ne pas utiliser de valeur par défaut ( `DEFAULT` ) pour les colonnes. Indiquer une erreur lors des commandes `INSERT` , si elle contient une colonne qui n'a pas de valeur par défaut.
Ajout des fonctions de groupement `ANY()` , `EVERY()` et `SOME()` . En ANSI SQL, elles ne fonctionnent que sur les colonnes de type booléens, mais nous pouvons étendre leur champ d,action aux colonnes et expression, en appliquant la règle suivante : value == 0 −> FALSE et value <> 0 −> TRUE.
Correction du type pour que `MAX(column)` ait le mêm type de colonne :

```
 mysql> CREATE TABLE t1 (a DATE);
mysql> INSERT INTO t1 VALUES (NOW());
mysql> CREATE TABLE t2 SELECT MAX(a) FROM t1;
mysql> SHOW COLUMNS FROM t2;
```

*Amélioration des performances*

Ne permettre qu'à un nombre défini de threads d'exécuter des réparations MyISAM simultanément.
Changer `INSERT ... SELECT` pour utiliser optionnellement des insertions simultanées.
Ajouter une option pour écrire sur le disque les pages de clés pour les tables à clés retardées, qui n'ont pas été sauvées dernièrement.
Permettre les jointures sur des parties de clés (problème d'optimisation).
Ajout de la simulation de `pread()` / `pwrite()` sur Windows pour permettre les insertions concurrentes.
Un analyseur de logs, qui permettrait d'extraire des informations sur les tables les plus souvent utilisées, les jointures, etc... Cela permettra aux utilisateurs d'identifier les tables qui peuvent être optimisées.

*Internationalisation*

*Ergonomie*

Retourner le type de champs original lors des requêtes `SELECT MIN(column) ... GROUP BY` .
Rendre possible la spécification de `long_query_time` avec une granularité de l'ordre de la micro seconde.
Intégrer le code de `myisampack` dans le serveur, activer les commandes PACK et COMPRESS depuis le serveur.
Ajouter un cache de buffer de clé temporaire durant les commandes `INSERT/DELETE/UPDATE` pour que nous puissions nous rétablir si jamais le fichier d'index est plein.
Si vous effectuez la commande `ALTER TABLE` sur une table qui est un lien symbolique sur un autre disque, les tables tempooraires seront aussi faites sur ce disque.
Implémenter un type `DATE/DATETIME` qui gère les fuseaux horaire correctement, pour résoudre le problème du décalage horaire.
Corriger la configuration pour que l'on puisse compiler toute les librairies (comme `MyISAM` ) sans les threads.
Permettre toutes les variables SQL dans la clause `LIMIT` , comme par exemple `LIMIT @a,@b` .
Affichage automatique de `mysql` vers un serveur web.
`LOCK DATABASES` (avec différentes options).

De nombreuses nouvelles variables dans `SHOW STATUS` . Les lignes lues et modifiées. Les sélections sur tables uniques et les jointures. Le nombre moyen de tables dans une sélection. Le nombre de rquêtes `ORDER BY` et `GROUP BY` . `mysqladmin copy database new-database` ; a besoin de la commande `COPY` dans `mysqld` .
La liste des processus doit afficher le nombre de requêtes et de threads. `SHOW HOSTS` affichera les informatiosn concernant le cache de noms d'hôtes.
Changer le nom des tables de chaîne vide à `NULL` pour les colonnes calculées.
Ne pas utiliser la fonction `Item_copy_string` sur les valeurs numériques, pour éviter les conversions nombre -> chaîne -> nombre en cas de : `SELECT COUNT(*)*(id+0) FROM table_name GROUP BY id`
Changer `ALTER TABLE` pour qu'il n'interrompe pas le client qui exécute des `INSERT DELAYED` .
Lorsque les colonnes sont spécifiées dans `UPDATE` , elles contiennent les anciennes valeurs d'avant la modification.

*Nouveaux systèmes d'exploitation.*

Port de MySQL vers QNX.
Port de MySQL vers BeOS.
Port des clients MySQL vers LynxOS.

## 1.9.5 Ce qui est prévu à moyen terme

- Implémenter la fonction : `get_changed_tables(timeout,table1,table2,...)` .
- Remplacer la lecture dans les tables par une zone mémoire aussi souvent que possible. Actuellement, seules les tables compressées utilisent des memmap.
- Rendre le code des timestamp automatiques bien plus pratique. Ajouter les timestamps dans le log avec `SET TIMESTAMP=#;` .
- Utiliser un mutex de lecture/écriture pour gagner de la vitesse.
- Support des vues simples (d'abord sur une table, ensuite sur une expression).
- Fermer automatiquement des tables si une table, une table temporaire ou un fichier temporaire reçoit une erreur 23 (plus assez de fichiers ouverts).
- Lors de la rencontre de field=#, changez toutes les occurrences du champ en #. Actuellement, cela n'est fait que pour les cas simples.
- Changer toutes les expressions constantes par des expressions calculées, si possible.
- Optimiser la relation clé = expression. Actuellement, seul les relations clé = champ et clé = constante sont optimisées.
- Fusionner les fonctions de copie pour améliorer le code.
- Changez `sql_yacc.yy` pour le remplacer par un analyseur de ligne de commande plus petit, et qui gère mieux les messages.
- Changer l'analyseur pour utiliser uniquement une règle pour tous les nombres d'arguments possibles dans une fonction.
- Utiliser les calculs de noms complets dans la clause ORDER (pour ACCESS97).
- `MINUS` , `INTERSECT` et `FULL OUTER JOIN` . (actuellement, `UNION` [en 4.0] et `LEFT OUTER JOIN` fonctionnent).
- `SQL_OPTION MAX_SELECT_TIME=#` pour donner une limite de temps à une requête.
- Diriger le log d'historique vers une base.
- Améliorer `LIMIT` pour permettre la lecture de données à la fin du résultat.
- Alertes lors des connexions/écritures/lectures du client.
- Notez ces modifications de `safe_mysqld` : selon la FSSTND (que Debian essaie de suivre) les fichiers PID devraient être placés dans `/var/run/<progname>.pid` et les fichiers de logs dans `/var/log` . Il serait bien si vous pouviez mettre le "DATADIR" dans la première déclaration de "pidfile" et "log", de façon à ce que l'emplacement de ces fichiers puisse être modifié en une seule ligne.
- Permettre au client de commander le log des actions.
- Ajouter l'utilisation de `zlib()` pour les fichiers `gzip` , avec la commande `LOAD DATA INFILE` .
- Corriger le tri et le groupage avec les colonnes `BLOB` (en partie résolu).

- Utiliser des sémaphores pour compter les threads. Il faut commencer par implémenter des sémaphores pour MIT−pthreads.
- Ne pas assigner de nouvelle valeur AUTO_INCREMENT lorsque l'on utilise la valeur 0. Utilisez NULL à la place.
- Ajouter le support complet pour les JOIN avec parenthèses.
- Comme alternative à la relation un thread, une connexion, gérer un groupe de threads pour répondre aux requêtes.
- Permettre la pose de plusieurs verrous avec GET_LOCK . Lors de ces verrous multiples, gérer le cas des blocages par verrous qui pourrait être introduit.

Le temps est indiqué en temps de travail et non pas en temps normal.

### 1.9.6 Ce qui n'est pas prévu

- Rien. Nous nous dirigeons vers la compatibilité complète avec ANSI 92/ANSI 99.

## 1.10 Comparatif de MySQL avec les autres serveurs SQL

Nos utilisateurs ont réussi à exécuter nos tests avec un grand nombre de serveurs SQL Open Source ou non. Nous connaissons des tests exécutés avec Oracle , DB/2 , Microsoft SQL Server et d'autres produits commerciaux. Pour des raisons légales, nous ne pouvons pas les publier dans notre manuel de référence.

Cette section inclut la comparaison de MySQL avec mSQL pour des raisons historiques, et avec PostgreSQL car c'est aussi une base Open Source . Si vous avez des résultats de tests que nous pouvons publier, contactez−nous à benchmarks@mysql.com .

Pour une liste comparative des fonctions et types supportés, voyez la page web de l'utilitaire crash−me à l'adresse http://www.mysql.com/information/crash−me.php .

### 1.10.1 MySQL face à mSQL

*Performance*

Pour un véritable test de vitesse, consultez la suite de test de MySQL . La suite de tests comparatifs de MySQL .

Comme il n'y a pas de création de threads, que l'analyseur est plus petit, et que les fonctionnalités sont moins nombreuses, mSQL devrait être plus rapide avec :

- les tests qui réalisent des connexions répétées et qui n'exécutent qu'une simple requête à chaque fois.
- les opérations d' INSERT dans des tables très simples, avec peu de colonnes et d'index.
- les commandes de CREATE TABLE et DROP TABLE .
- les commandes SELECT sur tout ce qui n'est pas indexé (une analyse de table est très facile).

Comme ces opérations sont très simples, il est difficile d'être meilleur lorsque les coûts d'administration sont bien plus forts. Une fois la connexion établie, le serveur MySQL devrait être bien plus rapide.D'un autre coté, le serveur MySQL est plus rapide que mSQL (et que les autres serveurs SQL) avec :

les opérations SELECT complexes.
les lectures de grands résultats (le serveur MySQL a un protocole plus rapide, plus sûr et bien meilleur).
les tables à taille de chaînes variable, car le serveur MySQL a une gestion plus efficace et utilise les colonnes VARCHAR indexées.
la gestion des tables avec de nombreuses colonnes.
la gestion des tables ayant des grandes lignes.
les commandes SELECT avec de nombreuses expressions.
les commandes SELECT sur de grandes tables.
la gestion de nombreuses connexions simultanées. Le serveur MySQL est complètement multi−threadé. Chaque connexion possède son propre thread, ce qui signifie qu'il y a pas d'attente entre les threads (à moins que l'un d'entre eux modifie une table que l'autre veut utiliser). En mSQL , une fois que la connexion a été établie, les autres doivent attendre que la première ait fini, indépendamment de la durée d'exécution de la requête. Lorsque la première connexion se termine, le suivant peut être servi, tandis que les autres attendent.
les jointures. mSQL peut devenir irrémédiablement lent si vous modifiez l'ordre des tables dans la colonne SELECT . Dans la suite de test, il s'est vu des tests 15000 fois plus lent que MySQL . Ceci est dû au manque d'optimiseurs chez mSQL . Toutefois, si vous avez placé les tables dans le bon ordre avec mSQL 2 et que la clause WHERE est simple et utilise les index, la jointure peut être plutôt rapide. La suite de tests comparatifs de MySQL .
les clauses ORDER BY et GROUP BY .
la clause DISTINCT .
l'utilisation des colonnes de type TEXT et BLOB .

*Support de SQL*

GROUP BY et HAVING . mSQL ne supporte pas du tout la clause GROUP BY . MySQL supporte totalement la clause GROUP BY avec les clauses HAVING et les fonctions suivantes : COUNT() , AVG() , MIN() , MAX() , SUM() et STD() . COUNT(*) est optimisée pour retourner très vite le résultat si SELECT lit les lignes dans une table, qu'aucune autre colonne n'est lue, et qu'il n'y a pas de clause WHERE . MIN() et MAX() accepte les chaînes de caractères comme arguments.
Les INSERT et UPDATE avec calculs d'expression. MySQL peut faire des calculs d'expression dans les commandes INSERT et UPDATE . Par exemple :

```
mysql> UPDATE SET x=x*10+y WHERE x<20;
```

Aliasing. MySQL supporte les alias de colonne.
Identification simple des colonnes. Avec MySQL , si un nom de colonne est unique dans les tables utilisées pour une jointure, vous n'avez pas à utiliser son identifiant total.
SELECT avec des fonctions MySQL dispose de nombreuses fonctions (trop nombreuses à lister ici : voir Fonctions ).

*Efficacité d'utilisation de l'espace disque*

C'est à dire, comment pouvez−vous réduire la taille de vos tables ? MySQL a des types très précis, et vous pouvez créer des tables avec un très petit espace. Un exemple de type de données pratique est le type MEDIUMINT qui occupe 3 octets. Si vous avez 100 millions d'enregistrement, économiser un octet par enregistrement est très important.

mSQL2 a un jeu de type de données plus limité, ses tables sont donc plus grosses.

*Stabilité*

Il est difficile de juger de cela objectivement. Pour un début sur la stabilité du serveur MySQL , voyez Degré de stabilité de MySQL .

Nous n'avons aucune expérience avec la stabilité du code de mSQL , alors nous ne nous prononceront pas la−dessus.

*Prix*

Un autre point important est la licence. MySQL dispose d'une licence plus souple que mSQL , est il est aussi moins cher que mSQL . Quelque soit le produit que vous utilisez, n'oubliez pas de prendre un contrat de support par email ou une licence commerciale.

*interface Perl*

MySQL a la même interface avec Perl que mSQL , avec des fonctionnalités supplémentaires.

*JDBC (Java)*

MySQL supporte de nombreux pilotes JDBC :

MySQL Connector/J est un pilote natif Java. La version 3.x est publiée sous une licence double (GPL et commerciale).

Le pilote Resin : un pilote JDBC commercial et Open Source. http://www.caucho.com/projects/jdbc−mysql/index.xtp

Le pilote gwe : une interface Java par GWE technologies (non supporté).

Le pilote jms : un pilote gwe amélioré par Xiaokun Kelvin ZHU X.Zhu@brad.ac.uk (non supporté).

Le pilote twz : un pilote JDBC de type 4, écrit par Terrence W. Zellers zellert@voicenet.com . Il est commercialisé, mais gratuit pour les utilisations personnelles et dans le cadre des études (non supporté).

Le pilote recommandé est le pilote mm. Le pilote Resin est aussi bon (les tests sont bons) mais nous n'avons pas reçu beaucoup d'informations à son propos.Nous savons que mSQL a un pilote JDBC, mais nous n'avons pas l'expérience pour le comparer.

*Vitesse de développement*

MySQL a une petite équipe de développeur, mais nous sommes très habitués à coder en C et C++, très rapidement. Comme les threads, les fonctions, GROUP BY et de nombreuses autres fonctionnalités ne sont pas encore codées pour mSQL , il y a beaucoup faire pour nous rattraper. Pour mesurer cela, vous pouvez consulter le fichier d'historique de mSQL HISTORY pour l'an dernier, et le comparer avec la section nouveautés du manuel de référence MySQL ( Historique des versions de MySQL ). Il est alors assez évident de voir qui se développe plus vite.

*Programmes utilitaires*

Les deux projets mSQL et MySQL ont des contributions de tiers très intéressantes. Comme il est plus facile de porter vers les versions plus récentes (depuis mSQL vers MySQL ), presque toutes les applications intéressantes de mSQL sont disponibles pour MySQL .

MySQL est livré avec un programme simple msql2mysql qui corrige les différences de syntaxe entre mSQL et MySQL pour la plupart des fonctions de l'API C. Par exemple, il modifie les occurrences de msqlConnect() en mysql_connect() . Convertir un programme client de mSQL vers MySQL requiert un effort minimal.

#### 1.10.1.1 Comment convertir des outils mSQL pour MySQL

Selon notre expérience, il ne prend pas longtemps pour convertir des outils tels que msql−tcl et msqljava qui utilisent l'API C de mSQL pour qu'ils fonctionnent avec l'API C de MySQL.

La procédure de conversion est la suivante :

- Exécutez le script shell msql2mysql sur la source. Ce script requiert le programme replace , qui est distribué avec le serveur MySQL.

- Compilez.
- Corrigez les erreurs de compilation.

Les différences entre l'API C de mSQL et de MySQL sont :

- MySQL utilise la structure MYSQL comme type de connexion, alors que mSQL utilise un int .
- mysql_connect() prend un pointeur sur la structure MYSQL comme paramètre. Il est facile d'en définir un globalement, ou d'utiliser la fonction malloc() pour en obtenir un. mysql_connect() prend aussi deux paramètres supplémentaires, qui sont le nom de l'utilisateur et son mot de passe. Vous pouvez aussi utiliser NULL, NULL pour retrouver l'utilisation classique.
- mysql_error() prend la structure MYSQL comme paramètre. Ajoutez simplement ce paramètre à votre vieux code msql_error() , si vous portez du vieux code.
- MySQL retourne un numéro d'erreur et un message d'erreur pour tous les messages d'erreurs. mSQL ne retourne que le texte d'erreur.
- Certaines incompatibilités existent du fait que le serveur MySQL supporte des connexions multiples, dans le même processus.

### 1.10.1.2 Différences entre les protocoles de communication de mSQL et de MySQL

Il y a tellement de différences qu'il est impossible (ou tout au moins très difficile) de ne pas les supporter toutes.

Les traits les plus caractéristiques de différences entre le protocole de communication de MySQL et de mSQL :

- Un buffer de message peut contenir plusieurs lignes.
- Le buffer de message est dynamiquement agrandi si la requête ou le résultat sont plus grands que le buffer courant, jusqu'à une taille limite configurable.
- Tous les paquets sont numérotés pour identifier les paquets doublons ou manquants.
- Toutes les valeurs de colonnes sont échangées en mode ASCII. La taille de la colonne est envoyée dans un paquet binaire (1,2 ou 3 octets).
- MySQL peut lire un résultat non bufferisé (sans avoir a stocker tout le résultat dans le client).
- Si une opération de lecture/ecriture prendre plus de 30 secondes, le serveur ferme la connexion.
- Si la connexion est inactive pendant 8 heures, le serveur ferme la connexion.

### 1.10.1.3 Comparatif des syntaxes SQL de mSQL 2.0 et MySQL

#### Types de colonnes

*Serveur MySQL*

Dispose des types traditionnels (entre autres : Syntaxe de CREATE TABLE ):

le type ENUM pour les chaînes à valeurs limitées.

le type SET pour les combinaisons de valeurs.

BIGINT pour les entiers 64−bit.

*MySQL supporte aussi les attributs de types suivants :*

option UNSIGNED pour les entiers et nombres à virgule flottante.

option ZEROFILL pour les entiers.

option AUTO_INCREMENT pour les colonnes qui sont PRIMARY KEY . mysql_insert_id() .

valeur par DEFAULT pour toutes les colonnes.

*mSQL2*

Les types de colonnes de mSQL correspondent aux types de MySQL dans cette table :

| mSQL type | Type MySQL correspondant |
|---|---|
| CHAR(len) | CHAR(len) |
| TEXT(len) | TEXT(len) . len est la taille maximale. Et LIKE fonctionne. |
| INT | INT . Avec de nombreuses options supplémentaires. |
| REAL | REAL . Ou FLOAT . Les versions 4 et 8 octets sont disponibles. |
| UINT | INT UNSIGNED |
| DATE | DATE . Utilise le format ANSI SQL plutôt que le format spécifique de mSQL . |
| TIME | TIME |
| MONEY | DECIMAL(12,2) . Un nombre à virgule fixe. |

### Création d'index

*Serveur MySQL*
: Les index peuvent être spécifiés au moment de la création de la table avec la commande `CREATE TABLE` .

*mSQL*
: Les index doivent être créés après la création de la table, avec une commande `CREATE INDEX` séparée.

### Insertion d'un identifiant unique dans une table

*Serveur MySQL*
: Utilisez l'option de colonne `AUTO_INCREMENT` . `mysql_insert_id()` .

*mSQL*
: Créez une `SEQUENCE` dans la table et sélectionnez la colonne `_seq` .

### Obtenir un identifiant unique pour une ligne

*Serveur MySQL*
: Ajoutez la clé `PRIMARY KEY` ou `UNIQUE` à la table et utilisez ceci. Nouveau en version 3.23.11 : si la clé `PRIMARY` ou `UNIQUE` est constituée d'une seule colonne et qu'il est de type entier, vous pouvez y faire référence avec `_rowid` .

*mSQL*
: Utilisez la colonne `_rowid` . Notez bien que `_rowid` peut changer durant ce temps, en fonction de nombreux facteurs.

### Lire la date de dernière modification d'une colonne

*Serveur MySQL*
: Ajoutez une colonne de type `TIMESTAMP` dans la table. Cette colonne prend automatiquement la date et l'heure courante lors des commandes `INSERT` ou `UPDATE` si vous ne lui affectez pas de valeur, ou si vous lui donnez la valeur `NULL` .

*mSQL*
: Utilisez la colonne `_timestamp` .

### Comparaisons avec la valeur `NULL`

*Serveur MySQL*
: MySQL suit la norme ANSI SQL, et la comparaison avec une valeur `NULL` retourne toujours `NULL` .

*mSQL*
: Avec `mSQL` , `NULL = NULL` est vrai (TRUE). Vous devez changer `=NULL` en `IS NULL` et `<>NULL` en `IS NOT NULL` lorsque vous portez du vieux code de `mSQL` vers MySQL.

### Comparaisons de chaînes

*Serveur MySQL*

Normalement, la comparaison de chaînes est faîte sans tenir compte de la casse, et l'ordre de tri est déterminé par le jeu de caractères courant (ISO−8859−1 Latin1 par défaut). Si vous ne le souhaitez pas, déclarez vos colonnes avec l'option `BINARY` , qui fera que les comparaisons peuvent être faites en fonction de l'ordre ASCII utilisé sur l'hôte MySQL.

*mSQL*

Toutes les comparaisons de chaînes sont faites en tenant compte de la casse, en fonction de l'ordre ASCII.

**Recherche insensible à la casse**

*Serveur MySQL*

`LIKE` est un opérateur sensible ou non à la casse, suivant les colonnes utilisées. Si possible, MySQL utilise des index si l'argument de `LIKE` ne commence pas par un caractère joker.

*mSQL*

Utilise `CLIKE` .

**Gestion des espaces finaux**

*Serveur MySQL*

Supprime tous les espaces de fin de chaînes dans les colonnes `CHAR` et `VARCHAR` . Utilisez les colonnes de type `TEXT` si ce comportement n'est pas souhaité.

*mSQL*

Conserve les espaces finaux.

**clause** `WHERE`

*Serveur MySQL*

MySQL établit correctement les priorités : `AND` est calculé avant les `OR` . Pour obtenir le comportement de `mSQL` avec MySQL, utilisez les parenthèses (tel que présenté dans l'exemple un peu plus loin).

*mSQL évalue les opérations de gauche à droite. Cela signifie que*

certaines opérations logiques exprimées avec trois arguments ou plus ne peuvent être exprimées. Cela signifie aussi que vous devez modifier des requêtes lorsque vous passez à MySQL : tout ce que vous avez à faire est d'ajouter des parenthèses. Supposez que vous avez le code `mSQL` suivant :

```
mysql> SELECT * FROM table WHERE a=1 AND b=2 OR a=3 AND b=4;
```

Pour le passer sur MySQL et le faire fonctionner comme sur `mSQL` , vous devez ajouter ces parenthèses :

```
mysql> SELECT * FROM table WHERE (a=1 AND (b=2 OR (a=3 AND (b=4))));
```

**Contrôle d'accès**

*Serveur MySQL*

Possède des tables qui stockent les droits de chaque utilisateur, hôte et base. Comment le système de droits fonctionne .

*mSQL*

Dispose d'un fichier `mSQL.acl` dans lequel vous pouvez ajouter et modifier les droits d'accès des utilisateurs.

## 1.10.2 Comparatif de MySQL avec PostgreSQL

Lorsque vous lirez ces sections, notez bien que les deux produits évoluent continuellement. Les équipes de MySQL AB et de PostGreSQL travaillent toutes deux à rendre leur base les meilleures possibles, et elles présentent toutes les deux des alternatives sérieuses aux bases de données commerciales.

La comparaison qui suit a été fait par MySQL AB. Nous avons essayé d'être aussi précis et objectifs que possible, mais même si nous connaissons MySQL sur le bout des doigts, nous n'avons pas une connaissance aussi pointues des possibilitées de PostgreSQL : nous pourrions être en erreur. Nous corrigerons ces erreurs dès qu'elles seront portées à notre attention.

Nous souhaitons noter que PostgreSQL et MySQL sont deux produits largement répandus, mais qui présentent des objectifs différents, même si tous les deux se dirigent vers la compatibilité ANSI SQL. Cela signifie que pour certaines applications, MySQL est bien plus optimisé, et pour d'autres, c'est PostGreSQL. Lorsque vous choisissez votre serveur de base de données, commencez par vérifier que le jeu de fonctionnalités de votre base correspond bien à votre application. Si vous avez besoin de vitesse pure, MySQL est probablement votre choix. Si vous avez besoin des fonctionnalités que PostGreSQL propose, utilisez `PostgreSQL` .

### 1.10.2.1 Stratégies de développement de MySQL et PostgreSQL

Lorsque nous ajoutons des fonctionnalités à MySQL, nous mettons un point d'honneur à le faire de manière optimale et définitive. Le code doit être tellement bon que vous n'y verrez rien à ajouter dans un futur accessible. Nous sommes peu enclins à sacrifier la vitesse aux fonctionnalités, mais nous faisons notre possible pour trouver une solution qui donnera le meilleur. Cela signifie que le développement prend plus de temps, mais que le résultat en vaut la peine. Ce type de développement n'est possible que parce que le code est vérifié par quelques (actuellement 2) personnes avant qu'il ne soit inclus dans le serveur.

Chez MySQL AB, nous croyons aux publications fréquentes car cela nous permet de transmettre nos nouveautés plus rapidement aux utilisateurs. Comme nous publions une nouvelle version mineure toutes les trois semaines, et une version majeure chaque année. Toutes les versions sont testées soigneusement, avec nos outils de tests et sur de nombreuses plate−formes.

PostgreSQL est basé sur un noyau de nombreux contributeurs. Avec cette configuration, il est logique d'ajouter de nombreuses fonctionnalités, au lieu de les implanter optimalement, car il sera toujours temps d'optimiser lorsque le besoin s'en fera sentir.

Une autre différence majeure entre MySQL et PostgreSQL est que presque tout le code de MySQL est fait par des développeurs employés par MySQL AB et qui travaillent toujours sur ce code. Les seules exceptions sont le moteur de transaction et la librairie d'expressions régulières.

Le contraste est saisissant avec le code de PostgreSQL, dont la majorité est réalisée par un grand groupe de personnes avec différentes sensibilités. Ce n'est que récemment que les développeurs de PostGreSQL ont annoncé qu'ils ont eu finalement le temps de faire le tour du code de la version courante.

Les deux stratégies de développement présentées ont leur mérites et inconvénients. Chez MySQL AB, nous pensons, bien sur, que notre modèle est le meilleur car il conduit à une meilleure cohérence du code, et, à notre avis, moins de bogues. Comme nous sommes les auteurs du serveur MySQL, nous sommes les mieux placés pour coordonner les publications.

### 1.10.2.2 Comparaison des fonctionnalités de MySQL et PostgreSQL

Sur la page `crash-me` ( http://www.mysql.com/information/crash-me.php ) vous pouvez trouver une liste des avantages et limitations que l'on peut détecter avec un programme. Notez bien que de nombreuses évaluations numériques peuvent être modifiées avec les options de démarrage des différentes bases de données. Cette page web est extrêmement pratique si vous voulez vous assurer que votre application fonctionnera sur différents serveurs SQL, ou si vous voulez convertir votre application d'un serveur à l'autre.

MySQL offre les avantages suivants sur PostgreSQL:

- `MySQL` est généralement plus rapide que PostgreSQL. MySQL 4.0.1 a aussi un cache de requêtes qui peut accélérer les requêtes pour les lectures de tables les plus fréquentes.
- MySQL a une base d'utilisateurs bien plus grande que PostgreSQL. Par conséquent, le code est testé dans de meilleures conditions, et il a été historiquement plus stable que PostgreSQL. MySQL est utilisé plus souvent en environnement de production que PostgreSQL, notamment grâce au fait que MySQL AB, ex TCX DataKonsult AB, a fourni un support commercial de haute qualité pour le serveur MySQL depuis la première publication, tandis que, jusqu'à récemment, PostgreSQL n'avait pas de support.
- MySQL fonctionne bien mieux sur Windows que PostgreSQL. MySQL fonctionne comme une application Windows native (un service sur NT/2000/XP), tandis que PostgreSQL fonctionne en mode émulation de `Cygwin` . Nous avons entendu dire que PostgreSQL n'est pas encore stable sous Windows mais non n'avons pas été capable de le vérifier nous−mêmes.
- MySQL a plus d'API avec d'autres langages, et est supporté par un plus grand nombre de programmes que PostgreSQL. Contributions .
- MySQL fonctionne avec les systèmes critiques, à forte charge, 24h/24 et 7j/7. Dans la plupart des cas, nous n'avons jamais à effectuer d'entretien de MySQL. PostgreSQL ne supporte par encore les systèmes 24h/24 et 7j/7, car il faut exécuter `VACUUM` une fois de temps en temps pour récupérer l'espace vide issu des commandes `UPDATE` et `DELETE` , et pour faire des analyses statistiques qui sont primordiales pour obtenir de bonnes performances avec PostgreSQL. `VACUUM` est aussi nécessaire après avoir ajouté un grand nombre de lignes dans une table. Sur un système très occupé avec de nombreuses modifications, `VACUUM` doit être exécuté très fréquemment, et parfois, plusieurs fois dans la journée. Durant l'exécution de `VACUUM` , qui peut prendre plusieurs heures si la base de données est importante, le serveur est, d'un point de vue production, quasiment mort. Notez qu'avec PostgreSQL version 7.2, l'entretien simple des tables ne nécessite plus le verrouillage des tables, et le niveau d'utilisation est alors à peu près normal. Une nouvelle commande `VACUUM FULL` effectue les entretiens traditionnels en verrouillant la table, et en réduisant sa taille sur le disque.
- Le système de réplication de MySQL a été testé soigneusement, et il est utilisé sur des sites comme :
    - Yahoo Finance ( http://finance.yahoo.com/ )
    - Mobile.de ( http://www.mobile.de/ )
    - Slashdot ( http://www.slashdot.org/ )
- Dans les deux distributions de MySQL, vous trouverez des suites de tests, `mysql-test-run` et `crash-me` ( http://www.mysql.com/information/crash−me.php ), ainsi que la suite de performances. Le système de tests est souvent modifié avec des nouveaux tests pour les nouvelles fonctionnalités, ainsi que les bogues qui nous sont rapportés. Nous testons chacune de nos versions sur un grand nombre de plate−formes. Ces tests sont plus sophistiqués que tout ce que nous avons pu voir chez PostgreSQL, et il nous assurent que le serveur MySQL reste de bonne qualité.
- Il y a bien plus de livres concernant MySQL que PostgreSQL. O'Reilly, SAMS, Que etNew Riders sont des éditeurs importants, avec des livres traitant de MySQL. Toutes les fonctionnalités de MySQL sont aussi documentées sur le manuel en ligne, car lorsqu'une nouvelle fonctionnalité est implémentée, les développeurs MySQL doivent la documenter

avant qu'elle soit inclue dans le source.

- MySQL support plus de fonctions ODBC que `PostgreSQL` .
- MySQL a une commande `ALTER TABLE` beaucoup plus puissante.
- MySQL supporte les tables sans transaction pour les applications qui ont besoin de plus de vitesse. Les tables peuvent être placées en mémoire, comme les tables `HEAP` , ou bien sur le disque, comme les tables `MyISAM` . Types de tables MySQL .
- MySQL supporte deux types de gestionnaires de tables, qui supportent les transactions : `InnoDB` et `BerkeleyDB` . Comme chaque moteur de transactions travaille dans différentes conditions, cela donne plus de possibilités aux auteurs d'applications pour choisir leur solution optimale pour la configuration du serveur, au besoin table par table. Types de tables MySQL .
- Les tables groupées `MERGE` vous donnent un moyen d'accéder instantanément a plusieurs tables de même structure, et de les utiliser ensemble, comme une seule. C'est le moyen parfait lorsque vous avez des tables de logs trop grosses, et que vous les séparez par mois. Tables `MERGE` .
- La possibilité de compresser les tables en lecture seule, mais tout en ayant un accès direct aux lignes de la table, vous donnent plus de performance en lecture, en réduisant la taille des données lues sur le disque. C'est très pratique pour les archives. `myisampack` , le générateur de tables MySQL compressées en lecture seule .
- MySQL possède le support d'outils de recherche en texte plein. Recherche full−text avec MySQL .
- Vous pouvez accéder à de nombreuses bases de données avec la même connexion, du moment que vous avez les droits pour le faire.
- MySQL est codé depuis le début pour être multi−threadé, tandis que PostgreSQL utilise les processus. Le temps de changement de contexte et d'accès aux zones de stockage est bien plus court avec des threads qu'entre des processus séparés. Cela donne un avantage de vitesse à MySQL dans les applications multi−utilisateurs, et aussi avec les systèmes à plusieurs processeurs symétriques.
- MySQL a un système de droits bien plus avancé que PostgreSQL. Alors que PostgreSQL ne supporte que des droits pour les commandes `INSERT` , `SELECT` et `UPDATE/DELETE` au niveau de l'utilisateur pour une base ou une table, MySQL vous permet de définir un jeu de droits complet pour une base, une table ou une colonne. MySQL vous permet aussi de spécifier des droits par hôte ou utilisateur. Syntaxe de `GRANT` et `REVOKE` .
- MySQL supporte un protocole client/serveur compressé qui améliore grandement les performances sur les connexions lentes.
- MySQL applique le concept des ``gestionnaires de table'', et est la seule base de données relationnelle que nous connaissons qui utilise ce concept. Cela permet différents types de tables depuis le moteur SQL, et chaque table peut être optimisée pour différentes performances.
- Tous les types de tables MySQL (excepté `InnoDB` ) sont implémentés avec des fichiers (une table par fichier), ce qui rend très souple les sauvegardes, les déplacements et les effacements, ou même l'utilisation des liens symboliques pour les tables et les bases, même lorsque le serveur est éteint.
- Les outils pour réparer et optimiser les tables `MyISAM` (le type de tables MySQL le plus courant). Un outil de réparation n'est nécessaire que pour les corruptions physiques dans un fichier de données, et généralement, pour corriger une défaillance du matériel. Elle permet à la majorité des données d'être récupérées.
- Changer de version de MySQL est très simple. Lorsque vous changer de version de MySQL, vous n'avez pas à exporter et réimporter vos données, comme vous devez le faire avec PostgreSQL.

Les inconvénients de MySQL face à PostgreSQL:

- Le support des transactions de MySQL n'est pas aussi bien testé que celui de PostgreSQL.

- Comme le serveur MySQL utilise les threads, qui ne sont pas encore parfaitement rodés sur tous les systèmes d'exploitation, il faut alors utiliser les exécutables binaires de http://www.mysql.com/downloads/ , ou bien lire avec beaucoup de soin les instructions dans la section Installer MySQL à partir des sources pour obtenir un exécutable qui fonctionne de manière optimale.
- Le verrouillage de table, tel qu'utilisé par les tables non transactionnelles `MyISAM` , est dans de nombreux cas plus rapide que les verrous de page, de ligne ou le versionnage. L'inconvénient est que si l'on ne prend pas garde au fonctionnement du verrouillage de table, une requête seule, à longue durée d'exécution peut bloquer la table en écriture pour un long moment. Cela peut être évité en concevant correctement l'application. Sinon, on peut toujours changer le gestionnaire de la table pour un modèle transactionnel. Problèmes avec le verrouillage de tables .
- Avec UDF (user−defined functions, fonction définies par les utilisateurs), il est possible de faire évoluer les fonctions du serveur MySQL aussi bien pour les fonctions SQL normales que pour les fonctions d'agrégation, mais ce système n'est pas aussi pratique que pour PostgreSQL. Ajout de nouvelles fonctions à MySQL .
- Les modifications qui s'effectuent sur plusieurs tables sont bien plus difficiles à faire avec MySQL. Toutefois, cela a été corrigé avec la version 4.0.2 et les commandes `UPDATE` multi−tables, et en MySQL version 4.1, avec les sous−sélections. En MySQL version 4.0, on peut utiliser les effacements multi−tables pour effacer plusieurs lignes dans différentes tables simultanément. Syntaxe des `DELETE` .

PostgreSQL offre actuellement les avantages suivants sur MySQL :

Notez que comme nous connaissons le plan de développement de MySQL, nous avons inclus la table suivante, avec les version du serveur MySQL où les fonctionnalités seront supportées. Malheureusement, nous ne pouvions pas faire les mêmes comparaisons avec les fonctionnalités à venir de PostgreSQL, car nous ne connaissons pas le plan de développement de PostgreSQL.

| **Fonctionnalité** | **version de MySQL** |
|---|---|
| Sous−sélections | 4.1 |
| Clé étrangères | 5.0 (3.23 avec InnoDB ) |
| Vues | 5.0 |
| Procédures stockées | 5.0 |
| Triggers | 5.0 |
| Unions | 4.0 |
| Jointures totales | 4.1 |
| Contraintes | 4.1 ou 5.0 |
| Curseurs | 4.1 ou 5.0 |
| R−trees | 4.1 (pour les tables MyISAM) |
| Tables héritées | Non prévu |
| Type de données extensible | Non prévu |

Autres raisons de considérer PostgreSQL :

- L'utilisation standard de PostgreSQL est parfois plus proche de la norme ANSI SQL.
- Il est possible d'accélérer les traitements de PostgreSQL en codant des procédures stockées.
- Pour les données géographiques, les R−trees font de PostgreSQL un meilleur choix que MySQL. (note : MySQL version 4.1 dispose des R−trees pour les tables MyISAM).
- L'optimiseur PostgreSQL fait des optimisations que l'optimiseur de MySQL ne peut faire. La plus importante est lors de jointures, lorsqu'il n'y a pas les clés nécessaires, ou lorsque les jointures se font avec des clés combinées par un OR. La suite de tests de MySQL http://www.mysql.com/information/benchmarks.html montre quels types de constructions vous devriez éviter lorsque vous utilisez différentes bases de données.
- PostgreSQL a une plus grande équipe de développement qui contribue au serveur.

Les inconvénients de PostgreSQL face à MySQL :

- L'utilitaire VACUUM rend PostgreSQL difficile à utiliser en environnement de production 24h/24 et 7j/7.
- Les tables sont obligatoirement transactionnelles.
- Les commandes INSERT , DELETE et UPDATE sont plus lentes.

Pour une liste complète des inconvénients, examinez la première table de cette section.

### 1.10.2.3 Performances comparées de MySQL et PostgreSQL

Le seul test de vitesse Open Source que nous connaisssons et qui peut être utilisé pour tester les performances de MySQL et de PostgreSQL (et d'autres bases de données) est le nôtre. Il est disponible à l'adresse http://www.mysql.com/information/benchmarks.html .

Nous avons demandé de nombreuses fois aux développeurs de PostgreSQL et aux utilisateurs de PostgreSQL de nous aider à développer et améliorer ce test, pour en faire le meilleur test possible, mais nous n'avons pas reçu de réponse de leur part.

Nous, les développeurs MySQL, avons, à cause de cela, passé de nombreuses heures à obtenir les meilleures performances de PostgreSQL pour ces tests, mais comme nous ne connaissons pas intimement le serveur PostgreSQL, nous sommes certains d'être passé à coté de certains points. Sur la page de test, nous avons documenté exactement comment nous avons exécuté le test, pour qu'il soit facile à tout le monde de reproduire et de confirmer nos résultats.

Les tests sont généralement exécutés avec et sans l'option --fast . Lorsque l'option --fast est utilisée, nous tentons d'utiliser tous les trucs connus du serveur pour que l'exécution soit aussi rapide que possible. L'idée est que le mode normal doit montrer comment le serveur devrait se comporter avec une configuration normale, et le mode --fast montre comment le serveur se comporte si le développeur utilise toutes les extensions du serveur pour accélérer son application.

Lorsque nous exécutons PostgreSQL avec l'option --fast , nous exécutons la commande VACUUM après chaque modification importante de la table, et après l'effacement de table, pour conserver la base dans un état parfait pour les prochaines sélections. Le temps nécessaire à l'exécution de la commande VACUUM est mesuré séparément.

Lorsque PostgreSQL 7.1.1 fonctionne, nous n'avons pu le faire fonctionner avec l'option --fast car durant les test d' INSERT , le postmaster (démon PostgreSQL) s'est arrété, et la base était tellement corrompue qu'il a été impossible de redémarrer le postmaster. Après un deuxième problème identique, nous avons décidé de reporter les tests --fast jusqu'à la prochaine version de PostgreSQL. Les détais sur la machine avec laquelle nous avons exécuté les tests sont disponible sur la page de tests.

Avant de passer aux autres tests de performances que nous connaissons, nous souhaitons vous donner des détails sur le contexte des tests.

Il est très facile d'écrire un test qui montre que **toutes** les bases du monde sont les meilleures, en se limitant simplement aux points forts de la base, et en ne testant rien qui soit une faiblesse. De plus, si la conclusion se ramène à un seul chiffre, c'est encore plus facile.

Cela revient à mesurer la vitesse de MySQL par rapport à PostgreSQL en regardant simplement les temps moyens des tests de MySQL sur notre page. En se basant sur ces informations, notre serveur MySQL serait 40 fois plus rapide que PostgreSQL, ce qui est, bien sûr, faux. Nous pourrions même rendre les choses encore pires en prenant les tests où PostgreSQL a les pires performances, et dire que MySQL est plus de 2000 fois plus rapide que PostgreSQL.

Le fait est que MySQL fait un certain nombre d'optimisation que PostgreSQL ne fait pas. C'est aussi vrai, bien sûr, dans le sens inverse. Un optimiseur SQL est un programme complexe, et une compagnie peut passer des années à le rendre de plus en plus rapide.

Lorsque vous regardez les résultats des tests, vous devriez vous limiter aux opérations que vous utilisez dans votre application, et utiliser simplement ces résultats pour décider quelle base de données est la plus adaptée à vos besoins. Les résultats de tests vous montrent quelles opérations sont le point faible de chaque base, et vous donnent des indications sur ce qu'il faudra éviter de faire.

Nous connaissons deux tests de performances qui indiquent que PostgreSQL est plus rapide que MySQL. Ces deux tests sont faits dans un environnement multi−utilisateur, un test que nous, chez MySQL AB, nous n'avons pas eu le temps de mettre en place car c'est une tâche énorme de le faire pour que ce soit équitable pour chaque base.

Le premier est le test payant, effectué pour Great Bridge, une entreprise qui a tenté durant 16 mois de bâtir un modèle d'affaires avec PostgreSQL mais qui a désormais cessé les opérations. C'est probablement le pire test que nous ayons jamais vu. Non seulement ce test était bâti pour ne tester que les points forts de PostgreSQL, mais il était totalement déloyal pour toutes les autres bases de données utilisées dans le test.

**Note** : nous savons aussi que même certains des développeurs principaux de PostgreSQL n'ont pas aimé la manière avec laquelle Great Bridge a dirigé le test, et nous ne blâmons pas l'équipe de PostgreSQL pour ces tests.

Ce test a été condamné dans de nombreux articles et groupes d'utilisateurs, et nous ne citerons que quelques−uns des points qui étaient incorrects.

- Les tests ont été exécutés avec un outil commercial très cher, qui rendent impossibles aux entreprises `Open Source` comme nous de vérifier les résultats ou même de voir comment les tests ont été mis en place. L'outil n'était même pas un véritable outil de test de performances, mais une application de mise en place et de tests. Dire que c'est un outil ``standard'' de tests est bien loin de la vérité.
- Great Bridge a admis qu'ils ont optimisé le serveur PostgreSQL (avec `VACUUM` avant le test) et optimisé le démarrage pour les tests, ce qu'ils n'ont pas fait pour les autres bases. Ils disent ``Ce processus optimise les index et libère un peu d'espace sur le disque. Les index optimisés améliorent les performances marginalement''. Nos tests indiquent clairement la différence lorsqu'on exécute une série de sélections sur une base qui a été entretenue avec `VACUUM` ou pas : le facteur peut atteindre 10.
- Les résultats des tests étaient eux−même très étranges. La documentation de tests AS3AP mentionne qu'ils sont des ``sélections, jointures simples, projections, agrégations, modification d'une ligne et modifications massives''.

  PostgreSQL est bon pour les `SELECT` s et les `JOIN` s (spécialement après l'exécution de la commande `VACUUM` ), mais n'est pas très performant pour les `INSERT` s ou les `UPDATE` . Les tests semblent indiquer que seuls des `SELECT` ont été exécutés. Cela peut facilement expliquer le nombre de bons résultats pour PostgreSQL dans ce test. Les mauvais résultats de MySQL seront évident un peu plus bas, dans ce document.
- Ils ont exécuté les soi−disant tests depuis une machine sous Windows vers un serveur Linux, via ODBC, une configuration que personne n'utiliserait pour un environnement multi−utilisateur. Cela testait surtout le pilote ODBC et le protocole Windows utilisé entre les clients et le serveur, plutôt que le serveur lui−même.
- Lorsque les tests ont été exécutés avec Oracle et MS−SQL (Great Bridge a indirectement indiqué que ces bases étaient utilisées dans les tests), ils n'ont pas utilisé le protocole natif

mais ODBC. Toute personne qui a utilisé Oracle sait que toutes les vraies applications utilisent l'interface native au lieu d'ODBC. Faire un test via ODBC et dire que les résultats sont comparables avec les situations d'utilisation de la base dans le monde réel ne peut être considéré comme équitable. Ils auraient dû faire deux tests, avec et sans ODBC, pour fournir des faits (après que leurs experts aient optimisé les serveurs, bien sûr).

- Ils font référence au tests TPC−C, mais ils ne mentionnent nulle part que le test n'était pas un vrai test TPC−C et qu'ils n'étaient pas autorisés à dire que c'était un tel test. Un test TPC−C ne peut être conduit que par les règles approuvées par le TPC Council ( http://www.tpc.org/ ). Great Bridge ne l'a pas fait. Ils ont donc violé la marque commerciale de TPC et discrédité leurs propres tests. Les règles dictées par le TPC Council sont très strictes pour s'assurer que personne ne produit de faux résultats ou tire des conclusions invérifiables. Apparemment, Great Bridge n'était pas interessé par cela.
- Après le premier test, nous avons contacté Great Bridge et nous leur avons mentionné les erreurs évidentes qu'ils avaient faits avec le serveur MySQL :
    - Travailler avec une version beta de notre pilote ODBC.
    - Travailler avec un système Linux qui n'était pas optimisé pour les threads.
    - Utiliser une vieille version de MySQL alors qu'il en existait une nouvelle.
    - Ne pas démarrer MySQL avec les bonnes options pour gérer les environnements fortement multi−utilisateurs (la configuration par défaut est optimisée pour consommer le moins de ressources).

  Great Bridge a exécuté un nouveau test, avec notre nouveau pilote ODBC, et avec de meilleures options de démarrage de MySQL mais ils ont refusé d'utiliser notre librairie glibc ou notre binaire standard (utilisé par 80% de nos utilisateurs), qui étaient statiquement compilé avec la librairie glibc.Selon nos sources, Great Bridge n'a rien fait pour s'assurer que les autres bases de données étaient correctement configurées pour leur environnement de tests. Nous sommes certains qu'ils n'ont pas contacté Oracle ou Microsoft pour leur demander des conseils de configurations.
- Le test était payé par Great Bridge, et ils ont décidé de publier une partie des résultats, bien choisis (au lieu de tous les publier).

Tim Perdue, un inconditionnel de PostgreSQL et un utilisateur réticent de MySQL, publia une comparaison sur PHPbuilder ( http://www.phpbuilder.com/columns/tim20001112.php3 ).Lorsque nous avons eu connaissance de la comparaison, nous avons téléphoné à Tim Perdue à ce propos, car il y avait de nombreux résultats étranges dans ses publications. Par exemple, il statuait que MySQL avait un problème avec 5 utilisateurs durant ses tests, alors que nous avons des utilisateurs avec des machines similaires, qui utilisent MySQL avec plus de 2000 utilisateurs simultanés, et 400 requêtes par secondes. (Dans ce cas, la limite est la bande passante du réseau, pas le serveur SQL).

Il semblait qu'il utilisait un noyau Linux qui avait des problèmes avec les threads, comme les noyaux avant le 2.4 qui avait des problèmes avec les threads sur les machines à plusieurs processeurs. Nous avons documenté dans le manuel comment corriger ce problème, et Tim devrait être bientôt au courant de ce problème.

Les autres problèmes possibles pourraient provenir d'une vieille librairie glibc ou du fait que MySQL n'utilise pas une version exécutable de notre site web, qui est bâtie avec une version corrigée de glibc library. Dans ces cas, les symptômes sont ceux que Tim a mesuré.

Nous avons demandé à Tim d'avoir accès a ses données pour que nous puissions reproduire le test, et s'il pouvait vérifier la version de MySQL sur sa machine pour voir ce qui n'allait pas. Il a promis qu'il nous contacterait mais n'a rien fait à ce jour.

A cause de cela, nous ne pouvons pas non plus avoir confiance en ce test.

Avec le temps qui passe, les technologies évoluent, et les tests précédents ne sont désormais plus valables. Le serveur MySQL dispose d'un jeu de gestionnaires de tables, avec des choix très différents sur la vitesse et les fonctionnalités. Types de tables MySQL . Il serait intéressant de voir comment les tests ci−dessus s'effectuent avec les différents gestionnaires de tables de MySQL. PostgreSQL a aussi, bien sûr, de nouvelles fonctionnalités depuis que ces tests ont été menés. Comme ces tests ne sont pas publiquement disponibles, il n'y a pas moyen pour nous de savoir comment les bases se comportent de nos jours avec ces tests.

Conclusion :

Les seuls tests qui existent aujourd'hui, disponibles au téléchargement pour tous, et que vous pouvez exécuter avec MySQL, PostgreSQL sont les tests de MySQL. Chez MySQL AB, nous croyons que les bases de données Open Source devraient être testées avec des outils Open Source ! C'est le seul moyen de nous assurer que personne ne peut faire de tests que personne ne peut reproduire, et les utiliser pour proclamer que sa base est meilleure que les autres. Sans connaître le fond des choses, il est impossible de répondre à une telle déclaration.

Ce que nous trouvons étrange est que chaque test que nous avons vu à propos de PostgreSQL est impossible à reproduire, et indique que PostgreSQL est le meilleur dans la plupart des tests, alors que les nôtres , que tout le monde peut reproduire, montrent le contraire. Avec cela, nous ne voulons pas dire que PostgreSQL n'est pas bon dans de nombreuses tâches (il l'est !!), ou qu'il n'est pas plus rapide que MySQL dans certaines conditions. Nous voulons juste voir un test équitable où PostgreSQL se comporte bien, pour que nous puissions avoir une compétition amicale !

Pour plus d'informations sur nos tests, voyez La suite de tests comparatifs de MySQL .

Nous travaillons actuellement sur une suite de tests de perfomance qui inclue des tests multi utilisateurs, et une meilleure documentation sur les tests individuels pour indiquer ce qu'ils font et comment ajouter d'autres tests à la suite.

# 2 Installation de MySQL

Ce chapitre décrit comment obtenir et installer MySQL :

- Pour une liste de sites à partir desquels vous pouvez obtenir MySQL, voyez How to Get MySQL .
- Pour savoir quelles plate−formes sont supportées, voyez Operating Systems Supported by MySQL . Veuillez noter que tous les systèmes supportés ne sont pas tous aussi bons pour faire fonctionner MySQL. Certains sont plus robustes et efficaces que d'autres, voyez Operating Systems Supported by MySQL pour davantage d'informations.
- Plusieurs versions de MySQL sont disponibles à la fois sous forme de binaires ou de code source. Nous fournissons aussi un accès public à notre arbre source aux personnes souhaitant suivre les derniers développements et nous aider à tester le nouveau code. Pour déterminer quelle version et quel type de distribution vous devriez utiliser, voyez Which MySQL Version to Use . En cas de doute, utilisez la distribution binaire.
- Les instructions d'installation à partir des binaires ou des sources sont données dans Installing a MySQL Binary Distribution , et Installer MySQL à partir des sources . Chacun de ces jeux d'instructions inclut une partie sur des problèmes spécifiques que vous pourriez rencontrer.
- Pour les procédures consécutives à l'installation, voyez Configuration et tests consécutifs à l'installation . Ces procédures s'appliquent à la fois si vous installez MySQL à partir des sources ou à partir des binaires.

## 2.1 Installation standard rapide de MySQL

### 2.1.1 Installer MySQL sous Linux

Il est recommandé d'installer MySQL sous Linux en utilisant un fichier RPM. Les RPM de MySQL sont actuellement compilé sur une RedHAt en version 6.2, mais devraient fonctionner sur toute autre version de Linux qui supporte `rpm` et utilise `glibc` .

Si vous avez des problèmes avec un fichier RPM, si vous obtenez par exemple l'erreur `` `Sorry, the host 'xxxx' could not be looked up` '', référez vous à Notes relatives à Linux pour les distribution binaires .

Les fichiers RPM dont vous pourriez avoir besoin sont :

- `MySQL−VERSION.i386.rpm`

  Le serveur MySQL. Vous en aurez besoin à moins que vous ne vouliez que vous connecter à un serveur MySQL tournant sur une autre machine.
- `MySQL−client−VERSION.i386.rpm`

  Les programmes clients MySQL standards. Vous avez certainement besoin d'installer ce paquet.
- `MySQL−bench−VERSION.i386.rpm`

  Tests et bancs d'essai. Nécessite Perl et les modules RPM msql−mysql−modules.
- `MySQL−devel−VERSION.i386.rpm`

  Librairies et fichiers d'inclusions dont vous aurez besoin pour compiler d'autres clients MySQL, tels que les modules Perl.
- `MySQL−VERSION.src.rpm` Celui−ci contient le code source de tous les paquets précédents. Il peut donc être utilisé pour construire des fichiers RPM pour d'autres architectures (par exemple,

l'Alpha ou le SPARC).

Pour voir tous les fichiers présents dans un paquet RPM, lancez :

```
shell> rpm -qpl MySQL-VERSION.i386.rpm
```

Pour effectuer une installation standard minimale, lancez :

```
shell> rpm -i MySQL-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

Pour installer uniquement le paquet du client MySQL, lancez :

```
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

Le fichier RPM place les données dans `/var/lib/mysql` . Le RPM crée aussi les entrées appropriées dans `/etc/rc.d/` pour lancer le serveur automatiquement au démarrage. (Cela signifie que, si vous avez déjà effectué une installation auparavant, vous pouvez avoir besoin de faire une sauvegarde de vos fichiers de démarrage précédents si vous les changez, de façon à ne pas les perdre.)Après l'installation des fichiers RPM, le démon `mysqld` devrait fonctionner et vous devriez être capables de commencer à utiliser MySQL. Configuration et tests consécutifs à l'installation .

Si quelque chose se passe mal, vous pouvez trouver davantage d'informations dans le chapitre d'installation des binaires. Installing a MySQL Binary Distribution .

## 2.1.2 Installer MySQL sous Windows

Le serveur MySQL pour Windows est disponible sous deux formes :

- La forme binaire contient un programme de configuration qui installe tout ce dont vous avez besoin de telle façon que vous puissiez démarrer le serveur immédiatement.
- La distribution sous forme de code source contient tout le code et les fichiers de support nécessaires à la compilation des exécutables en utilisant le compilateur VC++ 6.0. La distribution sous forme de sources pour Windows .

En règle générale, vous devriez utiliser la distribution binaire.Vous aurez besoin des choses suivantes :

- Un système d'exploitation Windows 32−bit tel que Windows 9x, Me, NT, 2000, ou XP. La famille NT (NT, Windows 2000 et XP) permet de faire fonctionner le serveur MySQL en tant que service. Démarrer MySQL avec Windows NT / 2000 / XP .Si vous voulez utiliser des tables de plus de 4 Go, vous devez installer MySQL sur un système de fichiers NTFS ou supérieur. N'oubliez pas d'utiliser `MAX_ROWS` et `AVG_ROW_LENGTH` quand vous créez les tables. Syntaxe de `CREATE TABLE` .
- Support du protocole TCP/IP.
- Une copie du binaire ou de la distribution MySQL pour WIndows, qui peut être téléchargée sur http://www.mysql.com/downloads/ .Note : les fichiers de la distribution sont dans un fichier au format zip et nous vous recommandons l'utilisation d'un client FTP adéquat qui supporte la reprise des transferts afin d'éviter des problèmes de corruption de fichiers pendant le téléchargement.
- Un programme d'extraction `ZIP` pour décompresser le fichier de la distribution.
- Suffisamment d'espace disque pour décompresser, installer et créer les bases de données en relation avec vos besoins.
- Si vous envisagez de vous connecter à MySQL via `ODBC` , vous aurez aussi besoin du pilote `MyODBC` . Support ODBC pour MySQL .

### 2.1.2.1 Installation des binaires

- Si vous travaillez avec les serveurs NT/2000/XP, loguez−vous en utilisateur avec les permissions d'administrateur.
- Si vous faites une mise à jour d'une version plus récente de MySQL, il est nécessaire d'arrêter le serveur. Si vous utilisez le serveur en tant que service, utilisez :
  ```
  C:\> NET STOP MySQL
  ```
  Dans les autres cas, utilisez :
  ```
  C:\mysql\bin> mysqladmin −u root shutdown
  ```
- Avec les serveurs NT/2000/XP, si vous voulez changer l'exécutable du serveur (e.g., −max or −nt), il est aussi nécessaire de stopper le service :
  ```
  C:\mysql\bin> mysqld−max−nt −−remove
  ```
- Dézipper les fichiers dans un emplacement temporaire.
- Exécutez le fichier `setup.exe` pour commencer l'installation. Si vous voulez l'installer à un autre emplacement que celui pas défaut `c:\mysql` , utilisez le bouton `Parcourir` pour spécifier un emplacement.
- Terminer le processus d'installation.

### 2.1.2.2 Préparation de l'environnement MySQL de Windows

A partir de la version 3.23.38 de MySQL, la distribution Windows inclut le binaire du serveur normal et le binaire du serveur MySQL−Max. Voici une liste des différents serveurs MySQL que vous pouvez utiliser :

| Binaire | Description |
|---|---|
| mysqld | Compilé avec débogage total et vérification automatique de l'allocation de mémoire, liens symboliques, tables InnoDB et BDB. |
| mysqld−opt | Binaire optimisé sans le support des tables transactionnelles. |
| mysqld−nt | Binaire optimisé pour NT/2000/XP sans support des tunnels nommés. Vous pouvez faire fonctionner cette version sur Windows 9x/Me, mais dans ce cas, les tunnels nommés ne sont pas crées et vous devez avoir TCP/IP installé. |
| mysqld−max | Binaire optimisé avec support des liens symboliques et des tables InnoDB et BDB. |
| mysqld−max−nt | Comme mysqld−max , mais compilé avec le support des tunnels nommés. |

A partir de la version 3.23.50, les tunnels nommés ne sont activés que si on démarre mysqld avec `−−enable−named−pipe` .

Tous les binaires précédents sont optimisés pour le processeur Pentium Pro mais devraient fonctionner avec n'importe quel processeur Intel >= i386.

Vous devrez utiliser un fichiers d'options pour spécifier votre configuration MySQL dans les circonstances suivantes :

- Le répertoire d'installation ou de données est différent de ceux par défaut ( `c:\mysql` et `c:\mysql\data` ).
- Vous voulez utilisez l'un de ces serveurs :
  - mysqld.exe
  - mysqld−max.exe
  - mysqld−max−nt.exe
- Vous avez besoin de paramétrer la configuration du serveur.

Normalement, vous pouvez utiliser l'outil `WinMySQLAdmin` pour éditer le fichier d'options `my.ini` . Dans ce cas, vous n'avez pas à vous soucier de ce qui suit.

Il y a deux fichiers d'options avec la même fonction : `my.cnf` et `my.ini` . Toutefois, pour éviter la confusion, il est préférable de n'en utiliser qu'un seul. Les deux fichiers sont en texte plein. Le fichier `my.cnf` , si utilisé, doit être crée dans le répertoire racine du lecteur C. Le fichier `my.ini` , si utilisé, doit

être crée dans le répertoire système de Windows. (Ce répertoire est souvent nommé `C:\WINDOWS` ou `C:\WINNT` . Vous pouvez déterminer sa valeur exacte en regardant la valeur de la variable d'environnement `windir` .) MySQL regarde en premier le fichier `my.ini` , puis le fichier `my.cnf` .

Si votre PC utilise un système de boot ou le lecteur C n'est pas le lecteur de boot, votre seule option est d'utiliser le fichier `my.ini` . Notez aussi que si vous utilisez l'outil `WinMySQLAdmin` , celui−ci n'utilisera que le fichier `my.ini` . Le répertoire `\mysql\bin` contient un fichier d'aide avec les instruction pour utiliser cet outil.

En utilisant `notepad.exe` , créez le fichier d'options et éditez la section `[mysqld]` pour spécifier les valeurs des paramètres `basedir` et `datadir` :

```
 [mysqld]
# set basedir to installation path, e.g., c:/mysql
basedir=le_dossier_d'installation
# set datadir to location of data directory,
# e.g., c:/mysql/data or d:/mydata/data
datadir=le_dossier_des_données
```

Notez que les chemins de Windows doivent être spécifiés dans les fichiers d'options en utilisant des slashes normaux ( **/** ) au lieu des backslashes ( **** ). Si vous utilisez des backslashes, vous devez les doubler.

Si vous voulez utiliser un répertoires de données autre que `c:\mysql\data` , vous devez copier le contenu entier du dossier `c:\mysql\data` au nouvel endroit.

Si vous voulez utiliser les tables transactionnelles `InnoDB` , vous aurez besoin de créer manuellement deux nouveaux répertoires pour contenir les les données et les logs InnoDB, `c:\ibdata` et `c:\iblogs` . Vous aurez aussi besoin d'ajouter quelques lignes supplémentaires dans le fichier d'options. Options de démarrage `InnoDB` .

Si vous ne voulez pas utiliser les tables `InnoDB` , ajoutez l'option `skip−innodb` au fichier d'options.

Maintenant, vous pouvez commencer à tester le démarrage du serveur.

### 2.1.2.3 Démarrer le serveur pour la première fois

Tester à partir d'une console DOS est la meilleure chose à faire car le serveur affiche des messages qui y apparaissent. Si quelque chose n'est pas bon dans votre configuration, ces messages vous aiderons à identifier et corriger le problème.

Assurez−vous d'être dans le répertoire où se situe le serveur, puis entrez cette commande :

```
C:\mysql\bin> mysqld−max −−standalone
```

Vous devriez voir ce qui suit pendant le démarrage du serveur :

```
 InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file c:\ibdata\ibdata1 size to 209715200
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
011024 10:58:25  InnoDB: Started
```

Pour plus d'informations à propos du démarrage de MySQL sous Windows, voyez Notes relatives à Windows .

## 2.1.3 Installer MySQL sur Mac OS X

Depuis MySQL 4.0.11, vous pouvez installer MySQL sur Mac OS X 10.2 ("Jaguar") avec le package binaire Mac OS X `PKG` au lieu d'utiliser la distribution binaire compressée. Notez que les anciennes versions de Mac OS X (i.e. 10.1.x) ne sont pas supportées par ce package.

Le package est situé dans une image disque ( `.dmg` ), que vous devez monter en double−cliquant son icône sur le Finder. Le disque devrait alors se monter, et afficher son contenu.

**NOTE** : Avant de lancer l'installation, assurez vous qu'il n'y a pas de serveur MySQL en fonctionnement!

Arrêtez tous les serveurs MySQL avant de continuer, soit en utilisant l'application manager (pour les serveurs Mac OS X) ou via `mysqladmin shutdown` en ligne de commande.

Pour installer le package MySQL, double−cliquez sur l'icône. Cela va lancer l'installeur de package MacOSX, qui vous guidera durant l'installation.

Le package Mac OS X de MySQL va s'installer lui−même dans le dossier `/usr/local/mysql−<version>` et va aussi installer un lien symbolique `/usr/local/mysql` , pointant sur la nouvelle installation. Si un dossier appelé `/usr/local/mysql` existe déjà, il sera renommé en `/usr/local/mysql.bak` . De plus, il va installer les tables de droits MySQL en exécutant le script `mysql_install_db` après l'installation.

Le schéma d'installation est semblable à celui de la distribution binaire, tous les programmes MySQL sont situés dans le dossier `/usr/local/mysql/bin` . Les sockets MySQL sont installées dans le fichier `/etc/mysql.sock` par défaut. Installation Layouts .

L'installation requiert un compte nommé `mysql` (qui existe par défaut en Mac OS X 10.2 et plus récent).

Si vous utilisez Mac OS X Server, vous devez avoir déjà une version de MySQL installée :

- Mac OS X Server 10.2−10.2.2 est livré avec MySQL 3.23.51
- Mac OS X Server 10.2.3 et 10.2.4 est livré avec MySQL 3.23.53

Cette section du manuel couvre l'installation du package binaire MySQL pour Mac OS X uniquement. Assurez vous de bien lire l'aide d'Apple concernant l'installation de MySQL (Lancer le visualiseur d'aid, sélectionnez la rubrique "Serveur Mac OS X", et faites une recherche sur "MySQL", puis lisez l'entrée appelée "Installing MySQL").Notez bien que la version pré−installée de MySQL sur Mac OS X Server peut être lancée avec la commande `safe_mysqld` au lieu de `mysqld_safe` !

Si vous avez utilisé auparavant les packages MySQL de Marc Liyanage pour Mac OS X, depuis le site de http://www.entropy.ch , vous pouvez simplement suivre le process de mise à jour, en utilisant les conseils de ses pages.

Si vous faites une mise à jour depuis les versions de Marc, ou depuis une ancienne version de MySQL pour Mac OS X Server, avec le nouveau package officiel, vous devrez convertir les tables de droits. Upgrading From Version 3.23 to 4.0 .

Après cette installation, vous opuvez lancer le serveur MySQL avec ces commandes, exécutées

dans terminal. Notez bien que vous devez avoir des droits d'administrateur pour cela!

```
 shell> cd /usr/local/mysql
shell> sudo ./bin/mysqld_safe
(Enter your password)
(Press CTRL+Z)
shell> bg
(Press CTRL+D to exit the shell)
```

Vous devez alors être capable de vous connecter au serveur MySQL, notamment en exécutant la commande `/usr/local/mysql/bin/mysql` .

Pour activer automatiquement le démarrage de MySQL au lancement du serveur, vous pouvez télécharger l'utilitaire de démarrage de Marc Liyanage sur sont site :

http://www2.entropy.ch/download/mysql−startupitem.pkg.tar.gz

Nous avons prévu d'ajouter une application de démarrage à la distribution officielle prochainement.

Notez que l'installation d'un nouveau package MySQL ne supprimer pas le dossier d'une vieille installation : l'installeur de Mac OS X n'offre pas encore les fonctionnaltés nécessaires pour mettre à jour proprement une vieille version. Après avoir copié les fichiers de bases depuis votre ancien dossier vers le nouveau, et que vous avez réussi à redémarrer MySAL avec la nouvelle version, vous devriez pensez à supprimer les vieilles versions des fichiers, situées dans `/Library/Receipts/mysql−<version>.pkg` .

## 2.1.4 Installer MySQL sur NetWare

Depuis la version 4.0.11, le serveur MySQL est disponible sur Novell NetWare, en format binaire. Pour pouvoir héberger un serveur MySQL, un serveur NetWare doit avoir les pré−requis suivants :

- NetWare version 6.0 ou plus récent (le système doit absolument suivre ce pré−requis).
- Le fichier `LIBC.NLM` doit être celui de la mise à jour de Novell, daté du 18 Février 2003, pour NetWare 6. Vous pouvez obtenir ce fichier sur http://developer.novell.com/ndk/libc.htm .
- Les données MySQL ainsi que les exécutables, doivent être installé sur un volume NSS; les volumes traditionnels ne sont pas supportés.

Les exécutables binaires pour NetWare sont disponibles sur http://www.mysql.com/downloads/ .

### 2.1.4.1 Installer les exécutables MySQL pour NetWare

- Si vous faites une mise à jour depuis une installation antérieure, arrêtez le serveur MySQL. Pour cela, faites :
  ```
  SERVER:  mysql −u root shutdown
  ```
- Connectez vous sur le serveur cible depuis une machine cliente, pour accéder au dossier où installer MySQL.
- Décompressez le fichier zip binaire sur le serveur. Assurez vous d'utiliser les chemins du fichiers zip. Il n'y a pas de problème à installer le fichier dans `SYS:\` .

  Si vous faites une mise à jour depuis une ancienne installation, vous devez copier les données (c'est à dire `SYS:MYSQL\DATA` ), de même que le fichier `my.cnf` , si vous l'avez personnalisé. Vous pouvez alors effacer l'ancienne copie de MySQL.
- Vous pouvez renommer le dossier avec un autre nom plus cohérent et facile à retenir. Nous vous recomandons d'utiliser `SYS:MYSQL` ; les exemples suivants utiliserons ce nom pour se référner au dossier d'installation en général.

- Dans la console du serveur, ajoutez un chemin de recherche pour le dossier contenant les NLM de MySQL. Par exemple :
```
SERVER:  SEARCH ADD SYS:MYSQL\BIN
```
- Installez la base initiale, si nécessaire, en utilisant le script `mysql_install_db` en ligne de commande.
- Démarrez le serveur MySQL en utilisant le script `mysqld_safe` dans la console.
- Pour finir l'installation, vous devriez aussi ajouter les commandes ci−dessus au fichier `autoexec.ncf` . Par exemple, si votre installation MySQL est dans le dossier `SYS:MYSQL` , et que vous voulez que MySQL démarre automatiquement, vous pouvez utiliser ces lignes :
```
#Démarre le serveur MySQL 4.0.x
SEARCH ADD SYS:MYSQL\BIN
MYSQLD_SAFE
```

Si il y avait une installation MySQL pré−éxistante, assurez vous de bien vérifier les commandes de démarrages dans `autoexec.ncf` , et de les adapter si nécessaire.

# 2.2 Notes générales à propos de l'installation

## 2.2.1 Comment obtenir MySQL ?

Visitez le site de MySQL ( http://www.mysql.com/ ) pour des informations à propos de la version courante et les instructions de téléchargement.

Notre miroir principal est situé sur http://mirrors.sunsite.dk/mysql/ .

Pour une liste complète et à jour des miroirs web/téléchargement de MySQL, voyez http://www.mysql.com/downloads/mirrors.html . Vous trouverez là des informations à propos des futurs miroirs et de quoi nous informer de la non−validité de l'un d'entre eux.

## 2.2.2 Vérifier l'intégrité des paquets avec MD5 ou GnuPG

Une fois que vous avez téléchargé le paquet MySQL qui vous convient, et avant de l'installer, vous devriez vous assrurer qu'il est intact, et n'a pas été altéré.

MySQL AB propose deux moyens de vérifier l'intégrité : les sommes de contrôles `MD5` et les signatures de `GnuPG` , de `GNU Privacy Guard` .

**Verifying the** `MD5 Checksum` Une fois que vous avez téléchargé le paquet, vous devez vérifier si la somme de contrôle MD5 correspond à celle qui est disponibles sur le site de MySQL. Chaque package a une somme de contrôle individuelle, que vous pouez obtenir avec la commande suivante :

```
shell> md5sum <package>
```

Notez que tous les systèmes d'exploitation ne supportent pas la commande `md5sum` : sur certains, elle s'appelle simplement `md5` , sur d'autre, elle n'est pas du tout disponible. Sur Linux, ella fait partie des `utilitaires textes GNU (GNU Text Utilities)` , qui sont disponbiles pour toute une gamme de plate−formes. Vous pouvez télécharger le code source sur le site http://www.gnu.org/software/textutils/ . Si vous avez installé `OpenSSL` , vous pouvez utiliser la commande `openssl md5 <package>` à la place. Une implémentation DOS/Windows de la commande `md5` est disponible sur le site http://www.fourmilab.ch/md5/ .Exemple :

```
 shell> md5sum mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz
155836a7ed8c93aee6728a827a6aa153
                mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz
```

Ainsi, vous devez vérifier si la somme de contrôle résultante correspond à cette qui est imprimée sur la page de téléchargement, en dessous du paquet téléchargé.La plupart des sites miroirs proposent aussi un fichier appelé `MD5SUMS` , qui inclut les sommes de contrôle MD5 pour tous les fichiers disponibles dans le dossier `Downloads` . Notez bien qu'il est très facile de modifier ce fichier, et que ce n'est pas une méthode sécuritaire. Dans le doute, consultez différents mirroirs, et comparez les résultats.

**Vérification de la signature avec** `GnuPG` Une méthode plus sûre pour vérifier l'intégrité d'un paquet est d'utiliser la signature. MySQL AB utilise `GNU Privacy Guard` ( `GnuPG` ), une alternative `Open Source` du très connu `Pretty Good Privacy` ( `PGP` ) par Phil Zimmermann. Voir http://www.gnupg.org/ et http://www.openpgp.org/ pour plus d'informations sur `OpenPGP` / `GnuPG` et comment obtenir et installer `GnuPG` sur votre système. La plupart des distributions Linux dispose d'une version de `GnuPG` installée par défaut.

Depuis MySQL 4.0.10 (Février 2003), MySQL AB signe les paquets en téléchargement avec `GnuPG` . Les signateurs chifféres représentent une méthode bien plus sûre pour vérifier l'intégrité et l'authenticitié d'un fichier.

Pour vérifier la signature d'un paquet spécifique, vous devez obtenir en premier lieu une copie de la clé publique GPG build@mysql.com . Vous pouvez soit la copier-coller directement depuis ce manuel, ou la réclamer sur le serveur http://www.keyserver.net/ .

```
 Key ID:
pub  1024D/5072E1F5 2003-02-03
     MySQL Package signing key (www.mysql.com) <build&#64;mysql.com>
Fingerprint: A4A9 4068 76FC BD3C 4567  70C8 8C71 8D3B 5072 E1F5

 Public Key (ASCII-armored):

 -----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.0.6 (GNU/Linux)
Comment: For info see http://www.gnupg.org

 mQGiBD4+owwRBAC14GIfUfCyEDSIePvEW3SAFUdJBtoQHH/nJKZyQT7h9bPlUWC3
RODjQReyCITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPkBdCk96+OmSLN9brZ
fw2vOUgCmYv2hW0hyDHuvYlQA/BThQoADgj8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3
BqOxRznNCRCRxAuAuVztHRcEAJooQK1+iSiunZMYD1WufeXfshc57S/+yeJkegNW
hxwR9pRWVArNYJdDRT+rf2RUe3vpquKNQU/hnEIUHJRQqYHo8gTxvxXNQc7fJYLV
K2HtkrPbP72vwsEKMYhhr0eKCbtLGfls9krjJ6sBgACyP/Vb7hiPwxh6rDZ7ITnE
kYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgtobZX9qnrAXw+uNDI
QJEXM6FSbi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPAnWqcyefep
rv0sxxqUEMcM3o7wwgfN83POkDasDbs3pjwPhxvhz6//62zQJ7Q7TXlTUUwgUGFj
a2FnZSBzaWduaW5nIGtleSAod3d3Lm15c3FsLmNvbSkgPGJ1aWxkQG15c3FsLmNv
bT6IXQQTEQIAHQUCPj6jDAUJCWYBgAULBwoDBAMVAwIDFgIBAheAAAoJEIxxjTtQ
cuH1cY4AnilUwTXn8MatQOiG0a/bPxrvK/gCAJ4oinSNZRYTnblChwFaazt7PF3q
zIhMBBMRAgAMBQI+PqPRBYMJZgC7AAoJEElQ4SqycpHyJOEAn1mxHijft00bKXvu
cSo/pECUmppiAJ41M9MRVj5VcdH/KN/KjRtW6tHFPYhMBBMRAgAMBQI+QoIDBYMJ
YiKJAAoJELb1zU3GuiQ/lpEAoIhpp6BozKI8p6eaabzF5MlJH58pAKCu/ROofK8J
Eg2aLos+5zEYrB/LsrkCDQQ+PqMdEAgA7+GJfxbMdY4wslPnjH9rF4N2qfWsEN/l
xaZoJYc3a6M02WCnHl6ahT2/tBK2w1QI4YFteR47gCvtgb6O1JHffOo2HfLmRDRi
Rjd1DTCHqeyX7CHhcghj/dNRlW2Z0l5QFEcmV9U0Vhp3aFfWC4Ujfs3LU+hkAWzE
7zaD5cH9J7yv/6xuZVw411x0h4UqsTcWMu0iM1BzELqX1DY7LwoPEb/O9Rkbf4fm
Le11EzIaCa4PqARXQZc4dhSinMt6K3X4BrRsKTfozBu74F47D8Ilbf5vSYHbuE5p
/1oIDznkg/p8kW+3FxuWrycciqFTcNz215yyX39LXFnlLzKUb/F5GwADBQf+Lwqq
a8CGrRfsOAJxim63CHfty5mUc5rUSnTslGYEIOCR1BeQauyPZbPDsDD9MZ1ZaSaf
anFvwFG6Llx9xkU7tzq+vKLoWkm4u5xf3vn55VjnSd1aQ9eQnUcXiL4cnBGoTbOW
I39EcyzgslzBdC++MPjcQTcA7p6JUVsP6oAB3FQWg54tuUo0Ec8bsM8b3Ev42Lmu
QT5NdKHGwHsXTPtl0klk4bQk4OajHsiy1BMahpT27jWjJlMiJc+IWJ0mghkKHt92
6s/ymfdf5HkdQ1cyvsz5tryVI3Fx78XeSYfQvuuwqp2H139pXGEkg0n6KdUOetdZ
Whe70YGNPw1yjWJT1IhMBBgRAgAMBQI+PqMdBQkJZgGAAAoJEIxxjTtQcuH17p4A
n3r1QpVC9yhnW2cSAjq+kr72GX0eAJ4295kl6NxYEuFApmr1+0uUq/SlsQ==
=YJkx
-----END PGP PUBLIC KEY BLOCK-----
```

Vous pouvez importer cette clé dans votre trousseau de clé publiques `GPG` avec la commande `gpg --import` . Voyez la documnetation `GPG` pour plus de détails sur comment travailler avec les clés publiques.

Une fois que vous avez téléchargé et importé la clé publique, vous pouvez télécharger le paquet MySQL et la signature qui lui est associée, sur la même page. Le fichier de signature a pour extension `.asc` . Par exemple, la signature pour le fichier `mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz` serait `mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz.asc` . Assurez vous que les deux fichiers sont stockés dans le même dossier, puis exécutez la commande suivante pour vérifier la signature du fichier :

```
 shell> gpg --verify <package>.asc

 Example:

 shell> gpg --verify mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz.asc
gpg: Warning: using insecure memory!
gpg: Signature made Mon 03 Feb 2003 08:50:39 PM MET using DSA key ID 5072E1F5
gpg: Good signature from
     "MySQL Package signing key (www.mysql.com) <build&#64;mysql.com>"
```

La mention "Good signature" (bonne signature) indique que le paquet est correct.

Pour les paquets `RPM` , il n'y a pas de signature séparée : les paquets `RPM` disposent d'une signature `GPG` intégrée, et d'une somme de contrôle `MD5` . Vous pouvez les vérifier avec la commande suivante :

```
 shell> rpm --checksig <package>.rpm

 Example:

 shell> rpm --checksig MySQL-server-4.0.10-0.i386.rpm
MySQL-server-4.0.10-0.i386.rpm: md5 gpg OK
```

**Note :** Si vous utilisez RPM 4.1 et qu'il se plaint que `(GPG) NOT OK (MISSING KEYS: GPG#5072e1f5)` (même si vous l'avez importé dans votre trousseau de clé), vous devez alors importer la clé dans votre trousseau RPM d'abord. RPM 4.1 n'utilise pas votre trousseau de clé GPG (ni GPG lui−même), car il entretient son propre trousseau de clé (car c'est une application de niveau système, et que le trousseau de clé est spécifique à chaque utilisateur). Pour importer la clé publique MySQL dans votre trousseau de clé RPM, utilisez la commande suivante :

```
 shell> rpm --import <pubkey>

 Example:

 shell> rpm --import mysql_pubkey.asc
```

Dans le cas où vous vous apercevez que la somme de contrôle `MD5 checksum` ou la signature `GPG` ne correspond pas, essayez de télécharger à nouveau le même paquet, éventuellement depuis un autre miroir. Si vous échouez plusieurs fois à vérifier l'intégrité du paquet, faites nous part de votre problème, en incluant le nom complet du paquet désiré, et les sites de téléchargement que vous avez utilisé. Mailez nous à l'adresse webmaster@mysql.com ou build@mysql.com .

## 2.2.3 Systèmes d'exploitation supportés par MySQL

Nous utilisons `GNU Autoconf` , alors il est possible de porter MySQL sur tous les systèmes modernes qui utilisent les threads Posix et un compilateur C++. Pour compiler uniquement le client, un compilateur C++ est simplement nécessaire. Nous utilisons et développons le logiciel nous−même, en commençant par Sun Solaris (Versions 2.5 − 2.7) et SuSE Linux version 7.x.

Notez que pour de nombreux systèmes d'exploitation, le support natif des threads ne fonctionne qu'avec les dernières versions. MySQL a été compilé avec succès sur les combinaisons système d'exploitation/package de threads suivants :

- AIX 4.x, 5.x avec les threads natifs. Notes sur IBM−AIX .
- Amiga.
- BSDI 2.x avec le package MIT−pthreads. Notes relatives aux versions 2.x de BSD/OS .
- BSDI 3.0, 3.1 et 4.x avec les threads natifs. Notes relatives aux versions 2.x de BSD/OS .

- DEC Unix 4.x avec les threads natifs. Notes sur Alpha−DEC−UNIX (Tru64) .
- FreeBSD 2.x avec le package MIT−pthreads. Notes relatives à FreeBSD .
- FreeBSD 3.x et 4.x avec les threads natifs. Notes relatives à FreeBSD .
- HP−UX 10.20 avec les threads DCE ou avec le package MIT−pthreads. Notes sur HP−UX Version 10.20 .
- HP−UX 11.x avec les threads natifs Notes sur HP−UX Version 11.x .
- Linux 2.0+ avec LinuxThreads 0.7.1+ ou `glibc` 2.0.7+. Notes relatives à Linux (toutes les versions de Linux) .
- Mac OS X. Notes relatives à Mac OS X .
- NetBSD 1.3/1.4 Intel et NetBSD 1.3 Alpha (requiert GNU make). Notes relatives à NetBSD .
- OpenBSD > 2.5 avec les threads natifs. OpenBSD < 2.5 avec le package MIT−pthreads. Notes relatives à la version 2.5 de OpenBSD .
- OS/2 Warp 3, FixPack 29 et OS/2 Warp 4, FixPack 4. Notes relatives à OS/2 .
- SGI Irix 6.x avec les threads natifs. Notes sur SGI Irix .
- Solaris 2.5 et plus récent, avec les threads natifs sur SPARC et x86. Notes relatives à Solaris .
- SunOS 4.x avec le package MIT−pthreads. Notes relatives à Solaris .
- Caldera (SCO) OpenServer avec un port récent du package FSU Pthreads. Notes sur Caldera .
- Caldera (SCO) UnixWare 7.0.1. Notes sur Caldera Unixware Version 7.0 .
- Tru64 Unix
- Windows 9x, Me, NT, 2000 et XP. Notes relatives à Windows .

Notez que toutes les plate−formes ne sont pas équipées de la même façon pour faire fonctionner MySQL. Les capacités d'une plateforme pour supporter de fortes charges avec MySQL est déterminé par ceci :

- Stabilité générale de la librairie de threads. Une plateforme qui a une excellente réputation en général, mais une librairie de threads instable, dont le code est utiisé par MySQL, même si le reste est parfait, fera de MySQL une application instable.
- La capacité du noyeau et/ou de la librairie de threads de profiter des capacités multi−processeurs, symétrique ou pas. En d'autres termes, lorsqu'un processus crée un thread, il doit être possible pour ce thread de s'exécuter sur différents processeurs.
- La capacité du noyau et/ou de la librairie de threads de faire fonctionner de nombreux threads qui posent et lévent des verrous mutex en peut de temps, fréquemment, sans changement de contexte excessif. En d'autres termes, si l'implémentation de `pthread_mutex_lock()` est trop soucieux du temps CPU, cela va ralentir sérieusement MySQL. Si ce problème n'est pas réglé, ajouter des processeurs supplémentaires va finalement ralentir MySQL.
- Performance et stabilité générale du système de fichiers.
- La capacité du système d'exploitation de gérer de grands fichiers, et de le faire efficacement, si vos tables sont grandes.
- Notre niveau d'expertise avec la plate−forme, chez MySQL AB. Si vous connaissons bien une plate−forme, vous pourront introduire des optimisations et des corrections spécifiques à la plate−forme, et activé lors de la compilation. Nous pouvons aussi fournir des conseils judicieux pour cnofigurer votre système optimalement pour MySQL.
- Le temps de tests que vous avons consacré à des configurations similaires, en interne.
- Le nombre d'utilisateur de MySQL qui font fonctionner MySQL avec succès sur cette plate−forme, avec des configurations similaires. SI ce nombre est grand, les chances de rencontrer un problème spécifique sont faibles.

En se basant sur les critères précédents, les meilleures plate−forme pour MySQL sont x86 avec SuSE Linux 7.1, 2.4 kernel, et ReiserFS (ou toute autre distribution Linux similaire) et SPARC avec Solaris 2.7 ou 2.8. FreeBSD vient en troisième, mais nous espérons bien le voir rejoindre le groupe de tête, une fois que la librairie de threads sera améliorée. Nous espérons aussi être bientôt

capables d'ajouter les autres plate−formes sur laquelle MySQL compile, et fonctionne correctement, mais pas toujours le bon niveau de stabilité et de performances. Cela réclame des efforts de notre part, en coopération avec les développeurs de ces plate−formes. Si vous êtes interessés par l'amélioration de ces composants, et que vous êtes en position pour influencer le développement, demandez des instructions détaillées à MySQL en envoyant un email à internals@lists.mysql.com .

Notez bien que la comparaison précédente ne signifie pas qu'un système d'exploitation est meilleur que l'autre, en général. Nous avons classé les systèmes en fonction de leur capacité à faire fonctionner un système MySQL, et nous nous limitons à cette comparaison. Avec cela en tête, le résultat de cette comparaison serait différent si nous y ajoutions d'autres problèmes. Et dans certains cas, la seule raison qui fait qu'un OS est meilleur que l'autre est parce que nous y avons consacré plus de temps, pour optimiser et tester. Nous nous bornons à exprimer notre point de vue pour vous aider à décider quelle plate forme choisir pour votre serveur MySQL.

## 2.2.4 Quelle version de MySQL utiliser ?

La première décision à prendre est de savoir si vous voulez utiliser la dernière version de développement ou la dernière version stable :

- En règle générale, si vous utilisez MySQL pour la première fois ou si vous essayer de le porter vers un système pour lequel il n'existe pas de distribution binaire, nous vous recommandons d'utiliser la dernière version stable (actuellement la version 3.23). Notez que toutes les versions de MySQL sont passées aux bancs de test MySQL avant chaque sortie (même les versions de développement).
- D'autre part, si vous utilisez un vieux système et que vous voulez procéder à une mise à jour, sans pour autant risquer de mettre à jour sans raison, vous devriez mettre à jour vers la dernière version de la même branche que celle que vous êtes en train d'utiliser (dans le cas où un numéro de version supérieur existe). Nous avons essayé de résoudre uniquement les bogues fatals et de produire des patchs petits et sûrs pour cette version.

La seconde décision est de déterminer si vous devez utiliser une distribution source ou binaire. Dans la plupart des cas, vous devrez utiliser une distribution binaire, si celle−ci existe pour votre plate−forme, car elle s'installera certainement plus facilement qu'une distribution source.

Dans les cas suivants, vous devrez opter pour une installation à partir des sources :

- Si vous voulez installer MySQL à un endroit particulier. (Les distributions binaires standards sont ``prêtes à l'emploi'' à n'importe quel endroit, mais vous pouvez vouloir davantage de flexibilité).
- Pour être apte à satisfaire les différents besoins des utilisateurs, nous fournissons deux distributions binaires : une compilée avec le support du gestionnaire de tables non transactionnel (un binaire petit et rapide), et une autre configurée avec la plupart des options avancées, telles que le support des transactions. Les deux versions sont compilées à partir des mêmes sources. Tous les clients natifs `MySQL` peuvent se connecter aux deux versions.La distribution binaire avancée est marquée du suffixe `−max` et est configurée avec les mêmes options que `mysqld−max` . `mysqld−max` , la version étendue du serveur `mysqld` .

  Si vous souhaitez utiliser des fichiers RPM de MySQL−Max, vous devez tout d'abord installer les RPM MySQL standards.
- Si vous voulez configurer `mysqld` avec certaines options supplémentaires qui ne sont pas présentes dans les distributions binaires standards. Ci−dessous, une liste des options les plus courantes que vous pourriez vouloir utiliser :
  - `−−with−innodb`
  - `−−with−berkeley−db`
  - `−−with−raid`

    - `--with-libwrap`
    - `--with-named-z-lib` (Ceci est appliqué à certains binaires)
    - `--with-debug[=full]`
    -
- La distribution binaire standard est normalement compilée avec le support de tous les jeux de caractères et sont supposés fonctionner sur nombre de processeurs de la même famille.Si vous souhaitez un serveur MySQL plus véloce, vous pouvez vouloir le recompiler juste avec le support des jeux de caractères dont vous avez besoin, utiliser un meilleur compilateur (comme `pgcc` ), ou utiliser des options de compilations davantage optimisées pour votre processeur.
- Si vous avez rencontré des bogues et que vous les avez rapporté à l'équipe de développement MySQL, vous recevrez probablement un patch que vous devrez appliquer à la distribution source pour que le bogue soit résolu.
- Si vous voulez consulter (et/ou modifier) le code C et C++ qui compose MySQL, vous devrez obtenir une distribution source. Le code source est toujours la solution ultime. Les distributions sources contiennent aussi davantage de tests et d'exemples que les distributions binaires.

La politique de nommage de MySQL utilise des numéros de version qui consiste en trois nombres suivis d'un suffixe. Par exemple, une version nommée `mysql-3.21.17-beta` doit être interprétée de la façon suivante :

- Le premier nombre ( `3` ) décrit le format de fichier. Toutes les versions 3 ont le même format de fichier.
- Le second nombre ( `21` ) correspond au niveau de version. Normalement, il y a le choix entre deux d'entre eux. L'un correspond à la version/branche stable (actuellement `23` ) et l'autre se réfère à la branche de développement (actuellement `4.0` ). Normalement, les deux versions sont stables, mais la version de développement peut comporter des lacunes, manquer de documentation sur des nouvelles fonctionnalités, ou peut ne pas compiler sur certains systèmes.
- Le troisième nombre ( `17` ) est le numéro de version au sein du niveau de version. Celui-ci est incrémenté à chaque nouvelle publication. En temps normal, vous souhaiterez utiliser la dernière version du niveau de version que vous avez choisi.
- Le suffixe ( `beta` ) indique le niveau de stabilité de la version. Les suffixes possibles sont :
    - `alpha` indique que la publication contient de grandes portions de nouveau code qui n'a pas été testé à 100%. Les bogues connus (d'ordinaire, il n'y en a aucun) doivent être documentés dans la section News. Historique des versions de MySQL . Il existe aussi de nouvelles commandes et extensions dans la plupart des versions alpha. Du développement actif qui inclut des changements majeurs dans le code peut concerner les versions alpha, mais tout sera testé avant de faire une publication. Il ne devrait pas y avoir de bogues connus dans les publications de MySQL.
    - `beta` signifie que tout le nouveau code a été testé. Aucune fonctionnalité majeure qui pourrait causer corruption du code n'est ajoutée. Il ne doit pas y avoir un seul bogue connu. Une version alpha passe en beta quand il n'y a pas eu de bogue fatal rapporté depuis au moins un mois et que nous ne prévoyons pas de nouvelle fonctionnalité qui pourrait corrompre d'anciennes commandes.
    - `gamma` est une version bêta qui existe depuis un certain temps et qui semble fonctionner correctement. Seulement des changements mineurs sont effectués. C'est ce que de nombreuses autres compagnies appellent une publication.
    - S'il n'y a pas de suffixe, cela signifie que la version fonctionne depuis un certain temps sur différents sites avec aucun rapport de bogue autre que des bogues spécifiques à une plate-forme. Seuls des corrections critiques sont appliquées à la publication. C'est ce que l'on appelle une version stable.

Toutes les versions de MySQL passent par nos tests et bancs d'essais standards pour nous assurer qu'elles peuvent être utilisées sans danger. Les séries de tests s'améliorent en permanence car les tests standards sont étendus dans le temps pour traquer tous les bogues précédemment

trouvées.

Notez bien que toutes les versions de MySQL ont été testées au moins avec :

*Une batterie de tests internes*
: C'est une reproduction d'un système de production client. Il comporte de nombreuses tables avec des centaines de Mo de données.

*Les bancs de tests MySQL*
: Ils effectuent une série de requêtes communes. C'est aussi un test pour savoir si le dernier processus d'optimisation rend le code plus rapide. La suite de tests comparatifs de MySQL .

*Le test* `crash-me`
: Il tente de déterminer de quelles fonctionnalités disposent les bases de données et quelles en sont les limites. La suite de tests comparatifs de MySQL .

Un autre test provient du fait que nous avons la version la plus récente de MySQL dans notre propre environnement de production interne, sur au moins une machine. Nous avons plus de 100 Go de données à manipuler.

## 2.2.5 Dispositions d'installation

Cette section décrit les répertoires par défaut crées en installant les distributions binaires et les distributions de sources.

Une distribution binaire est installée en la décompressant dans le répertoire d'installation que vous avec choisis (souvent `/usr/local/mysql` ) et crée les répertoires suivants à cet endroit :

| Répertoire | Contenu du répertoire |
|---|---|
| bin | Programmes clients et le serveur mysqld |
| data | Fichiers de log et bases de données |
| include | Fichiers (entêtes) inclus |
| lib | Librairies |
| scripts | mysql_install_db |
| share/mysql | Fichiers de messages d'erreurs |
| sql-bench | Benchmarks |

Une distribution des sources est installée après que vous l'ayez configuré et compilé. Par défaut, l'installation se fait dans `/usr/local` , dans les sous-répertoires suivants :

| Répertoire | Contenu du répertoire |
|---|---|
| bin | Programmes clients et scripts |
| include/mysql | Fichiers (entêtes) inclus |
| info | Documentation au format Info |
| lib/mysql | Librairies |
| libexec | Le serveur mysqld |
| share/mysql | Fichiers de messages d'erreurs |
| sql-bench | Benchmarks et test crash-me |
| var | Fichiers de log et bases de données |

Dans le répertoire d'installation, les dispositions d'une installation des sources diffère d'une installation binaire des façons suivantes :

- Le serveur `mysqld` est installé dans le répertoire `libexec` plutôt que dans le répertoire `bin` .
- Le répertoire des données est `var` plutôt que `data` .
- `mysql_install_db` est installé dans le répertoire `/usr/local/bin` plutôt que dans `/usr/local/mysql/scripts` .
- Le répertoire des fichier d'entête et les répertoires des librairies sont `include/mysql` et `lib/mysql` au lieu de `include` et `lib` .

Vous pouvez créer votre propre installation binaire à partir d'une distribution de sources compilées en exécutant le script `scripts/make_binary_distribution` .

## 2.2.6 Quand et comment sont publiées les nouvelles versions

MySQL évolue rapidement ici, à MySQL AB, et nous voulons le partager avec les autres utilisateurs de MySQL. Nous essayons de faire une nouvelle version à chaque fois que nous avons implanté des fonctionnalités qui seront utiles à d'autres.

Nous essayons aussi d'aider les utilisateurs dont les requêtes sont faciles à programmer. Nous prenons en considération tout ce que nos clients nous demandent, et nous accordons une attention particulière à nos clients qui ont pris une licence e−mail étendue.

Personne n'est obligé de télécharger une nouvelle version. La sections News vous indiquera si la nouvelle version contient une fonctionnalité que vous attendez. Historique des versions de MySQL .

Nous utilisons la politique suivante, lors de la mise à jour de MySQL :

- Pour chaque modification mineure, le dernier numéro de la chaîne de version est incrémenté. Lorsqu'il y a des nouvelles fonctionnalités importantes ou des incompatibilité mineures avec la version précédente, nous incrémentons le chiffre du milieu. Lorsque le format de fichier change, le premier numéro est incrémenté.
- Des versions tables et testées sont publiées une à deux fois dans l'année, mais si de petits bogues apparaissent, une version qui ne va corriger que ces bogues sera publiée.
- Des versions fonctionnelles avec des corrections de bogues pour les vieilles versions sont publiées toutes les 1 à 8 semaines.
- Les distributions binaires de certaines plate−formes seront compilées par nos soins pour les versions majeures. D'autres personnes font des versions binaires pour d'autres systèmes, mais probablement moins fréquemment.
- Nous rendons généralement public les correctifs, une fois que nous avons découverts de petits bogues. Ils sont postés sur la liste bugs@lists.mysql.com et seront ajoutés à la prochaine version.
- Pour les bogues non critiques mais ennuyant, nous allons les ajouter dans le serveur de sources de MySQL, et ils seront publiés dans la prochaine version.
- Si il y a un bogue fatal dans une version, nous publierons une nouvelle version aussitôt que possible. Nous apprécions que les autres éditeurs fasse la même chose.

La version stable actuelle est la version 3.23; nous avons déjà commencé à développer activement la version 4.0. Les bogues seront corrigés dans cette version. Nous ne croyons pas à un gel complet, car cela laisserait en plan des corrections de bogues, et des listes de souhaites. ``Un peu gelé'' signifie que nous pourrions apporter de petits ajouts qui ``n'affecterons pas presque sûrement pas le fonctionnement''.MySQL utilise un système de numérotation légèrement différent pour les autres produits. En général, il est relativement sûr d'utiliser une version qui a été publiée il y a quelques semaines sans être remplacée par une nouvelle version. Which MySQL Version to Use .

## 2.2.7 Politique de publication : aucun bug connu dans les versions

Nous mettons beaucoup d'efforts et de temps à la publication de version sans bugs. A notre connaissance, nous n'avons jamais publié une version de MySQL qui contienne un bug fatal **connu** et reproductible.

Un bug fatal est un problème qui crashe MySQL en utilisation normal, fournit des réponses erronnées à des requêtes classiques, ou a des problèmes de sécurité.

Nous documentons tous les problèmes ouverts, bugs et tout ce qui dépend des choix de conceptions. Erreurs connues et manques de MySQL .

Nous avons pour but de corriger tout ce qui est corrigeable, sans risquer la stabilité des versions de MySQL. Dans certains cas, cela signifie que nous pouvons corriger une erreur dans la version de développement, mais pas dans la version stable. Naturellement, nous documentons ces problèmes, pour que les utilisateurs soient avertis.Voici une description de notre processus de publication :

- Nous surveillons les bugs sur les listes de support utilisateur, les listes externesm et la base de données de bugs sur le site http://bugs.mysql.com/ .
- Tous les bugs rapportés pour les versions en production entrent dans la base de données des bugs.
- Lorsque nous corrigeons un bug, nous essayons de réaliser un cas de test, que nous incluons dans notre système de tests, pour nous assurer que le bugs ne reviendra jamais (environ 90% des bugs ont des cas de tests).
- Nous créons aussi des cas de tests pour toutes les nouvelles fonctionnalités que nous voulons ajouter à MySQL.
- Avant de commencer à compiler une nouvelle version de MySQL, nous nous assurons que tous les bugs reproductibles pour les versions anciennes (3.23.x, 4.0.x, etc...) sont corrigés. Si le bug ne peut être corrigé (pour des raisons de choix de conception), nous le documentons dans le manuel. Erreurs connues et manques de MySQL .
- Nous compilons MySQL sur toutes les plates−formes pour lesquelles nous distribuons des paquets binaires (plus de 15 plates−formes à ce jour), et nous exécutons notre suite de tests et notre suite de performances sur chacune d'entre elles.
- Nous ne publions pas un paquet binaire sur une plate−forme pour laquelle la suite de test ou de performances échoue. Si c'est une erreur générale, nous la corrigeons, et nous recommencons les tests sur toutes les plates− formes, à partir de zéro.
- Si nous recevons, durant le temps de compilation et de tests, qui peut prendre deux à trois jours, un rapport de bugs concernant un bug fatal (par exemple, un bug qui crée un coredump), nous le corrigeons, et nous recommencons le processus de tests.
- Après avoir publié les paquets binaires sur http://www.mysql.com/ , nous envoyons une annonce par email aux listest mysql@lists.mysql.com et announce@lists.mysql.com . Le message d'annonce contient une liste de toutes les changements de la version, et de tous les bugs connus. La section des problèmes connus, 'known problems', dans les notes de publications n'a été utilisé que dans quelques versions.
- Pour donner rapidement accàs aux dernières fonctionnalités de MySQL, nous réalisons une publication de MySQL toutes les 4 à 5 semaines.
- Si, après une publication, nous recevons des rapports de bugs qui prouvent qu'il y a, malgré tout, un bug critique dans une version spécifique à une plate−forme, nous le corrigeons rapidement, et nous annoncons une version `'a'` pour la plate−forme. Grâce à notre large communauté d'utilisateurs, les problèmes sont trouvés rapidement.
- Nos résultats de bonne publication sont excellents. Dans les dernières 150 verions, nous avons du reprendre la compilation moins de 10 fois (dans trois des cas, le bug était dû à glibc sur l'une de nos machines de tests, qu'il a été difficile de trouver.

## 2.2.8 Binaires compilés par MySQL AB

Comme service, nous, MySQL AB, proposons un jeu de distributions binaires de MySQL qui sont compilés sur nos machines, ou les machines auxquelles nos clients nous ont gentiment donné accès.

Ces distributions sont générées avec le script `scripts/make_binary_distribution` et sont configurés avec les compilateurs et options suivantes :

*Linux 2.4.xx i386 with* `gcc` *2.95.3*

```
CFLAGS="−O2 −mcpu=pentiumpro" CXX=gcc CXXFLAGS="−O2 −mcpu=pentiumpro −felide−constructors" ./configure
−−prefix=/usr/local/mysql −−with−extra−charsets=complex −−enable−thread−safe−client −−enable−local−infile
−−enable−assembler −−disable−shared −−with−client−ldflags=−all−static −−with−mysqld−ldflags=−all−static
```

*Linux 2.4.xx Intel Itanium 2 with ecc (Intel C++ Itanium Compiler 7.0)*

```
CC=ecc CFLAGS="-O2 -tpp2 -ip -nolib_inline" CXX=ecc CXXFLAGS="-O2 -tpp2 -ip -nolib_inline" ./configure
--prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
```

*Linux 2.4.xx Intel Itanium with ecc (Intel C++ Itanium Compiler 7.0)*

```
CC=ecc CFLAGS=-tpp1 CXX=ecc CXXFLAGS=-tpp1 ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile
```

*Linux 2.4.xx alpha with ccc (Compaq C V6.2-505 / Compaq C++ V6.3-006)*

```
CC=ccc CFLAGS="-fast -arch generic" CXX=cxx CXXFLAGS="-fast -arch generic -noexceptions -nortti" ./configure
--prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--with-mysqld-ldflags=-non_shared --with-client-ldflags=-non_shared --disable-shared
```

*Linux 2.2.xx sparc with egcs 1.1.2*

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --disable-shared
```

*Linux 2.4.xx s390 with gcc 2.95.3*

```
CFLAGS="-O2" CXX=gcc CXXFLAGS="-O2 -felide-constructors" ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile --disable-shared
--with-client-ldflags=-all-static --with-mysqld-ldflags=-all-static
```

*Sun Solaris 2.8 sparc with gcc 3.2*

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --with-named-z-libs=no
--with-named-curses-libs=-lcurses --disable-shared
```

*Sun Solaris 2.9 sparc with gcc 2.95.3*

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --with-named-curses-libs=-lcurses
--disable-shared
```

*Sun Solaris 2.9 sparc with cc-5.0 (Sun Forte 5.0)*

```
CC=cc-5.0 CXX=CC ASFLAGS="-xarch=v9" CFLAGS="-Xa -xstrconst -mt -D_FORTEC_ -xarch=v9" CXXFLAGS="-noex -mt
-D_FORTEC_ -xarch=v9" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --with-named-z-libs=no
--enable-thread-safe-client --disable-shared
```

*IBM AIX 4.3.2 ppc with gcc 3.2.1*

```
CFLAGS="-O2 -mcpu=powerpc -Wa,-many " CXX=gcc CXXFLAGS="-O2 -mcpu=powerpc -Wa,-many -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --with-named-z-libs=no --disable-shared
```

*IBM AIX 5.1.0 ppc with gcc 3.2.1*

```
CFLAGS="-O2 -mcpu=powerpc -Wa,-many" CXX=gcc CXXFLAGS="-O2 -mcpu=powerpc -Wa,-many -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--with-server-suffix="-pro" --enable-thread-safe-client --enable-local-infile --with-named-z-libs=no
--disable-shared --with-innodb
```

*HP-UX 10.20 pa-risc1.1 with gcc 3.1*

```
CFLAGS="-DHPUX -I/opt/dce/include -O3 -fPIC" CXX=gcc CXXFLAGS="-DHPUX -I/opt/dce /include -felide-constructors
-fno-exceptions -fno-rtti -O3 -fPIC" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --with-pthread --with-named-thread-libs=-ldce
--with-lib-ccflags=-fPIC --disable-shared
```

*HP-UX 11.11 pa-risc2.0 with aCC (HP ANSI C++ B3910B A.03.33)*

```
CC=cc CXX=aCC CFLAGS=+DD64 CXXFLAGS=+DD64 ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared
```

*Apple Mac OS X 10.2 powerpc with gcc 3.1*

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared
```

*FreeBSD 4.7 i386 with gcc 2.95.4*

```
CFLAGS=-DHAVE_BROKEN_REALPATH ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --with-named-z-libs=not-used --disable-shared
```

Les paquets binaires suivants sont compilés sur des systèmes que des tiers prêtent gracieusement à MySQL AB. Notez que ces paquets sont fournis gracieusement. Comme MySQL AB n'a pas le contrôle complet sur ces systèmes, nous ne pouvons proposer qu'un support limité sur ces systèmes.

*SCO Unix 3.2v5.0.6 i386 with gcc 2.95.3*

```
CFLAGS="-O3 -mpentium" LDFLAGS=-static CXX=gcc CXXFLAGS="-O3 -mpentium -felide-constructors" ./configure
--prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--with-named-z-libs=no --enable-thread-safe-client --disable-shared
```

*Caldera Open Unix 8.0.0 i386 with cc 3.2*

```
CC=cc CFLAGS="-O" CXX=CC ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --with-named-z-libs=no --enable-thread-safe-client
--disable-shared
```

*Compaq Tru64 OSF/1 V5.1 732 alpha with cc/cxx (Compaq C V6.3-029i / DIGITAL C++ V6.1-027)*

```
CC="cc -pthread" CFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all" CXX="cxx -pthread"
CXXFLAGS="-O4 -ansi_alias -fast -inline speed -speculate all -noexceptions -nortti" ./configure
--prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--with-prefix=/usr/local/mysql --with-named-thread-libs="-lpthread -lmach -lexc -lc" --disable-shared
--with-mysqld-ldflags=-all-static
```

*SGI Irix 6.5 IP32 with gcc 3.0.1*

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared
```

Les options suivantes de compilations ont été utilisées pour les paquets binaires de MySQL, qui étaient fournis auparavant. Ces paquets ne sont plus mis à jours, mais les options de compilation sont conservées ici pour mémoire.

*Linux 2.2.x with x686 with* `gcc` *2.95.2*

```
CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro -felide-constructors -fno-exceptions -fno-rtti"
./configure --prefix=/usr/local/mysql --enable-assembler --with-mysqld-ldflags=-all-static --disable-shared
--with-extra-charsets=complex
```

*SunOS 4.1.4 2 sun4c with* `gcc` *2.7.2.1*

```
CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors" ./configure --prefix=/usr/local/mysql --disable-shared
--with-extra-charsets=complex --enable-assembler
```

*SunOS 5.5.1 (and above) sun4u with* `egcs` *1.0.3a or 2.90.27 or gcc 2.95.2 and newer*

```
CC=gcc CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti" ./configure
--prefix=/usr/local/mysql --with-low-memory --with-extra-charsets=complex --enable-assembler
```

*SunOS 5.6 i86pc with* `gcc` *2.8.1*

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql --with-low-memory --with-extra-charsets=complex
```

*BSDI BSD/OS 3.1 i386 with* `gcc` *2.7.2.1*

```
CC=gcc CXX=gcc CXXFLAGS=-O ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
```

*BSDI BSD/OS 2.1 i386 with* `gcc` *2.7.2*

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
```

*AIX 2 4 with* `gcc` *2.7.2.2*

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
```

Si vous avez des options plus optimales pour l'une des configurations précédement listées, vous pouvez toujours nous en faire part sur la liste de distribution des développeurs ( internals@lists.mysql.com ).

Les distributions RPM antérieures à la version 3.22 de MySQL sont contribuées par nos utilisateurs. A partir de la version 3.22, les RPM sont générés par nous chez MySQL AB.

Si vous voulez compiler une version de débogage de MySQL, vous devez ajouter `--with-debug` ou `--with-debug=full` aux lignes de configurations précédentes et effacer les options `-fomit-frame-pointer` .

Pour les distributions Windows, voyez Installer MySQL sous Windows .

## 2.2.9 Installer MySQL à partir d'une distribution binaire

Vous pouvez aussi vous référer à Installer les binaires , Installer MySQL sous Linux , et Building Client Programs .

Vous aurez besoin des outils suivants pour installer les binaires MySQL :

- GNU `gunzip` pour décompresser la distribution.
- Un programme `tar` pour désarchiver la distribution. GNU `tar` est connu pour fonctionner. Le `tar` de Sun connaît quelques problèmes.

L'utilisation de fichiers RPM (RedHat Package Manager) est une autre façon d'installer MySQL sous Linux. Installer MySQL sous Linux .

Si vous rencontrez des problèmes, **utilisez toujours** `mysqlbug` pour poster des questions à mysql@lists.mysql.com . Même si le problème n'est pas un bogue, `mysqlbug` rassemble des informations sur le système qui aidera les autres à résoudre votre problème. En n'utilisant pas `mysqlbug` , vous amoindrissez vos chances d'obtenir une solution à votre problème ! Vous trouverez `mysqlbug` dans le répertoire `scripts` après avoir désarchivé la distribution. Comment rapporter des bogues ou problèmes .

Les commandes de base que vous devez lancer pour installer et utiliser MySQL à partir des binaires sont les suivantes :

```
 shell> groupadd mysql
shell> useradd -g mysql mysql
shell> cd /usr/local
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> scripts/mysql_install_db
shell> chown -R root  .
shell> chown -R mysql data
shell> chgrp -R mysql .
shell> bin/safe_mysqld --user=mysql &
or
shell> bin/mysqld_safe --user=mysql &
si vous utilisez MySQL 4.x
```

Vous pouvez ajouter des nouveaux utilisateurs en utilisant le script `bin/mysql_setpermission` si vous installez les modules Perl `DBI` et `Msql-Mysql-modules` .

Une description plus détaillée est disponible ci-dessous.

Pour installer une distribution binaire, suivez ces étapes et procédez à Configuration et tests consécutifs à l'installation , pour la configuration et les tests consécutifs à l'installation :

- Choisissez le répertoire ou vous voulez désarchiver la distribution et déplacez-vous y. Dans l'exemple suivant, nous désarchivons la distribution dans le répertoire `/usr/local` et créons un répertoire `/usr/local/mysql` dans lequel MySQL est installé. (Les instructions suivantes supposent bien sûr que vous avez les autorisations suffisantes pour créer des fichiers dans `/usr/local` . Si ce répertoire est protégé, vous aurez besoin de faire l'installation en tant que `root` .)
- Procurez-vous un fichier de distribution à partir d'un des sites cités dans How to Get MySQL .

  Les distributions binaires de MySQL sont fournies sous forme d'archives `tar` compressées et ont des noms de la forme `mysql-VERSION-OS.tar.gz` , où `VERSION` est un nombre (par exemple, `3.21.15` ), et `OS` indique le type de système d'exploitation pour lequel la distribution est compilée (par exemple, `pc-linux-gnu-i586` ).
- Si vous voyez une distribution binaire avec le suffixe `-max` , cela signifie que le binaire supporte les tables avec les transactions ainsi que d'autres fonctionnalités. `mysqld-max` , la version étendue du serveur `mysqld` . Veuillez noter que tous les binaires sont compilés à partir du même code source.
- Ajoutez un utilisateur et un groupe avec les droits desquels `mysqld` fonctionnera :

  ```
   shell> groupadd mysql
  shell> useradd -g mysql mysql
  ```

  Ces commandes ajoutent le groupe `mysql` group et l'utilisateur `mysql` . La syntaxe de `useradd` et de `groupadd` peut varier de façon significative suivant les versions d'Unix. Elles peuvent aussi s'appeler `adduser` et `addgroup` . Vous pouvez aussi souhaiter nommer le groupe et l'utilisateur autrement que `mysql` .
- Déplacez-vous dans le répertoire d'installation choisi :

  ```
   shell> cd /usr/local
  ```
- Décompressez la distribution et créez le répertoire d'installation :

  ```
   shell> gunzip < /chemin/de/mysql-VERSION-OS.tar.gz | tar xvf -
  shell> ln -s chemin-complet-de-mysql-VERSION-OS mysql
  ```

  La première commande crée un répertoire `mysql-VERSION-OS` . La seconde commande crée un lien symbolique vers ce répertoire. Cela vous permet de vous référer plus facilement au répertoire d'installation en appelant `/usr/local/mysql` .
- Déplacez vous dans le répertoire d'installation :

  ```
   shell> cd mysql
  ```

  Vous y trouverez de nombreux fichiers et sous-répertoires. Les plus importants lors de l'installation sont les sous-répertoires `bin` et `scripts` .

- Si vous souhaitez utiliser `mysqlaccess` et placer la distribution MySQL à un endroit non standard, vous devez changer l'endroit où `mysqlaccess` s'attend à trouver le client `mysql` . Editez le script `bin/mysqlaccess` aux environs de la ligne 18. Cherchez une ligne qui ressemble à ceci :

  ```
  $MYSQL     = '/usr/local/bin/mysql';    # path to mysql executable
  ```

  Changez le chemin pour avoir le bon endroit où `mysql` est actuellement stocké sur votre système. Si vous ne le faites pas, vous obtiendrez une erreur `Broken pipe` quand vous lancerez `mysqlaccess` .
- Créez les tables de privilèges MySQL (ceci est nécessaire si vous n'avez jamais installé MySQL auparavant) :

  ```
  shell> scripts/mysql_install_db
  ```

  Veuillez noter que les versions de MySQL antérieures à la version 3.22.10 démarraient le serveur MySQL quand vous exécutiez `mysql_install_db` . Ce n'est plus le cas.
- Nommez `root` en tant que propriétaire des binaires et l'utilisateur avec les droits duquel vous ferez fonctionner `mysqld` comme propriétaire du répertoire de données :

  ```
  shell> chown -R root  /usr/local/mysql
  shell> chown -R mysql /usr/local/mysql/var
  shell> chgrp -R mysql /usr/local/mysql
  ```

  La première commande change les attributs `owner` en faveur de l'utilisateur `root` , la seconde change les attributs `owner` du répertoire de données en faveur de l'utilisateur `mysql` , et la troisième change les attributs `group` en faveur du groupe `mysql` .
- Si vous voulez installer le support de l'interface Perl `DBI` / `DBD` , référez-vous à Commentaires sur l'installation de Perl .
- Si vous souhaitez que MySQL se lance automatiquement au démarrage de votre machine, vous pouvez copier `support-files/mysql.server` à l'endroit où votre système stocke ses fichiers de démarrage. Davantage d'informations sont disponibles dans le script `support-files/mysql.server` lui-même et dans Démarrer et arrêter MySQL automatiquement .

Après que tout soit installé, vous devez initialiser et tester votre distribution :

```
shell> /usr/local/mysql/bin/safe_mysqld --user=mysql &
```

Procédez maintenant à `safe_mysqld` , le script père de `mysqld` , et à la Configuration et tests consécutifs à l'installation .

## 2.3 Installer MySQL à partir des sources

Avant de procéder à l'installation à partir des sources, vérifiez auparavant que notre distribution binaire pour votre plate-forme ne fonctionne pas. Nous faisons un maximum d'efforts pour nous assurer que nos binaires sont compilés avec les meilleures options possibles.

Vous avez besoin des outils suivants pour compiler et installer MySQL à partir des sources :

- GNU `gunzip` pour décompresser la distribution.
- Un programme `tar` pour désarchiver la distribution. GNU `tar` est connu pour fonctionner. Le `tar` de Sun connaît quelques problèmes.
- Un compilateur C++ ANSI fonctionnel. `gcc` >= 2.95.2, `egcs` >= 1.0.2 ou `egcs 2.91.66` , SGI C++, et SunPro C++ sont quelques-uns des compilateurs réputés pour fonctionner. `libg++` n'est pas nécessaire si vous utilisez `gcc` . `gcc` 2.7.x souffre d'un bogue qui l'empêche de compiler quelques fichiers C++ correctement écrits, tels que `sql/sql_base.cc` . Si vous disposez seulement de `gcc` 2.7.x, vous devez mettre à jour votre `gcc` afin de compiler MySQL. `gcc` 2.8.1 est aussi reconnu pour rencontrer des problèmes sur certaines plate-formes, il devrait donc être désactivé si un autre compilateur existe pour la plate-forme. `gcc` >= 2.95.2 est recommandé pour compiler MySQL dans ses versions 3.23.x.
- Un bon programme `make` . GNU `make` est une fois de plus recommandé et est quelquefois requis. Si vous rencontrez des problèmes, nous vous recommandons d'essayer GNU `make`

3.75 ou supérieur.

Si vous utilisez une version récente de `gcc` , suffisamment récente pour reconnaître l'option `-fno-exceptions` , il est **très important** que vous l'utilisiez. Sinon, vous risquez de compiler un binaire qui crashe aléatoirement. Nous recommandons donc l'utilisation de `-felide-constructors` et `-fno-rtti` en même temps que `-fno-exceptions` . En cas de doute, faites la chose suivante :

```
CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions \
      -fno-rtti" ./configure --prefix=/usr/local/mysql --enable-assembler \
      --with-mysqld-ldflags=-all-static
```

Sur la plupart des systèmes, il en résultera un binaire rapide et stable.

Si vous rencontrez des problèmes, **utilisez toujours** `mysqlbug` pour poster des questions à mysql@lists.mysql.com . Même si le problème n'est pas un bogue, `mysqlbug` rassemble des informations sur le système qui aidera les autres à résoudre votre problème. En n'utilisant pas `mysqlbug` , vous amoindrissez vos chances d'obtenir une solution à votre problème ! Vous trouverez `mysqlbug` dans le répertoire `scripts` après avoir désarchivé la distribution. Comment rapporter des bogues ou problèmes .

## 2.3.1 Vue d'ensemble de l'installation rapide

Les commandes de base que vous devez exécuter pour installer MySQL à partir des sources sont :

```
 shell> groupadd mysql
shell> useradd -g mysql mysql
shell> gunzip < mysql-VERSION.tar.gz | tar -xvf -
shell> cd mysql-VERSION
shell> ./configure --prefix=/usr/local/mysql
shell> make
shell> make install
shell> scripts/mysql_install_db
shell> chown -R root  /usr/local/mysql
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> /usr/local/mysql/bin/safe_mysqld --user=mysql &
ou
shell> /usr/local/mysql/bin/mysqld_safe --user=mysql &
si vous utilisez MySQL 4.x.
```

Si vous voulez avoir le support des tables InnoDB, vous devez éditer le fichier `/etc/my.cnf` et enlever le caractère `#` avant le paramètre qui commence par `innodb_...` . Fichier d'options `my.cnf` , et Options de démarrage `InnoDB` .Si vous utilisez un fichier RPM source, faites :

```
 shell> rpm --rebuild MySQL-VERSION.src.rpm
```

Cela produira un fichier RPM binaire que vous pourrez installer.Vous pouvez ajouter des utilisateurs en lançant le script `bin/mysql_setpermission` si vous installez les modules Perl `DBI` et `Msql-Mysql-modules` .

Ci-dessous, une description plus détaillée.

Pour installer MySQL à partir des sources, suivez ces étapes et procédez à Configuration et tests consécutifs à l'installation , pour la configuration et les tests consécutifs à l'installation :

- Notez le répertoire où vous voulez décompacter les sources et déplacez vous y.
- Récupérez une distribution des sources à partir d'un des sites listés sur How to Get MySQL .
- Si vous souhaitez utiliser les tables Berkeley DB avec MySQL, vous devez obtenir une version modifiée du code source de Berkeley DB. Veuillez lire le chapitre relatif aux tables Berkeley DB avant de procéder. Tables `BDB` ou `BerkeleyDB` .Les distributions des sources MySQL sont fournies sous forme d'archive `tar` compressées et ont des noms comme `mysql-VERSION.tar.gz` , où `VERSION` est un nombre comme 4.0.5a.

- Ajoutez un utilisateur et un groupe avec les droits desquels `mysqld` fonctionnera :

```
shell> groupadd mysql
shell> useradd -g mysql mysql
```

  Ces commandes ajoutent le groupe `mysql` group et l'utilisateur `mysql` . La syntaxe de `useradd` et de `groupadd` peut varier de façon significative suivant les versions d'Unix. Elles peuvent aussi s'appeler `adduser` et `addgroup` . Vous pouvez aussi souhaiter nommer le groupe et l'utilisateur autrement que `mysql` .
- Décompressez la distribution dans le répertoire courant :

```
shell> gunzip < /path/to/mysql-VERSION.tar.gz | tar xvf -
```

  Cette commande crée un répertoire nommé `mysql-VERSION` .
- Déplacez-vous dans le répertoire racine de la distrbution décompressée :

```
shell> cd mysql-VERSION
```

  Notez bien que vous devez alors configurer et compiler MySQL depuis ce répertoire racine. Vous ne pouvez pas le compiler à partir d'un autre répertoire.
- Configurez votre version et compilez tout :

```
shell> ./configure --prefix=/usr/local/mysql
shell> make
```

  Quand vous lancez la commande `configure` , vous pouvez spécifier quelques options. Lancez `./configure --help` pour une liste des options disponibles. La partie Les options habituelles de `configure` , traite des options les plus utiles.Si `configure` échoue, et que vous êtes sur le point d'envoyer un mail à mysql@lists.mysql.com pour demander de l'aide, ajoutez s'il vous plait les quelques lignes de `config.log` qui pourraient selon vous aider à résoudre le problème. Ajoutez aussi les deux dernières lignes de sortie de `configure` si `configure` s'arrête précipitemment. Envoyez le rapport de bogue en utilisant le script `mysqlbug` . Comment rapporter des bogues ou problèmes .

  Si la compilation échoue, référez-vous à Des problèmes à la compilation ? pour de l'aide sur les problèmes les plus courants.
- Installez tout :

```
shell> make install
```

  Vous aurez certainement besoin de lancer cette commande en tant que `root` .
- Créez les tables de gestion des droits MySQL (ceci est nécessaire uniquement si vous n'avez pas installé MySQL auparavant):

```
shell> scripts/mysql_install_db
```

  Veuillez noter que les versions de MySQL antérieures à la version 3.22.10 démarraient le serveur MySQL quand vous exécutiez `mysql_install_db` . Ce n'est plus le cas!
- Nommez `root` en tant que propriétaire des binaires et l'utilisateur avec les droits duquel vous ferez fontionner `mysqld` comme propriétaire du répertoire de données :

```
shell> chown -R root  /usr/local/mysql
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql
```

  La première commande change les attributs `owner` en faveur de l'utilisateur `root` , la seconde change les attributs `owner` du répertoire de données en faveur de l'utilisateur `mysql` , et la troisième change les attributs `group` en faveur du groupe `mysql` .
- Si vous voulez installer le support de l'interface Perl `DBI` / `DBD` , référez-vous à Commentaires sur l'installation de Perl .
- Si vous souhaitez que MySQL se lance automatiquement au démarrage de votre machine, vous pouvez copier `support-files/mysql.server` à l'endroit où votre système stocke ses fichiers de démarrage. Davantage d'informations sont disponibles dans le script `support-files/mysql.server` lui-même et dans Démarrer et arrêter MySQL automatiquement .

Après que tout soit installé, vous devez initialiser et tester votre distribution :

```
shell> /usr/local/mysql/bin/safe_mysqld --user=mysql &
```

Si cette commande échoue immédiatement sur un `mysqld daemon ended` , vous pouvez trouvez des informations dans le fichier `mysql-data-directory/'hostname'.err` . La raison la plus courante est que vous avez déjà un autre serveur `mysqld` qui fonctionne. Faire fonctionner plusieurs serveurs MySQL sur la même machine .Procédez maintenant à la Configuration et tests consécutifs à l'installation .

## 2.3.2 Appliquer des patches

Parfois, des patches sont publiés sur la liste de diffusion ou sont placés sur la page des patches sur le site de MySQL ( http://www.mysql.com/Downloads/Patches/ ).

Pour appliquer un patch venant de la liste de diffusion, sauvegardez le message qui contient le patch, placez−le dans le répertoire racine de vos sources MySQL et lancez les commandes suivantes :

```
shell> patch −p1 < patch−file−name
shell> rm config.cache
shell> make clean
```

Les patches venant du site FTP sont distribués sous forme de fichiers en texte clair ou bien sous forme de fichiers compressés avec `gzip` . Appliquez un patch en texte clair de la même façon qu'avec les patches de la liste de diffusion. Pour appliquer un patch compressé, déplacez−vous dans le répertoire racine de vos sources et lancez les commandes suivantes :

```
shell> gunzip < patch−file−name.gz | patch −p1
shell> rm config.cache
shell> make clean
```

Après avoir appliqué un patch, suivez les instructions d'une installation normale à partir des sources, à commencer par l'étape du `./configure` . Après avoir lancé le `make install` , redémarrez votre serveur MySQL.

Vous aurez certainement besoin d'arrêter tout autre serveur MySQL avant de lancer `make install` . (Utilisez pour cela `mysqladmin shutdown` .) Certains systèmes n'autorisent pas l'installation d'une nouvelle version d'un programme ; elle remplace la version qui est en train de s'exécuter.

## 2.3.3 Options habituelles de configure

Le script `configure` vous donne un bon moyen de contrôler la configuration de votre distribution MySQL. Habituellement, vous faites cela en spécifiant les options dans la ligne de commande de `configure` . Vous pouvez aussi affecter le comportement de `configure` en utilisant certaines variables d'environnement. Variables d'environnement MySQL . Pour une liste des options supportées par `configure` , exécutez cette commande :

```
shell> ./configure −−help
```

Les options de `configure` les plus utilisées sont décrites ici :

- Pour ne compiler que les librairies et programmes clients, et non le serveur, utilisez l'option `−−without−server` :

  ```
  shell> ./configure −−without−server
  ```

  Si vous n'avez pas de compilateur C++, `mysql` ne compilera pas (c'est le programme client qui requière C++). Dans ce cas, vous pouvez supprimer la partie de code dans `configure` qui vérifie l'existence d'un compilateur C++, puis exécuter `./configure` avec l'option `−−without−server` . La compilation essaiera encore de construire `mysql` , mais vous pouvez ignorer les messages d'erreurs concernant `mysql.cc` . (Si `make` stoppe, essayez `make −k` pour dire de continuer même si on rencontre des erreurs.)
- Si vous voulez obtenir une librairie MySQL embarquée ( `libmysqld.a` ) vous devez utiliser l'option `−−with−embedded−server` .
- Si vous ne voulez pas que vos fichiers de log et bases de données soient dans `/usr/local/var` , utiliser une commande `configure` se rapprochant de l'une des commandes suivantes :

  ```
  shell> ./configure −−prefix=/usr/local/mysql
  shell> ./configure −−prefix=/usr/local \
           −−localstatedir=/usr/local/mysql/data
  ```

  La première change le préfixe de l'installation pour que tout soit installé dans `/usr/local/mysql` au lieu de `/usr/local` par défaut. La seconde commande préserve le préfixe d'installation par

défaut mais change le répertoire par défaut pour les bases de données (normalement `/usr/local/var` ) en `/usr/local/mysql/data` . Après que vous ayez compilé MySQL, vous pouvez changer ces options dans les fichiers d'options. Fichier d'options `my.cnf` .

- Si vous utilisez Unix et que vous voulez que la socket de MySQL soit à un autre endroit que celui par défaut (normalement `/tmp` ou `/var/run` ) utilisez une commande `configure` comme celle−ci :

```
shell> ./configure −−with−unix−socket−path=/usr/local/mysql/tmp/mysql.sock
```

  Notez que le fichier donné doit avoir un chemin absolu ! Vous pourrez aussi changer le chemin vers `mysql.sock` plus tard en utilisant les fichiers d'options de MySQL. Comment protéger `/tmp/mysql.sock` .
- Si vous voulez compiler des programmes liés statiquement (par exemple, pour créer une distribution binaire, pour obtenir plus de vitesse, ou pour résoudre des problèmes avec quelques distributions RedHat Linux), exécutez `configure` de la manière suivante :

```
shell> ./configure −−with−client−ldflags=−all−static \
          −−with−mysqld−ldflags=−all−static
```

- Si vous utilisez `gcc` et n'avez pas `libg++` ou `libstdc++` d'installés, vous pouvez dire à `configure` d'utiliser `gcc` en tant que compilateur C++ :

```
shell> CC=gcc CXX=gcc ./configure
```

  Quand vous utilisez `gcc` en tant que compilateur C++, aucune tentative de liaison avec `libg++` ou `libstdc++` ne sera effectuée. Il peut être bon d'utiliser cette méthode même si vous avez les librairies citées, car quelques versions de celles−ci ont causé des problèmes à des utilisateurs MySQL par le passé.Voici quelques variables d'environnement à définir selon le compilateur que vous utilisez :

| Compilateur | Options recommandées |
|---|---|
| gcc 2.7.2.1 | CC=gcc CXX=gcc CXXFLAGS="−O3 −felide−constructors" |
| egcs 1.0.3a | CC=gcc CXX=gcc CXXFLAGS="−O3 −felide−constructors −fno−exceptions −fno−rtti" |
| gcc 2.95.2 | CFLAGS="−O3 −mpentiumpro" CXX=gcc CXXFLAGS="−O3 −mpentiumpro \ −felide−constructors −fno−exceptions −fno−rtti" |
| pgcc 2.90.29 ou plus récent | CFLAGS="−O3 −mpentiumpro −mstack−align−double" CXX=gcc \ CXXFLAGS="−O3 −mpentiumpro −mstack−align−double −felide−constructors \ −fno−exceptions −fno−rtti" |

  Dans la plupart des cas, vous pouvez obtenir un binaire MySQL raisonnablement optimal en utilisant les options de la table précédente et en ajoutant les options suivantes aux lignes de configuration :

```
 −−prefix=/usr/local/mysql −−enable−assembler \
−−with−mysqld−ldflags=−all−static
```

  En d'autres termes, la ligne de configuration ressemble à ce qui suit pour les versions récentes de gcc :

```
 CFLAGS="−O3 −mpentiumpro" CXX=gcc CXXFLAGS="−O3 −mpentiumpro \
−felide−constructors −fno−exceptions −fno−rtti" ./configure \
−−prefix=/usr/local/mysql −−enable−assembler \
−−with−mysqld−ldflags=−all−static
```

  Les binaires que nous fournissons sur le site de MySQL à http://www.mysql.com/ sont tous compilés avec une optimisation totale et devraient être parfaits pour la plupart des utilisateurs. MySQL Binaries Compiled by MySQL AB . Il y a quelques choses que vous pouvez modifier pour rendre le binaire encore plus rapide, mais cela est réservé aux utilisateurs avancés. Comment la compilation et les liaisons affectent la vitesse de MySQL .

  Si la génération échoue et produit des erreurs disant que votre compilateur ou outil de liaison n'est pas capable de créer la librairie partagée `libmysqlclient.so.#` ( `'#'` étant un numéro de version), vous pouvez contourner ce problème en donnant l'option `−−disable−shared` à `configure` . Dans ce cas, `configure` ne générera pas de librairie partagée `libmysqlclient.so.#` .
- Vous pouvez configurer MySQL pour qu'il n'utilise pas les valeurs `DEFAULT` pour les colonnes non− `NULL` (c'est à dire, les colonnes qui ne peuvent être `NULL` ). Cela fait générer des erreurs aux requêtes `INSERT` si vous ne spécifiez pas explicitement une valeur pour toutes les colonnes non− `NULL` . Pour supprimer l'utilisation des valeurs par défaut, exécutez `configure` de cette façon :

```
shell> CXXFLAGS=−DDONT_USE_DEFAULT_FIELDS ./configure
```

- Par défaut, MySQL utilise le jeu de caractères ISO−8859−1 (Latin1). Pour changer le jeu par défaut, utilisez l'option `−−with−charset` :

```
shell> ./configure −−with−charset=CHARSET
```

`CHARSET` peut être l'un des `big5`, `cp1251`, `cp1257`, `czech`, `danish`, `dec8`, `dos`, `euc_kr`, `gb2312`, `gbk`, `german1`, `hebrew`, `hp8`, `hungarian`, `koi8_ru`, `koi8_ukr`, `latin1`, `latin2`, `sjis`, `swe7`, `tis620`, `ujis`, `usa7`, ou `win1251ukr`. Le jeu de caractères utilisé pour les données et le stockage .Si vous voulez convertir les caractères entre le serveur et le client, regardez du côté de la commande `SET CHARACTER SET`. Syntaxe de `SET`.

**Attention** : Si vous changez les jeux de caractères après avoir crée des tables, vous devrez exécuter `myisamchk -r -q --set-character-set=charset` sur chaque table. Vous index pourraient être stockés de manière incorrecte sinon. (Cela peut survenir si vous installez MySQL, créez quelques tables, puis reconfigurez MySQL pour qu'il utilise un jeu de caractères différent et le réinstallez.)

Avec l'option `--with-extra-charsets=LIST` vous pouvez définir les jeux de caractères additionnels à compiler dans le serveur.

Ici `LIST` est soit une liste de jeux de caractères séparés par des espaces, soit `complex` pour inclure tous les jeux de caractères ne pouvant être chargés dynamiquement, ou encore `all` pour inclure tous les jeux de caractères dans les binaires.

- Pour configurer MySQL avec le code de débogage, utilisez l'option `--with-debug` :
```
shell> ./configure --with-debug
```
Cela alloue un vérificateur d'allocation de mémoire qui peut trouver quelques erreurs et qui fournit des informations sur ce qui se produit. Déboguer un serveur MySQL .
- Si vos programmes clients utilisent les threads, vous avez besoin de compiler une version sûre pour les threads de la librairie du client MySQL avec l'option de configuration `--enable-thread-safe-client`. Cela créera une librairie `libmysqlclient_r` avec laquelle vous devez lier vos applications threadées. How to Make a Threaded Client .
- Les options relatives à un système d'exploitation particulier peuvent être trouvées dans la section spécifique aux systèmes de ce manuel. Remarques spécifiques aux systèmes d'exploitation .

## 2.3.4 Installer à partir de l'arbre source de développement

**Attention** : Vous devez lire cette partie seulement si vous voulez nous aider à tester notre nouveau code. Si vous souhaitez seulement faire fonctionner MySQL sur votre système, vous devriez utiliser la distribution d'une version standard (que ce soit une distribution sous forme de sources ou de binaire).

Pour obtenir notre arbre source de développement le plus réent, suivez les instructions suivantes :

- Téléchargez `BitKeeper` à partir de http://www.bitmover.com/cgi-bin/download.cgi . Vous aurez besoin de `Bitkeeper` 2.0 ou supérieur pour accéder à notre dépôt.
- Suivez les instructions pour l'installer.
- Après avoir installé `BitKeeper`, commencez par vous déplacer dans le répertoire à partir duquel vous voulez travailler, et lancez l'une des commandes suivantes pour dupliquer la branche MySQL de votre choix :Pour dupliquer la branche 3.23, utilisez cette commande :
```
shell> bk clone bk://work.mysql.com:7000 mysql-3.23
```
Pour dupliquer la branche 4.0, utilisez cette commande :
```
shell> bk clone bk://work.mysql.com:7001 mysql-4.0
```
Pour dupliquer la branche 4.1, utilisez cette commande :
```
shell> bk clone bk://work.mysql.com:7004 mysql-4.1
```
Dans les exemples précédents, l'arbre source sera déposé dans les sous-répertoires `mysql-3.23/`, `mysql-4.0/`, ou `mysql-4.1/` de votre répertoire courant.Le premier téléchargement de l'arbre source peut prendre un certain temps, selon la vitesse de votre connexion. Soyez patients.

- Vous aurez besoin de GNU `autoconf 2.52` , de `automake 1.5` , de `libtool 1.4` , et de `m4` pour lancer la prochaine série de commandes.

```
shell> cd mysql-4.0
shell> bk -r get -Sq
shell> aclocal; autoheader; autoconf;  automake;
shell> ./configure  # Ajoutez ici vos options favorites
shell> make
```

  Si vous obtenez des erreurs étranges pendant cette étape, vérifiez bien que vous avez vraiment installé `libtool` !

  Une collection de nos scripts de configuration les plus courants de trouve dans le sous-répertoire `BUILD/` . Si vous êtes fainéants, vous pouvez utiliser `BUILD/compile-pentium-debug` . Pour compiler sur une architecture différente, modifiez ce script en enlevant les drapeaux spécifiques au Pentium.
- Quand la compilation est achevée, lancez `make install` . Prenez garde sur des machines de production. Cette commande pourrait écraser votre installation actuelle. Si vous avez une autre installation de MySQL, nous vous recommandons de lancer `./configure` avec des valeurs des options `prefix` , `with-tcp-port` , et `unix-socket-path` différentes de celles de votre serveur de production.
- Torturez votre nouvelle installation et tentez de faire planter les nouvelles fonctionnalités. Commencez par lancer `make test` . Suites de tests MySQL .
- Si vous avez échoué avec l'étape `make` et que la distribution ne compile pas, envoyez un rapport de l'incident à bugs@lists.mysql.com . Si vous avez installé la dernière version des indispensables outils GNU, et qu'ils échouent dans l'analyse de vos fichiers de configuration, envoyez aussi un rapport. D'autre part, si vous exécutez `aclocal` et que vous obtenez l'erreur `command not found` ou un problème du même type, n'envoyez pas de rapport. A la place, assurez vous que les outils nécessaires sont bien installés et que votre variable `PATH` est configurée de telle façon que votre interpréteur de commandes les trouvent.
- Après la première opération `bk clone` pour obtenir l'arbre source, vous devez lancer régulièrement `bk pull` pour obtenir les mises à jour.
- Vous pouvez examiner l'historique des changements de l'arbre avec toutes les différences en utilisant `bk sccstool` . Si vous apercevez des différences anormales ou sur lesquelles vous avez des questions, n'hésitez pas à envoyer un e-mail à internals@lists.mysql.com . De même, si vous pensez avoir une meilleure méthode pour traiter un problème, envoyez un e-mail accompagné d'un patch à la même adresse. `bk diffs` vous fournira un patch après que vous ayez fait vos changements aux sources. Si vous n'avez pas le temps de coder votre idée, envoyez en juste une description.
- `BitKeeper` dispose d'une aide agréable à laquelle vous pouvez accéder via `bk helptool` .
- Veuillez noter que chaque commit ( `bk ci` ou `bk citool` ) postera un message avec un aperçu des changements à notre liste de diffusion interne, à la façon habituelle des propositions openlogging.org avec seulement les commentaires des changements. Généralement, vous n'aurez pas besoin d'utiliser commit (l'arbre public interdisant les `bk push` ), mais plutôt d'utiliser la méthode `bk diffs` décrite plus haut.

## 2.3.5 Problèmes de compilation?

Tous les programmes MySQL compilent proprement chez nous, sans aucune alerte sur Solaris avec `gcc` . Sur d'autres systèmes, des alertes peuvent apparaître à cause de différences dans le système d'inclusions. Voyez Remarques sur MIT-pthreads pour les alertes qui peuvent apparaître avec MIT-pthreads. Pour d'autres problèmes, voyez la liste suivante.

La solution à de nombreux problèmes implique une reconfiguration. Si vous avez besoin de faire une reconfiguration voici quelques conseils généraux :

- Si `configure` est exécuté après une première exécution, il peut utiliser des informations qui ont été rassemblées durant une première invocation. Ces informations sont stockées dans le fichier `config.cache` . Lorsque `configure` est lancé, il commence par regarder dans ce fichier, et lire le contenu qui existe, en supposant que ces données sont toujours correctes. Cette supposition est invalide si vous faîtes une reconfiguration.
- Chaque fois que vous exécutez `configure` , vous devez exécuter à nouveau `make` pour recompiler. Toutefois, vous devrez peut être supprimer les vieux fichiers d'objets qui ont été compilé en utilisant différentes configurations précédentes.

Pour éviter d'utiliser de vieilles informations de configuration, ou des vieux fichiers d'objet, vous pouvez utiliser ces commandes, avant `configure` :

```
 shell> rm config.cache
shell> make clean
```

Alternativement, vous pouvez aussi utiliser `make distclean` .La liste suivante décrit certains problèmes lors de la compilation de MySQL, qui surviennent souvent :

- Si vous avez des problèmes lors de la compilation de `sql_yacc.cc` , comme ceux qui sont décrits ci−dessous, vous avez probablement été à court de mémoire ou d'espace de swap :

  ```
   Internal compiler error: program cc1plus got fatal signal 11
    ou
  Out of virtual memory
    ou
  Virtual memory exhausted
  ```

  Le problème est que `gcc` requiert de grandes quantité de mémoire pour compiler `sql_yacc.cc` avec les fonctions inline. Essayez d'exécuter `configure` avec l'option `--with-low-memory` :

  ```
   shell> ./configure --with-low-memory
  ```

  Cette option ajoute `-fno-inline` dans la ligne de compilation, si vous utilisez `gcc` et `-O0` si vous utilisez autre chose. Vous pouvez essayer `--with-low-memory` même si il vous reste suffisamment de mémoire, et que vous ne pensez pas être limité. Ce problème a été observé sur des systèmes avec de généreuses configurations, et `--with-low-memory` résout ce problème.
- Par défaut, `configure` choisit `c++` comme compilateur, et GNU `c++` pour les liens avec `-lg++` . Si vous utilisez `gcc` , ce comportement peut poser les problèmes suivants :

  ```
   configure: error: installation or configuration problem:
  C++ compiler cannot create executables.
  ```

  Vous pourriez aussi observer des problèmes durant la compilation, avec `g++` , `libg++` ou `libstdc++` .La cause de ces problèmes est que vous avez peut être que vous n'avez pas `g++` , ou que vous avez `g++` mais pas `libg++` , ou `libstdc++` . Regardez le fichier de log `config.log` . Il va sûrement contenir la raison exacte du mauvais fonctionnement de votre compilateur. Pour contourner ce problème, vous pouvez utiliser `gcc` comme compilateur C++. Essayez de modifier la variable d'environnement `CXX` avec la valeur `"gcc -O3"` . Par exemple :

  ```
   shell> CXX="gcc -O3" ./configure
  ```

  Cela fonctionne car `gcc` compile les sources C++ aussi bien que `g++` , mais il n'est pas lié avec `libg++` ou `libstdc++` par défaut.Un autre moyen pour régler ces problèmes, bien sur, est d'installer `g++` , `libg++` et `libstdc++` . Nous vous recommandons toutefois de ne pas utiliser `libg++` ou `libstdc++` avec MySQL car cela ne fera qu'accroître la taille de votre exécutable binaire, sans vous apporter d'avantages. Par le passé, certaines versions de ces librairies ont posé des problèmes étranges aux utilisateurs MySQL.
- Si votre compilation échoue avec des erreurs, ou si l'une des erreurs suivantes apparaît, vous devez changer la version de `make` en GNU `make` :

  ```
   making all in mit-pthreads
  make: Fatal error in reader: Makefile, line 18:
  Badly formed macro assignment
    ou
  make: file `Makefile' line 18: Must be a separator (:
    ou
  pthread.h: No such file or directory
  ```

  Solaris et FreeBSD sont connus pour avoir des problèmes avec `make` .

  GNU `make` version 3.75 est reconnu pour fonctionner.

- Si vous voulez définir des options supplémentaires qui seront utilisées par votre compilateur C ou C++, faites le en ajoutant ces options aux variables d'environnement `CFLAGS` et `CXXFLAGS` . Vous pouvez aussi spécifier le nom du compilateur via les variables `CC` et `CXX` . Par exemple :

```
shell> CC=gcc
shell> CFLAGS=-O3
shell> CXX=gcc
shell> CXXFLAGS=-O3
shell> export CC CFLAGS CXX CXXFLAGS
```

  Voyez MySQL Binaries Compiled by MySQL AB , pour avoir une liste des définitions des options disponibles sur divers systèmes.
- Si vous obtenez un message d'erreur comme celui-ci, vous devrez mettre à jour votre version de `gcc` :

```
client/libmysql.c:273: parse error before `__attribute__'
```

  `gcc` 2.8.1 est connu pour fonctionner, mais nous recommandons l'utilisation de `gcc` 2.95.2 ou `egcs` 1.0.3a.
- Si vous obtenez des erreurs telles que celles qui sont affichées ci-dessous lors de la compilation de `mysqld` , c'est que `configure` n'a pas correctement détecter le dernier argument des fonctions `accept()` , `getsockname()` ou `getpeername()` :

```
cxx: Error: mysqld.cc, line 645: In this statement, the referenced
    type of the pointer value "&length" is "unsigned long", which
    is not compatible with "int".
new_sock = accept(sock, (struct sockaddr *)&cAddr, &length);
```

  Pour corriger ce problème, éditez le fichier `config.h` (qui est généré par le fichier `configure` ). Recherchez ces lignes :

```
/* Define as the base type of the last arg to accept */
#define SOCKET_SIZE_TYPE XXX
```

  Remplacez `XXX` par `size_t` ou `int` , suivant votre système d'exploitation. Notez que vous devrez faire cette manipulation à chaque fois que vous exécuterez le script `configure` car `configure` regénère `config.h` .
- Le fichier `sql_yacc.cc` est généré à partir du fichier `sql_yacc.yy` . Normalement, le processus de création ne s'occupe pas de `sql_yacc.cc` , car MySQL en a déjà une copie. Cependant, si vous devez le recréer, vous pouvez rencontrer cette erreur :

```
"sql_yacc.yy", line xxx fatal: default action causes potential...
```

  Cela indique que votre version de `yacc` est inadéquate. Vous devrez probablement réinstaller `bison` (la version GNU de `yacc` ) et l'utiliser à la place.
- Si vous avez besoin de déboguer `mysqld` ou un client MySQL, exécutez le script `configure` avec l'option `--with-debug` , puis recompilez vos clients avec la nouvelle librairie. Déboguer un client MySQL .

## 2.3.6 Notes relatives aux MIT-pthreads

Cette section décrit quelques informations concernant l'utilisation des MIT-pthreads.

Notez que sur Linux vous **ne** devez **pas** utiliser les MIT-pthreads mais installer LinuxThreads ! Notes relatives à Linux (toutes les versions de Linux) .

Si votre système ne fournit pas un support natif des threads, vous aurez besoin de construire MySQL en utilisant le package des MIT-pthreads. Cela inclut les anciens systèmes FreeBSD, SunOS 4.x, Solaris 2.4 et plus ancien, et quelques autres systèmes. Operating Systems Supported by MySQL .

Notez qu'à partir de la version 4.0.2 de MySQL les MIT-pthreads ne font plus partie de la distribution des sources ! si vous avez besoin de ce package, vous pouvez l'obtenir sur http://www.mysql.com/Downloads/Contrib/pthreads-1_60_beta6-mysql.tar.gz

Après l'avoir récupéré, décompressez l'archive dans le répertoire racine de votre répertoire des sources de MySQL. Cela créera le répertoire `mit-pthreads` .

- Sur la plupart des systèmes, vous pouvez forcer l'utilisation des MIT−pthreads en exécutant `configure` avec l'option `--with-mit-threads` :

  ```
  shell> ./configure --with-mit-threads
  ```

  La compilation dans un dossier non−sources n'est pas supporté lors de l'utilisation des MIT−pthreads car nous voulons minimiser les changements de leur code.
- La vérification pour l'utilisation des MIT−pthreads ne survient que durant la partie du processus de configuration qui s'occupe du code du serveur. Si vous avez configuré la distribution en utilisant `--without-server` pour ne construire que le client, les clients ne sauront pas si les MIT−pthreads sont utilisés et utiliserons les socket Unix pour les connexions par défaut. Puisque les sockets Unix ne fonctionnent pas avec les MIT−pthreads sur certaines plate−formes, cela signifie que vous devrez utiliser `-h` ou `--host` quand vous exécuterez les programmes clients.
- Lorsque MySQL est compilé en utilisant les MIT−pthreads, le verrouillage système est désactivé par défaut pour des soucis de performances. Vous pouvez demander au serveur d'utiliser les verrous systèmes avec l'option `--external-locking` . Cela n'est requis que si vous avez besoin de faire fonctionner deux serveurs MySQL avec les mêmes données (non recommandé).
- De temps en temps, la commande `bind()` des pthreads n'arrive pas à attacher une socket sans afficher d'erreurs (du mois, sous Solaris). Le résultat est que toutes les connexions au serveur échouent. Par exemple :

  ```
  shell> mysqladmin version
  mysqladmin: connect to server at '' failed;
  error: 'Can't connect to mysql server on localhost (146)'
  ```

  La solution est de terminer le serveur `mysqld` et de le redémarrer. Cela ne nous est arrivé que quand nous avons forcé le serveur à se terminer et que nous l'avons redémarré immédiatement après.
- Avec les MIT−pthreads, l'appel système à `sleep()` n'est pas interruptible avec `SIGINT` (break). On ne s'en rend compte que quand on exécute `mysqladmin --sleep` . Vous devez attendre que l'appel système à `sleep()` se termine avant que le processus ne s'arrête.
- Lors de la liaison, vous pouvez obtenir des messages d'erreurs comme ceux−ci (du moins sur Solaris); ils peuvent être ignorés :

  ```
  ld: warning: symbol `_iob' has differing sizes:
      (file /my/local/pthreads/lib/libpthread.a(findfp.o) value=0x4;
  file /usr/lib/libc.so value=0x140);
      /my/local/pthreads/lib/libpthread.a(findfp.o) definition taken
  ld: warning: symbol `__iob' has differing sizes:
      (file /my/local/pthreads/lib/libpthread.a(findfp.o) value=0x4;
  file /usr/lib/libc.so value=0x140);
      /my/local/pthreads/lib/libpthread.a(findfp.o) definition taken
  ```
- D'autres avertissements peuvent être ignorés :

  ```
  implicit declaration of function `int strtoll(...)'
  implicit declaration of function `int strtoul(...)'
  ```
- Nous n'avons pas réussi à faire fonctionner `readline` avec les MIT−pthreads. (Cela n'est pas nécessaire, mais peut être utile à quelqu'un.)

## 2.3.7 La distribution source Windows

Vous aurez besoin de :

- Un compilateur VC++ 6.0 (mis à jour avec 4 ou 5 SP et le paquet Pré−processeur) Le paquet Pré−processeur est nécessaire pour l'assembleur de macros. D'avantage de détails sur http://msdn.microsoft.com/vstudio/sp/vs6sp5/faq.asp .
- La distribution source MySQL pour Windows, qui peut être obtenue sur http://www.mysql.com/downloads/ .

Compiler MySQL

- Créez un répertoire de travail (par exemple, workdir)

- Décompressez la distribution source dans ce répertoire.
- Lancez le compilateur VC++ 6.0.
- Dans le menu `Fichier` , sélectionnez `Ouvrir un espace de travail` .
- Ouvrez l'espace de travail `mysql.dsw` que vous trouverez dans le répertoire de travail.
- Dans le menu `Générer` , sélectionnez le menu `Choisir la configuration active` .
- Choisissez `mysqld - Win32 Debug` puis cliquez sur OK.
- Appuyez sur `F7` pour commencer à compiler le serveur de débogage, les bibliothèques, et quelques applications clientes.
- Quand la compilation est terminée, copiez les bibliothèques et les exécutables dans un répertoire séparé.
- Compilez les versions que vous souhaitez de la même manière.
- Créez le répertoire pour les outils MySQL : par exemple, `c:\mysql`
- A partir du répertoire workdir, copiez les répertoires suivants dans c:\mysql :
    - Data
    - Docs
    - Share
- Créez le répertoire `c:\mysql\bin` et copier toutes les applications serveur et client compilées auparavant.
- Si vous le souhaitez, vous pouvez aussi créer le répertoire `lib` et y copier les librairies compilées auparavant.
- Faites un nettoyage en utilisant Visual Studio.

Configurez et lancez le serveur de la même façon que la distribution binaire Windows. Préparer l'environnement MySQL sous Windows .

## 2.4 Configuration après l'installation, et tests

Une fois que vous avez installé MySQL (que ce soit avec les sources ou avec la distribution binaire), vous devez initialiser les tables de droits, démarrer le serveur et vous assurer que tout fonctionne bien. Vous pouvez aussi vouloir configurer le démarrage et l'extinction automatique du serveur.

Normalement, vous installez les tables de droits et démarrez le serveur comme ceci, pour une distribution source :

```
shell> ./scripts/mysql_install_db
shell> cd mysql_installation_directory
shell> ./bin/safe_mysqld --user=mysql &
```

Pour une distribution binaire (pas un RPM ou un package), faites ceci :

```
shell> cd mysql_installation_directory
shell> ./bin/mysql_install_db
shell> ./bin/safe_mysqld --user=mysql &
```

Cela crée la base `mysql` qui va contenir tous les droits, ainsi que la base test, que vous pouvez utiliser pour tester MySQL, ainsi que les droits pour l'utilisateur qui fait fonctionner `mysql_install_db` et un utilisateur `root` (sans mots de passe). Cela va aussi démarrer le serveur `mysqld` .

`mysql_install_db` ne va pas écraser d'anciens droits installé, et il peut être utilisé en toutes circonstances. Si vous ne voulez pas de base `test` , vous pouvez la supprimer avec la commande `mysqladmin -u root drop test` .

Les tests sont plus facilement exécuté depuis le niveau supérieur de la distribution MySQL. Pour la distribution binaire, c'est votre dossier d'installation (typiquement, `/usr/local/mysql` ). Pour une distribution source, c'est le dossier principal de votre dossier source MySQL.

Dans les commandes présentées dans cette section et dans les sous sections suivantes, `BINDIR` est le chemin jusqu'au programmes binaires de la distribution, comme `mysqladmin` et `safe_mysqld` . Pour une distribution binaire, il s'agit du dossier `bin` dans la distribution. Pour une distribution source, `BINDIR` vaut sûrement `/usr/local/bin` , à moins que vous ne spécifiez un dossier d'installation différent que `/usr/local` avec le script `configure` . `EXECDIR` est le chemin jusqu'au serveur `mysqld` . Pour une distribution binaire, c'est la même valeur que `BINDIR` . Pour une distribution source, `EXECDIR` est probablement `/usr/local/libexec` .

Les tests sont décrits en détails :

- Si besoin, démarrez le serveur `mysqld` et modifiez les tables initiales de droits de MySQL qui contiennent les privilèges qui déterminent les droits de connexion des utilisateurs au serveur. Cela se fait normalement avec le script `mysql_install_db` :

  ```
  shell> scripts/mysql_install_db
  ```

  Typiquement, `mysql_install_db` doit être exécuté uniquement lors de votre première installation de MySQL. Par conséquent, si vous faites évoluer une installation, vous pouvez éviter cette étape. Cependant, `mysql_install_db` est tout à fait sécuritaire, et vous ne modifiera pas des tables existants, alors dans le doute, vous pouvez toujours utiliser `mysql_install_db` .

  `mysql_install_db` crée six tables ( `user` , `db` , `host` , `tables_priv` , `columns_priv` , and `func` ) dans la base `mysql` database. Une description des droits initiaux est proposé dans la section Mise en place des privilèges initiaux de MySQL . En bref, ces droits permettront au `root` MySQL de faire ce qu'il souhaite, et notamment, d'autoriser la création et l'utilisation de la base `test` ou dont le nom commence par `test_` .

  Si vous ne configurez pas les tables de droits, l'erreur suivante va apparaître dans le fichier de log, au démarrage du serveur :

  ```
  mysqld: Can't find file: 'host.frm'
  ```

  Cela peut aussi arriver avec la distribution binaire de MySQL, si vous ne démarrez pas MySQL exactement avec `./bin/safe_mysqld` ! `safe_mysqld` , le script père de `mysqld` .Vous pouvez avoir besoin d'exécuter `mysql_install_db` en tant que `root` . Toutefois, si vous pouvez exécuter le serveur MySQL en tant qu'utilisateur avec peu de droits (non−root), tant que cet utilisateur a les droits de lecture et d'écriture dans le dossier de données. Les instructions pour faire fonctionner MySQL avec un autre utilisateur sont données dans le chapitre Comment faire tourner MySQL en tant qu'utilisateur normal .

  Si vous avez des problèmes avec `mysql_install_db` , voyez Problèmes d'exécution de `mysql_install_db` .

  Il y des alternatives à l'utilisation de `mysql_install_db` tel qu'il est fourni dans la distribution MySQL :

  - Vous pouvez éditer le script `mysql_install_db` avant de l'exécuter, pour modifier les droits initiaux qui sont implantés dans les tables de droits. C'est pratique si vous voulez installer MySQL sur de nombreuses machines, avec les mêmes droits. Dans ce cas, vous aurez probablement à ajouter quelques insertions `INSERT` dans les tables `mysql.user` et `mysql.db` !
  - Si vous voulez modifier des éléments dans la table de droits après les avoir installé, vous pouvez exécuter `mysql_install_db` , puis utiliser `mysql −u root mysql` pour vous connecter aux tables de droit en tant que `root` MySQL, et exécuter les commandes SQL nécessaires, directement dans les tables de droits.
  - Il est possible de recréer complètement les tables de droits une fois qu'elles ont été créées. Vous pouvez faire cela si vous avez déjà installé les tables, mais que vous voulez les recréer après édition de `mysql_install_db` .

Pour plus d'informations sur ces alternatives, voyez Mise en place des privilèges initiaux de MySQL .

- Démarrez le serveur MySQL comme ceci :

```
shell> cd mysql_installation_directory
shell> bin/safe_mysqld &
```

Si vous avez des problèmes avec le démarrage du serveur, voyez Problèmes de démarrage de MySQL .

- Utilisez `mysqladmin` pour vérifier que le serveur fonctionne. Les commandes suivantes fournissent un accès simple pour tester si le serveur fonctionne correctement :

```
shell> BINDIR/mysqladmin version
shell> BINDIR/mysqladmin variables
```

Le résultat de `mysqladmin version` varie suivant votre plate−forme et votre version de MySQL, mais il doit être similaire à ceci :

```
shell> BINDIR/mysqladmin version
mysqladmin  Ver 8.14 Distrib 3.23.32, for linux on i586
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license.

Server version          3.23.32−debug
Protocol version        10
Connection              Localhost via Unix socket
TCP port                3306
UNIX socket             /tmp/mysql.sock
Uptime:                 16 sec

Threads: 1  Questions: 9  Slow queries: 0
Opens: 7  Flush tables: 2  Open tables: 0
Queries per second avg: 0.000
Memory in use: 132K  Max memory used: 16773K
```

Pour savoir ce que vous pouvez faire d'autre avec `BINDIR/mysqladmin` , appelez le avec l'option `−−help` .

- Vérifiez que vous pouvez éteindre le serveur :

```
shell> BINDIR/mysqladmin −u root shutdown
```

- Verifiez que vous pouvez redémarrer le serveur. Faites cela avec `safe_mysqld` ou directement avec `mysqld` . Par exemple :

```
shell> BINDIR/safe_mysqld −−log &
```

Si `safe_mysqld` échoue, essayez d'exécuter MySQL depuis le dossier d'installation (si vous n'y êtes pas déjà). Si cela ne fonctionne pas, voyez Problèmes de démarrage de MySQL .

- Exécutez un test simple pour vérifier que le serveur fonctionne. Le résultat devrait être similaire à ces lignes ci :

```
shell> BINDIR/mysqlshow
+−−−−−−−−−−−+
| Databases |
+−−−−−−−−−−−+
| mysql     |
+−−−−−−−−−−−+

shell> BINDIR/mysqlshow mysql
Database: mysql
+−−−−−−−−−−−−−−+
|    Tables    |
+−−−−−−−−−−−−−−+
| columns_priv |
| db           |
| func         |
| host         |
| tables_priv  |
| user         |
+−−−−−−−−−−−−−−+

shell> BINDIR/mysql −e "SELECT host,db,user FROM db" mysql
+−−−−−−+−−−−−−−−+−−−−−−+
| host | db     | user |
+−−−−−−+−−−−−−−−+−−−−−−+
| %    | test   |      |
| %    | test_% |      |
+−−−−−−+−−−−−−−−+−−−−−−+
```

Il y a aussi la suite de performances dans le dossier `sql−bench` (dans le dossier d'installation MySQL) que vous pouvez utiliser pour comparer les performances de MySQL avec d'autres plate−formes. Le dossier `sql−bench/Results` contient le résultat de nombreux tests avec différentes bases et plate−forme. Pour exécuter tous les tests, exécutez cette commande :

```
shell> cd sql-bench
shell> run-all-tests
```

Si vous n'avez pas de dossier `sql-bench` , vous utilisez probablement une distribution binaire au format RPM. Les distribution source RPM incluent le dossier de tests. Dans ce cas, vous devez d'abord installer la suite de test avant de l'utiliser. Depuis MySQL Version 3.22, il y a des fichiers de tests RPM, appelé `mysql-bench-VERSION-i386.rpm` qui contient les données de tests et les données.Si vous avez une distribution source, vous pouvez aussi exécuter les tests dans le dossier `tests` . Par exemple, pour exécuter `auto_increment.tst` , faites ceci :

```
shell> BINDIR/mysql -vvf test < ./tests/auto_increment.tst
```

Les résultats attendus sont présentés dans le fichier `./tests/auto_increment.res` .

## 2.4.1 Problèmes d'exécution de mysql_install_db

Le but du script `mysql_install_db` est de générer un nouveau système de droits pour MySQL. Il ne modifiera aucune autre donnée! Il ne fera rien du tout si vous avez des tables de droits installées.

Si vous voulez recréer vos tables de droits, vous devez éteindre le serveur `mysqld` , si il fonctionnait, et faire ceci :

```
mv mysql-data-directory/mysql mysql-data-directory/mysql-old
mysql_install_db
```

Cette section liste les problèmes que vous pourriez rencontrer lors de l'exécution du script `mysql_install_db` :

`mysql_install_db` *n'installe pas les tables de droits*

Vous réalisez que `mysql_install_db` n'arrive pas à installer les tables de droits, et se termine sur ce message :

```
starting mysqld daemon with databases from XXXXXX
mysql daemon ended
```

Dans ce cas, examinez le fichier de log très attentivement! Le fichier de log est situé dans le dossier `XXXXXX` indiqué dans le message d'erreur, et il indiquera pourquoi `mysqld` n'a pas démarré. SI vous ne comprenez pas ce qui est arrivé, incluez le log dans votre message, lors de l'envoi du rapport de bugs avec `mysqlbug` ! Comment rapporter des bogues ou problèmes .

*Un démon* `mysqld` *fonctionne déjà*

Dans ce cas, vous n'avez probablement pas exécuté `mysql_install_db` du tout. Vous avez exécuté `mysql_install_db` une fois, lorsque vous avez installé MySQL pour la première fois.

*Installer un second démon* `mysqld` *n'est pas possible lorsque le premier fonctionne.*

Cela arrive lorsque vous avez une installation MySQL pré-existantes, mais que vous voulez installer une autre version ailleurs (par exemple, pour faire des tests ou simplement pour avoir deux installations). Généralement, le problème survient lorsque le second serveur est démarré, et qu'il essaie d'utiliser les mêmes ports et sockets que le premier. Dans ce cas, vous recevez des message d'erreur tels que : `Can't start server: Bind on TCP/IP port: Address already in use` ou `Can't start server: Bind on unix socket...` .

Installer plusieurs serveurs sur la même machine .

*You don't have write access to* `/tmp`

Si vous n'avez pas les droits d'accès suffisant pour créer un fichier de socket à l'endroit prévu (dans `/tmp` ) ou les permissions pour créer un fichier temporaire dans `/tmp,` vous allez avoir une erreur lors de l'utilisation de `mysql_install_db` ou avec `mysqld` .

Vous pouvez spécifier une socket différente et un dossier temporaire différent avec les options suivantes :

```
shell> TMPDIR=/some_tmp_dir/
shell> MYSQL_UNIX_PORT=/some_tmp_dir/mysqld.sock
shell> export TMPDIR MYSQL_UNIX_PORT
```

Voir Comment protéger `/tmp/mysql.sock` .

`some_tmp_dir` doit être un chemin vers un dossier dans lequel vous avez les permissions d'écriture. Variables d'environnement MySQL .

Après cela, vous devriez être capable d'exécuter `mysql_install_db` et démarrer le serveur avec ces commandes :

```
shell> scripts/mysql_install_db
shell> BINDIR/safe_mysqld &
```

`mysqld` *crashes immediately*

Si vous utilisez RedHat version 5.0 avec une version de `glibc` plus ancienne que 2.0.7−5, assurez vous que vous avez installé les patchs `glibc` ! Il y a beaucoup de littérature à ce propos dans les archives de listes de diffusion. Des liens vers les archives sont disponibles à http://lists.mysql.com/ . Voir aussi Notes relatives à Linux (toutes les versions de Linux) .

Vous pouvez aussi démarrer `mysqld` manuellement avec l'option `--skip-grant-tables` puis ajouter les tables de droits avec S `mysql` :

```
shell> BINDIR/safe_mysqld --skip-grant-tables &
shell> BINDIR/mysql -u root mysql
```

Dans la base `mysql` , exécutez manuellement les commandes SQL disponibles dans `mysql_install_db` . Assurez vous que vous n'oubliez pas la commande `mysqladmin flush-privileges` ou `mysqladmin reload` pour dire au serveur de prendre en compte les tables de droits.

## 2.4.2 Problèmes de démarrage du serveur MySQL

Si vous allez utiliser des tables qui supportent les transactions (InnoDB, BDB), vous devez commencer par créer un fichier `my.cnf` et y placer les options de démarrage de ces tables. Types de tables MySQL .

Généralement, vous démarrez le serveur `mysqld` de l'une de ces façons :

- En utilisant le script `mysql.server` . Ce script est utilisé à la base au moment du démarrage et de l'extinction du système, et il est décrit avec plus de détail dans le chapitre Démarrer et arrêter MySQL automatiquement .
- En appelant le script `safe_mysqld` , qui va déterminer les options correctes pour `mysqld` puis utiliser ces options. `safe_mysqld` , le script père de `mysqld` .
- Pour Windows NT/2000/XP, voyez Démarrer MySQL avec Windows NT / 2000 / XP .
- En appelant `mysqld` directement.

Lorsque le démon `mysqld` démarre, il change le dossier de travail par le dossier de données. C'est là qu'il doit trouver les fichiers de log, et le fichier pid (ID de processus), ainsi que les dossiers de bases.

Le chemin du dossier de données est codé en dur lorsque la distribution est compilée. Cependant, si `mysqld` cherche le dossier de données ailleurs que là ou il est vraiment, il ne va pas fonctionner correctement. Si vous avez des problèmes avec les chemins, vous pouvez utiliser les options dont

`mysqld` dispose pour vous permettre de modifier dynamiquement le chemin du dossier de données : il suffit d'appeler `mysqld` avec l'option `--help` . Vous pouvez remplacer les valeurs par défaut en spécifiant les chemins corrects en ligne de commande avec `mysqld` . Ces options fonctionneront aussi avec `safe_mysqld` .

Normalement, vous devez appeler `mysqld` uniquement depuis le dossier d'installation de MySQL. Vous pouvez faire cela avec l'option `--basedir` . Vous pouvez aussi consulter l'affichage de `--help` pour vérifier le changement des options de chemin (notez que `--help` **doit** être l'option finale d'une commande `mysqld` . Par exemple :

```
shell> EXECDIR/mysqld --basedir=/usr/local --help
```

Une fois que vous avez déterminé le chemin que vous souhaitez, vous pouvez démarrer le serveur sans l'option `--help` .Quelque soit la méthode que vous utilisez pour démarrer le serveur, si elle échoue, vérifiez le fichier de log pour savoir pourquoi. Les fichiers de log sont situés dans le dossier de données (typiquement `/usr/local/mysql/data` pour une distribution binaire, `/usr/local/var` pour une distribution source, et `\mysql\data\mysql.err` sous Windows). Regardez dans le dossier de données et recherchez des fichiers de la forme `host_name.err` et `host_name.log` ou `host_name` est le nom de votre serveur. Vérifiez alors les dernières lignes de ce fichier :

```
shell> tail host_name.err
shell> tail host_name.log
```

Recherchez des lignes comme celles-ci :

```
000729 14:50:10  bdb:  Recovery function for LSN 1 27595 failed
000729 14:50:10  bdb:  warning: ./test/t1.db: No such file or directory
000729 14:50:10  Can't init databases
```

Cela signifie que vous n'avez pas démarré `mysqld` avec `--bdb-no-recover` et Berkeley DB a trouvé une erreur dans les fichiers de log lorsqu'il a essayé de restaurer votre base. Pour pouvoir continuer, vous devez déplacer le vieux fichier de log Berkeley DB vers un autre dossier, pour l'examiner plus tard. Les fichiers de logs sont nommés `log.0000000001` , et ce nombre augmente au fil du temps.Si vous exécutez `mysqld` avec les tables BDB et que `mysqld` fait des core dumps au démarrage, c'est peut être que vous avez des problèmes avec le fichier de restauration de BDB. Dans ce cas, essayez de démarrer `mysqld` avec `--bdb-no-recover` . Si cela aide, vous devriez alors retirer tous les fichiers de log `log.*` du dossier de données, et démarrer `mysqld` à nouveau.

Si vous obtenez l'erreur suivant, cela signifie que d'autres programmes (ou un autre serveur `mysqld` ) fonctionne déjà avec le port TCP/IP ou la socket que `mysqld` essaie d'utiliser :

```
Can't start server: Bind on TCP/IP port: Address already in use
  ou
Can't start server : Bind on unix socket...
```

Utilisez `ps` pour vous assurer que vous n'avez pas d'autre serveur `mysqld` qui fonctionne. SI vous ne pouvez pas trouver d'autre serveur en fonctionnement, essayer d'exécuter la commande `telnet votre-nom-d-hote numero-de-port-tcp` puis pressez la touche 'Entrée' plusieurs fois. SI vous n'obtenez pas de message d'erreur comme `telnet: Unable to connect to remote host: Connection refused` , alors un autre processus utilise le port TCP/IP de `mysqld` . Voyez Problèmes d'exécution de `mysql_install_db` et Faire fonctionner plusieurs serveurs MySQL sur la même machine .

Si `mysqld` est en fonctionnement, vous pouvez connaître les chemins qu'il utilise avec la commande suivante :

```
shell> mysqladmin variables
```

ou

```
shell> mysqladmin -h 'your-host-name' variables
```

Si vous obtenez une erreur `Errcode 13` , qui indique `Permission denied` , lors du démarrage de `mysqld` , cela signifie que vous n'avez pas les droits de lecture ou d'écriture sur le dossier de données de MySQL, ou dans le dossier de logs. Dans ce cas, vous devriez démarrer `mysqld` en tant que root, ou changer

les permissions des fichiers et dossiers utilisés.Si safe_mysqld démarre le serveur, mais que vous n'arrivez pas à vous y connecter, vous devriez vous assurer que vous avez une entrée dans le fichier /etc/hosts qui ressemble à ceci :

```
127.0.0.1       localhost
```

Ce problème survient uniquement sur les systèmes qui n'ont pas une librairie de threads fonctionnels, ou pour lesquels MySQL a été configuré pour utiliser les MIT−pthreads.SI vous ne pouvez pas faire démarrer mysqld , essayez de faire un fichier de trace pour identifier le problème. Créer des fichiers de traçage .

Si vous utilisez les tables InnoDB , reportez vous aux options de configuration spécifiques à InnoDB . Options de démarrage InnoDB .

Si vous utilisez les tables BDB (Berkeley DB), vous devriez vous familiariser avec les options de démarrage spécifiques à BDB. Options de démarrage BDB .

## 2.4.3 Lancer et arrêter MySQL automatiquement.

Les scripts mysql.server et safe_mysqld peuvent être utilisés pour démarrer le serveur automatiquement au moment du démarrage du serveur. mysql.server peut aussi servir à arrêter le serveur.

Le script mysql.server peut servir à démarrer ou arrêter le serveur en l'appelant avec les arguments start ou stop :

```
shell> mysql.server start
shell> mysql.server stop
```

mysql.server est installé dans le dossier share/mysql du dossier d'installation de MySQL, ou dans le dossier support−files de la distribution source.Avant que mysql.server ne démarre le serveur, il change de dossier pour aller dans le dossier d'installation et appelle safe_mysqld . Vous pourriez avoir à éditer mysql.server si vous avez une installation binaire dans une situation non standard. Modifiez la commande cd avec le dossier correct, avant qu'il n'exécute safe_mysqld . Si vous voulez que le serveur fonctionne avec un utilisateur spécifique, ajouter l'option user appropriée dans le fichier /etc/my.cnf , tel que présenté ultérieurement dans cette section.

mysql.server stop arrête le serveur en lui envoyant un signal. Vous pouvez éteindre le serveur manuellement avec la commande mysqladmin shutdown .

Vous pouvez ajouter ces commandes de démarrage et d'arrêt aux endroits appropriés dans votre fichier /etc/rc* lorsque vous démarrez MySQL dans les applications de production. Notez que si vous modifiez mysql.server , et que vous passez à une nouvelle version de MySQL, votre vesrion modifiée sera écrasée, et vous devriez faire une copie de sauvegarde de votre script.Si votre système utilise /etc/rc.local pour démarrer des scripts externes, vous devriez ajouter la ligne suivante :

```
/bin/sh −c 'cd /usr/local/mysql ; ./bin/safe_mysqld −−user=mysql &'
```

Vous pouvez aussi ajouter des options à mysql.server via le fichier global /etc/my.cnf file. Un fichier /etc/my.cnf typique peut ressembler à ceci :

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql

[mysql.server]
basedir=/usr/local/mysql
```

Le script mysql.server comprend les options suivantes : datadir , basedir et pid−file .

La table suivante montre quels groupes d'options chaque scripts de démarrage utilise :

| Script | Groupe d'options |
|---|---|
| mysqld | mysqld and server |
| mysql.server | mysql.server , mysqld , and server |
| safe_mysqld | mysql.server , mysqld , and server |

Fichier d'options `my.cnf` .

# 2.5 Changer de version de MySQL

Vous pouvez toujours les fichiers de structures et de données entre les différentes versions de MySQL. La version de base actuelle est la version 3. Si vous changez le jeu de caractères lors de l'utilisation de MySQL (ce qui va aussi affecter le tri), vous devez exécuter la commande `myisamchk -r -q --set-character-set=charset` sur toutes les tables. Sinon, vos index ne seront pas correctement triés.

Si vous avez peur des nouvelles versions, vous pouvez toujours renommer votre vieux `mysqld` avec un nom comme `mysqld-ancienne_version` . Si votre nouveau serveur `mysqld` se comportement bizarrement, vous pourrez toujours l'éteindre, et redémarrer avec votre vieux `mysqld` !

Lorsque vous faites une évolution de version, vous devriez toujorus faire une sauvegarde de vos anciennes données.

Si après un changement de version, vous rencontrez des problèmes avec les clients recompilés, comme `Commands out of sync` ou des core dumps inopinés, vous avez probablement utiliser un vieux fichier d'entête ou une vieille librairie lors de la compilation de vos programmes. Dans ce cas, vérifiez la date de votre fichier `mysql.h` , et de votre librairie `libmysqlclient.a` , pour vous assurer qu'ils proviennent bien de la nouvelle distribution MySQL. Si ce n'est pas le cas, recompilez vos programmes!

Si vous avez des problèmes comme si le nouveau serveur `mysqld` qui ne veut pas démarrer, ou que vous ne pouvez pas vous connecter sans un mot de passe, vérifiez que vous n'avez pas un vieux fichier `my.cnf` dans votre installation! Vous pouvez le vérifier comme ceci : `program-name --print-defaults` . Si cette commande affiche autre chose que le nom du programme, vous avez un fichier `my.cnf` actif, qui perturbe vos opérations.

C'est une bonne idée que de reconstruire et reinstaller le module `Msql-Mysql` à chaque fois que vous faites une nouvelle version de MySQL, en particulier si vous rencontrez des symptômes tels que les `DBI` qui font des core dump après votre mise à jour de MySQL.

## 2.5.1 Passer de la version 4.0 à la version 4.1

En général, voici ce que vous devez faire lorsque vous passez à la version 4.1, depuis une version plus ancienne :

- Exécutez le script `mysql_fix_privilege_tables` pour générer un nouveau champs de mot de passe, nécessaire à la gestion de ces derniers.

Voici une liste plus complète, vous indiquant ce à quoi vous devez faire attention lors du changement de version :

- Les fonctions qui retournent des DATE, DATETIME ou TIME sont désormais traitées lors de leur arrivée sur le client.

```
mysql> SELECT cast("2001-1-1" as DATE)
   -> '2001-01-01'
```

- Toutes les colonnes et tables ont désormais un jeu de caractère, qui apparait dans le résultat de la commande `SHOW CREATE TABLE` et `mysqldump` . (MySQL 4.0.6 et plus récent peuvent lire les nouveaux fichiers de dump, mais pas les plus anciennes versions de MySQL).
- Les `TIMESTAMP` sont désormais retournés sous la forme de chaîne, avec le format `'YYYY-MM-DD HH:MM:DD'` . Si vous voulez retrouver le nombre, ajoutez la valeur de 0 a la colonne timestamp. Différentes tailles de timestamp ne sont pas supportées.

  **Note** : Depuis la version 4.0.12, l'option `--new` est utilisable pour que le serveur se comporte comme la version 4.1. Cela peut permettre de préparer la mise à jour de 4.0 en 4.1.
- Si vous utilisez plusieurs serveurs sur la même machine Windows, vous devriez utiliser l'option `--shared_memory_base_name` avec des valeurs différentes sur toutes les machines.

**Notez** que le fichier de définition des tables (.frm) a été modifié légérement en 4.1. MySQL 4.0.11 peut désormais lire le nouveau format .frm mais les anciennes versions ne le peuvent pas. Si vous devez passer des tables en format 4.1 vers une version plus ancienne, vous devrez utiliser `mysqldump` . `mysqldump` , exporter les structures de tables et les données .

## 2.5.2 Préparer le passage de version 4.0 en 4.1

- En version 4.1, `TIMESTAMP` est retourné sous forme de chaîne, au format `'AAAA-MM-JJ HH:MM:SS'` . Si vous voulez l'obtenir sous forme de nombre (comme ne version 4.0), vous devrez ajouter +0 à la colonne de type timestamp. Différentes tailles de timestamp ne sont pas supportés. **Note** : Depuis la version 4.0.12, l'option `--new` peut être utilisée pour que le serveur se comporte comme en version 4.1. Cela vous aidera à préparer le passage de version 4.0 en 4.1.

## 2.5.3 Passer de la version 3.23 à la version 4.0

En général, ce que vous devez faire pour passer en version 4.0, à partir d'une version plus ancienne :

- Exécutez le script `mysql_fix_privilege_tables` pour ajouter de nouveaux droits et fonctionnalités à a table MySQL.
- Editez les scripts de démarrage MySQL pour les fichiers de configuration pour ne plus utiliser les options abandonnées, listées ci-dessous.
- Convertissez vos vieilles tables `ISAM` en tables `MyISAM` avec la commande : `mysql_convert_table_format database` . Notez que cela ne doit être fait que si toutes les tables de la base sont des tables `ISAM` ou `MyISAM` . Si ce n'est pas le cas, vous devrez alors utiliser la commande `ALTER TABLE table_name TYPE=MyISAM` sur toutes les tables `ISAM` .
- Assurez vous que vous n'avez pas de client MySQL qui utilise des librairies partagées (comme les modules Perl Msql-Mysql). Si vous en avez, vous devriez les recompiler car les structures utilisées dans `libmysqlclient.so` ont changées.

MySQL 4.0 va fonctionner même si vous ne suivez pas les instructions ci-dessus, mais il ne sera pas capable de profiter des nouveaux droits disponibles avec MySQL 4.0 et vous pourriez rencontrer des problèmes lors de l'évolution vers MySQL 4.1 ou plus récent. Les fichiers `ISAM` fonctionne toujours en MySQL 4.0 mais il est abandonné, et il sera désactivé en MySQL 5.0.

Les anciens clients doivent fonctionner avec le serveur version 4.0 sans aucun problème.

Même si vous suivez les instructions ci-dessus, vous pourrez retourner en version MySQL 3.23.52 ou plus récent, si vous rencontrez des difficultés avec MySQL 4.0. Dans ce cas, vous devez utiliser

la commande `mysqldump` sur toutes les tables qui utilisent un index en texte plein, et restaurer ces tables en 3.23 (car la version 4.0 utilise un nouveau format pour les index en texte plein).

Voici une liste plus complète de points à contrôler lorsque vous passez à la version 4.0 :

- MySQL 4.0 a de très nombreux nouveaux droits dans la table `mysql.user` . Syntaxe de `GRANT` et `REVOKE` .

  Pour faire fonctionner ces nouveaux droits, vous devez exécuter le script `mysql_fix_privilege_tables` . Jusqu'à ce que ce script soit exécuté, les utilisateurs auront les droits de `SHOW DATABASES` , `CREATE TEMPORARY TABLES` , et `LOCK TABLES` . Les droits de `SUPER` et `EXECUTE` héritent leur valeur du droit de `PROCESS` . `REPLICATION SLAVE` et `REPLICATION CLIENT` héritent leur valeur de `FILE` .

  Si vous avez un script qui crée automatiquement des nouveaux utilisateur, vous devez le modifier pour y inclure les nouveaux droits. Si vous n'utilisez pas la commande `GRANT` dans ces scripts, c'est une bonne idée que de les vérifier.

  En version 4.0.2, l'option `--safe-show-database` est abandonnée (et ne fait plus rien du tout). Options de démarrage de `mysqld` liées à la sécurité .

  Si vous obtenez des interdictions d'accès pour les nouveaux utilisateurs en version 4.0.2, vous devriez vérifier si vous avez besoin de nouveaux droits que vous n'utilisiez pas avant. En particulier, vous aurez besoin du droit de `REPLICATION SLAVE` (au lieu de `FILE` ) pour les nouveaux esclaves.
- Les paramètres de démarrage `myisam_max_extra_sort_file_size` et `myisam_max_extra_sort_file_size` sont désormais exprimés en octets, et non plus en Mo, comme cela était le cas jusqu'en version 4.0.3). Les systèmes externes de verrouillages des tables MyISAM/ISAM sont désormais désactivés par défaut. Vous pouvez les réactiver avec l'option `--external-locking` Pour la plupart des utilisateurs, ce n'est jamais nécessaire.
- Les options de démarrage suivantes ont été renommées :

| **Ancien nom** | **Nouveau nom .** |
|---|---|
| myisam_bulk_insert_tree_size | bulk_insert_buffer_size |
| query_cache_startup_type | query_cache_type |
| record_buffer | read_buffer_size |
| record_rnd_buffer | read_rnd_buffer_size |
| sort_buffer | sort_buffer_size |
| warnings | log-warnings |
| err-log | --log-error (for mysqld_safe ) |

  Les options de démarrage `record_buffer` , `sort_buffer` et `warnings` vont encore fonctionner avec MySQL 4.0 mais elles sont obsolètes.
- Les variables SQL suivantes ont changé de nom.

| **Ancien nom** | **Nouveau nom .** |
|---|---|
| SQL_BIG_TABLES | BIG_TABLES |
| SQL_LOW_PRIORITY_UPDATES | LOW_PRIORITY_UPDATES |
| SQL_MAX_JOIN_SIZE | MAX_JOIN_SIZE |
| SQL_QUERY_CACHE_TYPE | QUERY_CACHE_TYPE |

  Les anciens noms fonctionneront encore en MySQL 4.0, mais sont obsolètes.
- Vous devez utiliser la commande `SET GLOBAL SQL_SLAVE_SKIP_COUNTER=#` au lieu de `SET SQL_SLAVE_SKIP_COUNTER=#` .
- Renommez les options de démarrage `--skip-locking` en `--skip-external-locking` et `--enable-locking` en `--external-locking` .
- `SHOW MASTER STATUS` retourne désormais une liste vide si les logs binaires ne sont pas activés.
- `SHOW SLAVE STATUS` retourne désormais une liste vide si l'esclave n'est pas initialisé.
- `mysqld` dispose désormais de l'option `--temp-pool` activée par défaut, car cela donne de meilleurs performances sur certains systèmes d'exploitation.
- Les colonnes `DOUBLE` et `FLOAT` acceptent désormais l'option `UNSIGNED` pour le stockage (auparavant, `UNSIGNED` était ignoré pour ces colonnes).

- `ORDER BY column DESC` trie désormais les valeurs `NULL` en premier. En version 3.23, ce n'était pas toujours le cas.
- `SHOW INDEX` a 2 colonnes de plus ( `Null` et `Index_type` ) qu'il n'avait pas en version 3.23.
- `CHECK` , `SIGNED` , `LOCALTIME` et `LOCALTIMESTAMP` sont des mots réservés.
- Le résultat de toutes les opérations sur les bits, `|` , `, << , >> et ~ est maintenant non signé. Cela peut poser des problèmes si vous aviez un contexte dans lequel vous souhaitez un résultat signé. Fonctions de transtypage .`
- **Note** : lorsque vous utilisez la soustraction entre des entiers dont l'un est `UNSIGNED` , le résultat sera non signé! En d'autres termes, avant de passer à la version MySQL 4.0, vous devriez vérifier les situations où votre application soustrait une valeur d'un entier non signé, et que vous attendez une valeur négative, ou si vous soustrayez une valeur non signée d'une colonne. Vous pouvez désactiver ce comportement en utilisant l'option de démarrage `--sql-mode=NO_UNSIGNED_SUBTRACTION` lorsque vous démarrez `mysqld` . Fonctions de transtypage .
- Pour utiliser `MATCH ... AGAINST (... IN BOOLEAN MODE)` avec vos table,s vous devez les reconstruire avec `ALTER TABLE table_name TYPE=MyISAM` , **même** si la table est déjà au format `MyISAM` .
- `LOCATE()` et `INSTR()` sont sensibles à la casse, si l'un des arguments est une chaîne binaire. Sinon, ils sont insensibles à la casse.
- `STRCMP()` utilise désormais le jeu de caractères courant pour les comparaisons, ce qui signifie que le comportement par défaut des comparaisons est désormais insensible à la casse.
- `HEX(string)` retourne désormais les caractères convertis sous la forme d'une chaîne hexadécimale. Si vous voulez convertir un nombre en hexadécimal, vous devez vous assurer d'appeler `HEX()` avec un argument numérique.
- En version 3.23, `INSERT INTO ... SELECT` fonctionne toujours avec l'option `IGNORE` . En version 4.0.1, MySQL va s'arrêter (et peut être annuler la transaction) si vous ne spécifiez pas l'option `IGNORE` .
- `safe_mysqld` est renommée en `mysqld_safe` . Pour encore un peu de temps, nous allons inclure dans nos distributions binaires le script `safe_mysqld` vous la forme d'un lien symbolique vers `mysqld_safe` .
- Les fonctions de l'ancienne API C API `mysql_drop_db` , `mysql_create_db` et `mysql_connect` ne sont plus supportées, à moins que vous ne compiliez MySQL avec `CFLAGS=-DUSE_OLD_FUNCTIONS` . Au lieu de cela, il sera plus sage de changer vos programmes, pour qu'il utilisent la nouvelle API 4.0.
- Dans la structure `MYSQL_FIELD` , `length` et `max_length` ont évolué de `unsigned int` en `unsigned long` . Cela ne va pas causer de problèmes, hormis le fait qu'ils peuvent générer des messages d'alerte lorsqu'ils sont utilisé comme argument de fonctions comme `printf()` .
- Vous devriez utiliser la commande `TRUNCATE TABLE` lorsque vous voulez effacer toutes les lignes d'une table, et que vous ne souhaitez pas savoir combien de lignes ont été effacées de la table (car `TRUNCATE TABLE` est plus rapide que `DELETE FROM table_name` ).
- Vous allez rencontrer une erreur si vous avez un verrou actif ou une transaction active, et que vous essayez d'utiliser les commandes `TRUNCATE TABLE` ou `DROP DATABASE` .
- Vous devriez utiliser des entiers pour stocker les valeurs dans les colonnes de type BIGINT (au lieu d'utiliser des chaînes, comme vous le faisiez en MySQL 3.23). Utiliser des chaînes va toujours fonctionner, mais passer des entiers est bien plus efficace.
- Le format de `SHOW OPEN TABLE` a été changé.
- Les clients multi-threadés doivent utiliser `mysql_thread_init()` et `mysql_thread_end()` . How to Make a Threaded Client .
- Si vous voulez recompiler le module Perl DBD::mysql, vous devez prendre les versions Msql-Mysql-modules 1.2218 ou plus récente, car les anciennes versions des module DBD utilisent une fonction `drop_db()` abandonnée.
- `RAND(seed)` retourne un nombre différent en version 4.0 qu'en version 3.23 : cela est fait pour différencier plus fortement `RAND(seed)` de `RAND(seed+1)` .
- Le type par défaut retourné par `IFNULL(A,B)` est maintenant le plus général des deux types `A` et `B` . (L'ordre est `STRING` , `REAL` puis `INTEGER` ).

## 2.5.4 Passer de la version 3.22 à la version 3.23

MySQL version 3.23 supporte les nouvelles tables `MyISAM` et les anciennes tables `ISAM` . Vous n'avez pas à convertir vos anciennes tables pour utiliser la nouvelle version 3.23. Par défaut, toutes les nouvelles tables seront créées avec le type `MyISAM` (à moins que vous ne lanciez `mysqld` avec l'option `--default-table-type=isam` ). Vous pouvez changer la table `ISAM` en table `MyISAM` avec la commande `ALTER TABLE table_name TYPE=MyISAM` ou le script Perl `mysql_convert_table_format` .

Les clients des versions 3.22 et 3.21 vont fonctionner sans problèmes avec la verison 3.23 du serveur.

La liste suivante indique les points à vérifier lors de la migration :

- Toutes les tables qui utilisent le jeu de caractères `tis620` doivent être corrigées avec `myisamchk -r` ou `REPAIR TABLE` .
- Si vous exécutez une commande `DROP DATABASE` sur un lien symbolique, le lien et la base originale seront effacés. Cela n'arrivait pas en 3.22 car `configure` ne détectait pas les appels à `readlink` .
- `OPTIMIZE TABLE` ne fonctionne que pour les tables `MyISAM` . Pour les autres types de tables, vous devez utiliser `ALTER TABLE` pour optimiser la table. Durant la commande `OPTIMIZE TABLE` , la table est verrouillée.
- Le client MySQL `mysql` est démarré par défaut avec l'option option `--no-named-commands (-g)` . Cette option peut être désactivée avec `--enable-named-commands (-G)` . Cela peut causer des problèmes d'incompatibilité dans certains cas : par exemple, dans les scripts SQL qui utilisent des commandes nommées sans point virgule! Le format long de la commande devrait fonctionner correctement.
- Les fonctions de date qui travaille sur des parties de dates (comme `MONTH()` ) vont désormais retourner 0 pour la date `0000-00-00` . (MySQL 3.22 retournait `NULL` .)
- Si vous utilisez le jeu de caractères `allemand` pour les tris, vous devez réparer vos tables avec `isamchk -r` , car nous avons fait des modifications dans l'ordre de tri.
- Le type de retour par défaut de `IF` dépendant maintenant des deux arguments, et plus seulement du premier.
- `AUTO_INCREMENT` ne fonctionne pas sur les nombres négatifs. La raison pour cela est que les nombres négatifs posaient des problèmes d'écrasement entre −1 et 0. `AUTO_INCREMENT` pour les tables MyISAM est maintenant géré à un niveau plus bas, et il est bien plus rapide. Pour les tables MyISAM, les anciens numéros ne sont plus réutilisés, même si vous effacez des lignes dans la table.
- `CASE` , `DELAYED` , `ELSE` , `END` , `FULLTEXT` , `INNER` , `RIGHT` , `THEN` et `WHEN` sont de nouveaux mots réservés.
- `FLOAT(X)` est maintenant un véritable type de nombre à virgule flottante, avec un nombre défini de décimales.
- Lors de la déclaration de `DECIMAL(length,dec)` , la taille de l'argument n'inclut plus une place pour le signe ou le séparateur décimal.
- Une chaîne `TIME` doit être fournie au format suivant : `[[[DAYS] [H]H:]MM:]SS[.fraction]` ou `[[[[[H]H]H]H]MM]SS[.fraction]` .
- `LIKE` compare maintenant les chaînes en appliquant les mêmes règles que `=` . Si vous voulez l'ancien comportement, vous pouvez compiler MySQL avec l'option `CXXFLAGS=-DLIKE_CMP_TOUPPER` .
- `REGEXP` est maintenant insensible à la casse pour les chaînes normales (non binaires).
- Quand vous vérifiez/réparez des tables, vous devez utiliser `CHECK TABLE` ou `myisamchk` pour les tables `MyISAM` ( `.MYI` ) et `isamchk` pour les tables `ISAM` ( `.ISM` ).
- Si vous voulez que les fichiers d'export de `mysqldump` soit compatibles entre les versions MySQL 3.22 et 3.23, vous ne devez pas utiliser l'option `--opt` ou `--all` de `mysqldump` .
- Vérifiez tous vos appels à `DATE_FORMAT()` pour vous assurer qu'il y a un signe pourcentage `'%'` avant chaque caractère de format (MySQL version 3.22 et plus récent avait déjà cette syntaxe).

- `mysql_fetch_fields_direct` est maintenant une fonction (c'était une macro), qui retourne un pointeur sur `MYSQL_FIELD` au lieu de `MYSQL_FIELD` .
- `mysql_num_fields()` ne peut plus être utilisé sur les objets `MYSQL*` (c'est maintenant une fonction qui prend `MYSQL_RES*` comme argument. Il faut donc utiliser `mysql_field_count()` à la place).
- En MySQL version 3.22, le résultat de `SELECT DISTINCT ...` était toujours trié. En version 3.23, vous devez spécifier la clause `GROUP BY` ou `ORDER BY` pour obtenir un résultat trié.
- `SUM()` retourne désormais `NULL` , au lieu de 0, si il n'y a pas de lignes à calculer. Ceci s'accorde avec la norme SQL.
- `AND` ou `OR` avec les valeurs `NULL` vont désormais retourner `NULL` au lieu de 0. Cela affecte surtout les requêtes qui utilisait `NOT` ou une expression `AND/OR` telle que `NOT NULL` = `NULL` . `LPAD()` et `RPAD()` vont réduire la taille de la chaîne résultante, si elle est plus grand que l'argument de taille.

## 2.5.5 Passer de la version 3.21 à la version 3.22

Rien qui n'affecte la compatibilité n'a changé entre les versions 3.21 et 3.22. Le seul problème courant est que les nouvelles tables qui sont créées avec le type `DATE` vont désormais utiliser le nouveau format de stockage. Vous ne pourrez pas accéder à ces nouveaux formats depuis les vieilles versions de `mysqld` .

Après avoir installé MySQL version 3.22, vous devriez démarrer le nouveau serveur, et exécuter le script `mysql_fix_privilege_tables` . Il va ajouter les nouveaux droits à la commande `GRANT` . SI vous oubliez cela, vous obtiendrez des erreurs `Access denied` lorsque vous essayez d'utiliser les commandes `ALTER TABLE` , `CREATE INDEX` ou `DROP INDEX` . Si votre compte `root` MySQL utilise un mot de passe, vous devriez l'indiquer au script `mysql_fix_privilege_tables` .

L'interface C de `mysql_real_connect()` a changé. Si vous avez un vieux client qui appelle cette fonction, vous devez placer un `0` pour le nouvel argument `db` (ou réécrire le client pour qu'il envoie l'élément `db` , et accélère les connexions). Vous devez aussi appeler `mysql_init()` avant d'appeler `mysql_real_connect()` ! Ce changement a été fait pour permettre l'appel de la fonction `mysql_options()` , qui sauve les options dans la structure `MYSQL` .

La variable `key_buffer` de `mysqld` a changé de nom, et est devenue `key_buffer_size` , mais vous pouvez toujours utiliser l'ancien nom dans vos fichiers de démarrage.

## 2.5.6 Passer de la version 3.20 à la version 3.21

Si vous avez une version de MySQL plus ancienne que la version 3.20.28 et que vous voulez passer à la version 3.21, vous devez suivre ces étapes :

Vous pouvez démarrer le serveur `mysqld` version 3.21 avec le script `safe_mysqld --old-protocol` pour l'utiliser avec les clients de la version 3.20. Dans ce cas, la fonction `mysql_errno()` des nouveaux clients ne sera pas fonctionnelle, et seul `CR_UNKNOWN_ERROR` (mais il fonctionne pour les erreurs client), et le serveur utilisera l'ancienne fonction `password()` plutôt que la nouvelle.

Si vous **n'utilisez pas** l'option `--old-protocol` avec `mysqld` , vous devez suivre ces instructions :

- Tous les clients doivent être recompilés. Si vous utilisez ODBC, vous devez obtenir le nouveau pilote `MyODBC` 2.x.
- Le script `scripts/add_long_password` doit être utilisé pour convertir le champs `Password` de la table `mysql.user` en `CHAR(16)` .
- Tous les mots de passe doivent être réassignés dans la table `mysql.user` pour utiliser les mots de 62 bits au lieu de 31 bits.

- Le format de table n'a pas changé, ce qui vous évite d'avoir à convertir des tables.

MySQL version 3.20.28 et plus récent peut gérer les nouvelles tables `user` sans affecter les clients. Si vous avez une version plus ancienne que la 3.20.28, les mots de passe ne seront plus valide, si vous convertissez la table `user` . Pour être tranquille, commencez par faire passer votre version à la 3.20.28 puis passez en version 3.21. Le nouveau client fonctionne avec le serveur 3.20.x `mysqld` , alors si vous rencontrez des problèmes avec la version 3.21.x, vous pouvez toujours vous rabattre sur les vieux serveurs 3.20.x sans recompiler les clients.

Si vous n'utilisez pas l'option `--old-protocol` de `mysqld` , les vieux clients vont émettre une erreur :

```
ERROR: Protocol mismatch. Server Version = 10 Client Version = 9
```

La nouvelle interface Perl `DBI` / `DBD` supporte aussi l'ancienne interface `mysqlperl` . Le seul changement que vous devez faire si vous utilisez `mysqlperl` est de changer les arguments de la fonction `connect()` . Les nouveaux arguments sont : `host` , `database` , `user` , et `password` (les arguments `user` et `password` ont été échangés). L'interface `DBI` .

Les modifications actuelles affectent les requêtes des anciennes applications :

- `HAVING` doit être spécifié avant la clause `ORDER BY` .
- Les paramètres de la fonction `LOCATE()` ont été échangés.
- Il y a de nouveaux mots réservés. Les plus notables sont `DATE` , `TIME` et `TIMESTAMP` .

## 2.5.7 Migrer depuis une autre architecture

Si vous utilisez MySQL version 3.23, vous pouvez copier les fichiers `.frm` , `.MYI` et `.MYD` entre les différentes architectures qui supportent le même format de nombre à virgule flottante (MySQL prend en charge les échanges d'octets).

Les données MySQL des tables `ISAM` et les fichiers d'index ( `.ISD` et `*.ISM` , respectivement) sont dépendantes de l'architecture, et dans certains cas, dépendantes du système d'exploitation. Si vous voulez déplacer des applications vers une autre machine qui a une autre architecture, ou un autre système d'exploitation que votre machine courante, il est recommandé de ne pas faire une simple copie de base en copiant les fichiers vers leur nouvelle destination. Utilisez plutôt `mysqldump` .

Par défaut, `mysqldump` va créer un fichier de requêtes SQL. Vous pouvez alors transférer le fichier sur une autre machine, et le fournir comme script à un client `mysql` .

Essayez la commande `mysqldump --help` pour voir quelles options sont disponibles. Si vous envoyez les données vers une nouvelle version de MySQL, il recommandé d'utiliser l'option `mysqldump --opt` pour obtenir un export compact et plus rapide.

Le plus facile (mais pas le plus rapide) pour déplacer une base de données entre deux machines et d'exécuter les commandes suivantes sur la machine qui héberge la base :

```
 shell> mysqladmin -h 'other hostname' create db_name
shell> mysqldump --opt db_name \
        | mysql -h 'other hostname' db_name
```

Si vous voulez copier la base depuis une machine distante sur un réseau lent, vous pouvez utiliser :

```
 shell> mysqladmin create db_name
shell> mysqldump -h 'other hostname' --opt --compress db_name \
        | mysql db_name
```

Vous pouvez aussi stocker le résultat dans un fichier, et transférer le fichier sur la machine de destination, puis charger ce fichier dans le serveur. Par exemple, vous pouvez exporter la base vers un fichier source comme ceci :

```
shell> mysqldump --quick db_name | gzip > db_name.contents.gz
```

Le fichier créé est compressé. Transférez le fichier contenant le contenu de votre base sur la machine de destination, puis utilisez ces commandes :

```
shell> mysqladmin create db_name
shell> gunzip < db_name.contents.gz | mysql db_name
```

Vous pouvez aussi utiliser `mysqldump` et `mysqlimport` pour accomplir cette opération. Pour les grandes tables, c'est bien plus rapide que d'utiliser simplement `mysqldump` . Dans les commandes suivantes, `DUMPDIR` représente le chemin complet du dossier que vous utilisez pour stocker le résultat de `mysqldump` .Premièrement, créez un dossier pour les fichiers d'exportation, puis faites l'export :

```
shell> mkdir DUMPDIR
shell> mysqldump --tab=DUMPDIR db_name
```

Puis transférez les fichiers du dossier `DUMPDIR` dans un dossier correspondant, dans la machine de destination, puis chargez ces fichiers dans MySQL comme ceci :

```
shell> mysqladmin create db_name           # Création de la base
shell> cat DUMPDIR/*.sql | mysql db_name   # Création des tables dans la base
shell> mysqlimport db_name DUMPDIR/*.txt   # Chargement des données dans les tables
```

N'oubliez pas non plus de copier le contenu de votre base `mysql` car c'est là que résident les droits ( `user` , `db` , `host` ). Vous devrez alors exécuter les commandes en tant que `root` MySQL sur la nouvelle machine, jusqu'à ce que vous ayez réinstallé `mysql` .

Après l'importation de la base `mysql` sur la nouvelle machine, exécutez la commande `mysqladmin flush-privileges` pour que le serveur relise les droits.

# 2.6 Notes spécifiques aux systèmes d'exploitation

## 2.6.1 Notes relatives à Linux (toutes versions)

Les notes suivantes relatives à `glibc` ne s'appliquent que dans la situation où vous construisez MySQL vous-même. Si vous n'utilisez pas Linux sur une machine x86 machine, dans la plupart des cas, il sera mieux pour vous d'utiliser nos binaires. Nous lions nos binaires avec la meilleur version patchée de `glibc` que nous pouvons fournir et avec les meilleurs options du compilateur, en essayant de les rendre bons pour un serveur qui connait de fortes charges. Et donc, si vous lisez le texte suivant et que vous avez un doute sur ce que vous devez faire, essayez d'abord notre binaire pour voir s'il vous convient, et ne vous souciez de vosz propres constructions qu'après vous être assurés que notre binaire ne répond pas à vos attentes. Dans ce cas, nous apprécierons une note à propos de cela, pour que nous puissions faire mieux la prochaine fois. Pour les besoins d'un utilisateur de base, pour des configurations avec beaucoup de connexions et/ou des tables dépassant la limite des 2G, notre binaire est dans la plupart des cas le meilleur choix.

MySQL utilise LinuxThreads sur Linux. Si vous utilisez une vielle version de Linux qui ne possède pas `glibc2` , vous devz installer LinuxThreads avant d'essayer de compiler MySQL. Vous pouvez obtenir LinuxThreads à l'adresse suivante : http://www.mysql.com/Downloads/Linux/ .

**Note** : Nous avons rencontré quelques problèmes étranges avec Linux 2.2.14 et MySQL sur les systèmes SMP. Si vous avez un système SMP, nous vous recommandons de mettre à jour à Linux 2.4 dès que possible ! Votre système n'en sera que plus rapide et plus stable !

Notez que les versions de `glibc` inférieure ou égale à la 2.1.1 ont un bogue fatal dans la gestion de `pthread_mutex_timedwait` , qui est utilisé lors que vous exécutez `INSERT DELAYED` . Nous vous recommandons de ne pas utiliser `INSERT DELAYED` avant de mettre à jour glibc.

Si vous planifiez d'avoir plus de 1000 connexions simultanées, vous aurez besoin d'apporter quelques modifications à LinuxThreads, le recompiler, et relier MySQL avec le nouveau `libpthread.a` . Augmentez `PTHREAD_THREADS_MAX` dans `sysdeps/unix/sysv/linux/bits/local_lim.h` à 4096 et diminuez `STACK_SIZE` dans `linuxthreads/internals.h` à 256 KB. Les chemins sont relatifs à la racine de `glibc` . Notez que MySQL ne sera pas stable autour de 600−1000 connexions si `STACK_SIZE` est à 2 MB (par défaut).

Si MySQL n'arrive pas à ouvrir assez de fichiers, ou à créer assez de connexions, il se peut que vous n'ayez pas configuré Linux pour qu'il gère assez de fichiers.

Dans Linux 2.2 ou plus, vous pouvez connaitre le nombre de gestionnaires de fichiers alloués en faisant :

```
 cat /proc/sys/fs/file−max
cat /proc/sys/fs/dquot−max
cat /proc/sys/fs/super−max
```

Si vous avez plus de 16 MB de mémoire, vous devez ajouter qauelque chose comme ce qui suit dans vos scripts d'initialisation ( `/etc/init.d/boot.local` sur SuSE Linux) :

```
 echo 65536 > /proc/sys/fs/file−max
echo 8192 > /proc/sys/fs/dquot−max
echo 1024 > /proc/sys/fs/super−max
```

Vous pouvez aussi exécuter les commandes précédentes à partir de la ligne de commande en tant que root, mais les changements seront perdus au prochain redémarrage de l'ordinateur.

Vous pouvez sinon définir ces paramètres lors du démarrage de la machine en utilisant l'outil `sysctl` , qui est utilisé par plusieurs distributions Linux (SuSE l'a aussi ajouté, à partir de SuSE Linux 8.0). Ajoutez simplement les valeurs suivantes dans un fichier nommé `/etc/sysctl.conf` :

```
 # Increase some values for MySQL
fs.file−max = 65536
fs.dquot−max = 8192
fs.super−max = 1024
```

Vous devez aussi ajouter ce qui suit à `/etc/my.cnf` :

```
 [safe_mysqld]
open−files−limit=8192
```

Cela devrait permettre à MySQL de créer jusqu'à 8192 fichiers de conexion.La constante `STACK_SIZE` des LinuxThreads contrôle l'espacement des piles de threads dans l'espace d'adressage. Elle doit être assez grande pour qu'il y ait plusieurs chambres pour la pile de chaque thread individuel, mais assez petite pour empêcher les piles de certains threads d'agir sur les données globales de `mysqld` . Malheureusement, l'implémentation Linux de `mmap()` , comme nous l'avons découvert, will successfully unmap an already mapped region if you ask it to map out an address already in use, zeroing out the data on the entire page, instead of returning an error. So, the safety of `mysqld` or any other threaded application depends on the "gentleman" behaviour of the code that creates threads. The user must take measures to make sure the number of running threads at any time is sufficiently low for thread stacks to stay away from the global heap. With `mysqld` , you should enforce this "gentleman" behaviour by setting a reasonable value for the `max_connections` variable.

Si vous construisez MySQL vous−mêmes et ne voulez pas vous amuser à patcher LinuxThreads, vous ne devez pas dépasser 500 pour la valeur de `max_connections` . Cela devrait même être moins si vous avez un tampon de clefs assez large, de grosses tables heap, ou d'autres choses qui peuvent faire allouer beaucoup de mémoire à `mysqld` , ou si vous utilisez un noyau 2.2 avec un patch 2G. Si vous utilisez notre binaire ou RPM 3.23.25 ou plus, vous pouvez mettre `max_connections` à 1500 sans problèmes, en supposant que vous n'avez ni de grosses tables heap ni grands tampons de clefs. Plus vous réduirez `STACK_SIZE` dans LinuxThreads plus les threads créés seront sûrs. Nous recommandons une valeur entre 128K et 256K.

Si vous utilisez beaucoup de connexions simultanées vous souffrirez peut−être d'une "fonctionnalité" du noyau 2.2 qui pénalise un processus lors du fork ou du clonage d'un enfant en

essayant de prévenir un attaque du type fork bomb. This will cause MySQL not to scale well as you increase the number of concurrent clients. On single−CPU systems, we have seen this manifested in a very slow thread creation, which means it may take a long time to connect to MySQL (as long as 1 minute), and it may take just as long to shut it down. On multiple−CPU systems, we have observed a gradual drop in query speed as the number of clients increases. In the process of trying to find a solution, we have received a kernel patch from one of our users, who claimed it made a lot of difference for his site. The patch is available at http://www.mysql.com/Downloads/Patches/linux−fork.patch . We have now done rather extensive testing of this patch on both development and production systems. It has significantly improved `MySQL` performance without causing any problems and we now recommend it to our users who are still running high−load servers on 2.2 kernels. This issue has been fixed in the 2.4 kernel, so if you are not satisfied with the current performance of your system, rather than patching your 2.2 kernel, it might be easier to just upgrade to 2.4, which will also give you a nice SMP boost in addition to fixing this fairness bug.

We have tested MySQL on the 2.4 kernel on a 2−CPU machine and found MySQL scales **much** better−there was virtually no slowdown on queries throughput all the way up to 1000 clients, and the MySQL scaling factor (computed as the ratio of maximum throughput to the throughput with one client) was 180%. We have observed similar results on a 4−CPU system−virtually no slowdown as the number of clients was increased up to 1000, and 300% scaling factor. So for a high−load SMP server we would definitely recommend the 2.4 kernel at this point. We have discovered that it is essential to run `mysqld` process with the highest possible priority on the 2.4 kernel to achieve maximum performance. This can be done by adding `renice −20 $$` command to `safe_mysqld` . In our testing on a 4−CPU machine, increasing the priority gave 60% increase in throughput with 400 clients.

Nous essayons aussi actuellement d'obtenir des informations sur le bon fonctionnement de `MySQL` sur le noyaux 2.4 sur les systèmes 4−voies and 8−voies. Si vous avez accès à un tel système et que vous avez effectués quelques tests de performances, envoyez−nous un mail à docs@mysql.com avec les résultats, nous les inclurons dans le manuel.

There is another issue that greatly hurts MySQL performance, especially on SMP systems. The implementation of mutex in LinuxThreads in `glibc−2.1` is very bad for programs with many threads that only hold the mutex for a short time. On an SMP system, ironic as it is, if you link MySQL against unmodified `LinuxThreads` , removing processors from the machine improves MySQL performance in many cases. We have made a patch available for `glibc 2.1.3` to correct this behaviour ( http://www.mysql.com/Downloads/Linux/linuxthreads−2.1−patch ).

With `glibc−2.2.2` MySQL version 3.23.36 will use the adaptive mutex, which is much better than even the patched one in `glibc−2.1.3` . Be warned, however, that under some conditions, the current mutex code in `glibc−2.2.2` overspins, which hurts MySQL performance. The chance of this condition can be reduced by renicing `mysqld` process to the highest priority. We have also been able to correct the overspin behaviour with a patch, available at http://www.mysql.com/Downloads/Linux/linuxthreads−2.2.2.patch . It combines the correction of overspin, maximum number of threads, and stack spacing all in one. You will need to apply it in the `linuxthreads` directory with `patch −p0 </tmp/linuxthreads−2.2.2.patch` . We hope it will be included in some form in to the future releases of `glibc−2.2` . In any case, if you link against `glibc−2.2.2` you still need to correct `STACK_SIZE` and `PTHREAD_THREADS_MAX` . We hope that the defaults will be corrected to some more acceptable values for high−load MySQL setup in the future, so that your own build can be reduced to `./configure; make; make install` .

We recommend that you use the above patches to build a special static version of `libpthread.a` and use it only for statically linking against `MySQL` . We know that the patches are safe for `MySQL` and significantly improve its performance, but we cannot say anything about other applications. If you link other applications against the patched version of the library, or build a patched shared version and install

it on your system, you are doing it at your own risk with regard to other applications that depend on `LinuxThreads` .

Si vous rencontrez des problèmes étranges lors de l'installation de MySQL, ou quoi que ce soit qui s'en rapproche, il est fort possible que cela soit un problème de librairie ou de compilateur. Si c'est le cas, l'utilisation de notre binaire les résoudra.

Un problème connu avec la distribution binaire est que avec les anciens systèmes Linux qui utilisent `libc` (comme RedHat 4.x ou Slackware), vous obtiendrez quelques erreurs non−fatales de résolutions de noms d'hôtes. Notes relatives à Linux pour les distribution binaires .

Lors de l'utilisation des LinuxThreads vous verrez un minimum de trois processus en cours. Il s'agit en fait de threads, il y a a un pour le gestionnaire des LinuxThreads, un pour gérer les connexions, et un autre pour gérer les alarmes et les signaux.

Notez que le noyau Linux et la librairie LinuxThread ne peuvent avoir par défaut que 1024 threads. Cela signifie que vous ne pouvez avoir que 1021 connexions à MySQL sur un système non−patché. La page http://www.volano.com/linuxnotes.html contient des informations pour contourner cette limite.

Si vous voyez un processus de démon `mysqld` mort avec `ps` , cela signifie le plus souvent que vous avez trouvé un bogue dans MySQL ou que vous avez une table corrompue. Que faire si MySQL crashe constamment .

To get a core dump on Linux if `mysqld` dies with a `SIGSEGV` signal, you can start `mysqld` with the `−−core−file` option. Note that you also probably need to raise the `core file size` by adding `ulimit −c 1000000` to `safe_mysqld` or starting `safe_mysqld` with `−−core−file−size=1000000` . `safe_mysqld` , le script père de `mysqld` .

Si vous liez votre propre client MySQL et que vousobtenez l'erreur suivante :

```
 ld.so.1: ./my: fatal: libmysqlclient.so.4:
open failed: No such file or directory
```

lors de son exécution, le problème peut être contourné des façons suivantes :

- Liez le client avec les options suivantes (au lieu de `−Lpath` ) : `−Wl,r/path−libmysqlclient.so` .
- Copiez `libmysqclient.so` dans `/usr/lib` .
- Ajoutez le chemin vers le répertoire où se trouve `libmysqlclient.so` à la variable d'environnement `LD_RUN_PATH` avant de mettre en marche votre client.

Si vous utilisez le compilateur Fujitsu `(fcc / FCC)` vous aurez quelques problèmes en compilant MySQL car les fichiers d'entêtes Linux sont très orientés `gcc` .

La ligne suivante de `configure` devrait fonctionner avec `fcc/FCC` :

```
 CC=fcc CFLAGS="−O −K fast −K lib −K omitfp −Kpreex −D_GNU_SOURCE \
−DCONST=const −DNO_STRTOLL_PROTO" CXX=FCC CXXFLAGS="−O −K fast −K lib \
−K omitfp −K preex −−no_exceptions −−no_rtti −D_GNU_SOURCE −DCONST=const \
−Dalloca=__builtin_alloca −DNO_STRTOLL_PROTO \
'−D_EXTERN_INLINE=static __inline'" ./configure −−prefix=/usr/local/mysql \
−−enable−assembler −−with−mysqld−ldflags=−all−static −−disable−shared \
−−with−low−memory
```

### 2.6.1.1 Notes relatives à Linux pour les distributions binaires

MySQL requière au moins la version 2.0 de Linux.

**Attention** : Certains utilisateurs de MySQL nous ont avertis qu'ils ont rencontré de graves problèmes de stabilité avec MySQL et le noyau 2.2.14 de Linux. Si vous utilisez ce noyau, vous devez mettre à jour à la 2.2.19 (ou plus récent) ou a un noyau 2.4. Si vous utilisez un ordinateur multi−processeurs, vous devriez sérieusement songer à passer au noyau 2.4 qui vous aportera de grandes performances niveau vitesse.

La version binaire est liée avec `-static` , ce qui signifie que normalement vous n'avez pas besoin de vous soucier des versions des librairies système que vous avez. Vous n'avez pas besoin d'installer LinuxThreads non plus. Un programme lié avec `-static` est légèrement plus grand qu'un programme liée dynamiquement mais aussi un peu plus rapide (3−5%). Un problème, toutefois, est que vous ne pouvez utiliser de fonctions définies par l'utilisateur avec un programme lié statiquement. Si vous allez écrire ou utiliser des fonctions UDF (c'est réservé aux développeurs C ou C++), vous devez compiler MySQL vous−mêmes, en utilisant les liaisons dynamiques.

Si vous utilisez un système basé sur `libc` (au lieu de `glibc2` ), vous aurez probablement quelques problèmes de résolution des noms d'hôtes et des problèmes avec `getpwnam()` avec les versions binaires. (Cela vient du fait que `glibc` dépend malheureusement de quelques librairies externes pour résoudre les noms d'hôtes et `getpwent()` , même quand elle est compilée avec `-static` ). Dans ce cas, vous obtiendrez probablement l'erreur suivante quand vous exécuterez `mysql_install_db` :

```
Sorry, the host 'xxxx' could not be looked up
```

ou l'erreur suivante quand vous essayez de démarrer `mysqld` avec l'option `--user` :

```
getpwnam: No such file or directory
```

Vous pouvez résoudre ce problème de la façon suivante :

- Obtenez une distribution des sources MySQL (un distribution RPM ou le `tar.gz` ) et installez la à la place.
- Exécutez `mysql_install_db --force` ; cela n'exécutera pas le test `resolveip` dans `mysql_install_db` . Le mauvais côté est que vous ne pourrez pas utiliser de noms d'hôtes dans les tables de droits; vous devez utiliser les adresses IP à la place (sauf pour `localhost` ). Si vous utilisez une vielle version de MySQL qui ne supporte pas `--force` , vous devez supprimer le test `resolveip` dans `mysql_install` à l'aide d'un éditeur.
- Démarrez `mysqld` avec `su` au lieu d'utiliser `--user` .

Le binaire Linux−Intel et les RPM de MySQL sont configurés pour la vitesse la plus grande possible. Nous essayons toujours d'utiliser le compilateur le plus rapide disponible.Le support Perl de MySQL requière la version 5.004_03 de Perl ou plus récent.

Sur quelques version de Linux 2.2, vous pouvez obtenir l'erreur `Resource temporarily unavailable` quand vous faites beaucoup de nouvelles connexions à un serveur `mysqld` en utilisant TCP/IP.

Le problème est que Linux possède un délai entre votre fermeture de la socket TCP/IP et sa libération par le système. Vu qu'il y a un nombre fini de places pour les branchements TCP/IP, vous obtiendrez l'erreur précédente si vous essayez de faire beaucoup de connexions TCP/IP en peu de temps, comme quand vous exécutez le benchmark MySQL `test-connect` via TCP/IP.

Nous avons envoyé des questions plusieurs fois à propos de ce problème à différentes listes de diffusions Linux mais n'avons jamais réussi à résoudre ce problème proprement.

Le seul 'correctif' connu pour ce problème est d'utiliser des connexions persistantes dans vos clients ou d'utiliser les sockets, si vous utilisez le serveur de bases de données et le client sur la même machine. Nous espérons que le noyau de `Linux 2.4` corrigera ce problème bientôt.

### 2.6.1.2 Notes relatives à Linux x86

MySQL requière la version 5.4.12 de `libc` ou plus récent. Il est connu pour fonctionner avec `libc` 5.4.46. La version 2.0.6 de `glibc` ou plus récente devrait aussi fonctionner. Il y a eu quelques problèmes avec les RPM de `glibc` de RedHat, et donc, si vous avez des problèmes, vérifiez s'il existe des mises à jour ! Les RPM de `glibc` 2.0.7−19 et 2.0.7−29 snot connus pour fonctionner.

Si vous utilisez gcc 3.0 ou plus récent pour compiler MySQL, vous devez installer la librairie `libstdc++v3` avant de compiler MySQL; si vous ne le faites pas vous obtiendrez une erreur à propos d'un symbole `__cxa_pure_virtual` manquant durant la liaison !

Sur quelques vieilles distributions de Linux, `configure` peut produire une erreur comme celle qui suit :

```
 Syntax error in sched.h. Change _P to __P in the /usr/include/sched.h file.
See the Installation chapter in the Reference Manual.
```

Faites ce que le message d'erreur dit et ajoutez un _ à la macro `_P` qui n'en a qu'un, puis essayez à nouveau.Vous pouvez obtenir quelques avertissements en compilant; celles qui suivent peuvent être ignorées :

```
 mysqld.cc -o objs-thread/mysqld.o
mysqld.cc: In function `void init_signals()':
mysqld.cc:315: warning: assignment of negative value `-1' to
`long unsigned int'
mysqld.cc: In function `void * signal_hand(void *)':
mysqld.cc:346: warning: assignment of negative value `-1' to
`long unsigned int'
```

Sur Debian GNU/Linux, si vous voulez que MySQL démarre automatiquement lors du démarrage de votre système, faites ce qui suit :

```
 shell> cp support-files/mysql.server /etc/init.d/mysql.server
shell> /usr/sbin/update-rc.d mysql.server defaults 99
```

`mysql.server` peut être trouvé dans le dossier `share/mysql` dans le dossier d'installation de MySQL ou dans le dossier `support-files` de l'arborescence des sources de MySQL.

Si `mysqld` provoque toujours un core dump au démarrage, le problème peut être que vous avez un vieux `/lib/libc.a` . Renommez le, puis supprimez `sql/mysqld` et faites à nouveau un `make install` puis réassayez. Ce problème a été reporté sur quelques installations de Slackware.

Si vous obtenez l'erreur suivante en liant `mysqld` , cela signifie que votre `libg++.a` n'est pas installé correctement :

```
 /usr/lib/libc.a(putc.o): In function `_IO_putc':
putc.o(.text+0x0): multiple definition of `_IO_putc'
```

Vous pouvez éviter d'utiliser `libg++.a` en exécutant `configure` comme suit :

```
 shell> CXX=gcc ./configure
```

### 2.6.1.3 Notes relatives à Linux SPARC

Sur quelques implémentations, `readdir_r()` est cassé. Le symptome est que `SHOW DATABASES` retourne toujours un résultat vide. Cela peut être corrigé en supprimant `HAVE_READDIR_R` de `config.h` après avoir configuré et avant de commencer à compiler.

Quelques problèmes demanderont le patchage de votre installation Linux. Le patch peut être trouvé sur http://www.mysql.com/Downloads/patches/Linux−sparc−2.0.30.diff . Ce patch est pour la distribution `sparclinux-2.0.30.tar.gz` de Linux qui est disponible sur `vger.rutgers.edu` (une version de Linux qui n'a jamais été mélangée avec la 2.0.30 officielle). Vous devez aussi installer la version 0.6 des LinuxThreads ou une version plus récente.

### 2.6.1.4 Notes relatives à Linux Alpha

La version 3.23.12 de MySQL est la première version de MySQL à être testée sur Linux−Alpha. Si vous voulez utiliser MySQL sur Linux−Alpha, vous devez vous assurer d'avoir cette version ou une version plus récente.

Nous avons testé MySQL sur Alpha avec nos tests de performance et notre suite de tests : tout semble fonctionner correctement.

Nous construisons actuellement les paquets binaires de MySQL sur `SuSE Linux 7.0` pour `AXP` , `kernel 2.4.4−SMP` , `Compaq C compiler (V6.2−505)` et `Compaq C++ compiler (V6.3−006)` sur une machine `Compaq DS20` avec un processeur `Alpha EV6` .

Vous pouvez trouver les précédents compilateurs sur http://www.support.compaq.com/alpha−tools/ ). En utilisant ces compilateurs, au lieu de `gcc` , nous obtenons environs 9 à 14% d'améliorations des performances avec MySQL.

Notez que jusqu'aux versions 3.23.52 et 4.0.2 de MySQL nous avons optimisé l'application pour le processeur courant seulement (en utilisant l'option de compilation `−fast` ); cela signifiait que vous ne pouviez utiliser nos binaires si vous n'aviez pas un processeur Alpha EV6.

Avec les versions suivantes nous avons ajouté l'option `−arch generic` à nos options de compilation, ce qui assure que le binaire fonctionne sur tout les processeurs Alpha. Nous compilons aussi statiquement pour éviter les problèmes de librairies.

```
 CC=ccc CFLAGS="−fast −arch generic" CXX=cxx \
CXXFLAGS="−fast −arch generic −noexceptions −nortti" \
./configure −−prefix=/usr/local/mysql −−disable−shared \
−−with−extra−charsets=complex −−enable−thread−safe−client \
−−with−mysqld−ldflags=−non_shared −−with−client−ldflags=−non_shared
```

Si vous voulez utiliser `egcs` la ligne de configuration suivante a fonctionné pour nous :

```
 CFLAGS="−O3 −fomit−frame−pointer" CXX=gcc \
CXXFLAGS="−O3 −fomit−frame−pointer −felide−constructors \
−fno−exceptions −fno−rtti" ./configure −−prefix=/usr/local/mysql \
−−disable−shared
```

Quelques problèmes connus lors de l'utilisation de MySQL sur Linux−Alpha:

- Le débogage d'applications threadées comme MySQL ne fonctionnera pas avec `gdb 4.18` . Vous devez télécharger et utiliser `gdb 5.1` à la place !
- Si vous essayez de lier statiquement `mysqld` en utilisant `gcc` , l'image résultante videra son noyau ( `core dump` ) au démarrage. En d'autres termes, **n'** utilisez **pas** `−−with−mysqld−ldflags=−all−static` avec `gcc` .

### 2.6.1.5 Note relative à Linux PowerPC

MySQL devrait fonctionner sur MkLinux avec le dernier package `glibc` (testé avec `glibc` 2.0.7).

### 2.6.1.6 Notes relatives à Linux MIPS

Pour faire fonctionner MySQL sur Qube2, (Linux Mips), vous aurez besoin de la librairie `glibc` la plus récente ( `glibc−2.0.7−29C2` est connue pour marcher). Vous devez aussi utiliser le compilateur `egcs` C++ ( `egcs−1.0.2−9` , `gcc 2.95.2` ou plus récent).

### 2.6.1.7 Notes relatives à Linux IA64

Pour pouvoir compiler MySQL sous Linux IA64, nous utilisons les lignes de compilation suivante : En utilisant `gcc-2.96` :

```
 CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc \
CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors \
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
"--with-comment=Official MySQL binary" --with-extra-charsets=complex
```

Sur IA64 les binaires des clients MySQL utilisent des librairies partagées. Cela signifie que si vous installez notre distribution binaire à un autre endroit que `/usr/local/mysql` vous devez modifier le fichier `/etc/ld.so.conf` ou ajouter le chemin vers le répertoire où vous avez `libmysqlclient.so` à la variable d'environnement `LD_LIBRARY_PATH` .

Problèmes en reliant avec la librairie du client MySQL .

## 2.6.2 Notes relatives à Windows

Cette section décrit l'utilisation de MySQL sur Windows. Ces informations sont aussi fournies dans le fichier `README` disponible avec la distribution Windows de MySQL. Installer MySQL sous Windows .

### 2.6.2.1 Démarrer MySQL sous Windows 95, 98 ou Me

MySQL utilise TCP/IP pour connecter le client au serveur (cela permet à une machine distante de se connecter à votre serveur MySQL). A cause de cela, vous devez installer TCP/IP sur votre machine avant de démarrer MySQL. Vous pouvez trouver TCP/IP sur votre CDRom Windows.

Notez que si vous utilisez une vieille version de Windows 95 (par exemple, OSR2), il est probable que vous ayez une vieille version du Winsock; MySQL requiert Winsock 2! Vous pouvez télécharger le dernier Winsock sur http://www.microsoft.com/ . Windows 98 dispose de la nouvelle librairie Winsock 2, et ce paragraphe ne s'applique pas.

Pour démarrer le serveur `mysqld` , vous devez utiliser une fenêtre MS-DOS et y taper :

```
C:\> C:\mysql\bin\mysqld
```

Cela va démarrer `mysqld` en tâche de fond, sans fenêtre.Vous pouvez arrêter MySQL avec la commande :

```
C:\> C:\mysql\bin\mysqladmin -u root shutdown
```

Ce script appelle l'utilitaire d'administration MySQL sous le nom de `root', qui est l'administrateur par défaut. Notez bien que les noms d'utilisateurs de MySQL sont totalement indépendants des utilisateurs Windows.Notez que Windows 95/98/Me ne supporte pas la création des pipes nommés. Sur ces plate-formes, vous ne pouvez utiliser que des pipes pour vous connecter à un serveur MySQL distant, qui tournerait sur un hôte Windows NT/2000/XP (le serveur MySQL soit aussi supporter les pipes nommés, bien sur. Par exemple, utiliser `mysqld-opt` sous NT/2000/XP ne permettra pas d'utiliser les pipes nommés. Vous devez utiliser soit `mysqld-nt` , soit `mysqld-max-nt` ).

Si `mysqld` ne démarre pas, vérifiez le fichier `\mysql\data\mysql.err` pour voir si le serveur n'a pas placé un message d'erreur indiquant la cause du problème. Vous pouvez aussi essayer de démarrer le serveur avec la commande `mysqld --standalone` ; dans ce cas, vous pourriez voir apparaître des informations sur l'écran qui vous aideront à résoudre le problème.

La dernière option est de démarrer `mysqld` avec l'option `--standalone --debug` . Dans ce cas, `mysqld` va écrire le fichier de log `C:\mysqld.trace` qui devrait contenir l'explication du problème. Créer des fichiers de

traçage .

Utilisez `mysqld --help` pour afficher toutes les options que `mysqld` comprend!

### 2.6.2.2 Démarrer MySQL sur Windows NT, 2000 ou XP

Pour faire fonctionner MySQL avec TCP/IP sous Windows NT 4, vous devez installer le service pack 3 (ou plus récent)!

Normalement, vous devez installer MySQL comme un service Windows NT/2000/XP. Dans le cas où le serveur fonctionnerait déjà, commencez par le stopper, en utilisant cette commande :

```
C:\mysql\bin> mysqladmin -u root shutdown
```

Cela appelle l'utilitaire d'administation MySQL, avec l'utilisateur `` root '', qui est par défaut un `Administrateur` de MySQL. Notez bien que le système de droits de MySQL est totalement indépendant du système d'utilisateur de Windows.Maintenant, l'installation du serveur en tant que service :

```
C:\mysql\bin> mysqld-max-nt --install
```

Si des options sont nécessaires, vous devez les spécifier comme `` Start parameters '' dans l'utilitaire Windows `Services` avant de démarrer le service MySQL.L'utilitaire `Services` ( `Windows Service Control Manager` ) est disponible dans le `Windows Control Panel` (sous les `Outils d'administration` de Windows 2000). Il est recommandé de fermer les utilitaires de services durant l'installation `--install` ou la suppression `--remove` , pour éviter des erreurs étranges.

Pour des informations sur quel exécutable serveur utiliser, voyez Préparer l'environnement MySQL sous Windows .

Notez bien que depuis MySQL version 3.23.44, vous avez le choix de configurer le serveur en `Manual` au lieu d'automatique (si vous ne voulez pas que le service démarre en même temps que le serveur) :

```
C:\mysql\bin> mysqld-max-nt --install-manual
```

Le service est installé avec le nom `MySQL` . Une fois installé, il peut être immédiatement démarré depuis l'utilitaire `Services` , ou avec la commande en ligne `NET START MySQL` .Une fois qu'il fonctionne, `mysqld-max-nt` peut être arrété avec `mysqladmin` , depuis l'utilitaire de services, ou en utilisant la commande en ligne `NET STOP MySQL` .

Lorsqu'il fonctionne comme service, le système d'exploitation va automatiquement arréter MySQL au moment de l'exticntion. Avec MySQL versions inférieures à 3.23.47, Windows attendait simplement quelques secondes l'extinction, puis si cela durait trop, il tuait le processus, ce qui causait parfois des problèmes. Par exemple, au démarrage suivant, le gestionnaire de table `InnoDB` devait faire une restauration après crash. Depuis MySQL version 3.23.48, Windows attend plus longtemps que MySQL s'arrête. Si vous notez que ce temps n'est pas suffisant pour votre installation, il est recommandé de ne pas utiliser MySQL sous forme de service, mais à partir de la ligne de commande, et de l'éteindre manuellement avec `mysqladmin shutdown` .

Il y a ce problème avec Windows NT (mais pas Windows 2000/XP), qui, par défaut, attend 20 secondes l'extinction d'un service, puis tue le processus. Vous pouvez augmenter ce délai en ouvrant l'éditeur de base de registre `\winnt\system32\regedt32.exe` et en modifiant la valeur de `WaitToKillServiceTimeout` dans `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control` . Spécifiez une nouvelle valeur en millisecondes, par exemple 120000 pour que Windows NT attende désormais jusqu'à 120 secondes.

Notez que lorsqu'il fonctionne comme un service, `mysqld-max-nt` n'a pas d'accès à la console, et ainsi, aucun mesasge n'est disponible. Les erreurs sont alors stockées dans le fichier `c:\mysql\data\mysql.err` .

Si vous avez des problèmes à installer `mysqld-max-nt` comme service, essayer de le démarrer en utilisant le chemin complet :

```
C:\> C:\mysql\bin\mysqld-max-nt --install
```

Si cela ne fonctionne pas, vous pouvez faire démarrer proprement `mysqld-max-nt` en modifiant le chemin dans la base de registre.Si vous ne voulez pas démarrer `mysqld-max-nt` comme service, vous pouvez le démarrer comme ceci :

```
C:\> C:\mysql\bin\mysqld-max-nt --standalone
```

ou

```
C:\> C:\mysql\bin\mysqld --standalone --debug
```

Cette dernière méthode vous donnera une trace de débogage dans le fichier `C:\mysqld.trace` . Créer des fichiers de traçage .

### 2.6.2.3 Faire fonctionner MySQL sous Windows

MySQL supporte TCP/IP sur toutes les plate−formes Windows et les tunnels nommés sur NT/2000/XP. Par défaut, les tunnels nommés sont utilisés sur NT/2000/XP pour les connexions locales et TCP/IP pour tout les autres cas si le client à TCP/IP d'installé. Le nom d'hôte spécifie le protocole à utiliser :

| Nom d'hôte | Protocole |
|---|---|
| NULL (aucun) | Sur NT/2000/XP, essayer les tunnels nommés avant; si cela ne fonctionne pas, utiliser TCP/IP. Sur 9x/Me, TCP/IP est utilisé. |
| . | Tunnels nommés |
| localhost | TCP/IP vers l'hôte courant |
| hostname | TCP/IP |

Vous pouvez forcer un client MySQL à utiliser les tunnels nommés en spécifiant l'option `--pipe` ou en spécifiant le nom d'hôte `.` en tant que nom d'hôte. Utilisez l'option `--socket` pour spécifier le nom du tunnel.

Notez qu'à partir de la version 3.23.50, les tunnels nommés ne sont activés que si mysqld est démarré avec `--enable-named-pipe` . Il en est ainsi car certains utilisateurs ont rencontré des problèmes en coupant le serveur MySQL lorsque celui−ci utilise des tunnels nommés.

Vous pouvez vérifier si MySQL fonctionne en exécutant les commandes suivantes :

```
C:\> C:\mysql\bin\mysqlshow
C:\> C:\mysql\bin\mysqlshow -u root mysql
C:\> C:\mysql\bin\mysqladmin version status proc
C:\> C:\mysql\bin\mysql test
```

Si `mysqld` est lent à répondre aux connexions sur Windows 9x/Me, il y a probablement un problème avec vos DNS. Dans ce cas, démarrez `mysqld` avec `--skip-name-resolve` et utilisez `localhost` et les adresses IP dans les tables de droits MySQL. Vous pouvez aussi éviter les DNS lors de la connexion à un serveur `mysqld-nt` MySQL tournant sur NT/2000/XP en utilisant l'argument `--pipe` pour spécifier l'utilisation des tunnels nommés. Cela fonctionne pour la plupart des clients MySQL.Il y a deux versions de l'outil en ligne de commande MySQL :

| Binaire | Description |
|---|---|
| mysql | Compilé nativement sur Windows, ce qui fournit des possiblités très limitées d'édition de texte. |
| mysqlc | Compilé avec le compilateur Cygnus GNU et librairies, ce qui fournit l'édition readline . |

Si vous voulez utiliser `mysqlc.exe` , vous devez copier `C:\mysql\lib\cygwinb19.dll` dans le répertoire système de Windows ( `\windows\system` ou un endroit similaire).

Les privilèges par défaut sur Windows donnent un accès à tous les utilisateurs locaux pour toutes les bases de données sans spécifier de mot de passe. Pour rendre MySQL un peu plus sécurisé, vous devez définir un mot de passe pour tout les utilisateurs et supprimer la ligne dans la table

`mysql.user` qui contient `Host='localhost'` et `User=''` .

Vous devez aussi ajouter un mot de passe pour le nouvel utilisateur `root` . L'example suivant commence par supprimer l'utilisateur anonyme qui possède tous les privilèges, puis configure un mot de passe pour l'utilisateur `root` :

```
 C:\> C:\mysql\bin\mysql mysql
mysql> DELETE FROM user WHERE Host='localhost' AND User='';
mysql> QUIT
C:\> C:\mysql\bin\mysqladmin reload
C:\> C:\mysql\bin\mysqladmin -u root password votre_mot_de_passe
```

Après avoir changé le mot de passe, si vous voulez couper le serveur `mysqld` , vous pouvez le faire en utilisant la commande :

```
 C:\> mysqladmin --user=root --password=votre_mot_de_passe shutdown
```

Si vous utilisez la vielle version 3.21 de MySQL sous Windows, la commande précédente échouera avec l'erreur : `parse error near 'SET password'` . La solution est de mettre à jour à la dernière version de MySQL qui est maintenant disponible.

Avec les versions courantes de MySQL vous pouvez facilement ajouter des utilisateurs et changer les privilèges avec les commandes `GRANT` et `REVOKE` . Syntaxe de `GRANT` et `REVOKE` .

### 2.6.2.4 Connexion à un serveur MySQL distants, sous Windows avec SSH

Voici une note pour connecter un serveur MySQL avec une connexion sécurisée grâce à SSH (de David Carlson dcarlson@mplcomm.com ) :

- Installez un client SSH pour votre machine Windows. En tant qu'utilisateur, le meilleur que je connaisse est celui de `SecureCRT` de http://www.vandyke.com/ . Une autre option est `f-secure` de http://www.f-secure.com/ . Vous pouvez aussi en trouver d'autres de gratuit avec `Google` à http://directory.google.com/Top/Computers/Security/Products_and_Tools/Cryptography/SSH/Clients/Wi
.
- Démarrez votre client SSH Windows. Spécifiez `Host_Name = yourmysqlserver_URL_or_IP` . Spécifiez `userid=your_userid` pour vous loger dans votre serveur (probablement avec un mot de passe et un nom d'utilisateur différent).
- Configurez le forward de port. Faites soit un forward distant (spécifiez `local_port: 3306` , `remote_host: yourmysqlservername_or_ip` , `remote_port: 3306` ) soit un forward local (spécifiez `port: 3306` , `host: localhost` , `remote port: 3306` ).
- Sauvez le tout, sinon vous devrez le refaire la prochaine fois.
- Connectez vous à votre serveur avec la session SSH que vous venez de créer.
- Sous votre machine Windows, démarrez une application ODBC (comme Access).
- Créez un nouveau fichier dans Windows et reliez le avec MySQL en utilisant le pilote ODBC de la même façon que vous le feriez habituellement, hormis le fait que vous devrez taper `localhost` comme hôte serveur au lieu de `yourmysqlservername` .

Vous avez maintenant une connexion ODBC avec un serveur MySQL distant, et sécurisée avec SSH.

### 2.6.2.5 Paratger les données entre plusieurs disque sous Windows

A partir de la version 3.23.16 de MySQL, les serveurs `mysqld-max` et `mysql-max-nt` dans les distributions MySQL sont compilés avec l'option `-DUSE_SYMDIR` . Cela vous permet de placer une base de données sur un disque différent en lui ajoutant un lien symbolique (c'est d'une certaine manière similaire au fonctionnement des liens symboliques sous Unix).

Sous Windows, vous créez un lien symbolique vers la base de données en créant un fichier qui contient le cheminvers le répertoire de destination et en le sauvant dans le répertoire `mysql_data` sous le nom `database.sym` . Notez que le lien symbolique ne sera utilisé que si le répertoire `mysql_data_dir\database` n'existe pas.

Par exemple, si le répertoire des données de MySQL est `C:\mysql\data` et que vous voulez avoir une base de données `foo` située dans `D:\data\foo` , vous devez créer le fichier `C:\mysql\data\foo.sym` qui contient le texte `D:\data\foo\` . Après cela, toutes les tables crées dans `foo` le seront dans `D:\data\foo` .

Notez qu'à cause de la pénalité sur la vitesse que vous obtenez en ouvrant toutes les tables, nous n'avons pas activé cela par défaut, même si vous avez compilé MySQL avec le support des liens symboliques. Pour les activer, vous devez placer dans votre fichier `my.cnf` ou `my.ini` l'entrée suivante :

```
 [mysqld]
use-symbolic-links
```

En MySQL 4.0 nous activerons les liens symboliques par défaut. Vous devrez alors utiliser l'option `skip-symlink` pour les désactiver.

### 2.6.2.6 Compiler les clients MySQL sous Windows

Dans vos fichiers sources, vous devez inclure `windows.h` avant `mysql.h` :

```
 #if defined(_WIN32) || defined(_WIN64)
#include <windows.h>
#endif
#include <mysql.h>
```

Vous pouvez soit lier votre code avec la librairie dynamique `libmysql.lib` , qui est juste une interface pour charger `libmysql.dll` à la demande, soit lier avec la librairie statique `mysqlclient.lib` .

Notez que puisque les librairies mysqlclient sont compilées en tant que librairies threadées, vous devez aussi compiler votre code pour qu'il soit multi-thréadé !

### 2.6.2.7 MySQL pour Windows face à MySQL pour Unix

MySQL pour Windows a prouvé qu'il était très stable. Cette version de MySQL a les mêmes fonctionnalités que la version Unix, a quelques exceptions :

*Windows 95 et les threads*
: Windows 95 perf environs 200 octets de mémoire central lors de la création de chaque thread. Chaque connexion MySQL crée un nouveau thread, ce qui fait qu'il n'est pas recommandé d'exécuter `mysqld` pour des durées longues sur Windows 95 si votre serveur gère de nombreuses connexions. Les autres versions de Windows ne souffrent pas du même problème.

*Lectures concurrentes*
: MySQL dépends des fonctions `pread()` et `pwrite()` pour être capable de mêler des `INSERT` et des `SELECT` . Actuellement, nous utilise mutexes pour émuler les fonctions `pread()` / `pwrite()` . Nous allons, à long terme, remplacer ce niveau d'interface par une interface virtuelle de façon à ce que nous puissions utiliser l'interface `readfile()` / `writefile()` de Windows NT/2000/XP pour gagner de la vitesse. L'implémentation courante limite le nombre de fichiers ouverts par MySQL à 1024, ce qui signifie que vous ne pouvez pas utiliser d'aussi nombreux threads concurrents sur NT/2000/XP que sur Unix.

*Blocking read*
: MySQL utilise une lecture bloquée pour chaque connexion. Cela signifie que :

Une connexion ne sera pas déconnectée automatiquement après 8 heures d'inactivité, comme c'est le cas sous Unix.
Si une connexion se bloque, il est impossible de la détruire sans tuer MySQL. `mysqladmin kill` ne fonctionne pas sur une connexion endormie.
`mysqladmin shutdown` ne peut pas s'exécuter tant qu'il y a des connexions qui dorment.

Nous envisageons de corriger ce problème, lorsque les développeurs Windows auront fourni un palliatif.

`DROP DATABASE`

Vous ne pouvez pas détruire une base qui est utilisée par un autre thread.

*Killing MySQL from the task manager*

Vous ne pouvez pas tuer MySQL depuis le gestionnaire de tâche ou avec un utilitaire d'extinction de Windows 95. Vous devez l'éteindre avec `mysqladmin shutdown` .

*Noms sensibles à la casse*

Les noms de fichiers sont insensibles à la casse sous Windows, ce qui fait que les noms de tables et de bases ne sont pas sensibles à la casse pour MySQL sur Windows. La seule restriction est que les noms de tables et de bases doivent être donnés avec même casse dans le nom (tout en majuscules, ou en minuscules). Sensibilité à la casse pour les noms .

*Le caractère* `'\'`

Les composants d'un chemin sont séparés par le caractère `'\'` sous Windows, qui est aussi le caractère de protection de MySQL. Si vous utilisez la commande `LOAD DATA INFILE` ou `SELECT ... INTO OUTFILE` , vous devez doubler le caractère `'\'` :

```
 mysql> LOAD DATA INFILE "C:\\tmp\\skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:\\tmp\\skr.txt' FROM skr;
```

Alternativement, vous pouvez utiliser les noms de fichiers au format Unix, avec le caractère `'/'` :

```
 mysql> LOAD DATA INFILE "C:/tmp/skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:/tmp/skr.txt' FROM skr;
```

`Can't open named pipe` *erreur*

Si vous utilisez MySQL 3.22 sous NT avec les derniers clients mysql, vous allez obtenir cette erreur :

```
 error 2017: can't open named pipe to host: . pipe...
```

Ceci est dû au fait que les versions modernes de MySQL utilisent des pipes nommés sous NT, par défaut. Pour éviter cette erreur, vous devez utiliser l'option `--host=localhost` sur les nouveaux clients, ou bien créer le fichier d'options `C:\my.cnf` , qui contiendra les informations suivantes :

```
 [client]
host = localhost
```

Depuis la version 3.23.50, les pipes nommés sont les seuls activés si `mysqld` est démarré avec `--enable-named-pipe` .

*Erreur* `Access denied for user`

Si vous rencontrez l'erreur `Access denied for user: 'utilisateur@unknown' to database 'mysql'` lors de l'accès au serveur MySQL sur la même machine, cela signifie que MySQL ne peut résoudre proprement votre nom d'hôte.Pour corriger cela, vous devriez créer un fichier `\windows\hosts` dans l'information suivante :

```
 127.0.0.1       localhost
```

`ALTER TABLE`

Lorsque vous exécutez la commande `ALTER TABLE` , la table est verrouillé, empêchant les autres threads d'y accéder. Cela est lié au fait que sous Windows, vous ne pouvez pas effacer un fichier qui est en cours d'utilisation par d'autres threads : à l'avenir, vous pourrions trouver un moyen de contourner ce problème.

*La commande* `DROP TABLE` *sur une table qui est utilisée dans le cadre*

d'un `MERGE` ne fonctionne pas sous Windows, car le gestionnaire de `MERGE` garde la carte des tables cachée de la couche supérieure de MySQL. Comme Windows ne vous autorise pas à effacer des fichiers qui sont ouverts, vous devez d'abord vider de la mémoire toutes les tables du `MERGE` (avec la commande `FLUSH TABLES` ) puis effacer la table `MERGE` avant d'effacer les tables. Nous allons corriger cela lorsque nous introduirons la

notion de `VIEW` s.

*Les directives* `DATA DIRECTORY` *et* `INDEX DIRECTORY` *de* `CREATE TABLE` sont ignorées sous Windows, car Windows ne supporte pas les liens symboliques.

Voici quelques problèmes connus et pas encore corrigés, si jamais quelqu'un souhaite nous aider sur la version Windows :

- Mettre en place une librairie pour un utilisateur simple `MYSQL.DLL` . Elle inclurait toutes les caractéristiques d'un serveur MySQL standard, hormis la création de threads. Cela permettra d'inclure facilement MySQL dans des applications qui n'ont pas vraiment besoin du support client, ou qui n'ont pas besoin d'accéder à un serveur distant.
- Ajouter des icônes sympas pour le démarrage et l'arrêt de MysQL, dans l'installateur.
- Lorsque `mysqld` est enregistré comme service avec `--install` (sous NT) il serait bien de pouvoir ajouter les options par défaut de la ligne de commande. Pour le moment, le palliatif consiste à lister les paramètres dans le fichier `C:\my.cnf` .
- Il serait vraiment pratique de pouvoir arrêter le processus `mysqld` depuis le gestionnaire de tâches. Pour le moment, il faut passer par `mysqladmin shutdown` .
- Le port de `readline` sur Windows pour pouvoir l'utiliser avec l'outil de ligne de commande `mysql` .
- Des versions graphiques des clients standards MySQL ( `mysql` , `mysqlshow` , `mysqladmin` et `mysqldump` ) seraient bien.
- Il serait bien si les fonctions de lecture et d'écriture sur les sockets de `net.c` pouvaient être interrompues. Cela rendrait possible l'arrêt des threads en court avec `mysqladmin kill` sous Windows.
- `mysqld` se lance toujours avec la locale "C" et non pas avec la locale par défaut. Nous voulons que `mysqld` utilise la configuration de localisation pour l'ordre de tri.
- Ajouter des macros pour utiliser les méthodes rapides d'incrément/décrément compatibles avec les threads, fourni par Windows.

D'autres spécificités de Windows sont décrites dans le fichier `README` qui est livré avec la distribution Windows de MySQL.

## 2.6.3 Remarques pour Solaris

Sous Solaris, vous pouvez renontrer des problèmes avant même d'avoir désarchivé la distribution MySQL! Le programme `tar` de Solaris ne peut pas manipuler de noms de fichiers longs, provoquant les messages suivants quand vous décompressez MySQL :

```
 x mysql-3.22.12-beta/bench/Results/ATIS-mysql_odbc-NT_4.0-cmp-db2,\
informix,ms-sql,mysql,oracle,solid,sybase, 0 bytes, 0 tape blocks
tar: directory checksum error
```

Dans ce cas, vous devez utiliser GNU `tar` ( `gtar` ) pour désarchiver la distribution. Vous pouvez en trouver une copie précompilée pour Solaris sur http://www.mysql.com/Downloads/ .La gestion native des threads Sun fonctionne uniquement depuis Solaris 2.5. Pour les versions 2.4 et antérieures, MySQL utilisera automatiquement les MIT-pthreads. Remarques sur MIT-pthreads .

Vous pouvez rencontrer les erreurs suivantes lors du configure:

```
 checking for restartable system calls... configure: error can not run test
programs while cross compiling
```

Cela signifie que l'installation de votre compilateur est défectueuse! Dans ce cas, vous devez mettre à jour votre compilateur en faveur d'une version plus récente. Vous pouvez aussi résoudre le problème en inérant la ligne suivante dans le fichier `config.cache` :

```
ac_cv_sys_restartable_syscalls=$&#123;ac_cv_sys_restartable_syscalls='no'&#125;
```

Si vous utilisez Solaris sur une architecture SPARC, nous recommandons `gcc` 2.95.2 comme compilateur. Vous pouvez le trouver sur http://gcc.gnu.org/ . Notez que `egcs` 1.1.1 et `gcc` 2.8.1 ne fonctionnent pas correctement sur SPARC!La ligne `configure` recommandée dans le cas de l'utilisation de `gcc` 2.95.2 est:

```
 CC=gcc CFLAGS="-O3" \
CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory --enable-assembler
```

Si vous avez une machine UltraSPARC, vous pouvez gagner 4% de performances supplémentaires en ajoutant "−mcpu=v8 −Wa,−xarch=v8plusa" à CFLAGS et CXXFLAGS.Si vous utilisez le compilateur Forte 5.0 (et supérieur) de Sun, vous pouvez lancer `configure` de la façon suivante :

```
 CC=cc CFLAGS="-Xa -fast -native -xstrconst -mt" \
CXX=CC CXXFLAGS="-noex -mt" \
./configure --prefix=/usr/local/mysql --enable-assembler
```

Vous pouvez créer un binaire 64 bits avec :

```
 CC=cc CFLAGS="-Xa -fast -native -xstrconst -mt -xarch=v9" \
CXX=CC CXXFLAGS="-noex -mt -xarch=v9" \
./configure --prefix=/usr/local/mysql --enable-assembler
```

Lors de bancs de tests MySQL, nous avons gagné 4% en vitesse sur une UltraSPARC en utilisant Forte 5.0 en mode 32 bits plutôt que gcc 3.2 avec les marqueurs −mcpu.

Si vous créez un binaire 64 bits, il est de 4% plus lent que le binaire 32 bits, mais en contrepartie vous pouvez gérer davantage de treads et de mémoire.

Si vous rencontrez des problèmes avec `fdatasync` ou `sched_yield` , vous pouvez les résoudre en ajoutant `LIBS=-lrt` à la ligne configure.

Le paragraphe suivant ne s'applique qu'aux compilateurs plus anciens que WorkShop 5.3 :

Vous pouvez avoir à modifier le script `configure` et changer la ligne :

```
#if !defined(__STDC__) || __STDC__ != 1
```

en :

```
#if !defined(__STDC__)
```

Si vous activez `__STDC__` avec l'option `-Xc` , le compilateur Sun ne peut pas compiler avec le fichier d'entêtes `pthread.h` de Solaris. C'est un bogue de Sun (compilateur ou fichier d'inclusion défectueux).

Si `mysqld` génère les messages d'erreur suivants lorsque vous le lancez, cela est du au fait que vous avez compilé MySQL avec le compilateur de Sun sans activer l'option multi−threads ( `-mt` ):

```
libc internal error: _rmutex_unlock: rmutex not held
```

Ajoutez `-mt` à `CFLAGS` et `CXXFLAGS` puis réessayez.

Si vous utilisez la version SFW de gcc (fournie avec Solaris 8), vous devez ajouter `/opt/sfw/lib` à la variable d'environnement `LD_LIBRARY_PATH` avant de lancer le configure.

Si vous utilisez le gcc disponible sur `sunfreeware.com` , vous pouvez rencontrer de nombreux problèmes. Vous devriez recompiler gcc et les GNU binutils sur la machine à partir de laquel vous les utiliserez, afin d'éviter tout souci.

Si vous obtenez l'erreur suivante lorsque vous compilez MySQL avec `gcc` , cela signifie que votre `gcc` n'est pas configuré pour votre version de Solaris :

```
 shell> gcc -O3 -g -O2 -DDBUG_OFF  -o thr_alarm ...
./thr_alarm.c: In function `signal_hand':
./thr_alarm.c:556: too many arguments to function `sigwait'
```

La meilleure chose à faire dans ce cas est d'obtenir la version la plus récente de `gcc` et de compiler

avec votre `gcc` actuel! Au moins pour Solaris 2.5, la plupart des versions binaires de `gcc` ont d'anciens fichiers d'inclusion inutilisables qui planteront les programmes qui utilisent les threads (ainsi probablement d'autres programmes)!

Solaris ne fournit pas de versions statiques de toutes les librairies système ( `libpthreads` et `libdl` ), vous ne pouvez donc pas compiler MySQL avec `--static` . Si vous tentez de le faire, vous obtiendrez l'erreur :

```
ld: fatal: library -ldl: not found

ou

undefined reference to `dlopen'

ou

cannot find -lrt
```

Si de nombreux processus essaient de se connecter très rapidement à `mysqld` , vous verrez cette erreur dans le journal MySQL :

```
Error in accept: Protocol error
```

Pour éviter cela, vous pouvez lancer le serveur avec l'option `--set-variable back_log=50` . Veuillez noter que `--set-variable` est déprécié depuis MySQL 4.0, utilisez uniquement `--back_log=50` . Options en ligne de commande de `mysqld` .Si vous liez votre propre client MySQL, vous pouvez avoir l'erreur suivante quand vous le lancez :

```
 ld.so.1: ./my: fatal: libmysqlclient.so.#:
open failed: No such file or directory
```

Le problème peut être évité avec l'une des méthodes suivantes :

- Liez le client avec le marqueur suivant (à la place de `-Lpath` ) : `-Wl,r/full-path-to-libmysqlclient.so` .
- Copiez `libmysqclient.so` dans `/usr/lib` .
- Ajoutez le chemin du répertoire où `libmysqlclient.so` est installé à la variable d'environnement `LD_RUN_PATH` avant de lancer votre client.

Si vous avez des soucis avec configure qui essaie de lier avec `-lz` et que vous n'avez pas installé `zlib` , vous avez deux solutions :

- Si vous voulez utiliser le protocole compresseé de communication, vous devrez vous procurer et installer zlib sur ftp.gnu.org.
- Configurez avec `--with-named-z-libs=no` .

Si vous utilisez gcc et rencontrez des problèmes en chargeant la fonction `UDF` dans MySQL, essayez d'ajouter `-lgcc` à la ligne de liaison de la fonction `UDF` .

Si vous voulez que MySQL se lance automatiquement, vous pouvez copier `support-files/mysql.server` dans `/etc/init.d` et créer un lien symbolique pointant dessus et s'appelant `/etc/rc3.d/S99mysql.server` .

Comme Solaris ne supporte pas les fichiers core pour les applications `setuid()` , vous ne pouvez pas obtenir un fichier core de `mysqld` si vous utilisez l'option `--user` .

### 2.6.3.1 Notes relatives à Solaris 2.7/2.8

Vous pouvez normalement utiliser les binaires Solaris 2.6 sur Solaris 2.7 et 2.8. La plupart des fonctionnalitées de Solaris 2.6 s'appliquent aussi à Solaris 2.7 et 2.8.

Notez que la version 3.23.4 de MySQL et plus doivent être capables de détecter automatiquement les nouvelles versions de Solaris et d'activer les parades pour résoudre les problèmes suivants !

Solaris 2.7 / 2.8 ont quelques bogues dans les fichiers inclus. Vous verrez peut−être l'erreur suivante en utilisant `gcc` :

```
 /usr/include/widec.h:42: warning: `getwc' redefined
/usr/include/wchar.h:326: warning: this is the location of the previous
definition
```

Si cela arrive, vous pouvez faire ce qui suit pour résoudre ce problème :Copiez `/usr/include/widec.h` vers `.../lib/gcc−lib/os/gcc−version/include` et changez la ligne 41 de :

```
 #if     !defined(lint) && !defined(__lint)

 en

 #if     !defined(lint) && !defined(__lint) && !defined(getwc)
```

Alternativement, vous pouvez éditer directement le fichier `/usr/include/widec.h` . De toutes façons, après avoir apporté la correction, vous devez effacer `config.cache` et exécuter `configure` à nouveau !Si vous obtenez des erreurs comme celles qui suivent quand vous exécutez `make` , c'est parce que `configure` n'a pas détecté le fichier `curses.h` (probablement à cause de l'erreur dans `/usr/include/widec.h` ) :

```
 In file included from mysql.cc:50:
/usr/include/term.h:1060: syntax error before `,'
/usr/include/term.h:1081: syntax error before `;'
```

La solution est de faire l'une des choses qui suit :

- Configurez avec `CFLAGS=−DHAVE_CURSES_H CXXFLAGS=−DHAVE_CURSES_H ./configure` .
- Editez `/usr/include/widec.h` comme indiqué plus haut et ré−exécutez configure.
- Effacez la ligne `#define HAVE_TERM` di fichier `config.h` et exécutez `make` à nouveau.

Si vous obtenez une erreur disant que votre programme de liaison ne peut trouver `−lz` lors de la liaison du programme de votre client, le problème est probablement que votre fichier `libz.so` est installé dans `/usr/local/lib` . Vous pouvez corriger ceci en utilisant l'une des méthodes suivantes :

- Ajoutez `/usr/local/lib` à `LD_LIBRARY_PATH` .
- Ajoutez un lien vers `libz.so` à partir de `/lib` .
- Si vous utilisez Solaris 8, vous pouvez installer la zlib optionnelle à partir de votre CD Solaris 8.
- Configurez MySQL avec l'option `−−with−named−z−libs=no` .

### 2.6.3.2 Remarques pour Solaris x86

Sous Solaris 2.8 sur x86, `mysqld` va crasher (core dump) si vous l'exécutez 'strip'.

Si vous utilisez `gcc` ou `egcs` sous Solaris x86 et que vous rencontrez des problèmes avec des coredumps, lorsqu'il y a de la charge, il est recommandé d'utiliser la commande de `configure` suivante :

```
 CC=gcc CFLAGS="−O3 −fomit−frame−pointer −DHAVE_CURSES_H" \
CXX=gcc \
CXXFLAGS="−O3 −fomit−frame−pointer −felide−constructors −fno−exceptions \
−fno−rtti −DHAVE_CURSES_H" \
./configure −−prefix=/usr/local/mysql
```

Cela va éviter les problèmes avec la librairie `libstdc++` et avec les exceptions C++.

Si cela ne vous aide pas, il est recommandé de compiler une version de débogage, et de l'exécuter avec un fichier de trace sous `gdb` . Déboguer mysqld sous gdb .

## 2.6.4 Notes relatives à BSD

Cette section fournit des informations pour les différentes variétés de BSD, ainsi que les versions spécifiques de celles−ci.

### 2.6.4.1 Notes relatives à FreeBSD

FreeBSD 3.x est recommandé pour exécuter MySQL vu que le package des threads est plus intégré.

La façon la plus facile et la plus conseillée d'installer est d'utiliser les ports du serveur et du client MySQL disponibles sur http://www.freebsd.org/ .

Les utiliser vous donnera :

- Un MySQL fonctionnant avec toutes les optimisations connues pour votre version active de FreeBSD.
- Configuration et construction automatique.
- Scripts de démarrage installés dans /usr/local/etc/rc.d.
- La possibilité de voir tous les fichiers installés avec pkg_info −L. Et la possibilité de les effacer tous avec pkg_delete si vous ne voulez plus de MySQL sur cette machine.

Il est recommandé d'utiliser les MIT−pthreads sur FreeBSD 2.x et les threads natifs sur les versions 3 et plus. Il est possible de faire fonctionner le tout avec les threads natifs sur les dernières versions 2.2.x mais vous rencontrerez probablement des problèmes en coupant `mysqld` .Le `Makefile` de MySQL requière GNU make ( `gmake` ) pour fonctionner. Si vous voulez compiler MySQL vous devez d'abord installer GNU `make` .

Assurez−vous que votre configuration de la résolution des noms est bonne. Sinon, vous airez peut−être quelques problèmes lors de la connexion à `mysqld` .

Assurez vous que l'entrée `localhost` dans le fichier `/etc/hosts` est correcte (sinon, vous aurez des problèmes pour vous connecter à la base de données). Le fichier `/etc/hosts` doit commencer par :

```
127.0.0.1       localhost localhost.votre.domaine
```

La manière recommandée de compiler et d'installer MySQL sur FreeBSD avec gcc (2.95.2 et plus) est :

```
CC=gcc CFLAGS="−O2 −fno−strength−reduce" \
CXX=gcc CXXFLAGS="−O2 −fno−rtti −fno−exceptions −felide−constructors \
−fno−strength−reduce" \
./configure −−prefix=/usr/local/mysql −−enable−assembler
gmake
gmake install
./scripts/mysql_install_db
cd /usr/local/mysql
./bin/mysqld_safe &
```

Si vous vous apercevez que `configure` utilisera les MIT−pthreads, vous devriez lire les notes relatives aux MIT−pthreads. Remarques sur MIT−pthreads .

Si vous obtenez une erreur de `make install` disant qu'il ne peut trouver `/usr/include/pthreads` , `configure` n'a pas détecté que vous avez besoin des MIT−pthreads. Cela est corrigé en exécutant ces commandes :

```
shell> rm config.cache
shell> ./configure −−with−mit−threads
```

FreeBSD est aussi connu pour avoir une petite limite de gestionnaires de fichiers par défaut. Fichier non trouvé . Décommentez la section ulimit −n dans safe_mysqld ou enlevez la limite pour

l'utilisateur `mysqld` dans /etc/login.conf (et regénérez le avec cap_mkdb). Assurez−vous aussi de définir la classe appropriée pour cet utilisateur dans le fichier des mots de passe si vous n'utilisez pas celui par défaut. (utilisez : chpass nom−utilisateur−mysqld). `safe_mysqld` , le script père de `mysqld` .

Si vous avez beaucoup de mémoire, vous devriez penser à recompiler le noyau pour permettre à MySQL d'utiliser plus de 512M de RAM. Regardez l' `option MAXDSIZ` dans le fichier de configuration de LINT pour plus d'informations.

Si vous avez des problèmes avec la date courante dans MySQL, configurer la variable d'environnement `TZ` aidera sûrement. Variables d'environnement MySQL .

Pour obtenir un système sécurisé et stable, vous ne devez utiliser que les noyaux FreeBSD marqués `−RELEASE` .

### 2.6.4.2 Notes concernant NetBSD

Pour compiler sur NetBSD vous aurez besoin de GNU `make` . Sinon, la compilation stoppera lorsque `make` essayera d'exécuter `lint` sur les fichiers C++.

### 2.6.4.3 Notes relatives à OpenBSD 2.5

Dans la version 2.5 de OpenBSD, vous pouvez compiler MySQL avec les threads natifs avec les options suivantes :

```
CFLAGS=−pthread CXXFLAGS=−pthread ./configure −−with−mit−threads=no
```

### 2.6.4.4 Notes relatives à OpenBSD 2.8

Nos utilisateurs nous ont informé que OpenBSD 2.8 comporte un bogue des threads qui pose quelques problèmes avec MySQL. Les développeurs d'OpenBSD ont résolu ce problème, mais depuis le 25 janvier 2001 ce n'est disponible que dans la branche ``−current''. Les symptômes de ce bogue sont : réponses lentes, beaucoup de charge, grande utilisation du CPU, et crashes.

Si vous obtenez une erreur comme `Error in accept:: Bad file descriptor` ou erreur 9 en essayant d'ouvrir les tables ou les dossiers, le problème est probablement que vous n'avez pas alloué assez de descripteurs de fichiers à MySQL.

Dans ce cas, essayez de démarrer `safe_mysqld` en tant que root avec les options suivantes :

```
−−user=mysql −−open−files−limit=2048
```

### 2.6.4.5 Notes relatives aux versions 2.x de BSD/OS

Si vous obtenez l'erreur suivante lors de la compilation de MySQL, votre valeur de `ulimit` pour la mémoire vortuelle est trop petite :

```
 item_func.h: In method `Item_func_ge::Item_func_ge(const Item_func_ge &)':
item_func.h:28: virtual memory exhausted
make[2]: *** [item_func.o] Error 1
```

Essayez d'utiliser `ulimit −v 80000` et exécutez `make` à nouveau. Si cela ne fonctionne pas et que vous utilisez `bash` , essayez de passer à `csh` ou `sh` ; quelques utilisateurs de BSDI ont reporté des problèmes

avec `bash` et `ulimit` .

Si vous utilisez `gcc` , vous aurez peut−être aussi à utiliser l'option `--with-low-memory` de `configure` pour pouvoir compiler `sql_yacc.cc` .

Si vous avez des problèmes avec la date courante dans MySQL, configurer la variable `TZ` vous aidera probablement. Variables d'environnement MySQL .

### 2.6.4.6 Notes relatives aux versions 3.x de BSD/OS

Mettez à jour à la version 3.1 de BSD/OS. Si cela n'est pas possible, installez le patch BSDIpatch M300−038.

Utilisez la commande suivante lors de la configuration de MySQL :

```
shell> env CXX=shlicc++ CC=shlicc2 \
      ./configure \
          --prefix=/usr/local/mysql \
          --localstatedir=/var/mysql \
          --without-perl \
          --with-unix-socket-path=/var/mysql/mysql.sock
```

Ce qui suit fonctionne aussi :

```
shell> env CC=gcc CXX=gcc CXXFLAGS=-O3 \
      ./configure \
          --prefix=/usr/local/mysql \
          --with-unix-socket-path=/var/mysql/mysql.sock
```

Vous pouvez changer les répertoires si vous voulez, ou utiliser les valeurs par défaut en ne spécifiant pas de chemins.Si vous avez des problèmes de performances alors que la charge est petite, essayez d'utiliser l'option `--skip-thread-priority` de `mysqld` ! Cela exécutera tous les threads avec la même priorité; Sur la version 3.1 de BSDI, cela donne de meilleures performances (en attendant que BSDI corrige sont gestionnaire de threads).

Si vous obtenez l'erreur `virtual memory exhausted` durant la compilation, vous devez essayer en utilisant `ulimit -v 80000` et exécutant `make` à nouveau. Si cela ne fonctionne pas et que vous utilisez `bash` , essayez de passer à `csh` ou `sh` ; quelques utilisateurs de BSDI ont reporté des problèmes avec `bash` et `ulimit` .

### 2.6.4.7 Notes relatives aux versions 4.x de BSD/OS

Les versions 4.x de BSDI ont quelques bogues relatifs aux threads. Si vous voulez utiliser MySQL sur ce système, vous devez installer tous les patches liés aux threads. vous devez au moins installer M400−023.

Sur quelques systèmes avec une version 4.x de BSDI, vous pouvez rencontrer des problèmes avec les librairies partagées. Le symptôme est que vous ne pouvez utiliser aucun programme client, comme par exemple, `mysqladmin` . Dans ce cas, vous devez le reonfigurer pour qu'il n'utilise pas les librairies paratgées avec l'option `--disable-shared` de configure.

Quelques utilisateurs ont eu avec BSDI 4.0.1 un problème faisant qu'après un bout de temps, le binaire `mysqld` ne peut plus ouvrir de tables. Cela est du au fait qu'un bogue relatif au système ou à la librairie fait changer de répertoire à `mysqld` sans qu'on ne l'ait demandé !

La solution est soit de mettre à jour vers la version 3.23.34 ou de supprimer la ligne `#define HAVE_REALPATH` de `config.h` après avoir exécuté `configure` et avant d'exécuter `make` .

Notez que ce qui précède signifie que vous ne pouvez pas créer de lien symbolique sur un dossier de bases de données vers un autre dossier de bases de données ou lier une table symboliquement vers une autre base de données sur BSDI ! (Créer un lien symbolique vers un autre disque fonctionne).

## 2.6.5 Notes relatives à Mac OS X

### 2.6.5.1 Mac OS X 10.x

MySQL devrait fonctionner sans problème avec les versions Mac OS X 10.x (Darwin). Vos n'avez pas besoin du patch pour les pthread sur cet OS!

Cela s'applique aussi à Mac OS X 10.x Server. La compilation pour la plate−forme serveur est identique à la compilation pour pa version client de Mac OS X. Toutes fois, notez que MySQL est pré−installé sur la version Server!

Installer MySQL sous Mac OS X .

### 2.6.5.2 Mac OS X Server 1.2 (Rhapsody)

Avant d'essayer de configurer MySQL sur Mac OS X Server, vous devez d'abord installer le paquet pthread depuis http://www.prnet.de/RegEx/mysql.html .

Notre programme binaire Mac OS X est compilé sur Darwin 6.3 avec la liste suivante d'options de configuration :

```
 CC=gcc CFLAGS="−O3 −fno−omit−frame−pointer" CXX=gcc \
CXXFLAGS="−O3 −fno−omit−frame−pointer −felide−constructors \
−fno−exceptions −fno−rtti" ./configure −−prefix=/usr/local/mysql \
−−with−extra−charsets=complex −−enable−thread−safe−client \
−−enable−local−infile −−disable−shared
```

Vous pouvez aussi ajouter des alias dans votre fichier de ressources Shell, pour accéder à `mysql` et `mysqladmin` depuis la ligne de commande :

```
 alias mysql '/usr/local/mysql/bin/mysql'
alias mysqladmin '/usr/local/mysql/bin/mysqladmin'
```

Alternativement, vous pouvez simplement ajouter `/usr/local/mysql/bin` à votre variable d'environnement `PATH` , i.e. en ajouter la ligne suivante à votre fichier `$HOME/.tcshrc` :

```
 setenv PATH $PATH:/usr/local/bin
```

## 2.6.6 Notes sur les autres Unix

### 2.6.6.1 Notes relatives à HP−UX pour les distributions binaires

Quelques distributions binaires de MySQL pour HP−UX sont distribués en tant que fichier depot HP et archive tar. Pour utiliser le fichier depot, vous devez avoir au moins HP−UX 10.x pour avoir accès aux outils depot de HP.

La version HP de MySQL a été compilée sur un serveur HP 9000/8xx sous HP−UX 10.20, et utilise les MIT−pthreads. Elle est connue pour bien fonctionner sous cette configuration. A partir de la version 3.22.26 de MySQL vous pouvez aussi compiler avec le package de threads natifs HP.

Autres configurations pouvant fonctionner :

- HP 9000/7xx avec HP−UX 10.20+
- HP 9000/8xx avec HP−UX 10.30

Les configurations suivantes semblent ne jamais vouloir fonctionner :

- HP 9000/7xx ou 8xx avec HP−UX 10.x où x < 2
- HP 9000/7xx ou 8xx avec HP−UX 9.x

Pour installer la distribution, utilisez l'une des commandes suivantes, où `/chemin/vers/depot` est le chemin complet vers le fichier dépôt :

- Pour tout installer, serveur, client et outils de développement :
```
shell> /usr/sbin/swinstall −s /chemin/vers/depot mysql.full
```
- Pour n'installer que le serveur :
```
shell> /usr/sbin/swinstall −s /chemin/vers/depot mysql.server
```
- Pour n'installer que le package client :
```
shell> /usr/sbin/swinstall −s /chemin/vers/depot mysql.client
```
- Pour n'installer que les outils de développement :
```
shell> /usr/sbin/swinstall −s /chemin/vers/depot mysql.developer
```

Le depot place les binaires et les librairies dans `/opt/mysql` et les données dans `/var/opt/mysql` .Le depot crée aussi les entrées appropriées dans `/etc/init.d` et `/etc/rc2.d` pour démarrer le serveur automatiquement lors du boot. Evidemment, il faut être `root` pour installer.

Pour installer la distribution HP−UX tar.gz, vous devez avoir une copie de GNU `tar` .

### 2.6.6.2 Notes relatives à la version 10.20 de HP−UX

Il y a quelques petits problèmes que vous pourrez rencontrer lors de la compilation de MySQL sur HP−UX. Nous recommandons l'utilisation de `gcc` au lieu du compilateur natif de HP−UX, car `gcc` produit un meilleur code !

Nous recommandons l'utilisation de gcc 2.95 sur HP−UX. N'utilisez pas les options de haute optimisation (comme −O6) car cela pourrait ne pas être sûr sur HP−UX.

La ligne de configuration suivante devrait fonctionner avec gcc 2.95 :

```
 CFLAGS="−I/opt/dce/include −fpic" \
CXXFLAGS="−I/opt/dce/include −felide−constructors −fno−exceptions \
−fno−rtti" CXX=gcc ./configure −−with−pthread \
−−with−named−thread−libs='−ldce' −−prefix=/usr/local/mysql −−disable−shared
```

La ligne de configuration suivante devrait fonctionner avec gcc 3.1 :

```
 CFLAGS="−DHPUX −I/opt/dce/include −O3 −fPIC" CXX=gcc \
CXXFLAGS="−DHPUX −I/opt/dce/include −felide−constructors −fno−exceptions \
−fno−rtti −O3 −fPIC" ./configure −−prefix=/usr/local/mysql \
−−with−extra−charsets=complex −−enable−thread−safe−client \
−−enable−local−infile  −−with−pthread \
−−with−named−thread−libs=−ldce −−with−lib−ccflags=−fPIC
−−disable−shared
```

### 2.6.6.3 HP−UX Version 11.x Notes

Pour les version 11.x de HP−UX nous recommandons MySQL 3.23.15 ou plus récent.

A cause de quelques bogues critiques dans les librairies standard de HP−UX, vous devez installer les patches suivants avant d'essayer de faire fonctionner MySQL sous HP−UX 11.0 :

```
 PHKL_22840 Streams cumulative
PHNE_22397 ARPA cumulative
```

Cela résoudra le problème de l'obtention de `EWOULDBLOCK` à partir de `recv()` et `EBADF` à partir de `accept()` dans les applications threadées.

Si vous utilisez `gcc` 2.95.1 sur un système HP−UX 11.x non−patché, vous obtiendrez l'erreur :

```
 In file included from /usr/include/unistd.h:11,
                 from ../include/global.h:125,
                 from mysql_priv.h:15,
                 from item.cc:19:
/usr/include/sys/unistd.h:184: declaration of C function ...
/usr/include/sys/pthread.h:440: previous declaration ...
In file included from item.h:306,
                 from mysql_priv.h:158,
                 from item.cc:19:
```

Le problème est que HP−UX ne définit pas `pthreads_atfork()` avec consistence. Il possède des prototypes en conflit dans `/usr/include/sys/unistd.h` :184 et `/usr/include/sys/pthread.h` :440 (détails ci−dessous).

Une solution est de copier `/usr/include/sys/unistd.h` dans `mysql/include` et éditer `unistd.h` en le changeant pour qu'il corresponde à la définition dans `pthread.h` . Voici les modifications :

```
 183,184c183,184
<      extern int pthread_atfork(void (*prepare)(), void (*parent)(),
<                                               void (*child)());
−−−
>      extern int pthread_atfork(void (*prepare)(void), void (*parent)(void),
>                                               void (*child)(void));
```

Après cela, la ligne de configuration suivante devrait fonctionner :

```
 CFLAGS="−fomit−frame−pointer −O3 −fpic" CXX=gcc \
CXXFLAGS="−felide−constructors −fno−exceptions −fno−rtti −O3" \
./configure −−prefix=/usr/local/mysql −−disable−shared
```

Si vous utilisez MySQL 4.0.5 avec le compilateur HP−UX, vous pouvez utiliser : (testé avec cc B.11.11.04):

```
 CC=cc CXX=aCC CFLAGS=+DD64 CXXFLAGS=+DD64 ./configure −−with−extra−character−set=complex
```

Vous pouvez ignorer toutes les erreurs de ce type :

```
 aCC: warning 901: unknown option: `−3': use +help for online documentation
```

Si vous obtenez l'erreur suivante de `configure` :

```
 checking for cc option to accept ANSI C... no
configure: error: MySQL requires a ANSI C compiler (and a C++ compiler).
Try gcc. See the Installation chapter in the Reference Manual.
```

Vérifiez que le chemin vers le compilateur KRne précède pas le chemin vers le compilateur C et C++ HP−UX.

Une autre raison qui pourrait vous empêcher de compiler, et le fait de n'avoir pas définit l'option `+DD64` ci−dessus.

### 2.6.6.4 Notes relatives à IBM−AIX

La détection automatique de `xlC` est absente de Autoconf, ce qui fait qu'un commande `configure` comme celle qui suit est requise lors de la compilation de MySQL (Cet exemple utilise le compilateur IBM) :

```
 export CC="xlc_r −ma −O3 −qstrict −qoptimize=3 −qmaxmem=8192 "
export CXX="xlC_r −ma −O3 −qstrict −qoptimize=3 −qmaxmem=8192"
export CFLAGS="−I /usr/local/include"
export LDFLAGS="−L /usr/local/lib"
```

```
export CPPFLAGS=$CFLAGS
export CXXFLAGS=$CFLAGS

 ./configure --prefix=/usr/local \
        --localstatedir=/var/mysql \
        --sysconfdir=/etc/mysql \
        --sbindir='/usr/local/bin' \
        --libexecdir='/usr/local/bin' \
        --enable-thread-safe-client \
        --enable-large-files
```

Ce sont les options utilisées pour compiler la distribution de MySQL qui peut être trouvée sur http://www-frec.bull.com/ .Si vous changez le `-O3` en `-O2` dans la ligne précédente, vous devez aussi enlever l'option `-qstrict` (c'est une limitation du compilateur IBM C).

Si vous utilisez `gcc` ou `egcs` pour compiler MySQL, vous **devez** utiliser l'option `-fno-exceptions` , vu que la gestion des exceptions de `gcc` / `egcs` n'est pas sûre pour les threads ! (Cela est testé avec `egcs` 1.1.) Il y a aussi quelques problèmes connus avec l'assembleur d'IBM, qui peuvent lui faire générer du mauvais code lors de son utilisation avec gcc.

Nous recommandons la ligne de `configure` suivante avec `egcs` et `gcc 2.95` sur AIX :

```
 CC="gcc -pipe -mcpu=power -Wa,-many" \
CXX="gcc -pipe -mcpu=power -Wa,-many" \
CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory
```

Le `-Wa,-many` est nécessaire pour que la compilation se passe sans problèmes. IBM est au courant de ce problème mais n'est pas pressé de le corriger à cause de l'existence du palliatif. Nous ne savons pas si `-fno-exceptions` est requise avec `gcc 2.95` , mais comme MySQL n'utilise pas les exeptions et que l'option en question génère un code plus rapide, nous vous recommandons de toujours utiliser cette option avec `egcs / gcc` .Si vous obtenez un problème avec le code de l'assembleur essayez en changeant l'option −mcpu=xxx pour l'adapter à votre processeur. Le plus souvent, on a besoin de power2, power, ou powerpc, et sinon 604 ou 604e. Je ne suis pas positif mais je pense que l'utilisation de "power" sera sûre la plupart du temps, même sur une machine power2.

Si vous ne savez pas quel est votre processeur, exécutez "uname −m", cela vous renverra une chaîne comme "000514676700", avec un format xxyyyyyymmss où xx et ss sont toujours des zéros, yyyyyy est un identifiant unique du système et mm est l'identifiant du CPU Planar. Une liste de ces valeurs peut être trouvée sur http://publib.boulder.ibm.com/doc_link/en_US/a_doc_lib/cmds/aixcmds5/uname.htm . Cela vous donnera un type et un modèle de machine que vous pouvez utiliser pour déterminer quel type de processeur vous avez.

Si vous avez des problèmes avec les signaux (MySQL se termine de manière imprévue lors des montées en charge) vous avez peut−être trouvé un bogue du système avec les threads et les signaux. Dans ce cas, vous pouvez demander à MySQL de ne pas utiliser les signaux en configuration avec :

```
 shell> CFLAGS=-DDONT_USE_THR_ALARM CXX=gcc \
       CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti \
       -DDONT_USE_THR_ALARM" \
       ./configure --prefix=/usr/local/mysql --with-debug --with-low-memory
```

Cela n'affecte pas les performances de MySQL, mais comporte un effet secondaire faisant en sorte que vous ne pourrez tuer les clients en état ``sleeping'' sur une connexion avec `mysqladmin kill` ou `mysqladmin shutdown` . A la place, le client se terminera lorsqu'il émmetra sa prochaine commande.Sur quelques versions de AIX, lier avec `libbind.a` fait vider son noyau à `getservbyname` (core dump). Il s'agit d'un bogue AIX et doit être remonté à IBM.

Pour AIX 4.2.1 et gcc vous devez apporter les modifications suivantes :

Après la configuration, éditez `config.h` et `include/my_config.h` et changez la ligne qui comporte

```
#define HAVE_SNPRINTF 1
```

en

```
#undef HAVE_SNPRINTF
```

Et finalement, dans `mysqld.cc` vous devez ajouter un prototype pour initgoups.

```
#ifdef _AIX41
extern "C" int initgroups(const char *,int);
#endif
```

Si vous avez besoin d'allouer beaucoup de mémoire au processus mysqld, il ne suffit pas de configurer 'ulimit −d unlimited'. Vous aurez aussi à configurer dans `mysqld_safe` quelque chose comme :

```
export LDR_CNTRL='MAXDATA=0x80000000'
```

Vous trouverez plus d'informations sur l'utilisation d'une grande quantité de mémoire sur : http://publib16.boulder.ibm.com/pseries/en_US/aixprggd/genprogc/lrg_prg_support.htm .

### 2.6.6.5 Notes relatives à SunOS 4

Avec SunOS 4, les MIT−pthreads sont requis pour compiler MySQL, ce qui signifie que vous aurez besoin de GNU `make` .

Quelques systèmes SunOS 4 ont des problèmes avec les librairies dynamiques et `libtool` . Vous pouvez utiliser la ligne suivante de `configure` pour éviter ce problème :

```
shell> ./configure --disable-shared --with-mysqld-ldflags=-all-static
```

Lors de la compilation de `readline` , vous pouvez obtenir des avertissements à propos de définitions dupliquées. Vous pouvez les ignorer.Lors de la compilation de `mysqld` , il y aura quelques avertissements `implicit declaration of function` . Vous pouvez les ignorer.

### 2.6.6.6 Notes pour Alpha−DEC−UNIX (Tru64)

Si vous utilisez `egcs` 1.1.2 sur Digital Unix, vous devez passer à `gcc` 2.95.2, car `egcs` connait de sérieux bugs sur DEC!

Lorsque vous compilez des programmes threadés sous Digital Unix, la documentation recommande l'utilisation de l'option `-pthread` avec `cc` et `cxx` et les librairies `-lmach -lexc` (en plus de `-lpthread` ). Vous devriez exécuter le script `configure` comme ceci :

```
CC="cc -pthread" CXX="cxx -pthread -O" \
./configure --with-named-thread-libs="-lpthread -lmach -lexc -lc"
```

Lorsque vous compilez `mysqld` , vous pouvez voir apparaître des alertes comme celles−ci :

```
mysqld.cc: In function void handle_connections()':
mysqld.cc:626: passing long unsigned int *' as argument 3 of
accept(int,sockadddr *, int *)'
```

Vous pouvez les ignorer tranquillement. Elles apparaissent car `configure` ne peut détecter que des erreurs, et pas des alertes.Si vous démarrez le serveur directement en ligne de commande, vous pouvez rencontrer des problèmes d'interruption si vous vous déconnectez. Lorsque vous vous déconnectez, les processus en cours reçoivent le signal `SIGHUP` . Si c'est le cas, essayez de démarrer le serveur comme ceci :

```
shell> nohup mysqld [options] &
```

`nohup` fait que la commande suivante va ignorer les signaux `SIGHUP` envoyés par le terminal. Alternativement, vous pouvez démarrer le serveur avec le script `safe_mysqld` , qui appelle le démon `mysqld` avec l'option `nohup` pour vous. `safe_mysqld` , le script père de `mysqld` .SI vous avez des problèmes pour compiler `mysys/get_opt.c` , vous pouvez simplement supprimer la ligne `#define _NO_PROTO` au début du fichier!

Si vous utilisez le compilateur `cc` de Compaq, la ligne de configuration suivante devrait fonctionner :

```
 CC="cc -pthread"
CFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host"
CXX="cxx -pthread"
CXXFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host \
-noexceptions -nortti"
export CC CFLAGS CXX CXXFLAGS
./configure \
--prefix=/usr/local/mysql \
--with-low-memory \
--enable-large-files \
--enable-shared=yes \
--with-named-thread-libs="-lpthread -lmach -lexc -lc"
gnumake
```

Si vous avez un problème avec `libtool` , lorsque vous compilez les librairies partagées, ou lorsque vous compilez `mysql` , vous devriez pouvoir résoudre ce problème avec :

```
 cd mysql
/bin/sh ../libtool --mode=link cxx -pthread  -O3 -DDBUG_OFF \
-O4 -ansi_alias -ansi_args -fast -inline speed \
-speculate all \ -arch host  -DUNDEF_HAVE_GETHOSTBYNAME_R \
-o mysql  mysql.o readline.o sql_string.o completion_hash.o \
../readline/libreadline.a -lcurses \
../libmysql/.libs/libmysqlclient.so  -lm
cd ..
gnumake
gnumake install
scripts/mysql_install_db
```

### 2.6.6.7 Notes pour Alpha−DEC−OSF/1

Si vous avez des problèmes de compilation et que le `cc` de DEC et `gcc` sont installés, essayez d'utiliser le script `configure` comme ceci :

```
 CC=cc CFLAGS=-O CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Si vous avez des problèmes avec le fichier `c_asm.h` , vous pouvez créer un fichier inerte `c_asm.h` avec :

```
 touch include/c_asm.h
CC=gcc CFLAGS=-I./include \
CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Notez que les problèmes suivants avec le programme `ld` peuvent être corrigés en téléchargeant le dernier kit de patch de DEC (Compaq) à : http://ftp.support.compaq.com/public/unix/ .Su OSF/1 V4.0D et avec le compilateur "DEC C V5.6−071 on Digital Unix V4.0 (Rev. 878)" le compilateur présente un comportement étrange ( `undefined asm symbols` ). `/bin/ld` apparaît aussi comme incorrect (problèmes avec des erreurs `_exit undefined` survenant lors du link de `mysqld` ). Sur ce système, nous avons réussi à compiler MySQL avec le script `configure` suivant, après avoir remplacé `/bin/ld` par la version de OSF 4.0C:

```
 CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
```

Avec le compilateur Digital "C++ V6.1−029", la ligne suivante doit fonctionner :

```
 CC=cc -pthread
CFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all \
       -arch host
CXX=cxx -pthread
CXXFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all \
          -arch host -noexceptions -nortti
export CC CFLAGS CXX CXXFLAGS
./configure --prefix=/usr/mysql/mysql --with-mysqld-ldflags=-all-static \
            --disable-shared --with-named-thread-libs="-lmach -lexc -lc"
```

Avec certaines versions de OSF/1, la fonction `alloca()` est buguée. Corrigez cela en supprimant la ligne du fichier `config.h` qui définit `'HAVE_ALLOCA'` .

La fonction `alloca()` a aussi un prototype incorrect dans `/usr/include/alloca.h` . L'alerte en résultant peut être ignorée.

Le script `configure` va utiliser automatiquement les librairies de threads suivantes : `--with-named-thread-libs="-lpthread -lmach -lexc -lc"` .

En utilisant `gcc` , vous pouvez aussi essayer le script `configure` avec ceci :

```
shell> CFLAGS=-D_PTHREAD_USE_D4 CXX=gcc CXXFLAGS=-O3 ./configure ...
```

Si vous avez des problèmes avec les signaux (MySQL s'arrête inopinément sous forte charge), vous pouvez avoir rencontré un bogue de l'OS avec les threads, et les signaux. Dans ce cas, vous pouvez indiquer à MySQL de ne pas utiliser les signaux avec la configuration suivante :

```
shell> CFLAGS=-DDONT_USE_THR_ALARM \
      CXXFLAGS=-DDONT_USE_THR_ALARM \
      ./configure ...
```

Cela ne modifie pas les performances de MySQL, mais vous ne pourrez plus terminer les clients qui sont en mode ``sleeping'' sur une connexion avec la commande `mysqladmin kill` ou `mysqladmin shutdown` . Au lieu de cela, le client sera interrompu lorsqu'il émettra la prochaine commande.Avec `gcc` 2.95.2, vous aurez probablement les problèmes de compilation suivants :

```
sql_acl.cc:1456: Internal compiler error in `scan_region', at except.c:2566
Please submit a full bug report.
```

Pour corriger cela, vous devez aller dans le dossier `sql` et faire un ``copier coller'' de la dernière ligne `gcc` , tout en remplaçant le code `-O3` par le code `-O0` ou ajouter le code `-O0` immédiatement après `gcc` si vous n'avez aucune option `-O` sur votre ligne de compilation). Après cela, vous pouvez retourner au niveau de la racine de MySQL, et tenter à nouveau un `make` .

### 2.6.6.8 Notes relatives à SGI Irix

Si vous utilisez la version 6.5.3 d'Irix ou plus récente, `mysqld` ne pourra créer de threads que si vous l'exécutez en tant qu'utilisateur possédant le privilège `CAP_SCHED_MGT` (comme `root` ) ou que vous donnez au serveur `mysqld` ce privilège avec la commande suivante :

```
shell> chcap "CAP_SCHED_MGT+epi" /opt/mysql/libexec/mysqld
```

Vous devrez peut−être supprimer quelques définitions dans `config.h` après avoir exécuté `configure` et avant de compiler.

Sur quelques implémentations d'Irix, la fonction `alloca()` ne marche pas. Si le serveur `mysqld` se stoppe sur quelques requêtes `SELECT` , supprimez les lignes de `config.h` qui définissent `HAVE_ALLOC` et `HAVE_ALLOCA_H` . Si `mysqladmin create` ne fonctionne pas, supprimez la ligne qui définit `HAVE_READDIR_R` dans `config.h` . Vous devrez peut−être supprimer la ligne de `HAVE_TERM_H` aussi.

SGI recommande que vous installiez tous les patches de cette page :
http://support.sgi.com/surfzone/patches/patchset/6.2_indigo.rps.html

Vous devrez, au moins, installer la dernière version du noyau, de `rld` et de `libc` .

Vous avez besoin de tous les patches POSIX sur cette page, pour le support des pthreads :

http://support.sgi.com/surfzone/patches/patchset/6.2_posix.rps.html

Si vous obtenez une erreur se rapprochant de la suivante lors de la compilation de `mysql.cc` :

```
"/usr/include/curses.h", line 82: error(1084): invalid combination of type
```

Tapez ce qui suit dans le répertoire racine de votre source MySQL :

```
shell> extra/replace bool curses_bool < /usr/include/curses.h \
> include/curses.h
shell> make
```

Un problème de planification a aussi été signalé. Si seul un thread est en cours, les choses ralentissent. Evitez cela en démarrant un autre client. Cela pourra accélérer l'exécution de l'autre thread de 2 à 10 fois. Ceci est un problème pas encore très clair avec les threads Irix; vous devrez

improviser pour trouver des solutions en attendant que cela soit corrigé.

Si vous compilez avec `gcc` , vous pouvez utiliser la commande `configure` suivante :

```
 CC=gcc CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql --enable-thread-safe-client \
--with-named-thread-libs=-lpthread
```

Sous Irix 6.5.11 avec les compilateurs natifs Irix C et C++ versions 7.3.1.2, ce qui suit est connu pour fonctionner :

```
 CC=cc CXX=CC CFLAGS='-O3 -n32 -TARG:platform=IP22 -I/usr/local/include \
-L/usr/local/lib' CXXFLAGS='-O3 -n32 -TARG:platform=IP22 \
-I/usr/local/include -L/usr/local/lib' ./configure \
--prefix=/usr/local/mysql --with-innodb --with-berkeley-db \
--with-libwrap=/usr/local \
--with-named-curses-libs=/usr/local/lib/libncurses.a
```

### 2.6.6.9 Notes pour Caldera (SCO)

Le port courant a été testé avec les systèmes ``sco3.2v5.0.4'' et ``sco3.2v5.0.5''. Il y a eu aussi beaucoup de progrès sur le port de ``sco 3.2v4.2''.

Pour le moment, le compilateur recommandé sur OpenServer est `gcc` 2.95.2. Avec lui, vous devriez pouvoir compiler MySQL simplement avec :

```
 CC=gcc CXX=gcc ./configure ... (options)
```

- Pour OpenServer 5.0.X, vous devez utiliser `gcc-2.95.2p1` ou plus récent, de Skunkware. http://www.caldera.com/skunkware/ et choisissez le visualiseur de packages OpenServer ou via FTP à ftp://ftp2.caldera.com/pub/skunkware/osr5/devtools/gcc .
- Vous avez besoin du port `GCC` 2.5.x pour ce produit, et du système de développement. Ils sont nécessaires sur cette version de Caldera (SCO) Unix. Vous ne pouvez pas juste utiliser le système `gcc` dev.
- Vous devez obtenir le package FSU Pthreads et l'installer en premier. Il est disponible à http://www.cs.wustl.edu/~schmidt/ACE_wrappers/FSU-threads.tar.gz . Vous pouvez aussi obtenir un package précompilé à http://www.mysql.com/Downloads/SCO/FSU-threads-3.5c.tar.gz .
- FSU Pthreads peut être compilé sur Caldera (SCO) Unix 4.2 avec tcpip. Ou OpenServer 3.0 ou Open Desktop 3.0 (OS 3.0 ODT 3.0), avec le Caldera (SCO) Development System installé, utilisant le bon port de `GCC` 2.5.x ODT ou OS 3.0, vous aurez besoin du bon port de `GCC` 2.5.x Il y a beaucoup de problèmes sans un bon port. Le port de ce produit requiert le système SCO Unix Development. Sans cela, vous manquerez des librairies et des linkeurs nécessaires.
- Pour compiler FSU Pthreads sur votre système, faites ceci :
  - Exécutez `./configure` dans le dossier `threads/src` et sélectionnez l'option SCO OpenServer. Cette commande copie `Makefile.SCO5` dans `Makefile` .
  - Exécutez `make` .
  - Pour faire l'installation dans le dossier par défaut `/usr/include` , connectez vous en `root` , puis utilisez `cd` vers le dossier `thread/src` , et exécutez la commande `make install` .
- N'oubliez pas d'utiliser GNU `make` pour compiler MySQL.
- Si vous ne démmarez pas `safe_mysqld` en tant que `root` , vous aurez probablement la limite de 110 fichiers ouverts par processus. `mysqld` va écrire une note à ce propos de le fichier de log.
- Avec SCO 3.2V5.0.5, vous devez utiliser FSU Pthreads version 3.5c ou plus récent. Vous devez aussi utiliser `gcc` 2.95.2 ou plus récent!La commande `configure` suivante doit fonctionner :
  ```
  shell> ./configure --prefix=/usr/local/mysql --disable-shared
  ```
- Avec SCO 3.2V4.2, vous devez utiliser FSU Pthreads version 3.5c ou plus récent. La commande `configure` suivante doit fonctionner :

```
shell> CFLAGS="-D_XOPEN_XPG4" CXX=gcc CXXFLAGS="-D_XOPEN_XPG4" \
      ./configure \
          --prefix=/usr/local/mysql \
          --with-named-thread-libs="-lgthreads -lsocket -lgen -lgthreads" \
          --with-named-curses-libs="-lcurses"
```

Vous pouvez avoir des problèmes avec certains fichiers d'inclusion. Dans ce cas, vous pouvez trouver un nouveau fichier d'inclus spécifique à SCO à http://www.mysql.com/Downloads/SCO/SCO-3.2v4.2-includes.tar.gz . Vous devrez décompresser de fichier dans le dossier `include` de votre dossier source MySQL.

Notes de développement Caldera (SCO):

- MySQL devrait détecter automatiquement les FSU Pthreads et compiler `mysqld` avec `-lgthreads -lsocket -lgthreads` .
- Les librairies de développement Caldera (SCO) development sont ré-entrantes avec les FSU Pthreads. Caldera dit que ses librairies de fonctions sont ré-entrantes, et donc, que FSU Pthreads aussi. FSU Pthreads sur OpenServer essaie d'utiliser le schéma SCO pour en faire des librairies ré-entrantes.
- Les FSU Pthreads (tout au moins la version de http://www.mysql.com/ ) est livré compilé avec GNU `malloc` . Si vous avez des problèmes d'utilisation de mémoire, assurez vous que `gmalloc.o` est inclus dans `libgthreads.a` et `libgthreads.so` .
- Avec les FSU Pthreads, les appels systèmes suivant sont compatibles avec pthreads : `read()` , `write()` , `getmsg()` , `connect()` , `accept()` , `select()` et `wait()` .
- Le CSSA-2001-SCO.35.2 (le patch est indiqué comme un patch de sécurité dans erg711905-dscr_remap (version 2.0.0)) rend bogué les FSU threads et MySQL instable. Vous devez supprimer ce patch si vous voulez exécuter `mysqld` sur un serveur OpenServer 5.0.6.

Si vous voulez installer DBI sur Caldera (SCO), vous devez éditer le fichier `Makefile` dans DBI-xxx et chaque sous-dossier.

Notez que les lignes suivantes présupposent que `gcc` 2.95.2 ou plus récent est installé :

```
 Ancien :                                  Nouveau :
CC = cc                                CC = gcc
CCCDLFLAGS = -KPIC -W1,-Bexport        CCCDLFLAGS = -fpic
CCDLFLAGS = -wl,-Bexport               CCDLFLAGS =

 LD = ld                                LD = gcc -G -fpic
LDDLFLAGS = -G -L/usr/local/lib        LDDLFLAGS = -L/usr/local/lib
LDFLAGS = -belf -L/usr/local/lib       LDFLAGS = -L/usr/local/lib

 LD = ld                                LD = gcc -G -fpic
OPTIMISE = -Od                         OPTIMISE = -O1

 Ancien :
CCCFLAGS = -belf -dy -w0 -U M_XENIX -DPERL_SCO5 -I/usr/local/include

 Nouveau :
CCFLAGS = -U M_XENIX -DPERL_SCO5 -I/usr/local/include
```

Ceci est dû au fait que le `Perl dynaloader` ne va pas charger les modules `DBI` si ils ont été compilés avec `icc` ou `cc` .

Perl fonctionne au mieux avec `cc` .

### 2.6.6.10 Notes relatives à la version 7.0 fr Caldera (SCO) Unixware

Vous devez utiliser une version de MySQL supérieure ou égale à la 3.22.13 car cette version corrige quelques problèmes de port vers Unixware.

Nous avons réussi à compiler MySQL avec la ligne suivante de `configure` sur la version 7.0.1 de Unixware :

```
CC=cc CXX=CC ./configure --prefix=/usr/local/mysql
```

Si vous voulez utiliser `gcc` , vous devez utiliser la version 2.95.2 de `gcc` ou plus récente.Caldera fournit libsocket.so.2 disponible sur ftp://stage.caldera.com/pub/security/tools pour des corrections de sécurité pre−OSR506. Le correctif de telnetd est aussi disponible sur ftp://stage.caldera.com/pub/security/openserver/CSSA−2001−SCO.10/ comme libsocket.so.2 et libresolv.so.1 avec des instructions pour installer sur les systèmes pre−OSR506.

Il est bon d'installer les patches précédents avant d'essayer de compiler/utiliser MySQL.

## 2.6.7 Notes relatives à OS/2

MySQL utilise un certain nombre de fichiers ouverts. A cause de cela, vous devez ajouter un ligne se rapprochant de la suivante dans votre fichier `CONFIG.SYS` :

```
SET EMXOPT=-c -n -h1024
```

Si vous ne le faites pas, vous obtiendrez probablement l'erreur :

```
File 'xxxx' not found (Errcode: 24)
```

Lors de l'utilisation de MySQL avec OS/2 Warp 3, FixPack 29 ou plus est requis. Avec OS/2 Warp 4, FixPack 4 ou plus est requis. C'est un besoin de la librairie des Pthreads. MySQL doit être installé sur une partition qui supporte les noms de fichiers longs, tel que HPFS, FAT32, etc.Le script `INSTALL.CMD` doit être exécuté à partir du `CMD.EXE` d'OS/2 et ne fonctionnera probablement pas avec des substitus tel que `4OS2.EXE` .

Le script `scripts/mysql-install-db` a été renommé. Il est maintenant nommé `install.cmd` et est un script REXX, qui mettra en place les configurations de sécurité par défaut de MySQL et créera les icônes WorkPlace Shell pour MySQL.

Le support des module dynamiques est compilé, mais n'est pas assez testé. Les modules dynamiques doivent être compilés en utilisant la librairie run−time Pthreads.

```
 gcc -Zdll -Zmt -Zcrtdll=pthrdrtl -I../include -I../regex -I.. \
    -o exemple udf_exemple.cc -L../lib -lmysqlclient udf_exemple.def
mv exemple.dll exemple.udf
```

**Note** : A cause des limitations de OS/2, les noms des modules UDF ne doivent pas dépasser 8 caractères. Les modules sont stockés dans le répertoire `/mysql2/udf` ; le script `safe-mysqld.cmd` placera ce répertoire dans la variable d'environnement `BEGINLIBPATH` . Lors de l'utilisation des modules UDF, les extensions spécifiées sont ignorées. Elle est supposée être `.udf` . Par exemple, sous Unix, le module partagé peut se nommer `exemple.so` et vous chargeriez une de ses fonctions de la façon suivante :

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "exemple.so";
```

Sous OS/2, me module s'appelera `exemple.udf` , mais vous n'aurez pas à spécifier son extension :

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "exemple";
```

## 2.6.8 Notes relatives à BeOS

Nous sommes vraiment intéressés par le port de MySQL sur BeOS, mais malheureusement, nous n'avons personne qui s'y connaisse en BeOS ou qui ait le temps de s'en occuper.

Nous sommes intéressés par quelqu'un qui serait prêt à faire le port, et nous l'aiderions pour toutes les questions techniques qu'il pourrait se poser durant le processus.

Nous avons déjà eu des contacts avec des développeurs BeOS qui ont dit que MySQL était porté à 80% sur BeOS, mais nous n'avons plus entendu parler d'eux depuis.

### 2.6.9 Notes relatives à Novell NetWare

Nous sommes vraiment intéressés par le port de MySQL sur NetWare, mais malheureusement, nous n'avons personne qui s'y connaisse en NetWare ou qui ait le temps de s'en occuper.

Nous sommes intéressés par quelqu'un qui serait prêt à faire le port, et nous l'aiderions pour toutes les questions techniques qu'il pourrait se poser durant le processus.

## 2.7 Commentaires sur l'installation de Perl

### 2.7.1 Installer Perl sur Unix

Le support Perl pour MySQL est fournit par les interfaces clientes `DBI` / `DBD` . API Perl pour MySQL . Le code client des modules `DBD` / `DBI` de Perl, requière une version de celui−ci supérieure ou égale à la version 5.004. L'interface **ne fonctionnera pas** si vous avez une ancienne version de Perl.

Le support Perl de MySQL requière aussi que vous ayez installé le support de programmation de clients pour MySQL. Si vous avez installé MySQL à partir de fichiers RPM, les programmes clients sont dans le RPM client, mais le support de la programmation de clients est dans le RPM des développeurs. Assurez−vous d'avoir installé le dernier RPM.

Depuis la version 3.22.8, le support Perl est distribué séparément du reste de la distribution MySQL. Si vous voulez installer le support Perl, les fichiers dont vous aurez besoin peuvent être trouvés sur http://www.mysql.com/downloads/api−dbi.html .

Les distributions de Perl sont fournies en tant qu'archives compréssées `tar` et ont des noms comme `MODULE−VERSION.tar.gz` , où `MODULE` est le nom du module et `VERSION` est le numéro de version. Vous devez obtenir les distributions `Data−Dumper` , `DBI` , et `Msql−Mysql−modules` et les installer dans cet ordre. La procédure d'installation est décrite ici. L'exemple montré ici concerne le module `Data−Dumper` , mais la procédure est la même pour les trois distributions :

- Décompressez la distribution dans le dossier courant :
  ```
  shell> gunzip < Data−Dumper−VERSION.tar.gz | tar xvf −
  ```
  Cette commande crée un dossier appelé `Data−Dumper−VERSION` .
- Mettez vous dans le répertoire racine de la distribution décompressée :
  ```
  shell> cd Data−Dumper−VERSION
  ```
- Construisez la distribution et compilez tout :
  ```
  shell> perl Makefile.PL
  shell> make
  shell> make test
  shell> make install
  ```

La commande `make test` est importante car elle vérifie que le module fonctionne. Notez que quand vous exécutez cette commande durant l'installation de `Msql−Mysql−modules` pour tester le code de l'interface, le serveur MySQL doit être en marche sinon le test échouera.

Il est bon de reconstruire et réinstaller la distribution `Msql−Mysql−modules` à chaque fois que vous réinstallez une nouvelle version de MySQL, particulièrement si vous avez des problèmes avec vos scripts `DBI` après avoir avoir mis à jour MySQL.

Si vous n'avez pas le droit d'installer des modules Perl dans le dossier système ou que vous voulez installer des modules locaux de Perl, la référence suivante pourra vous aider :

```
http://www.iserver.com/support/contrib/perl5/modules.html
```

Regardez le paragraphe `Installing New Modules that Require Locally Installed Modules` .

## 2.7.2 Installer ActiveState Perl sur Windows

Pour installer le module `DBD` MySQL avec ActiveState Perl sous Windows, vous devez faire ce qui suit :

- Obtenez ActiveState Perl à partir de http://www.activestate.com/Products/ActivePerl/ et installez le.
- Ouvrez un terminal DOS.
- Si requis, définissez la variable HTTP_proxy. Par exemple, vous pouvez faire :
  ```
  set HTTP_proxy=my.proxy.com:3128
  ```
- Démarrez le programme PPM :
  ```
  C:\> c:\perl\bin\ppm.pl
  ```
- Installez `DBI` , si ce n'est pas déjà fait :
  ```
  ppm> install DBI
  ```
- Si cela fonctionne, exécutez la commande suivante :
  ```
  install \
ftp://ftp.de.uu.net/pub/CPAN/authors/id/JWIED/DBD-mysql-1.2212.x86.ppd
  ```

Ce qui suit devrait fonctionner avec la version 5.6 d'ActiveState Perl.Si ce qui précède ne veut pas fonctionner, vous devez à la place installer le pilote `MyODBC` et vous connecter au serveur MySQL via ODBC :

```
 use DBI;
$dbh= DBI->connect("DBI:ODBC:$dsn","$utilisateur","$motdepasse") ||
  die "Obtenu l'erreur $DBI::errstr lors de la connexion à $dsn\n";
```

## 2.7.3 Installer la distribution Perl de MySQL sous Windows

Cette distribution Perl de MySQL contient `DBI` , `DBD:MySQL` et `DBD:ODBC` .

- Obtenez la distribution Perl pour Windows à partir de http://www.mysql.com/downloads/os-win32.html .
- Décompressez la distribution dans `C:` pour obtenir le dossier `C:\PERL` .
- Ajoutez le répertoire `C:\PERL\BIN` à votre path.
- Ajoutez le répertoire `C:\PERL\BIN\MSWIN32-x86-thread` ou `C:\PERL\BIN\MSWIN32-x86` à votre path.
- Vérifiez que `perl` fonctionne en exécutant `perl -v` dans un terminal DOS.

## 2.7.4 Problèmes lors de l'utilisation des interfaces Perl DBI et DBD

Si Perl vous informe qu'il ne peut trouver le module `../mysql/mysql.so` , il se trouve probablement que Perl n'arrive pas à trouver la librairie partagée `libmysqlclient.so` .

Vous pouvez corriger cela en suivant l'une des méthodes suivantes :

- Compilez la distribution `Msql-Mysql-modules` avec `perl Makefile.PL -static -config` au lieu de `perl Makefile.PL` .
- Copiez `libmysqlclient.so` dans le dossier où se situent vos autres librairies partagées (souvent `/usr/lib` ou `/lib` ).

- Sous Linux vous pouvez ajouter le chemin vers le dossier dans lequel se trouve `libmysqlclient.so` au fichier `/etc/ld.so.conf` .
- Ajoutez le chemin complet vers le dossier où se situe `libmysqlclient.so` à la variable d'environnement `LD_RUN_PATH` .

Si vous obtenez l'erreur suivante de `DBD-mysql` , vous utilisez probablement `gcc` (ou un vieux binaire compilé avec `gcc` ) :

```
 /usr/bin/perl: can't resolve symbol '__moddi3'
/usr/bin/perl: can't resolve symbol '__divdi3'
```

Ajoutez `-L/usr/lib/gcc-lib/... -lgcc` à la commande de liaison lorsque la librairie `mysql.so` est construite (vérifiez l'affichage de `make` concernant `mysql.so` quand vous compilez le client Perl). L'option `-L` doit spécifier le chemin vers le dossier où se situe `libgcc.a` sur votre système.Une autre cause du problème peut être que Perl et MySQL ne sont pas tous deux compilés avec `gcc` . Dans ce cas là, vous devrez faire en sorte qu'ils le soient.

Si vous obtenez les erreurs suivantes de la part de `Msql-Mysql-modules` quand vous exécutez ces tests :

```
 t/00base...........install_driver(mysql) failed:
Can't load '../blib/arch/auto/DBD/mysql/mysql.so' for module DBD::mysql:
../blib/arch/auto/DBD/mysql/mysql.so: undefined symbol:
uncompress at /usr/lib/perl5/5.00503/i586-linux/DynaLoader.pm line 169.
```

cela signifie que vous avez besoin d'inclure la librairie dynamique, -lz, dans la ligne de liaison. Cela peut se faire en changeant ce qui suit dans `lib/DBD/mysql/Install.pm` :

```
 $sysliblist .= " -lm";

 en

 $sysliblist .= " -lm -lz";
```

Après cela, vous **devez** exécuter 'make realclean' et reprendre l'installation dès le début.Si vous voulez utiliser le module de Perl sur un système qui ne supporte pas les liaisons dynamiques (comme Caldera/SCO) vous pouvez générer une version statique de Perl incluant `DBI` et `DBD-mysql` . L'approche est de générer une version de Perl avec le code de `DBI` lié et de l'installer au dessus de votre Perl courant. Puis vous utilisez cette version pour en créer à nouveau une autre qui comporte le code de `DBD` lié et d'installer cette version ci.

Sur Caldera (SCO), vous devez définir les variables d'environnement suivantes :

```
 shell> LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/usr/progressive/lib
ou
shell> LD_LIBRARY_PATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:\
/usr/progressive/lib:/usr/skunk/lib
shell> LIBPATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:\
/usr/progressive/lib:/usr/skunk/lib
shell> MANPATH=scohelp:/usr/man:/usr/local1/man:/usr/local/man:\
/usr/skunk/man:
```

D'abord, créez un Perl incluant un `DBI` lié statiquement en exécutant des commandes dans le dossier où se situe votre distribution `DBI` :

```
 shell> perl Makefile.PL -static -config
shell> make
shell> make install
shell> make perl
```

Ensuite, vous devez installer le nouveau Perl. Les affichages de `make perl` vous indiqueront les commandes `make` exactes que vous aurez besoin d'exécuter pour faire l'installation. Sur Caldera (SCO), il s'agit de `make -f Makefile.aperl inst_perl MAP_TARGET=perl` .Puis, utilisé le Perl qui vient d'être créé pour en créer un nouveau qui inclut un `DBD::mysql` lié statiquement en exécutant ces commandes dans le dossier où votre distribution de `Msql-Mysql-modules` se situe :

```
 shell> perl Makefile.PL -static -config
shell> make
shell> make install
```

```
shell> make perl
```

Finalement, vous devez installer ce nouveau Perl. Une fois de plus, l'affichage de `make perl` vous indiquera la commande à utiliser.

# 3 Tutoriels d'introduction

Ce chapitre fournit un tutoriel d'introduction à MySQL en montrant comment utiliser le client `mysql` pour créer et utiliser une simple base de données. `mysql` (quelques fois nommé ``moniteur terminal'' ou juste ``moniteur'') est un programme interactif qui vous permet de vous connecter à un serveur MySQL, exécuter des requêtes et voir les résultats. `mysql` peut aussi être utilisé en mode batch : vous placez vos requêtes dans un fichier, puis vous faites exécuter à `mysql` le contenu de ce fichier. Les deux manières d'utiliser `mysql` sont expliquées ici.

Pour voir une liste d'options fournies par `mysql` , invoquez−le avec l'option `--help` :

```
shell> mysql --help
```

Ce chapitre assume que `mysql` est installé sur votre machine et qu'un serveur MySQL est disponible pour que vous vous y connectiez. Si ce n'est pas le cas, contactez votre administrateur MySQL. (Si **vous** êtes l'administrateur, vous aurez besoin de consulter d'autres sections de ce manuel.)Ce chapitre décrit le processus d'installation et d'utilisation d'une base de données en entier. Si vous n'êtes intéressés que par l'accès à une base de données existante, vous pouvez sauter les sections décrivant la création de la base et des tables.

Ce chapitre n'est qu'un tutoriel, beaucoup de détails ne sont pas approfondis. Consultez les sections appropriées du manuel pour plus d'informations sur les sujets abordés.

## 3.1 Connexion et déconnexion au serveur

Pour vous connecter au serveur, vous aurez dans la plupart des cas à fournir un nom d'utilisateur à MySQL, et, sûrement, un mot de passe. Si le serveur fonctionne sur une autre machine que la vôtre, vous devrez spécifier son adresse. Contactez votre administrateur pour connaître les paramètres à utiliser lors de la connexion (hôte, nom d'utilisateur, mot de passe à utiliser...). Une fois que vous aurez les bons paramètres, vous pourrez vous connecter de la façon suivante :

```
shell> mysql -h hote -u utilisateur -p
Enter password: ********
```

`********` représente votre mot de passe, entrez−le lorsque `mysql` affiche `Enter password:` .Si tout fonctionne, vous devrez voir quelques informations d'introduction suivies d'une invite de commande `mysql>` :

```
shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a-log

Type 'help' for help.

mysql>
```

L'invite vous dit que `mysql` attend que vous entriez des commandes.Quelques installations de MySQL autorisent les connexions anonymes au serveur tournant sur l'hôte local. Si c'est la cas sur votre machine, vous devriez arriver à vous connecter à ce serveur en invoquant la commande `mysql` sans aucune option :

```
shell> mysql
```

Après vous être connecté avec succès, vous pouvez vous déconnecter à tout moment en entrant `QUIT` dans l'invite `mysql>` :

```
mysql> QUIT
Bye
```

Vous pouvez aussi le faire en appuyant sur Ctrl−D.

La plupart des exemples dans les sections suivantes supposent que vous êtes connecté au serveur. Cela se voit à l'invite `mysql>` .

# 3.2 Entrer des requêtes

Assurez−vous d'être connecté au serveur, comme expliqué précédemment dans cette section. Faire ceci ne sélectionnera pas une base par lui même, mais c'est normal. A ce stade, il est important de découvrir la façon dont sont publiées les requêtes, pour ensuite pouvoir créer des tables, y insérer et rechercher des données. Cette section décrit les principes de base pour entrer une commande, en utilisant plusieurs requêtes que vous pouvez essayer pour vous familiariser avec la façon dont `mysql` fonctionne.

Voilà une commande simple qui demande au serveur de vous donner son numéro de version et la date courante. Entrez−la comme suit, juste après l'invite `mysql>` puis pressez Enter :

```
 mysql> SELECT VERSION(), CURRENT_DATE;
+−−−−−−−−−−−−−−+−−−−−−−−−−−−−−+
| VERSION()    | CURRENT_DATE |
+−−−−−−−−−−−−−−+−−−−−−−−−−−−−−+
| 3.22.20a−log | 1999−03−19   |
+−−−−−−−−−−−−−−+−−−−−−−−−−−−−−+
1 row in set (0.01 sec)
mysql>
```

La requête révèle plusieurs choses à propos de `mysql` :

- Une commande consiste normalement en une commande SQL suivie d'un point−virgule. (Il y a quelques cas ou le point−virgule n'est pas requis. `QUIT` , mentionnée plus tôt, en fait partie. Nous verrons les autres plus tard.)
- Lorsque vous entrez une commande, `mysql` l'envoi au serveur pour l'exécution et affiche le résultat, puis affiche un autre `mysql>` pour indiquer qu'il attend une autre commande.
- `mysql` affiche le résultat des requêtes dans une table (lignes et colonnes). La première ligne contient le nom des colonnes. Les lignes suivantes constituent le résultat de la requête. Normalement, les titres des colonnes sont les noms des champs des tables de la base de données que vous avez récupérés. Si vous récupérez la valeur d'une expression au lieu d'une colonne (comme dans l'exemple précédent), `mysql` nomme la colonne en utilisant l'expression elle−même.
- `mysql` vous indique combien de lignes ont été retournées et combien de temps d'exécution la requête a pris, ce qui vous donnera une approximation des performances du serveur. Ces valeurs sont imprécises car elles représentent le temps soft (et non le temps CPU ou hard), et qu'elles sont affectées par des facteurs tels que la charge du serveur ou l'accessibilité du réseau. (Dans un soucis de brièveté, la ligne contenant ``rows in set'' n'est plus montrée dans les exemples suivants de ce chapitre.)

Les mots−clefs peuvent être entrés sous n'importe quelle forme de casse. Les requêtes suivantes sont équivalentes :

```
 mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Voilà une autre requête. Elle montre que vous pouvez utiliser `mysql` en tant que simple calculatrice :

```
 mysql> SELECT SIN(PI()/4), (4+1)*5;
+−−−−−−−−−−−−−+−−−−−−−−−+
| SIN(PI()/4) | (4+1)*5 |
+−−−−−−−−−−−−−+−−−−−−−−−+
|    0.707107 |      25 |
+−−−−−−−−−−−−−+−−−−−−−−−+
```

Les commandes vues jusqu'à présent ont été relativement courtes, et tenaient sur une seule ligne. Vous pouvez même entrer plusieurs requêtes sur une seule ligne. Il suffit de terminer chacune d'elle par un point−virgule :

```
 mysql> SELECT VERSION(); SELECT NOW();
+−−−−−−−−−−−−−−+
| VERSION()    |
+−−−−−−−−−−−−−−+
| 3.22.20a−log |
+−−−−−−−−−−−−−−+
```

```
 +---------------------+
| NOW()               |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

Une commande ne doit pas être obligatoirement sur une seule ligne ; les commandes qui exigent plusieurs lignes ne sont pas un problème. `mysql` détermine où se situe la fin de votre commande en recherchant le point−virgule de terminaison, et pas l'extrémité de la commande entrée. (Dans d'autres termes, `mysql` accepte des formats libres d'entrée : il collecte les lignes entrées mais ne les exécute qu'une fois le point−virgule trouvé.) Voilà une seule requête sur plusieurs lignes :

```
 mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+--------------------+--------------+
| USER()             | CURRENT_DATE |
+--------------------+--------------+
| joesmith&#64;localhost | 1999-03-18   |
+--------------------+--------------+
```

Dans cet exemple, notez comment l'invite change de `mysql>` à `->` après avoir entré la première ligne d'une requête multi−lignes. C'est la façon dont `mysql` indique qu'il n'a pas vu de requête complète et qu'il attend la fin de celle−ci. L'invite est votre ami en vous fournissant la rétroactivité. Si vous utilisez cette rétroactivité, vous vous rendrez toujours compte de ce que `mysql` attend. Si vous décidez d'annuler une commande que vous êtes en train de taper, faites−le en entrant `\c` :

```
 mysql> SELECT
    -> USER()
    -> \c
mysql>
```

Ici aussi, portez votre attention sur l'invite. Elle se transforme à nouveau en `mysql>` après que vous ayez entré `\c` , vous informant que `mysql` est prêt pour une nouvelle requête.Le tableau suivant montre les différentes invites que vous pourrez voir et résume leur signification quand à l'état dans lequel se trouve `mysql` :

| Invite | Signification |
|---|---|
| mysql> | Prêt pour une nouvelle commande. |
| -> | En attente de la ou des lignes terminant la commande. |
| '> | En attente de la prochaine ligne collectant une chaîne commencée par un guillemet simple ( ''' ). |
| "> | En attente de la prochaine ligne collectant une chaîne commencée par un guillemet double ( '"' ). |

Les commandes sur plusieurs lignes sont la plupart du temps des accidents, lorsque vous voulez faire une commande sur une seule ligne et que vous oubliez le point−virgule de fin. Dans ce cas, `mysql` attend la suite de votre saisie :

```
 mysql> SELECT USER()
    ->
```

Si cela vous arrive (vous pensez que votre requête est complète mais la seule réponse est l'invite `->` ), il est fort probable que `mysql` attende le point−virgule. Si vous ne notez pas ce que l'invite vous indique, vous pourriez patienter pendant longtemps avant de réaliser ce que vous devez faire. Entrez un point−virgule pour compléter la requête, et `mysql` devrait l'exécuter :

```
 mysql> SELECT USER()
    -> ;
+--------------------+
| USER()             |
+--------------------+
| joesmith&#64;localhost |
+--------------------+
```

L'invite `'>` ainsi que `">` apparaissent durant l'entrée de chaîne. Dans MySQL, vous pouvez écrire une chaîne entourée du caractère `'''` ou bien `'"'` (par exemple, `'Bonjour'` or `"Au Revoir"` ), et `mysql` vous laisse entrer une chaîne qui peut être sur plusieurs lignes. Lorsque vous voyez une invite comme `'>` ou `">` , cela signifie que vous avez entré une ligne contenant le caractère `'''` ou `'"'` , mais vous n'avez pas encore entré le caractère correspondant qui termine votre chaîne. C'est pratique si vous entrez réellement une chaîne à lignes multiples, mais est−ce probable ? Pas vraiment. Plus souvent, les invites `'>` et `">` indiquent que vous avez, par inadvertance, oublié un caractère de fermeture. Par exemple :

```
 mysql> SELECT * FROM ma_table WHERE nom = "Smith AND age < 30;
    ">
```

Si vous entrez cette requête `SELECT` , puis appuyez sur Enter et attendez le résultat, rien ne se passera. Au lieu de vous demander pourquoi la requête met si longtemps à s'exécuter, remarquez que l'invite de commande s'est transformée en `">` . Cela indique que `mysql` attend de voir la fin d'une

chaîne de caractères non−terminée. (Voyez−vous l'erreur dans cette requête ? Il manque le second guillemet à la suite de "Smith .) Que faire ? Le plus simple est d'annuler la commande. Toutefois, vous ne pouvez vous contenter de taper \c dans ce cas−là, car mysql l'interprète comme une partie de la chaîne qu'il est en train de collecter ! A la place, entrez le second guillemet (pour que mysql sache que vous avez fini la chaîne), puis entrez \c :

```
 mysql> SELECT * FROM my_table WHERE name = "Smith AND age < 30;
    "> "\c
mysql>
```

L'invite se change à nouveau en mysql> , indiquant que mysql est prêt pour une nouvelle requête.Il est important de savoir ce que les invites '> et "> signifient, car si vous avez entré par erreur une chaîne non terminée, toutes les lignes suivantes que vous entrerez seront ignorées par mysql , même une ligne contenant QUIT ! Cela peut prêter à confusion, spécialement si vous ne savez pas que vous devez fournir le guillemet fermant avant de pouvoir annuler la commande courante.

## 3.3 Création et utilisation d'une base de données

Maintenant que vous savez entrer des commandes, il est temps d'accéder à une base.

Supposons que vous avec plusieurs animaux chez vous (dans votre ménagerie) et que vous voulez garder diverses informations les concernant. Vous pouvez le faire en créant des tables pour stocker vos données et y charger vos informations. Vous pourrez alors répondre à différentes sortes de questions à propos de vos animaux en récupérant les données à partir des tables. Cette section vous montre comment :

- Créer une base de données
- Créer une table
- Charger des données dans vos tables
- Récupérer des donneées à partir des tables de différentes façons
- Utiliser plusieurs tables

La base de données de la ménagerie va être simple (délibérément), mais il n'est pas difficile de penser à des situations courantes de la vie où vous aurez à utiliser un tel type de base de données. Par exemple, une telle base pourrait être utilisée par un éleveur pour gérer sa boutique, ou par un vétérinaire pour garder des traces de ses patients. Une distribution de la ménagerie contenant quelques requêtes et des exemples de données utilisées dans la section suivante peuvent être trouvés sur le site web de MySQL. Ils sont disponibles au format compressé tar ( http://www.mysql.com/Downloads/Contrib/Examples/menagerie.tar.gz ) ou au format Zip ( http://www.mysql.com/Downloads/Contrib/Examples/menagerie.zip ).Utilisez la commande SHOW pour trouver quelles bases existent déjà sur le serveur :

```
 mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

La liste des bases de données est probablement différente sur votre machine, mais les bases mysql et test y figurent sûrement. La base mysql est requise car elle gère les accès et les privilèges. La base test est souvent fournie pour que les utilisateurs y effectuent leurs tests.Notez que vous ne pourrez voir toutes les bases de données si vous n'avez pas le privilège SHOW DATABASES . Syntaxe de GRANT et REVOKE .

Si la base de données test existe, essayez d'y accéder :

```
 mysql> USE test
Database changed
```

Notez que `USE` , comme `QUIT` , ne requiert pas de point−virgule. (Vous pouvez terminer ces commandes avec un point−virgule ; cela ne posera pas de problèmes.) La commande `USE` est spéciale d'un autre point de vue : elle doit être donnée sur une seule ligne.

Vous pouvez utiliser la base de données `test` (si vous y avez accès) pour les exemples qui suivent, mais tout ce que vous créerez dans cette base pourra être éffacé par quiconque y a accès. Pour cette raison, vous feriez mieux de demander à votre administrateur MySQL la permission d'utiliser une base de données rien que pour vous. Supposez que vous voulez nommer la votre `menagerie` . L'administrateur a besoin d'exécuter une commande telle que :

```
mysql> GRANT ALL ON menagerie.* TO votre_nom_mysql;
```

où `votre_nom_mysql` est le nom d'utilisateur MySQL qui vous est assigné.

## 3.3.1 Créer et sélectionner une base de données

Si l'administrateur vous a créé une base de données lors du paramétrage de vos droits, vous pouvez commencer à l'utiliser. Sinon, vous aurez besoin de la créer par vous−même :

```
mysql> CREATE DATABASE menagerie;
```

Sous Unix, les noms des bases de données sont sensibles à la casse (ce qui diffère des mots réservés de SQL), ce qui fait que vous devez tuojours vous référer à votre base de données avec `menagerie` , non avec `Menagerie` , `MENAGERIE` , ou d'autres variantes. Cela est aussi valable pour les noms de tables. (Sous Windows, cette restriction n'est pas appliquée, même si vous devez vous référer à une table ou une base de la même façon dans une même requête).

La création d'une base de données ne la sélectionne pas pour l'utilisation ; vous devez le faire explicitement. Pour rendre `menagerie` la base courante, utilisez cette commande :

```
mysql> USE menagerie
Database changed
```

Votre base a besoin d'être créée juste une fois, mais vous devez la sélectionner pour l'utiliser, chaque fois que vous débutez une session `mysql` . Vous pouvez le faire en publiant une requête `USE` comme ci−dessus. Sinon, vous pouvez sélectionner la base directement dans la ligne de commande lorsque vous invoquez `mysql` . Vous devez juste spécifier son nom après les paramètres de connexion dont vous avez besoin. Par exemple :

```
shell> mysql −h hote −u utilisateur −p menagerie
Enter password: ********
```

Notez que `menagerie` n'est pas votre mot de passe dans la commande que nous venons de montrer. Si vous voulez le fournir dans la ligne de commande après l'option `−p` , vous devez le faire sans espace entre les deux (par exemple, tapez `−pmonmotdepasse` , et non `−p monmotdepasse` ). Toutefois, mettre le mot de passe en ligne de commande n'est pas recommandé, car le faire permettrait à d'autres utilisateurs connectés sur votre machine de l'obtenir.

## 3.3.2 Création d'une table

Créer la base de données est la partie facile, mais jusque−là elle est vide, comme vous le montre `SHOW TABLES` :

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

La partie la plus difficile est le choix de la structure de la base de données : de quelles tables aurez vous besoin et quelles colonnes devront figurer dans chacune d'elles.Vous voudrez une table qui contient un enregistrement pour chaque animal. On peut l'appeler la table `animal` , et elle devra contenir, au minimum, le nom de chaque animal. Puisque le nom tout seul n'est pas intéressant, la table devra contenir d'autres informations. Par exemple, si plus d'une personne de votre famille possède un animal, vous voudrez lister le nom du maître de chaque animal. Vous voudrez peut−être aussi enregistrer une description basique comme l'espèce ou le sexe.

Et pour l'âge ? C'est intéressant, mais n'est pas bon pour un stockage en base de données. L'âge change chaque jour, vous devrez donc mettre à jour vos enregistrements assez souvent. Il est préférable de stocker une valeur fixe, comme la date de naissance. Dans ce cas−là, à chaque fois que vous aurez besoin de l'âge, vous pourrez l'obtenir en faisant la différence entre la date courante et la date enregistrée. MySQL fournit des fonctions de calcul sur les dates, cela ne sera donc pas difficile. Enregistrer la date de naissance, au lieu de l'âge a d'autres avantages :

- Vous pouvez utiliser la base de données pour des tâches, comme la génération d'un rappel pour les prochains anniversaires d'animaux. (Si vous trouvez que ce type de requêtes est quelque peu idiot, notez que c'est la même question que vous vous poseriez dans le contexte d'une base de données d'affaires pour identifier les clients à qui vous aurez besoin d'envoyer un message de voeux, pour cette touche informatiquement assistée d'humanisme.)
- Vous pouvez calculer l'âge à partir d'autres dates que la date du jour. Par exemple, si vous stockez la date de la mort dans la base de données, vous pourrez facilement calculer l'âge qu'avait un animal à sa mort.

Vous trouverez probablement d'autres informations qui pourront être utiles dans la table `animal` , mais celles identifiés jusqu'à maintenant sont largement suffisantes pour l'instant : nom, maitre, espece, sexe, naissance, et mort.Utilisez une requête `CREATE TABLE` pour spécifier la structure de votre table :

```
mysql> CREATE TABLE animal (nom VARCHAR(20), maitre VARCHAR(20),
    -> espece VARCHAR(20), sexe CHAR(1), naissance DATE, mort DATE);
```

`VARCHAR` est un bon choix pour les colonnes `nom` , `maitre` , et `espece` car leurs valeurs varient en longueur. La longueur de ces colonnes ne doit pas nécessairement être la même, et n'a pas besoin d'être forcement `20` . Vous pouvez choisir une taille entre `1` et `255` , celle qui vous semblera la plus raisonnable. (Si vous faites un mauvais choix et que vous vous apercevez plus tard que vous avez besoin d'un champ plus long, MySQL fournit la commande `ALTER TABLE` .)Le sexe des animaux peut être représenté de plusieurs façons, par exemple, `"m"` et `"f"` , ou bien `"male"` and `"femelle"` . Il est plus simple d'utiliser les caractères simples `"m"` et `"f"` .

L'utilisation du type de données `DATE` pour les colonnes `naissance` et `mort` est un choix plutôt judicieux.

Maintenant que vous avez créé une table, `SHOW TABLES` devrait produire de l'affichage :

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| animal              |
+---------------------+
```

Pour vérifier que la table a été créée de la façon que vous vouliez , utilisez la commande `DESCRIBE` :

```
mysql> DESCRIBE animal;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| nom       | varchar(20) | YES  |     | NULL    |       |
| maitre    | varchar(20) | YES  |     | NULL    |       |
| espece    | varchar(20) | YES  |     | NULL    |       |
| sexe      | char(1)     | YES  |     | NULL    |       |
| naissance | date        | YES  |     | NULL    |       |
| mort      | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Vous pouvez utiliser `DESCRIBE` quand vous voulez, par exemple, si vous avez oublié les noms des colonnes dans votre table ou leurs types.

### 3.3.3 Charger des données dans une table

Après la création de votre table, vous aurez besoin de la remplir. Les commandes `LOAD DATA` et `INSERT` sont utiles pour cela.

Supposons que les enregistrements de vos animaux peuvent être décrits comme suit. (Observez que MySQL attend les dates au format `YYYY-MM-DD` ; cela peut différer de ce à quoi vous êtes habitué.)

| nom | maitre | especes | sexe | naissance | mort |
|---|---|---|---|---|---|
| Fluffy | Harold | chat | f | 1993-02-04 | |
| Claws | Gwen | chat | m | 1994-03-17 | |
| Buffy | Harold | chien | f | 1989-05-13 | |
| Fang | Benny | chien | m | 1990-08-27 | |
| Bowser | Diane | chien | m | 1998-08-31 | 1995-07-29 |
| Chirpy | Gwen | oiseau | f | 1998-09-11 | |
| Whistler | Gwen | oiseau | | 1997-12-09 | |
| Slim | Benny | serpent | m | 1996-04-29 | |

Puisque vous commencez avec une table vide, il est facile de la remplir en créant un fichier texte contenant une ligne pour chaque animal que vous avez, puis charger son contenu à l'aide d'une seule commande.

Vous pouvez créer un fichier `pet.txt` contenant un enregistrement par ligne, avec les valeurs séparés par des tabulations, et ordonnées comme les champs l'étaient dans la requête `CREATE TABLE` . Pour les données manquantes (comme un sexe inconnu ou la date de mort d'un animal toujours en vie), vous pouvez utiliser les valeurs `NULL` . Pour les représenter dans votre fichier texte, utilisez `\N` . Par exemple, l'enregistrement de Whistler l'oiseau ressemblera à ça (l'espace entre les valeurs est une tabulation) :

| nom | maitre | especes | sexe | naissance | mort |
|---|---|---|---|---|---|
| Whistler | Gwen | bird | \N | 1997-12-09 | \N |

Pour charger le fichier `pet.txt` dans la table `pet` , utilisez cette commande :

```
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

Vous pouvez spécifier la valeur du séparateur de colonnes et le marqueur de fin de lignes explicitement dans la commande `LOAD DATA` si vous le voulez, mais les valeurs par défaut sont la tabulation et le retour à la ligne. Ceux-là sont suffisants pour que la commande lise le fichier `pet.txt` correctement.

Lorsque vous voulez ajouter des enregistrements un par un, la commande `INSERT` est utile. Dans sa forme la plus simple, où vous spécifiez une valeur pour chaque colonne, dans l'ordre où les colonnes sont listées dans la requête `CREATE TABLE` . Supposons que Diane achète un nouvel hamster nommé Puffball. Vous pourriez ajouter ce nouvel enregistrement en utilisant un `INSERT` de la façon suivante :

```
mysql> INSERT INTO pet
   -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

Notez que les chaînes de caractères et les valeurs de dates sont spécifiées en tant que chaînes protégées par des guillemets. De plus, avec `INSERT` vous pouvez insérer la valeur `NULL` directement pour représenter une valeur manquante. Vous n'utilisez pas `\N` comme vous le faites avec `LOAD DATA` .

A partir de cet exemple, vous devriez être capable de voir qu'il y a beaucoup plus de commandes à taper lorsque vous utilisez la commande `INSERT` au lieu de `LOAD DATA` .

## 3.3.4 Récupérer des informations à partir d'une table

La commande `SELECT` est utilisée pour récupérer des informations à partir d'une table. La forme usuelle est :

```
 SELECT quoi_selectionner
FROM quel_table
WHERE conditions_a_satisfaire
```

`quoi_selectionner` indique ce que vous voulez voir. Cela peut être une liste de colonnes, ou `*` pour indiquer ``toutes les colonnes''. `quel_table` indique la table à partir de laquelle récupérer les données. La clause `WHERE` est optionnelle. Si elle est présente, `conditions_a_satisfaire` spécifie les conditions que les lignes doivent satisfaire pour être séléctionnées.

### 3.3.4.1 Sélectionner toutes les données

La plus simple forme de `SELECT` récupère toutes les données d'une table :

```
 mysql> SELECT * FROM animal;
+----------+--------+---------+------+------------+------------+
| nom      | maitre | espece  | sexe | naissance  | mort       |
+----------+--------+---------+------+------------+------------+
| Fluffy   | Harold | chat    | f    | 1993-02-04 | NULL       |
| Claws    | Gwen   | chat    | m    | 1994-03-17 | NULL       |
| Buffy    | Harold | chien   | f    | 1989-05-13 | NULL       |
| Fang     | Benny  | chien   | m    | 1990-08-27 | NULL       |
| Bowser   | Diane  | chien   | m    | 1998-08-31 | 1995-07-29 |
| Chirpy   | Gwen   | oiseau  | f    | 1998-09-11 | NULL       |
| Whistler | Gwen   | oiseau  | NULL | 1997-12-09 | NULL       |
| Slim     | Benny  | serpent | m    | 1996-04-29 | NULL       |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Cette forme de `SELECT` est utile si vous voulez récupérer la table entière. par exemple, après l'avoir juste remplie avec vos données d'origine. Il apparaît alors qu'une erreur s'était glissée dans votre fichier de données : Bowser a l'air d'être né après sa mort ! En consultant le papier original de son pedigree, vous trouvez que la date correcte est 1989 et non pas 1998.

Il y a au moins deux façons de corriger cela :

- Corriger le fichier `animal.txt` pour corriger l'erreur, puis vider et recharger à nouveau la table en utilisant `DELETE` et `LOAD DATA` :
  ```
   mysql> SET AUTOCOMMIT=1;  # Utilisé pour une recréation rapide de la table
  mysql> DELETE FROM animal;
  mysql> LOAD DATA LOCAL INFILE "animal.txt" INTO TABLE animal;
  ```
  Toutefois, si vous choisissez cette méthode, vous devrez aussi rentrer à nouveau l'enregistrement de Puffball.
- Corriger uniquement l'enregistrement erroné avec une requête `UPDATE` :
  ```
   mysql> UPDATE animal SET naissance = "1989-08-31" WHERE nom = "Bowser";
  ```

Comme nous l'avons montré, il est facile de récupérer toutes les données d'une table. Toutefois, vous ne voudrez sûrement pas le faire, surtout si la table devient imposante. A la place, vous serez plus intéressé par répondre à une question particulière, dans ce cas−là, vous spécifiez quelques contraintes pour les informations que vous voulez. Regardons quelques requêtes de sélection qui répondent à des questions à propos de vos animaux.

### 3.3.4.2 Sélectionner des lignes particuliéres

Vous pouvez sélectionner des lignes particulières de votre table. Par exemple, si vous voulez vérifier la modification que vous avez effectuée sur la date de naissance de Bowser, sélectionnez son enregistrement comme suit :

```
mysql> SELECT * FROM animal WHERE nom = "Bowser";
+--------+--------+--------+------+------------+------------+
| nom    | maitre | espece | sexe | naissance  | mort       |
+--------+--------+--------+------+------------+------------+
| Bowser | Diane  | chien  | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+--------+------+------------+------------+
```

L'affichage confirme que la date est correcte maintenant : 1989, et non 1998.La comparaison des chaînes de caractères se fait normalement avec sensibilité à la casse, vous pouvez donc spécifier le nom `"bowser"` , `"BOWSER"` , etc. Le résultat de la requête sera le même.

Vous pouvez spécifier des conditions sur toutes les colonnes, pas seulement `nom` . Par exemple, si vous voulez savoir quels animaux sont nés après 1998, testez la colonne naissance :

```
mysql> SELECT * FROM animal WHERE naissance >= "1998-1-1";
+----------+--------+---------+------+------------+-------+
| nom      | maitre | espece  | sexe | naissance  | mort  |
+----------+--------+---------+------+------------+-------+
| Chirpy   | Gwen   | oiseau  | f    | 1998-09-11 | NULL  |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL  |
+----------+--------+---------+------+------------+-------+
```

Vous pouvez combiner plusieurs conditions, par exemple, pour trouver les chiennes :

```
mysql> SELECT * FROM animal WHERE espece = "chien" AND sexe = "f";
+-------+--------+--------+------+------------+-------+
| nom   | maitre | espece | sexe | naissance  | mort  |
+-------+--------+--------+------+------------+-------+
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL  |
+-------+--------+--------+------+------------+-------+
```

La requête précédente utilise l'opérateur logique `AND` . L'opérateur `OR` existe aussi :

```
mysql> SELECT * FROM animal WHERE espece = "serpent" OR espece = "oiseau";
+----------+--------+---------+------+------------+-------+
| nom      | maitre | espece  | sexe | naissance  | mort  |
+----------+--------+---------+------+------------+-------+
| Chirpy   | Gwen   | oiseau  | f    | 1998-09-11 | NULL  |
| Whistler | Gwen   | oiseau  | NULL | 1997-12-09 | NULL  |
| Slim     | Benny  | serpent | m    | 1996-04-29 | NULL  |
+----------+--------+---------+------+------------+-------+
```

`AND` et `OR` peuvent être utilisés ensemble. Si vous le faites, une bonne idée est d'utiliser les parenthèses pour indiquer comment les conditions doivent être regroupées :

```
mysql> SELECT * FROM animal WHERE (espece = "chat" AND sexe = "m")
    -> OR (espece = "chien" AND sexe = "f");
+-------+--------+--------+------+------------+-------+
| nom   | maitre | espece | sexe | naissance  | mort  |
+-------+--------+--------+------+------------+-------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL  |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL  |
+-------+--------+--------+------+------------+-------+
```

### 3.3.4.3 Sélectionner des colonnes particulières

Si vous ne voulez pas voir les lignes entières de votre table, nommez les colonnes qui vous intéressent, en les séparant par des virgules. Par exemple, si vous voulez savoir quand vos animaux sont nés, sélectionnez les colonnes `nom` et `naissance` :

```
mysql> SELECT nom, naissance FROM animal;
+----------+------------+
| nom      | naissance  |
+----------+------------+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
```

```
| Puffball | 1999-03-30 |
+----------+------------+
```

Pour trouver qui possède les animaux, utilisez cette requête :

```
mysql> SELECT maitre FROM animal;
+--------+
| maitre |
+--------+
| Harold |
| Gwen   |
| Harold |
| Benny  |
| Diane  |
| Gwen   |
| Gwen   |
| Benny  |
| Diane  |
+--------+
```

Toutefois, remarquez que la requête récupère le champ `maitre` de chaque enregistrement, et certains apparaissent plus d'une fois. Pour minimiser l'affichage, récupérez chaque résultat unique une seule fois en ajoutant le mot−clef `DISTINCT` :

```
mysql> SELECT DISTINCT maitre FROM animal;
+--------+
| maitre |
+--------+
| Benny  |
| Diane  |
| Gwen   |
| Harold |
+--------+
```

Vous pouvez utiliser une clause `WHERE` pour combiner la sélection des lignes avec celle des colonnes. Par exemple, pour obtenir les dates de naissance des chiens et chats uniquement, utilisez cette requête :

```
mysql> SELECT nom, espece, naissance FROM animal
   -> WHERE espece = "chien" OR espece = "chat";
+--------+--------+------------+
| nom    | espece | naissance  |
+--------+--------+------------+
| Fluffy | chat   | 1993-02-04 |
| Claws  | chat   | 1994-03-17 |
| Buffy  | chien  | 1989-05-13 |
| Fang   | chien  | 1990-08-27 |
| Bowser | chien  | 1989-08-31 |
+--------+--------+------------+
```

### 3.3.4.4 Trier les enregistrements

Vous avez sûrement noté dans les exemples précédents que les lignes de résultat sont affichées sans ordre particulier. Cependant, il est souvent plus facile d'examiner les résultats lorsqu'ils sont triés d'une manière significative. Pour trier un résultat, vous devez utiliser une clause `ORDER BY` .

L'exemple suivant présente les dates d'anniversaire des animaux, triées par date :

```
mysql> SELECT nom, naissance FROM animaux ORDER BY naissance;
+----------+------------+
| nom      | naissance  |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Sur les noms de colonnes, le tri − comme toutes les opérations de comparaison − est normalement exécuté sans tenir compte de la casse. Cela signifie que l'ordre sera indéfini pour les colonnes qui sont identiques, excepté leur casse. Vous pouvez forcer le tri sensible à la casse en utilisant la clause BINARY : `ORDER BY BINARY(champ)` . Pour trier dans l'ordre inverse, ajoutez le mot−clé `DESC` (décroissant) au nom de la colonne à trier :

```
mysql> SELECT nom, naissance FROM animaux ORDER BY naissance DESC;
+----------+------------+
| nom      | naissance  |
+----------+------------+
| Puffball | 1999-03-30 |
```

```
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

Vous pouvez effectuer un tri sur plusieurs colonnes. Par exemple, pour trier par types d'animaux, puis par la date d'anniversaire des animaux, en plaçant les plus jeunes en premier, utilisez la requête suivante :

```
mysql> SELECT nom, espece, naissance FROM animaux ORDER BY espece, naissance DESC;
+----------+---------+------------+
| nom      | espece  | naissance  |
+----------+---------+------------+
| Chirpy   | oiseau  | 1998-09-11 |
| Whistler | oiseau  | 1997-12-09 |
| Claws    | chat    | 1994-03-17 |
| Fluffy   | chat    | 1993-02-04 |
| Fang     | chien   | 1990-08-27 |
| Bowser   | chien   | 1989-08-31 |
| Buffy    | chien   | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim     | serpent | 1996-04-29 |
+----------+---------+------------+
```

Notez que le mot-clé `DESC` est appliqué uniquement au nom de la colonne qui le précède ( `naissance` ) ; les valeurs `espece` continuent à être triées dans l'ordre croissant.

### 3.3.4.5 Calcul sur les Dates

MySQL fournit plusieurs fonctions que vous pouvez utiliser pour effectuer des calculs sur les dates, par exemple, pour calculer l'âge ou pour extraire des parties de date.

Pour déterminer quel âge a chacun de vos animaux, vous devez calculer la différence entre l'année en cours et l'année de naissance, puis soustraire à la date courante si la date du jour se produit plus tôt dans l'année civile que la date de naissance. La requête suivante montre, pour chaque animal, la date de naissance, la date courante, ainsi que l'âge en années.

```
mysql> SELECT nom, naissance, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(naissance))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux;
+----------+------------+--------------+------+
| nom      |naissance   | CURRENT_DATE | age  |
+----------+------------+--------------+------+
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
+----------+------------+--------------+------+
```

Ici, `YEAR()` extrait l'année de la date et `RIGHT()` extrait les 5 caractères les plus à droite de la date qui représentent `MM-DD` (année civile). La partie de l'expression qui compare les valeurs de `MM-DD` évalue à 1 ou à 0, qui ajustent la différence d'année à la baisse, si `CURRENT_DATE` se produit plus au début de l'année que la `naissance` . L'expression complète est un peu plus fine en utilisant un alias ( `age` ) pour produire un nom de colonne un peu plus significatif.La requête fonctionne, mais le résultat pourrait être lu plus facilement si les lignes étaient présentées dans le même ordre. Cela peut être obtenu en ajoutant une clause `ORDER BY nom` pour trier le résultat par nom :

```
mysql> SELECT nom, naissance, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(naissance))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux ORDER BY nom;
+----------+------------+--------------+------+
| nom      |naissance   | CURRENT_DATE | age  |
+----------+------------+--------------+------+
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
```

```
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
+----------+------------+--------------+------+
```

Pour trier le résultat par l' `age` plutôt que par le `nom` , utilisez simplement une clause `ORDER BY` différente :

```
mysql> SELECT nom, naissance, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(naissance))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux ORDER BY age;
+----------+------------+--------------+------+
| nom      |naissance   | CURRENT_DATE | age  |
+----------+------------+--------------+------+
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
+----------+------------+--------------+------+
```

Une requête similaire peut être utilisée pour déterminer l'âge qu'avait un animal à sa mort. Vous determinez les animaux qui le sont en regardant les valeurs `mort` qui ne valent pas `NULL` . Alors, pour ceux dont la valeur est non `NULL` , calculez la différence entre la `mort` et la `naissance` :

```
mysql> SELECT nom, naissance, mort,
    -> (YEAR(death)-YEAR(naissance)) - (RIGHT(mort,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux WHERE mort IS NOT NULL ORDER BY age;
+--------+------------+------------+------+
| nom    |naissance   | mort       | age  |
+--------+------------+------------+------+
| Bowser | 1989-08-31 | 1995-07-29 |    5 |
+--------+------------+------------+------+
```

Cette requête utilise `mort IS NOT NULL` plutôt que `mort <> NULL` parce que `NULL` est une valeur spéciale. Cela sera expliqué plus tard. Travailler avec les valeurs `NULL` .Vous désirez savoir quels sont les animaux qui ont leur anniversaire le mois prochain ? Pour effectuer ce type de calculs, l'année et le jour ne sont pas utiles ; vous voulez simplement extraire le mois de la colonne `naissance` . MySQL fournit plusieurs fonctions d'extraction de parties de dates, comme `YEAR()` , `MONTH()` , et `DAYOFMONTH()` . `MONTH()` est la fonction appropriée dans notre cas. Pour voir comment cette fonction travaille, exécutez une requête simple qui retourne l' `naissance` et le `MONTH(naissance)` :

```
mysql> SELECT nom, naissance, MONTH(naissance) FROM animaux;
+----------+------------+------------------+
| nom      | naissance  | MONTH(naissance) |
+----------+------------+------------------+
| Fluffy   | 1993-02-04 |                2 |
| Claws    | 1994-03-17 |                3 |
| Buffy    | 1989-05-13 |                5 |
| Fang     | 1990-08-27 |                8 |
| Bowser   | 1989-08-31 |                8 |
| Chirpy   | 1998-09-11 |                9 |
| Whistler | 1997-12-09 |               12 |
| Slim     | 1996-04-29 |                4 |
| Puffball | 1999-03-30 |                3 |
+----------+------------+------------------+
```

Trouver les animaux qui ont leur anniversaire dans le mois suivant est aisé. Supposez que le mois courant est Avril. Donc, la valeur du mois est `4` et vous cherchez les animaux nés en Mai (mois 5) comme ceci :

```
mysql> SELECT nom, naissance FROM animaux WHERE MONTH(naissance) = 5;
+-------+------------+
| nom   |naissance   |
+-------+------------+
| Buffy | 1989-05-13 |
+-------+------------+
```

Il y a une petite complication si le mois courant est Décembre, bien sûr. Vous ne pouvez pas uniquement ajouter 1 au numéro du mois courant ( `12` ) et chercher les animaux qui sont nés le mois numéro 13, parce qu'il n'existe pas. A la place, vous cherchez les animaux nés en Janvier (mois numéro 1).

Vous pouvez toujours écrire une requête qui fonctionne quelque soit le mois courant. Comme cela, vous n'avez pas à utiliser un numéro de mois particulier dans votre requête. `DATE_ADD()` vous permet

d'ajouter un intervalle de temps à une date donnée. Si vous ajoutez un mois à la valeur de `NOW()` , et que vous extrayez le mois à l'aide de `MONTH()` , le résultat produit le mois dans lequel vous devez chercher un anniversaire :

```
mysql> SELECT nom, naissance FROM animaux
   -> WHERE MONTH(naissance) = MONTH(DATE_ADD(NOW(), INTERVAL 1 MONTH));
```

Une manière différente d'arriver au même résultat est d'ajouter `1` pour trouver le mois prochain après le mois courant (après l'usage de la fonction ( `MOD` ) pour ajouter à la valeur du mois la valeur `0` si il est de `12` ) :

```
mysql> SELECT nom, naissance FROM animaux
   -> WHERE MONTH(naissance) = MOD(MONTH(NOW()), 12) + 1;
```

Notez que `MONTH` retourne un nombre entre 1 et 12. `MOD(quelquechose,12)` retourne un nombre entre 0 et 11. Donc, l'addition doit être faite après l'utilisation de la fonction `MOD()` , sinon, nous aurions un intervalle entre Novembre (11) et Janvier (1).

### 3.3.4.6 Travailler avec la valeur NULL

La valeur `NULL` peut être surprenante jusqu'à ce que vous vous y habituiez. Conceptuellement, `NULL` représente une valeur qui manque, ou une valeur inconnue, et elle est traitée différemment des autres valeurs. Pour tester la présence de la valeur `NULL` , vous ne pouvez pas utiliser les opérateurs arithmétiques habituels comme `=` , `<` , ou `<>` . Pour le voir, il suffit d'essayer ceci :

```
mysql> SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 <> NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
|     NULL |      NULL |     NULL |     NULL |
+----------+-----------+----------+----------+
```

Clairement, vous n'obtiendrez aucun résultat valable pour ces comparaisons. Utilisez les opérateurs `IS NULL` et `IS NOT NULL` à la place :

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----------+---------------+
| 1 IS NULL | 1 IS NOT NULL |
+-----------+---------------+
|         0 |             1 |
+-----------+---------------+
```

Notez que deux `NULL` sont considérés comme égaux lors que vous utilisez la clause `GROUP BY` .

Avec MySQL, 0 et `NULL` représentent le booléen faux, et tout le reste représente le booléen vrai. La valeur par défaut du booléen vrai issue d'une comparaison est 1.

Lorsque vous utilisez la clause `ORDER BY` , les valeurs `NULL` sont toujours triées en premier, même si vous utilisez l'attribut `DESC` .

Ce traitement particulier de `NULL` explique pourquoi, dans la section précédente, il était nécessaire de déterminer quel animal ne vivait plus en utilisant la fonction `mort IS NOT NULL` au lieu de `mort <> NULL` .

### 3.3.4.7 Recherche de modèles

MySQL fournit le standard SQL des recherches de modèles, basé sur une extension des expressions régulières similaires à celles utilisées par les utilitaires Unix comme `vi` , `grep` , et `sed` .

La recherche de modèles SQL vous permet d'utiliser le caractère `'_'` pour trouver n'importe quel caractère et le caractère `'%'` pour trouver un nombre arbitraire de caractères (y compris aucun caractère). Dans MySQL, la recherche de modèles est sensible à la casse par défaut. Quelques exemples vous sont présentés ici. Notez que vous n'utilisez ni `=` ni `<>` lorsque vous utilisez la

recherche de modèles SQL ; utilisez les opérateurs de comparaison `LIKE` ou `NOT LIKE` à la place.

Pour trouver les noms commençant par la lettre `'b'` :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "b%";
+--------+--------+--------+------+------------+------------+
| nom    | maitre | espece | sexe | naissance  | mort       |
+--------+--------+--------+------+------------+------------+
| Buffy  | Harold | chien  | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | chien  | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+--------+------+------------+------------+
```

Pour trouver les noms finissant par `'fy'` :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "%fy";
+--------+--------+---------+------+------------+-------+
| nom    | maitre | espece  | sexe | naissance  | mort  |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | chat    | f    | 1993-02-04 | NULL  |
| Buffy  | Harold | chien   | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

Pour trouver les noms contenant le caractères `'w'` :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "%w%";
+----------+--------+---------+------+------------+------------+
| nom      | maitre | espece  | sexe | naissance  | mort       |
+----------+--------+---------+------+------------+------------+
| Claws    | Gwen   | chat    | m    | 1994-03-17 | NULL       |
| Bowser   | Diane  | chien   | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen   | oiseaux | NULL | 1997-12-09 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Pour trouver les noms contenant exactement 5 caractères, utilisez le caractère de recherche `'_'` :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "_____";
+-------+--------+--------+------+------------+------+
| nom   | maitre | espece | sexe | naissance  | mort |
+-------+--------+--------+------+------------+------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL |
+-------+--------+--------+------+------------+------+
```

L'autre type de recherche de modèles fourni par MySQL utilise les expressions régulières étendues. Lorsque vous testez une recherche avec ce type de modèle, utilisez les opérateurs `REGEXP` et `NOT REGEXP` (ou `RLIKE` et `NOT RLIKE` qui sont des synonymes).

Quelques caractéristiques des expressions régulières étendues sont :

- Le caractère `'.'` trouve n'importe quel caractère.
- Une classe de caractères `'[...]'` trouve n'importe quel caractère contenu entre les crochets. Par exemple, la classe de caractères `'[abc]'` trouve le caractère `'a'` , `'b'` , ou `'c'` . Pour définir un intervalle de caractères, utilisez un trait d'union. La classe de caractères `'[a-z]'` trouvera n'importe quel caractère minuscule, tout comme la classe `'[0-9]'` trouvera n'importe quel nombre.
- Le caractère `'*'` trouvera aucune ou plus d'instances du caractère qui le précède. Par exemple, `'x*'` trouvera n'importe quel nombre de fois le caractère `'x'` , `'[0-9]*'` trouvera n'importe quel nombre, et `'.*'` trouvera n'importe quel nombre de fois n'importe quel caractère.
- Le modèle est trouvé s'il se produit n'importe où dans la valeur testée. (Les modèles SQL ne sont trouvés que s'ils sont présents en valeur entière.)
- Pour ancrer un modèle de sorte qu'il soit trouvé au début ou à la fin de valeur testée, utilisez `'^'` au début ou bien `'$'` à la fin du modèle.

Pour démontrer comment les expressions régulières fonctionnent, les requêtes `LIKE` vues précédemment ont été réécrites pour utiliser `REGEXP` .

Pour trouver les noms qui commencent par la lettre `'b'` , utilisez `'^'` pour trouver le début du nom :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^b";
+--------+--------+--------+------+------------+------------+
| nom    | maitre | espece | sexe | naissance  | mort       |
+--------+--------+--------+------+------------+------------+
| Buffy  | Harold | chien  | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | chien  | m    | 1989-08-31 | 1995-07-29 |
```

```
+--------+--------+--------+------+------------+------------+
```

Avant la version 3.23.4 de MySQL, `REGEXP` était sensible à la casse, et la requête précédente ne retournait aucune ligne. Pour trouver la lettre `'b'` minuscule ou majuscule, utilisez cette requête à la place :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^[bB]";
```

Depuis MySQL 3.23.4, pour forcer `REGEXP` à être sensible à la casse, utilisez le mot−clé `BINARY` pour faire de la chaîne, une chaîne binaire. Cette requête trouvera uniquement la lettre minuscule `'b'` au début du nom :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP BINARY "^b";
```

Pour trouver les noms finissant par `'fy'` , utilisez `'$'` pour trouver la fin du nom :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "fy$";
+---------+--------+--------+-------+------------+------+
| nom     | maitre | espece | sexe  | naissance  | mort |
+---------+--------+--------+-------+------------+------+
| Fluffy  | Harold | chat   | f     | 1993-02-04 | NULL |
| Buffy   | Harold | chien  | f     | 1989-05-13 | NULL |
+---------+--------+--------+-------+------------+------+
```

Pour trouver les noms contenant la lettre `'w'` minuscule ou majuscule, utilisez la requête suivante :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "w";
+----------+--------+---------+------+------------+------------+
| nom      | maitre | espece  | sexe | naissance  | mort       |
+----------+--------+---------+------+------------+------------+
| Claws    | Gwen   | chat    | m    | 1994-03-17 | NULL       |
| Bowser   | Diane  | chien   | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen   | oiseaux | NULL | 1997-12-09 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Parce qu'une expression régulière est trouvée si le modèle se trouve n'importe où dans la valeur, il n'est pas nécessaire dans la requête précédente de mettre un joker de chaque côté du modèle recherché pour trouver la valeur entière comme cela aurait été le cas en utilisant les modèles de recherche SQL.

Pour trouver les noms contenant exactement 5 caractères, utilisez `'^'` et `'$'` pour trouver le début et la fin du nom, et 5 instances de `'.'` au milieu :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^.....$";
+-------+--------+--------+------+------------+------+
| nom   | maitre | espece | sexe | naissance  | mort |
+-------+--------+--------+------+------------+------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL |
+-------+--------+--------+------+------------+------+
```

Vous pouvez aussi écrire la requête suivante en utilisant l'opérateur `'{n'` } ``répéter− `n` −fois'' :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^.&#123;5&#125;$";
+-------+--------+--------+------+------------+------+
| nom   | maitre | espece | sexe | naissance  | mort |
+-------+--------+--------+------+------------+------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL |
+-------+--------+--------+------+------------+------+
```

### 3.3.4.8 Compter les lignes

Les bases de données sont souvent employées pour répondre à la question : ``Combien de fois un certain type de données se trouve dans la table ?'' Par exemple, vous aimeriez savoir combien d'animaux vous avez, ou bien combien d'animaux chaque propriétaire possède, ou encore savoir différentes choses concernant vos animaux.

Savoir combien vous avez d'animaux revient à se poser la question : ``Combien de lignes y a−t−il dans la table `animaux` ?'' parcequ'il y a un enregistrement par animaux. La fonction `COUNT()` compte le nombre de résultats non `NULL` , donc, la requête pour compter les animaux ressemble à ceci :

```
mysql> SELECT COUNT(*) FROM animaux;
+----------+
```

```
| COUNT(*) |
+----------+
|        9 |
+----------+
```

Vous pouvez trouver également les noms des propriétaires des animaux. Vous pouvez utiliser `COUNT()` si vous voulez trouver combien d'animaux possède chaque propriétaire :

```
 mysql> SELECT maitre, COUNT(*) FROM animaux GROUP BY maitre;
+--------+----------+
| maitre | COUNT(*) |
+--------+----------+
| Benny  |        2 |
| Diane  |        2 |
| Gwen   |        3 |
| Harold |        2 |
+--------+----------+
```

Notez l'utilisation de la clause `GROUP BY` pour grouper tous les enregistrements par propriétaire. Sans cela, vous auriez le message d'erreur suivant :

```
 mysql> SELECT maitre, COUNT(maitre) FROM animaux;
ERROR 1140 at line 1: Mixing of GROUP columns (MIN(),MAX(),COUNT()...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

`COUNT()` et `GROUP BY` sont utiles pour caractériser vos données de diverses façons. Les exemples suivants montrent différentes manières pour obtenir des statistiques sur les animaux.

Nombre d'animaux par espèce :

```
 mysql> SELECT espece, COUNT(*) FROM animaux GROUP BY espece;
+---------+----------+
| espece  | COUNT(*) |
+---------+----------+
| oiseau  |        2 |
| chat    |        2 |
| chien   |        3 |
| hamster |        1 |
| serpent |        1 |
+---------+----------+
```

Nombre d'animaux par sexe :

```
 mysql> SELECT sexe, COUNT(*) FROM animaux GROUP BY sexe;
+------+----------+
| sexe | COUNT(*) |
+------+----------+
| NULL |        1 |
| f    |        4 |
| m    |        4 |
+------+----------+
```

(Dans ce résultat, `NULL` indique les sexes inconnus.)Nombre d'animaux par espèce et sexe :

```
 mysql> SELECT espece, sexe, COUNT(*) FROM animaux GROUP BY espece, sexe;
+---------+------+----------+
| espece  | sexe | COUNT(*) |
+---------+------+----------+
| oiseau  | NULL |        1 |
| oiseau  | f    |        1 |
| chat    | f    |        1 |
| chat    | m    |        1 |
| chien   | f    |        1 |
| chien   | m    |        2 |
| hamster | f    |        1 |
| serpent | m    |        1 |
+---------+------+----------+
```

Vous n'avez pas besoin de rechercher une table entière quand vous employez `COUNT()` . Par exemple, la requête précédente, si vous voulez trouver uniquement les chiens et les chats, ressemble à cela :

```
 mysql> SELECT espece, sexe, COUNT(*) FROM animaux
    -> WHERE espece = "chien" OR espece = "chat"
    -> GROUP BY espece, sexe;
+---------+------+----------+
| espece  | sexe | COUNT(*) |
+---------+------+----------+
| chat    | f    |        1 |
| chat    | m    |        1 |
| chien   | f    |        1 |
| chien   | m    |        2 |
+---------+------+----------+
```

Ou bien, si vous voulez trouver le nombre d'animaux par sexe, uniquement pour les animaux dont le sexe est connu :

```
mysql> SELECT espece, sexe, COUNT(*) FROM animaux
    -> WHERE sexe IS NOT NULL
    -> GROUP BY espece, sexe;
+---------+------+----------+
| espece  | sexe | COUNT(*) |
+---------+------+----------+
| oiseau  | f    |        1 |
| chat    | f    |        1 |
| chat    | m    |        1 |
| chien   | f    |        1 |
| chien   | m    |        2 |
| hamster | f    |        1 |
| serpent | m    |        1 |
+---------+------+----------+
```

### 3.3.4.9 Utiliser plus d'une table

La table `animal` garde les enregistrements de vos animaux. Si vous voulez enregistrer d'autres informations concernant vos animaux, comme les événements de leurs vies, les visites chez le vétérinaire, ou encore lorsqu'ils ont mis bas, vous avez besoin d'une autre table. De quoi a besoin cette table ? Elle doit :

- Contenir le nom de l'animal pour savoir à quel animal cet événement se rattache.
- Une date pour savoir quand a eu lieu l'événement.
- Un champ qui décrit l'événement.
- Un champ de type événement, si vous voulez être capable de cataloguer les événements.

En prenant cela en considération, le code `CREATE TABLE` pour la table `evenement` doit ressembler à ceci :

```
mysql> CREATE TABLE evenement (nom VARCHAR(20), date DATE,
    -> type VARCHAR(15), remarque VARCHAR(255));
```

Tout comme la table `animal` , il est facile d'enregistrer les enregistrements initiaux en créant un fichier texte délimité par des tabulations, contenant l'information :

| nom | date | type | remarque |
|---|---|---|---|
| Fluffy | 1995-05-15 | mise bas | 4 chattons, 3 femelles, 1 mâles |
| Buffy | 1993-06-23 | mise bas | 5 chiots, 2 femelles, 3 mâles |
| Buffy | 1994-06-19 | mise bas | 3 chiots, 3 femelles |
| Chirpy | 1999-03-21 | vétérinaire | needed beak straightened |
| Slim | 1997-08-03 | vétérinaire | broken rib |
| Bowser | 1991-10-12 | chenil | |
| Fang | 1991-10-12 | chenil | |
| Fang | 1998-08-28 | anniversaire | Don d'un nouvel objet de mastication |
| Claws | 1998-03-17 | anniversaire | Don d'un nouveau collier anti-puces |
| Whistler | 1998-12-09 | anniversaire | Premier anniversaire |

Chargez ces enregistrements comme cela :

```
mysql> LOAD DATA LOCAL INFILE "evenement.txt" INTO TABLE evenement;
```

En se basant sur ce que vous avez appris des requêtes effectuées sur la table `animal` , vous devriez être capable de faire des recherches sur les enregistrements de la table `evenement` ; le principe est le même. Quand devez-vous vous demander si la table `evenement` est seule suffisante pour répondre à votre question ?

Supposez que vous voulez trouver l'âge de chaque animal lorsqu'il a mis bas. La table `evenement` indique quand cela s'est produit, mais pour le calcul de l'âge de la mère, vous avez besoin de sa date de naissance. Parce que ces informations sont stockées dans la table `animal` , vous avez besoin des deux tables pour cette requête :

```
mysql> SELECT animal.nom,
    -> (TO_DAYS(date) - TO_DAYS(naissance))/365 AS age,
    -> remarque
    -> FROM animal, evenement
    -> WHERE animal.nom = evenement.nom AND type = "mise bas";
+--------+------+-------------------------------+
| nom    | age  | remarque                      |
+--------+------+-------------------------------+
| Fluffy | 2.27 | 4 chattons, 3 femelles, 1 mâle |
```

```
| Buffy  | 4.12 | 5 chiots, 2 femelles, 3 mâles  |
| Buffy  | 5.10 | 3 chiots, 3 femelles           |
+--------+------+--------------------------------+
```

Il y a plusieurs choses à noter concernant cette requête :

- La clause `FROM` liste les deux tables parce que la requête a besoin d'informations contenues dans ces deux tables.
- Lorsque vous combinez (joignez) des informations provenant de plusieurs tables, vous devez spécifier quels enregistrements d'une table peuvent être associés à quels enregistrements des autres tables. C'est aisé parce qu'elles ont toutes les deux une colonne `nom` . La requête utilise la clause `WHERE` pour faire correspondre les enregistrements des deux tables sur les valeurs de la colonne `nom` .
- Parce que la colonne `nom` apparaît dans les deux tables, vous devez être explicite concernant la table que vous utilisez lorsque vous vous référez à cette colonne. C'est fait en faisant précéder le nom de la colonne par le nom de la table.

Vous n'avez pas besoin de deux tables différentes pour effectuer une jointure. Quelques fois, c'est plus facile de joindre une table sur elle−même, si vous voulez comparer des enregistrements dans une table avec d'autres enregistrements de la même table. Par exemple, pour trouver des paires multiples parmi vos animaux, vous pouvez joindre la table `animal` sur elle−même pour trouver les paires mâles / femelles par rapport à l'espèce :

```
 mysql> SELECT p1.nom, p1.sexe, p2.nom, p2.sexe, p1.espece
    -> FROM animal AS p1, animal AS p2
    -> WHERE p1.espece = p2.espece AND p1.sexe = "f" AND p2.sexe = "m";
+--------+------+--------+------+---------+
| nom    | sexe | nom    | sexe | espece  |
+--------+------+--------+------+---------+
| Fluffy | f    | Claws  | m    | chat    |
| Buffy  | f    | Fang   | m    | chien   |
| Buffy  | f    | Bowser | m    | chien   |
+--------+------+--------+------+---------+
```

Dans cette requête, nous avons spécifié des alias pour les noms de tables dans l'ordre de référence des colonnes et ainsi maintenir directement à quelle instance de la table chaque colonne est associée.

## 3.4 Obtenir des informations à propos des bases de données et des tables

Que faire si vous oubliez le nom d'une base de données ou d'une table, ou bien encore la structure d'une table donnée (par exemple, comment se nomment ses colonnes) ?

MySQL répond à ce problème en fournissant plusieurs commandes qui renvoient des informations à propos des tables et des bases de données les contenant.

Vous avez déjà vu `SHOW DATABASES` qui liste les bases de données gérées par le serveur. Pour trouver quelle base de données est actuellement sélectionnée, utilisez la fonction `DATABASE()` :

```
 mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+
```

Si vous n'avez encore sélectionné aucune base de données, le résultat est vide.

Pour trouver quelles sont les tables que la base contient (par exemple, quand vous n'êtes pas sûr du nom d'une table), utilisez cette commande :

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| evenement           |
| animal              |
+---------------------+
```

Si vous voulez en savoir d'avantage sur la structure d'une table, la commande `DESCRIBE` est utile ; elle fournit des informations sur chaque colonne de la table :

```
mysql> DESCRIBE animal;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| nom       | varchar(20) | YES  |     | NULL    |       |
| maitre    | varchar(20) | YES  |     | NULL    |       |
| espece    | varchar(20) | YES  |     | NULL    |       |
| sexe      | char(1)     | YES  |     | NULL    |       |
| naissance | date        | YES  |     | NULL    |       |
| mort      | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

`Field` indique le nom de la colonne, `Type` est son type de données, `NULL` indique si la colonne peut contenir des valeurs `NULL` , `Key` indique si la colonne est indexée et `Default` spécifie la valeur par défaut de la colonne.Si vous avez des index sur une table, `SHOW INDEX FROM nom_de_table` vous fournira des informations sur elles.

## 3.5 Exemples de requêtes usuelles

Voilà des exemples qui vous serviront à résoudre les problèmes communs avec MySQL.

Certains exemples utilisent la table `shop` pour sauvegarder le prix de chaque article (numéro de l'élément) pour certains vendeurs (dealers). En supposant que chaque vendeur à un prix fixe pour chaque article, le couple ( `article` , `dealer` ) est une clef primaire pour les enregistrements.

Démarrez le client en ligne de commande `mysql` et sélectionnez une base de données :

```
mysql nom-base-données
```

(Dans la plupart des installations de MySQL, vous pouvez utiliser la base de données 'test').

Vous pouvez créer la table d'exemple de la façon suivante :

```
CREATE TABLE shop (
article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
dealer  CHAR(20)                 DEFAULT ''     NOT NULL,
price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL,
PRIMARY KEY(article, dealer));

INSERT INTO shop VALUES
(1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),(3,'C',1.69),
(3,'D',1.25),(4,'D',19.95);
```

Ok, les données d'exemple sont :

```
mysql> SELECT * FROM shop;

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | A      |  3.45 |
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | B      |  1.45 |
|    0003 | C      |  1.69 |
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

### 3.5.1 La valeur maximale d'une colonne

``Quel est le numéro du plus grand élément ?''

```
 SELECT MAX(article) AS article FROM shop

 +---------+
| article |
+---------+
|       4 |
+---------+
```

### 3.5.2 La ligne contenant le maximum d'une certaine colonne

``Trouvez le numéro, vendeur et prix de l'article le plus cher.''

En ANSI SQL cela est facilement fait avec une sous−requête :

```
 SELECT article, dealer, price
FROM   shop
WHERE  price=(SELECT MAX(price) FROM shop)
```

En MySQL (qui ne gère pas encore les sous−requêtes), vous devez le faire en deux temps :

- Obtenir le plus grand prix de la table avec une requête `SELECT` .
- Utiliser cette valeur avec cette requête :

```
 SELECT article, dealer, price
FROM   shop
WHERE  price=19.95
```

Une autre solution est de trier toutes les lignes en ordre décroissant des prix et ne choisir que la première ligne avec la clause `LIMIT` qui est spécifique à MySQL :

```
 SELECT article, dealer, price
FROM   shop
ORDER BY price DESC
LIMIT 1
```

**NOTE** : s'il y a beaucoup d'articles chers (par exemple, chaque 19.95) la solution avec `LIMIT` n'en montre qu'un !.

### 3.5.3 Maximum d'une colonne par groupe

``Quel est le plus grand prix par article ?''

```
 SELECT article, MAX(price) AS price
FROM   shop
GROUP BY article

 +---------+-------+
| article | price |
+---------+-------+
|    0001 |  3.99 |
|    0002 | 10.99 |
|    0003 |  1.69 |
|    0004 | 19.95 |
+---------+-------+
```

### 3.5.4 La ligne contenant la plus grande valeur d'un certain champ par rapport à un groupe

``Pour chaque article, trouvez le ou les vendeurs avec le plus haut prix.''

En ANSI SQL, je l'aurais fait de cette façon avec une sous−requête :

```
 SELECT article, dealer, price
FROM   shop s1
WHERE  price=(SELECT MAX(s2.price)
              FROM shop s2
              WHERE s1.article = s2.article);
```

En MySQL il vaut mieux le faire en plusieurs étapes :

- Récupérer la liste de (article, plusgrandprix).
- Pour chaque article, récupérer la ligne qui a le plus grand prix stocké.

Cela se fait facilement avec une table temporaire :

```
 CREATE TEMPORARY TABLE tmp (
        article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
        price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL);

 LOCK TABLES shop read;

 INSERT INTO tmp SELECT article, MAX(price) FROM shop GROUP BY article;

 SELECT shop.article, dealer, shop.price FROM shop, tmp
WHERE shop.article=tmp.article AND shop.price=tmp.price;

 UNLOCK TABLES;

 DROP TABLE tmp;
```

Si vous n'utilisez pas une table `TEMPORARY` , vous devez aussi verrouiller celle−ci.``Peut−on le faire avec une seule requête ?''

Oui, mais en utilisant une astuce inefficace que j'appelle ``astuce du MAX−CONCAT'':

```
 SELECT article,
       SUBSTRING( MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 7) AS dealer,
  0.00+LEFT(      MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 6) AS price
FROM   shop
GROUP BY article;

 +---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | C      |  1.69 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

Le dernier exemple peut, bien sûr, être amélioré en découpant les colonnes concaténées dans le client.

## 3.5.5 Utiliser les variables utilisateur

Vous pouvez utiliser les variables utilisateur de MySQL pour garder des résultats en mémoire sans avoir à les enregistrer dans des variables temporaires du client. Variables définies par l'utilisateur .

Par exemple, pour trouver l'article avec le plus haut et le plus bas prix, vous pouvez faire :

```
 mysql> SELECT &#64;min_price:=MIN(price),&#64;max_price:=MAX(price) FROM shop;
mysql> SELECT * FROM shop WHERE price=&#64;min_price OR price=&#64;max_price;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

## 3.5.6 Utiliser les clefs étrangères

Depuis la version 3.23.44 de MySQL, les tables `InnoDB` supportent les contraintes des clefs étrangères. Tables `InnoDB` . Consultez aussi Clés étrangères .

Actuellement, vous n'avez pas besoin de clefs étrangères pour réaliser des jointures entre les tables. La seule chose que MySQL ne fait pas encore (avec les types autres que `InnoDB` ), est `CHECK` pour s'assurer que que la clef que vous utilisez existe bien dans la ou les tables que vous référencez et il n'efface pas automatiquement les lignes d'une table avec une définition de clef étrangère. Si vous utilisez vos clefs comme une clef normale, tout marchera parfaitement :

```
 CREATE TABLE person (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);

 CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL REFERENCES person(id),
    PRIMARY KEY (id)
);
 INSERT INTO person VALUES (NULL, 'Antonio Paz');

 INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', LAST_INSERT_ID()),
(NULL, 'dress', 'white', LAST_INSERT_ID()),
(NULL, 't-shirt', 'blue', LAST_INSERT_ID());


 <P>
INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');

 INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', LAST_INSERT_ID()),
(NULL, 'polo', 'red', LAST_INSERT_ID()),
(NULL, 'dress', 'blue', LAST_INSERT_ID()),
(NULL, 't-shirt', 'white', LAST_INSERT_ID());

 </P>

SELECT * FROM person;
+----+---------------------+
| id | name                |
+----+---------------------+
|  1 | Antonio Paz         |
|  2 | Lilliana Angelovska |
+----+---------------------+

 SELECT * FROM shirt;
+----+---------+--------+-------+
| id | style   | color  | owner |
+----+---------+--------+-------+
|  1 | polo    | blue   |     1 |
|  2 | dress   | white  |     1 |
|  3 | t-shirt | blue   |     1 |
|  4 | dress   | orange |     2 |
|  5 | polo    | red    |     2 |
|  6 | dress   | blue   |     2 |
|  7 | t-shirt | white  |     2 |
+----+---------+--------+-------+
 SELECT s.* FROM person p, shirt s
 WHERE p.name LIKE 'Lilliana%'
   AND s.owner = p.id
   AND s.color <> 'white';

 +----+-------+--------+-------+
| id | style | color  | owner |
+----+-------+--------+-------+
|  4 | dress | orange |     2 |
|  5 | polo  | red    |     2 |
|  6 | dress | blue   |     2 |
+----+-------+--------+-------+
```

## 3.5.7 Recherche sur deux clefs

MySQL n'optimise pas encore quand vous effectuez des recherches sur deux clefs différentes combinées avec `OR` (la recherche sur une clef avec différentes parties `OR` est elle pas mal optimisée) :

```
 SELECT champ1_index, champ2_index FROM test_table WHERE champ1_index = '1'
OR  champ2_index = '1'
```

La raison est que nous n'avons pas trouvé le temps suffisant pour parvenir à un moyen efficace de gérer cela dans un cas général. (En comparaison, la gestion de `AND` est maintenant complètement générale et fonctionne très bien.) Pour le moment, vous pouvez résoudre ce problème efficacement en utilisant une table temporaire ( `TEMPORARY` ). Ce type d'optimisation est très utile si vous utilisez des requêtes très complexes et que le serveur SQL fait une optimisation dans le mauvais ordre.

```
 CREATE TEMPORARY TABLE tmp
SELECT champ1_index, champ2_index FROM test_table WHERE champ1_index = '1';
INSERT INTO tmp
SELECT champ1_index, champ2_index FROM test_table WHERE champ2_index = '1';
SELECT * from tmp;
DROP TABLE tmp;
```

La méthode ci−dessus pour résoudre cette requête est en effet une `UNION` de deux requêtes. Syntaxe de `UNION` .

## 3.5.8 Calcul du nombre de visites par jour

Ce qui suit donne une idée d'une utilisation des fonctions de bits pour calculer le nombre de jours par mois où un utilisateur a visité une page web.

```
 CREATE TABLE t1 (annee YEAR(4), mois INT(2) UNSIGNED ZEROFILL,
            jour INT(2) UNSIGNED ZEROFILL);
INSERT INTO t1 VALUES(2000,1,1),(2000,1,20),(2000,1,30),(2000,2,2),
           (2000,2,23),(2000,2,23);
SELECT annee,mois,BIT_COUNT(BIT_OR(1<<jour)) AS jours FROM t1
       GROUP BY annee,mois;

 Qui retourne :

 +-------+-------+-------+
| annee | mois  | jours |
+-------+-------+-------+
|  2000 |    01 |     3 |
|  2000 |    02 |     2 |
+-------+-------+-------+
```

Ce qui précède calcule le nombre de jours différents qui a été utilisé pour une combinaison année/mois, avec suppression automatique des doublons.

## 3.5.9 Utiliser AUTO_INCREMENT

L'attribut `AUTO_INCREMENT` peut être utilisé pour générer un identifiant unique pour les nouvelles lignes :

```
 CREATE TABLE animals (
            id MEDIUMINT NOT NULL AUTO_INCREMENT,
            name CHAR(30) NOT NULL,
            PRIMARY KEY (id)
            );
INSERT INTO animals (name) VALUES ("dog"),("cat"),("penguin"),
                                  ("lax"),("whale");
SELECT * FROM animals;

 Qui retourne :

 +----+---------+
| id | name    |
+----+---------+
|  1 | dog     |
|  2 | cat     |
|  3 | penguin |
|  4 | lax     |
|  5 | whale   |
+----+---------+
```

Vous pouvez obtenir la valeur utilisée de la clef `AUTO_INCREMENT` avec la fonction SQL `LAST_INSERT_ID()` ou la fonction d'API `mysql_insert_id()` .

Note : Pour une insertion multi−lignes, `LAST_INSERT_ID()` / `mysql_insert_id()` retourneront la clef `AUTO_INCREMENT` de la **première** ligne insérée. Cela permet de reproduire les insertions multi−lignes sur d'autres

services.

Pour les tables MyISAM et BDB vous pouvez spécifier `AUTO_INCREMENT` sur une colonne secondaire d'une clef multi−colonnes. Dans ce cas, la valeur générée pour la colonne auto−incrementée est calculée de la façon suivante : `MAX(auto_increment_column)+1) WHERE prefix=given−prefix` . C'est utile lorsque vous voulez placer des données dans des groupes ordonnés.

```
 CREATE TABLE animals (
             grp ENUM('fish','mammal','bird') NOT NULL,
             id MEDIUMINT NOT NULL AUTO_INCREMENT,
             name CHAR(30) NOT NULL,
             PRIMARY KEY (grp,id)
             );
INSERT INTO animals (grp,name) VALUES("mammal","dog"),("mammal","cat"),
                  ("bird","penguin"),("fish","lax"),("mammal","whale");
SELECT * FROM animals ORDER BY grp,id;

 Qui retourne :

 +--------+----+---------+
| grp    | id | name    |
+--------+----+---------+
| fish   |  1 | lax     |
| mammal |  1 | dog     |
| mammal |  2 | cat     |
| mammal |  3 | whale   |
| bird   |  1 | penguin |
+--------+----+---------+
```

Notez que dans ce cas, la valeur d' `AUTO_INCREMENT` sera réutilisée si vous effacez la ligne avec la plus grande valeur d' `AUTO_INCREMENT` tous groupes confondus.

## 3.6 Utilisation de mysql en mode batch

Dans les sections précédentes, vous avez utilisé `mysql` inter activement pour entrer vos requêtes et voir les résultats. Vous pouvez aussi utiliser `mysql` en mode batch. Pour ce faire, placez les commandes que vous voulez exécuter dans un fichier, puis dites à `mysql` de lire les entrées à partir de celui−ci :

```
 shell> mysql < fichier−batch
```

Si vous utilisez `mysql` sous windows et que vous avez des caractères spéciaux dans le fichier qui posent problèmes, vous pouvez faire :

```
 dos> mysql −e "source fichier−batch"
```

Si vous devez spécifier les paramètres de connexion en ligne de commande, la commande ressemblera à ca :

```
 shell> mysql −h hôte −u utilisateur −p < fichier−batch
Enter password: ********
```

Lorsque vous utilisez `mysql` de cette façon, vous créez un fichier de script, puis vous l'exécutez.Si vous voulez que le script continue, même si il y a des erreurs, vous devez utiliser l'option `−−force` de la ligne de commande.

Pourquoi utilisez un script ? Voici quelques raisons :

- Si vous utilisez une requête de façon répétitive (c'est à dire, chaque jour, ou chaque semaine), en faire un script vous évitera de la réécrire chaque fois.
- Vous pouvez générer de nouvelles requêtes à partir de requêtes existantes et similaires en copiant et éditant des fichiers de scripts.
- Ce mode peut aussi être utile lors du développement d'une requête, particulièrement pour les commandes sur plusieurs lignes ou plusieurs séquences de commandes. Si vous commetez une erreur, vous n'avez pas à tout récrire. Editez juste votre script pour corriger l'erreur et dites à `mysql` de l'exécuter à nouveau.
- Si vous avez une requête qui produit beaucoup d'affichage, vous pouvez le rediriger vers un visionneur plutôt que de le regarder défiler sur votre écran :

```
shell> mysql < fichier-batch | more
```

- Vous pouvez capturer l'affichage dans un fichier pour un traitement ultérieur :

```
shell> mysql < fichier_batch > mysql.out
```

- Vous pouvez distribuer votre script à d'autres personnes pour qu'elles l'exécutent.
- Quelques situations ne permettent pas une utilisation interactive, par exemple, quand vous exécutez une requête à partir d'une tâche `cron` . Dans ce cas, vous devez utiliser le mode batch.

Le format d'affichage par défaut est différent (plus concis) lorsque vous exécutez `mysql` en mode batch de celui utilisé inter activement. Par exemple, le résultat de `SELECT DISTINCT espece FROM animal` ressemble à ça inter activement :

```
 +---------+
| espece  |
+---------+
| oiseau  |
| chat    |
| chien   |
| hamster |
| serpent |
+---------+
```

Mais à ça en mode batch :

```
 espece
oiseau
chat
chien
hamster
serpent
```

Si vous voulez le format d'affichage interactif en mode batch, utilisez `mysql -t` . Pour écrire les commandes exécutez dans la sortie, utilisez `mysql -vvv` .

Vous pouvez aussi utiliser un script à partir de l'invite `mysql` en utilisant la commande `source` :

```
mysql> source nom_fichier;
```

## 3.7 Requêtes du projet Twin

A Analytikerna et Lentus, nous avons eu à mettre en place le partie système et base de données d'un grand projet de recherche. Ce projet est une collaboration entre l'institut de médecine environnementale de l'institut de Karolinska Stockholm et la section de recherche clinique d'âge et de psychologie à l'université de la Californie du sud.

Le projet nécessite une partie de récolte d'informations où tous les jumeaux en Suède de plus de 65 ans sont contactés par téléphone. Ceux qui répondent à certains critères sont admis à la seconde étape. Dans celle−ci, les jumeaux qui veulent participer rencontrent une équipe de médecins/infirmiers. Les examens incluent des examens physiques et neuropsychologiques, des tests en laboratoire, de la neuro−imagerie, des études psychologiques et de la collecte d'informations relatives à la famille. En plus de tout cela, les données à propos des facteurs de risques médicaux et environnementaux sont collectées.

Plus d'informations à propos de l'étude Twin peuvent être trouvées sur : http://www.imm.ki.se/TWIN/TWINUKW.HTM

La dernière partie de ce projet est administrée avec une interface web écrite en utilisant Perl et MySQL.

Chaque nuit, toutes les informations des interviews sont stockées dans une base de données MySQL.

## 3.7.1 Trouver tous les jumeaux répondant aux critères

La requête suivante a été utilisée pour déterminer qui participerait à la seconde partie du projet :

```
 SELECT
        CONCAT(p1.id, p1.tvab) + 0 AS tvid,
        CONCAT(p1.christian_name, " ", p1.surname) AS Name,
        p1.postal_code AS Code,
        p1.city AS City,
        pg.abrev AS Area,
        IF(td.participation = "Aborted", "A", " ") AS A,
        p1.dead AS dead1,
        l.event AS event1,
        td.suspect AS tsuspect1,
        id.suspect AS isuspect1,
        td.severe AS tsevere1,
        id.severe AS isevere1,
        p2.dead AS dead2,
        l2.event AS event2,
        h2.nurse AS nurse2,
        h2.doctor AS doctor2,
        td2.suspect AS tsuspect2,
        id2.suspect AS isuspect2,
        td2.severe AS tsevere2,
        id2.severe AS isevere2,
        l.finish_date
FROM
        twin_project AS tp
        /* For Twin 1 */
        LEFT JOIN twin_data AS td ON tp.id = td.id
                  AND tp.tvab = td.tvab
        LEFT JOIN informant_data AS id ON tp.id = id.id
                  AND tp.tvab = id.tvab
        LEFT JOIN harmony AS h ON tp.id = h.id
                  AND tp.tvab = h.tvab
        LEFT JOIN lentus AS l ON tp.id = l.id
                  AND tp.tvab = l.tvab
        /* For Twin 2 */
        LEFT JOIN twin_data AS td2 ON p2.id = td2.id
                  AND p2.tvab = td2.tvab
        LEFT JOIN informant_data AS id2 ON p2.id = id2.id
                  AND p2.tvab = id2.tvab
        LEFT JOIN harmony AS h2 ON p2.id = h2.id
                  AND p2.tvab = h2.tvab
        LEFT JOIN lentus AS l2 ON p2.id = l2.id
                  AND p2.tvab = l2.tvab,
        person_data AS p1,
        person_data AS p2,
        postal_groups AS pg
WHERE
        /* p1 gets main twin and p2 gets his/her twin. */
        /* ptvab is a field inverted from tvab */
        p1.id = tp.id AND p1.tvab = tp.tvab AND
        p2.id = p1.id AND p2.ptvab = p1.tvab AND
        /* Just the sceening survey */
        tp.survey_no = 5 AND
        /* Skip if partner died before 65 but allow emigration (dead=9) */
        (p2.dead = 0 OR p2.dead = 9 OR
         (p2.dead = 1 AND
          (p2.death_date = 0 OR
           (((TO_DAYS(p2.death_date) − TO_DAYS(p2.birthday)) / 365)
            >= 65))))
        AND
        (
        /* Twin is suspect */
        (td.future_contact = 'Yes' AND td.suspect = 2) OR
        /* Twin is suspect − Informant is Blessed */
        (td.future_contact = 'Yes' AND td.suspect = 1
                                   AND id.suspect = 1) OR
        /* No twin − Informant is Blessed */
        (ISNULL(td.suspect) AND id.suspect = 1
                            AND id.future_contact = 'Yes') OR
        /* Twin broken off − Informant is Blessed */
        (td.participation = 'Aborted'
         AND id.suspect = 1 AND id.future_contact = 'Yes') OR
        /* Twin broken off − No inform − Have partner */
        (td.participation = 'Aborted' AND ISNULL(id.suspect)
                                      AND p2.dead = 0))
        AND
        l.event = 'Finished'
        /* Get at area code */
        AND SUBSTRING(p1.postal_code, 1, 2) = pg.code
        /* Not already distributed */
        AND (h.nurse IS NULL OR h.nurse=00 OR h.doctor=00)
        /* Has not refused or been aborted */
        AND NOT (h.status = 'Refused' OR h.status = 'Aborted'
        OR h.status = 'Died' OR h.status = 'Other')
ORDER BY
        tvid;
```

Quelques explications :

`CONCAT(p1.id, p1.tvab) + 0 AS tvid`
: Nous voulons trier la concaténation de `id` et `tvab` dans un ordre numérique. Ajouter `0` au résultat force MySQL à le considérer comme un nombre.

*colonne* `id`
: Identifie une paire de jumeaux. C'est un clef dans toutes les tables.

*colonne* `tvab`
: Identifie un jumeau dans une paire. Valeur `1` ou `2` .

*colonne* `ptvab`
: Inverse de `tvab` . Si `tvab` est `1` c'est égal à `2` , et vice−versa. Elle existe pour diminuer la frappe et faciliter la tâche à MySQL lors de l'optimisation de la requête.

Cette requête montre, entre autres, comment faire pour consulter une table depuis cette même table en utilisant une jointure ( `p1` et `p2` ). Dans cet exemple, est utilisé pour chercher quel partenaire du projet est décédé avant l'âge de 65 ans. Si c'est le cas, la ligne n'est pas retournée.Tout ce qui précède existe dans toutes les tables avec des informations relatives aux jumeaux. Nous avons une clé sur les champs `id,tvab` (toutes les tables), et sur les champs `id,ptvab` ( `person_data` ) pour accélérer les requêtes.

Sur notre machine de production (un 200MHz UltraSPARC), cette requête retourne près de 150−200 lignes et prend moins d'une seconde.

Le nombre d'enregistrements dans les tables utilisées plus haut :

| **Table** | **Lignes** |
|---|---|
| person_data | 71074 |
| lentus | 5291 |
| twin_project | 5286 |
| twin_data | 2012 |
| informant_data | 663 |
| harmony | 381 |
| postal_groups | 100 |

## 3.7.2 Afficher une table avec l'état des paires de jumeaux

Chaque entrevue se finit avec un code d'état, appelé `event` . La requête ci−dessous montre comment afficher une table avec toutes les paires, rassemblées par code d'état. Elle indique combien de paires ont terminé, combien de paires ont à moitié terminé et combien on refusé, etc.

```
 SELECT
        t1.event,
        t2.event,
        COUNT(*)
FROM
        lentus AS t1,
        lentus AS t2,
        twin_project AS tp
WHERE
        /* We are looking at one pair at a time */
        t1.id = tp.id
        AND t1.tvab=tp.tvab
        AND t1.id = t2.id
        /* Just the sceening survey */
        AND tp.survey_no = 5
        /* This makes each pair only appear once */
        AND t1.tvab='1' AND t2.tvab='2'
GROUP BY
        t1.event, t2.event;
```

# 3.8 Utilisation de MySQL avec Apache

Il existe des programmes vous permettant d'identifier vos utilisateurs à l'aide d'une base MySQL et qui vous permettent aussi de créer des journaux de log dans vos table MySQL.

Vous pouvez changer le format d'archivage d'Apache pour le rendre plus facilement lisible par MySQL en mettant ce qui suit dans le fichier de configuration d'Apache :

```
 LogFormat \
        "\"%h\",%&#123;%Y%m%d%H%M%S&#125;t,%>s,\"%b\",\"%&#123;Content-Type&#125;o\",  \
        \"%U\",\"%&#123;Referer&#125;i\",\"%&#123;User-Agent&#125;i\""
```

Avec MySQL, vous pouvez exécuter une requête de la sorte :

```
 LOAD DATA INFILE '/local/access_log' INTO TABLE table_name
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '\\'
```

# 4 Administration du serveur

## 4.1 Configuration de MySQL

### 4.1.1 Options de ligne de commande de mysqld

Dans la plupart des cas, vous devrez modifier les options de mysqld dans le fichier d'options. Fichier d'options `my.cnf` .

`mysqld` et `mysqld.server` lisent les options des groupes `mysqld` et `server` . `mysqld_safe` lit les options des groupes `mysqld` , `server` , `mysqld_safe` et `safe_mysqld` . Un serveur MySQL intégré lit généralement les options dans les groupes `server` , `embedded` et `xxxxx_SERVER` , où `xxxxx` est le nom de l'application.

`mysqld` accepte les options de ligne de commande suivantes :

*−−ansi*
: Utilise la syntaxe ANSI SQL au lieu de la syntaxe MySQL. Exécuter MySQL en mode ANSI .

*−b, −−basedir=path*
: Chemin jusqu'au dossier d'installation. Tous les chemins sont généralement relatifs à celui−ci.

*−−big−tables*
: Autorise la sauvegarde de grands résultats dans des fichiers temporaires. Cela résout le problème des erreurs 'table full', mais ralentit les requêtes alors que des tables en mémoire suffirait. Depuis la version version 3.23.2, MySQL est capable de résoudre automatiquement ce problème en utilisant de la mémoire pour toutes les tables temporaires de petite taille, et en passant sur le disque au besoin.

*−−bind−address=IP*
: L'adresse IP à utiliser.

*−−character−sets−dir=path*
: Dossier contenant les jeux de caractères. Le jeu de caractères utilisé pour les données et le stockage .

*−−chroot=path*
: Met le démon `mysqld` en environnement chroot au démarrage. Recommandé pour la sécurité. Cela limite les commandes `LOAD DATA INFILE` et `SELECT ... INTO OUTFILE` .

*−−core−file*
: Ecrire le fichier core lorsque `mysqld` s'arrête inopinéement. Pour certains fichiers, vous devez aussi spécifier `−−core−file−size` à `safe_mysqld` . `safe_mysqld` , le script père de `mysqld` . Notez que sur certains systèmes, comme Solaris, vous n'aurez pas de fichier de core si vous avez aussi utilisé l'option `−−user` .

*−h, −−datadir=path*
: Chemin jusqu'au dossier des bases.

*−−debug[...]=*
: Si MySQL est configuré avec `−−with−debug` , vous pouvez utiliser cette option pour obtenir un fichier de trace de ce que `mysqld` fait. Créer des fichiers de traçage .

*−−default−character−set=charset*
: Spécifie le jeu de caractères par défaut. Le jeu de caractères utilisé pour les données et le stockage .

*−−default−table−type=type*
: Spécifie le type de table par défaut. Types de tables MySQL .

*−−delay−key−write[= OFF | ON | ALL]*

Comment l'option des tables MyISAM `DELAYED KEYS` doit être utilisé. Choix des paramètres du serveur .

*−−delay−key−write−for−all−tables; En MySQL 4.0.3 vous devez utiliser*
*−−delay−key−write=ALL à la place.*
Ne vide pas les buffers de clés entre deux écritures pour les tables `MyISAM` . Choix des paramètres du serveur .

*−−des−key−file=filename*
Lit les clés par défaut utilisées par `DES_ENCRYPT()` et `DES_DECRYPT()` dans ce fichier.

*−−enable−external−locking (was −−enable−locking)*
Active le verrouillage système. Notez que si vous utilisez cette option sur un système pour qui `lockd` ne fonctionne pas (comme Linux), vous allez bloquer rapidement mysqld avec les verrous.

*−−enable−named−pipe*
Active le support des tunnels nommés (seulement sur NT/Win2000/XP).

*−T, −−exit−info*
Cette option est la combinaison d'options que vous pouvez utiliser pour le débogage du serveur mysqld; Si vous ne savez pas ce que ca fait exactement, ne les utilisez pas !

*−−flush*
Ecrit toutes les données sur le disque après chaque requête SQL. Normalement, MySQL fait des écritures sur le disque après chaque requête, et laisse le système d'exploitation assurer la synchronisation avec le disque. Que faire si MySQL crashe constamment .

*−?, −−help*
Affiche l'aide courte et termine le programme.

*−−init−file=file*
Lit les commandes SQL dans ce fichier au démarrage.

*−L, −−language=...*
Spécifie la langue utilisée pour les messages d'erreur du client. Le chemin complet doit être utilisé. Messages d'erreurs non−anglophones .

*−l, −−log[=file]*
Enregistre les connexions et les requêtes dans ce fichier. Le log général de requêtes .

*−−log−isam[=file]*
Enregistre toutes les modifications des tables ISAM/MyISAM dans ce fichier (uniquement nécessaire pour déboguer ISAM/MyISAM).

*−−log−slow−queries[=file]*
Enregistre toutes les requêtes qui ont pris plus de `long_query_time` secondes a s'exécute, dans ce fichier. Le log des requêtes lentes .

*−−log−update[=file]*
Enregistre les modifications dans le fichier `file.#` où `#` est un nombre unique si il n'est pas précisé. Le log des mises à jour .

*−−log−long−format*
Enregistre des informations supplémentaires dans le fichier d'historique. Si vous utilisez l'option `−−log−slow−queries` , alors les requêtes que vous qui n'utilisent pas les index sont enregistrées dans le log de requêtes longues.

*−−low−priority−updates*
Les opérations de modifications de table ( `INSERT` / `DELETE` / `UPDATE` ) auront une priorité inférieure aux sélections. Cela peut être aussi fait via l'attribut `{INSERT | REPLACE | UPDATE | DELETE} LOW_PRIORITY ...` pour baisser la priorité d'une requête, ou avec `SET LOW_PRIORITY_UPDATES=1` pour changer la priorité dans plus d'un thread. Problèmes avec le verrouillage de tables .

*−−memlock*
Verrouille le processus `mysqld` en mémoire. Cela fonctionne si votre système support la

fonction `mlockall()` (comme Solaris). Ceci peut être utile si vous avez des problèmes avec le système d'exploitation qui force `mysqld` a utiliser le swap sur le disque.

*−−myisam−recover [=option[,option...]]]*

Cette option est la combinaison de `DEFAULT` , `BACKUP` , `FORCE` et `QUICK` . Vous pouvez aussi lui donner la valeur explicite de `""` si vous voulez désactiver cette option. Si cette option est utilisée, `mysqld` va vérifier si la table est marquée comme corrompue à l'ouverture de chaque table (cette dernière option ne fonctionne que si vous utilisez l'option `−−skip−external−locking` ). Si c'est le cas, `mysqld` va essayer de vérifier la table. Si la table était corrompue, `mysqld` essaie alors de la réparer.

L'option suivante va affecter la manière avec la quelle la réparation s'effectue.

| Option | Description |
|---|---|
| DEFAULT | Identique à ne pas donner d'option à −−myisam−recover . |
| BACKUP | Si la table a été modifiée durant la réparation, sauver une copie du fichier table_name.MYD , sous le nom de table_name−datetime.BAK . |
| FORCE | Exécute une réparation même si nous allons perdre une ou plusieurs lignes dans le fichier .MYD. |
| QUICK | Ne vérifie pas les lignes dans la table si il n'y a pas eu d'effacement. |

Avant que la table ne soit automatiquement réparée, MySQL va ajouter une note dans le fichier de log d'erreurs. Si vous voulez être capable de restaurer la plupart des erreurs sans intervention de l'utilisateur, il vaut utiliser les options `BACKUP,FORCE` . Cela va forcer la réparation de la table, même si quelques lignes sont effacées, et conserve le vieux fichier de données comme sauvegarde, pour examen ultérieur.

*−−pid−file=path*

Le chemin jusqu'au fichier de pid utilisé par `safe_mysqld` .

*−P, −−port=...*

Numéro de port utilisé pour attendre les connexion TCP/IP.

*−o, −−old−protocol*

Utilise le protocole 3.20, pour la compatibilité avec de très vieux clients. Upgrading from Version 3.20 to 3.21 .

*−−one−thread*

Utilise uniquement un thread (pour débogage sous Linux). Déboguer un serveur MySQL .

*−O, −−set−variable var=option*

Donne une valeur à une variable. `−−help` liste ces variables. Vous pouvez trouver une description complète des variables dans la section sur la commande `SHOW VARIABLES` de ce manuel. `SHOW VARIABLES` . La section de paramétrage inclut des informations sur comment exploiter ces variables. Notez que `−−set−variable` est obsolète depuis MySQL 4.0, utilisez simplement la syntaxe `−−var=option` . Choix des paramètres du serveur .

En MySQL 4.0.2, il est possible de changer une variable directement avec la syntaxe `−−variable−name=option` et `set−variable` n'est plus nécessaire dans le fichier de configuration.

Si vous voulez restreindre la valeur maximale que peut prendre une option via la commande `SET` , vous pouvez définir une limite en utilisant l'option de ligne de commande `−−maximum−variable−name` . Syntaxe de `SET` .

Notez que lorsque vous donnez une valeur à une variable, MySQL peut corriger automatiquement votre valeur pour rester dans un intervalle donné, et ajuster un peu la valeur pour qu'elle soit optimale.

*−−safe−mode*

Ignore certains étapes d'optimisation.

*−−safe−show−database*

Avec cette option, la commande `SHOW DATABASES` retourne uniquemnt les bases pour lesquelles l'utilisateur a des droits. Depuis la version 4.0.2, cette option est

abandonnée, et ne fait plus rien (l'option est activée par défaut) car nous avons désormais le droit de `SHOW DATABASES` . Syntaxe de `GRANT` et `REVOKE` .

*−−safe−user−create*
: Si cette option est activée, un utilisateur ne peut pas créer de nouveaux utilisateurs avec la commande GRANT si l'utilisateur n'a pas les droits de `INSERT` dans la table `mysql.user` ou dans aucune colonne de cette table.

*−−skip−bdb*
: Désactive l'utilisation des tables BDB. Cela va économiser de la mémoire et accélérer le serveur un peu.

*−−skip−concurrent−insert*
: Désactive la possibilité de sélectionner et insérer en même temps dans les tables `MyISAM` (cela n'est utile que si vous pensez que vous avez trouvé un bug dans cette fonctionnalité).

*−−skip−delay−key−write; En MySQL 4.0.3, il est recommandé d'utiliser l'option* −−delay−key−write=OFF à la place. Ignorez l'option `DELAY_KEY_WRITE` de toutes les tables. Choix des paramètres du serveur .

*−−skip−grant−tables*
: Cette option force le serveur à ne pas utiliser le système de privilège du tout. Cela donne à tous l' **accès complet** à toutes les bases de données ! Vous pouvez demander à un serveur en exécution d'utiliser à nouveau les tables de droits en exécutant la commande `mysqladmin flush−privileges` ou `mysqladmin reload` ).

*−−skip−host−cache*
: Ne pas utiliser le cache de nom de domaine pour une résolution des IP plus rapide, mais interroger le serveur DNS à chaque connexion. Comment MySQL utilise les DNS .

*−−skip−innodb*
: Désactive l'utilisation des tables `InnoDB` . Cela va économiser de la mémoire et accélérer le serveur un peu.

*−−skip−external−locking (ancien −−skip−locking)*
: Ne pas utiliser le verrouillage du système. Pour utiliser les utilitaires `isamchk` ou `myisamchk` vous devez alors éteindre le système. Degré de stabilité de MySQL . Notez qu'en MySQL version 3.23 vous pouvez utiliser la commande `REPAIR` et `CHECK` pour réparer ou vérifier des tables `MyISAM` tables.

*−−skip−name−resolve*
: Les noms d'hôtes ne sont pas résolus. Toutes les colonnes `Host` dans vos tables de droits doivent être des IP numériques ou le mot `localhost` . Comment MySQL utilise les DNS .

*−−skip−networking*
: Ne pas attendre les connexions TCP/IP du tout. Toutes les interactions du serveur `mysqld` seront faites avec les sockets Unix. Cette option est particulièrement recommandée pour les systèmes qui utilisent des requêtes locales. Comment MySQL utilise les DNS .

*−−skip−new*
: Ne pas utiliser les nouvelles routines qui sont possiblement erronées.

*−−skip−symlink*
: Ne pas effacer ou renommer les fichiers qui ont un lien symbolique dans le dossier de données.

*−−skip−safemalloc*
: Si MySQL est configuré avec `−−with−debug=full` , tous les programmes vérifieront la mémoire pour rechercher les écrasment de zone lors des allocations et libérations de mémoire. Comme ce test est lent, vous pouvez l'éviter, si vous n'avez pas besoin de tester la mémoire, en utilisant cette option.

*−−skip−show−database*
: Ne pas autoriser la commande `SHOW DATABASES` , a moins que l'utilisateur n'ait les droits de

`SHOW DATABASES` . Depuis la version 4.0.2, vous n'avez plus besoin de cette option, car les droits pour ce faire sont distribués avec le droit de `SHOW DATABASES` .

*−−skip−stack−trace*
: Ne pas écrire les piles de traces. Cette option est pratique lorsque vous utilisez `mysqld` avec un débogueur. Sur certains systèmes, vous devez aussi utiliser cette option pour obtenir un fichier de core. Déboguer un serveur MySQL .

*−−skip−thread−priority*
: Désactive la priorisation des threads pour améliorer la vitesse de réponse.

*−−socket=path*
: Le fichier de socket à utiliser pour les connexions locales, au lieu du fichier par défaut `/tmp/mysql.sock` .

*−−sql−mode=option[,option[,option...]]*
: Cette option peut être la combinaison de : `REAL_AS_FLOAT` , `PIPES_AS_CONCAT` , `ANSI_QUOTES` , `IGNORE_SPACE` , `SERIALIZE` et `ONLY_FULL_GROUP_BY` . Ellep peut aussi être vide ( `""` ) si vous voulez remettre cette option à 0.En spécifiant toutes les options ci−dessus, vous obtiendrez le même effet qu'avec l'option −−ansi. Avec cette option, vous pouvez activer les modes SQL dont vous avez besoin. Exécuter MySQL en mode ANSI .

*−−temp−pool*
: En utilisant cette option, vous allez réduire le jeu de noms qui sont utilisés lors de la création de fichier temporaires, plutôt qu'un nom unique à chaque fois. Ceci est un palliatif au noyau Linux qui crèe plusieurs fichiers nouveaux avec des noms différents. Avec l'ancien comportement, Linux semble 'perdre de la mémoire', car ils sont alloués au cache d'entrées du dossier au lieu de celui du disque.

*−−transaction−isolation= { READ−UNCOMMITTED | READ−COMMITTED | REPEATABLE−READ | SERIALIZABLE }*
: Configure le niveau d'isolation des transactions. Syntaxe de `SET TRANSACTION` .

*−t, −−tmpdir=path*
: Chemin vers les fichiers temporaires. Il peut s'avérer pratique si votre dossier par défaut `/tmp` réside dans une partition qui est trop petite pour absorber les tables temporaires.

*−u, −−user= [user_name | userid]*
: Exécute le démon `mysqld` avec l'utilisateur `user_name` ou `userid` (numérique). Cette option est **obligatoire** lorsque vous démarrez `mysqld` en tant que root.

*−V, −−version*
: Affiche les informations de version.

*−W, −−log−warnings (Was −−warnings)*
: Enregistre les alertes comme `Aborted connection...` dans le fichier `.err` . Erreurs de communication / connexion annulée .

Il est possible de modifier la plupart des valeurs durant l'exécution, avec la commande `SET` command. Syntaxe de `SET` .

## 4.1.2 Fichier d'options my.cnf

MySQL peut, depuis la version 3.22, lire des options de démarrage par défaut pour le serveur en ligne de commande, et, par le client, dans un fichier.

MySQL lit les options par défaut dans les fichiers suivants sous Unix :

| Fichier | Objet |
|---|---|
| /etc/my.cnf | Options globales |
| DATADIR/my.cnf | Options spécifiques au serveur |
| defaults−extra−file | Le fichier spécifié par −−defaults−extra−file=# |
| ~/.my.cnf | Options spécifiques à l'utilisateur |

`DATADIR` est le dossier de données de MySQL (typiquement `/usr/local/mysql/data` pour les installation binaires ou `/usr/local/var` pour une installation source). Notez que c'est ce dossier qui a été spécifié au moment de la configuration et non pas le dossier de l'option `--datadir` lorsque `mysqld` démarre ! ( `--datadir` n'a aucun effet sur le serveur, car le serveur recherche les données avant de traiter les options de ligne de commande).

MySQL lit les fichiers d'options suivants sous Windows :

| Fichier | Contenu |
|---|---|
| windows-system-directory\my.ini | Options globales |
| C:\my.cnf | Options globales |

Notez que sous Windows, vous devez spécifier les chemins avec `/` plutôt que `\` . Si vous utilisez `\` , vous devez le spécifier deux fois, car `\` est un caractère de protection pour MySQL.

MySQL essaie de lire les fichiers d'options dans l'ordre dans lequel ils sont présentés ci-dessus. Si des options sont spécifiées plusieurs fois, la dernière occurrence utilisée prend la préséeance sur les options spécifiées avant. Les options de ligne de commande ont la priorité sur les options spécifiées dans les fichiers. Certaines options peuvent être spécifiées en utilisant des variables d'environnement. Les options spécifiées en ligne de commande ou en fichier ont la priorité sur les options qui le sont via une variable d'environnement. Variables d'environnement MySQL .

Les programmes suivants utilisent les fichiers d'options : `mysql` , `mysqladmin` , `mysqld` , `mysqld_safe` , `mysql.server` , `mysqldump` , `mysqlimport` , `mysqlshow` , `mysqlcheck` , `myisamchk` et `myisampack` .

Toute option longue qui doit être spécifiée en ligne de commande lorsque MySQL fonctionne, peut être aussi configurée dans le fichier d'options ( sans les doubles tirets). Exécutez le programme avec l'option `--help` pour avoir une liste des options disponibles.

Un fichier d'options contient des lignes ayant la forme suivante :

*#comment*
: Les lignes de commentaires commencent avec `'#'` ou `';'` . Les lignes vides sont ignorées.

*[group]*
: `group` est le nom du programme ou du groupe pour lequel vous souhaitez configurer des options. Après une ligne de groupe, toutes les `option` et `set-variable` s'appliqueront au groupe nommé, jusqu'à la fin du fichier d'option ou du démarrage d'un autre groupe.

*option*
: Ceci est équivalent à `--option` sur la ligne de commande.

*option=value*
: Ceci est équivalent à `--option=value` sur la ligne de commande.

*set-variable = variable=value*
: Ceci est équivalent à `--set-variable variable=value` sur la ligne de commande. Cette syntaxe doit être utilisée pour spécifier la valeur d'une variable `mysqld` . Notez que `--set-variable` est obsolète depuis MySQL 4.0, utilisez simplement `--variable=value` comme tel.

Le groupe `client` vous permet de spécifier des options qui ne s'appliquent qu'aux clients MySQL et non pas au serveur `mysqld` . C'est le groupe idéal pour spécifier des mots de passe de connexion au serveur (mais assurez-vous que vous êtes le seul à accéder à ce fichier !!).

Notez que pour les options et les valeurs, tous les caractères blancs de début et de fins seront automatiquement effacés. Vous pouvez utiliser les séquences d'échappement `'\b'` , `'\t'` , `'\n'` , `'\r'` , `'\\'` et `'\s'` dans votre chaîne à la place ( `'\s'` == espace).

Voici un exemple typique de fichier d'options globales :

```
 [client]
port=3306
socket=/tmp/mysql.sock

 [mysqld]
port=3306
socket=/tmp/mysql.sock
set-variable = key_buffer_size=16M
set-variable = max_allowed_packet=1M

 [mysqldump]
quick
```

Voici un exemple typique de fichier d'options utilisateur :

```
 [client]
# Le mot de passe suivant va être utilisé avec le serveur
password=mon_mot_de_passe

 [mysql]
no-auto-rehash
set-variable = connect_timeout=2

 [mysqlhotcopy]
interactive-timeout
```

Si vous avez une distribution source, vous trouverez des exemples de configuration dans les fichiers nommés `my-xxxx.cnf` dans le dossier `support-files` . Si vous avez une distribution binaire, regardez dans le dossier `DIR/support-files` , où `DIR` est le chemin de l'installation MySQL (typiquement `/usr/local/mysql` ). Actuellement, il y a des exemples de configuration pour des systèmes petits, moyens, grands et très grands. Vous pouvez copier l'un des fichiers `my-xxxx.cnf` dans votre dossier utilisateur (renommez le fichier en `.my.cnf` ) pour le tester.

Tous les clients MySQL qui supportent les fichiers d'options, acceptent les options suivantes :

| Option | Description |
|---|---|
| `--no-defaults` | Ne lire aucun fichier d'options. |
| `--print-defaults` | Affiche le nom du programme et toutes les options qui s'y trouvent. |
| `--defaults-file=full-path-to-default-file` | Utilise uniquement le fichier de configuraiton donné. |
| `--defaults-extra-file=full-path-to-default-file` | Lit ce fichier de configuration après le fichier de configuration global, mais avant le fichier de configuration utilisateur. |

Notez que les options ci-dessus doivent être en ligne de commande pour être utilisées ! `--print-defaults` peut quand même être utilisé directement après la commande `--defaults-xxx-file` .

Note pour les développeurs : la gestion des fichiers d'options est implémentée simplement en traitant toutes les options qui correspondent (c'est à dire, toutes les options appropriées du groupe), avant les arguments de ligne de commande. Cela fonctionne bien pour les programmes qui utilisent la dernière occurrence comme valeur d'option, si elle est spécifiée plusieurs fois. Si vous avez un vieux programme qui traite les options multiples de cette façon mais ne lit pas les fichiers d'options, vous n'avez besoin que de deux lignes pour qu'il accepte cette fonctionnalité. Récupérez le code source de n'importe quel client MySQL standard pour voir comment le faire.

En scripts shell, vous pouvez utiliser la commande `my_print_defaults` pour analyser les fichiers de configuration :

```
 shell> my_print_defaults client mysql
--port=3306
--socket=/tmp/mysql.sock
--no-auto-rehash
```

La ligne ci-dessus affiche toutes les options pour les groupes 'client' et 'mysql'.

## 4.1.3 Installer plusieurs serveurs sur la même machine

Dans certains cas, vous aurez besoin de plusieurs démons `mysqld` sur la même machine. Vous pouvez, par exemple, faire tourner une vieille version de MySQL pour la tester avec une nouvelle.

Vous pouvez aussi donner des accès différents à des utilisateurs de différents serveurs `mysqld` , qu'il gèrent eux−mêmes.

Une méthode pour avoir plusieurs serveurs différents sur la même machine est de le configurer avec différents sockets et ports comme suit :

```
 shell> MYSQL_UNIX_PORT=/tmp/mysqld-new.sock
shell> MYSQL_TCP_PORT=3307
shell> export MYSQL_UNIX_PORT MYSQL_TCP_PORT
shell> scripts/mysql_install_db
shell> bin/safe_mysqld &
```

L'annexe sur les variables d'environnement inclut une liste des variables d'environnement que vous pouvez utiliser pour paramétrer `mysqld` . Variables d'environnement MySQL .La méthode ci−dessus est immédiate et peu propre pour ceux qui font des tests. Ce qui est bien avec cette méthode, c'est que les connexions que vous réalisez avec le shell ci−dessus seront automatiquement redirigées vers le serveur en fonctionnement.

Si vous avez besoin d'une méthode plus permanente, il est recommandé de créer un fichier d'options pour chaque serveur. Fichier d'options `my.cnf` . Dans votre script de démarrage du serveur, vous pourriez spécifier tous les serveurs :

```
safe_mysqld --defaults-file=path-to-option-file
```

Enfin, les options suivantes doivent être différentes pour chaque serveur :

- port=#
- socket=path
- pid−file=path

Les options suivantes doivent être différentes, si elles sont utilisées :

- log=path
- log−bin=path
- log−update=path
- log−isam=path
- bdb−logdir=path

Si vous voulez plus de performances, vous pouvez aussi configurer différemment les options suivantes :

- tmpdir=path
- bdb−tmpdir=path

Options en ligne de commande de `mysqld` .

Si vous installez une version binaire de MySQL (fichiers .tar) et que vous les démarrez avec `./bin/safe_mysqld` , alors dans la plupart des cas, la seule option que vous devez modifier est la socket `socket` et le port `port` dans le script `safe_mysqld` .

Faire fonctionner plusieurs serveurs MySQL sur la même machine .

## 4.1.4 Faire fonctionner plusieurs serveurs MySQL sur la même machine

Il y a des situations où vous souhaiterez avoir plusieurs serveurs MySQL sur la même machine. Par exemple, si vous voulez tester une nouvelle version du serveur avec votre configuration de production sans perturber votre installation de production. Ou bien, vous êtes un fournisseur de

services Internet, et vous voulez fournir des installations distinctes pour des clients différents.

Si vous voulez exécuter plusieurs serveurs MySQL, le plus simple et de compiler les serveurs avec différents ports TCP/IP et fichiers de sockets pour qu'ils ne soient pas tous à l'écoute du même port ou de la même socket. `mysqld_multi` , un programme pour gérer plusieurs serveurs MySQL .

Supposons que votre serveur est configuré avec le numéro de port par défaut, et le numéro de socket par défaut. Alors vous pouvez configurer le nouveau serveur avec la commande `configure` suivante :

```
shell> ./configure  --with-tcp-port=numero_port \
            --with-unix-socket-path=nom_fichier \
            --prefix=/usr/local/mysql-3.22.9
```

Ici, les options `numero_port` et `nom_fichier` doivent être différentes des valeurs par défaut, et avec l'option `--prefix` vous allez spécifier un autre dossier d'installation que celui qui est déjà utilisé par la première installation.

Vous pouvez vérifier le nom de la socket qui est utilisée par un serveur MySQL avec cette commande :

```
shell> mysqladmin -h hostname --port=numero_port variables
```

Notez que si vous spécifiez `` `localhost` '' comme nom d'hôte, `mysqladmin` va utiliser les sockets Unix plutôt que TCP/IP.

Si vous avez un serveur MySQL qui utilise déjà le port, vous obtiendrez une liste des variables de configuration les plus importantes de MySQL, y compris le nom de la socket.

Vous n'avez pas à recompiler un nouveau serveur MySQL juste pour le démarrer sous un autre port et une autre socket. Vous pouvez changer le port et la socket utilisée au moment du démarrage, avec `safe_mysqld` :

```
shell> /path/to/safe_mysqld --socket=nom_fichier --port=numero_port
```

`mysqld_multi` peut aussi prendre `safe_mysqld` (ou `mysqld` ) comme argument, et passer les options depuis un fichier de configuration à `safe_mysqld` , et ainsi, à `mysqld` .Si vous exécutez un nouveau serveur sur les mêmes données qu'un autre serveur, avec le log activé, vous devez spécifier le nom du fichier de log à `safe_mysqld` avec l'option `--log` , `--log-update` , ou `--log-slow-queries` . Sinon, les deux serveurs risquent d'écrire dans le même fichier de log.

**Attention** : normalement, vous ne devez pas avoir deux serveurs qui modifient en même temps les données dans les mêmes bases. Si votre OS ne supporte pas le verrouillage sans échec, cela peut vous mener à de déplaisantes surprises !

Si vous voulez utiliser un autre dossier de fichiers pour le second serveur, vous pouvez utiliser l'option `--datadir=path` de `safe_mysqld` .

**Notez** aussi que démarrer plusieurs serveurs MySQL ( `mysqlds` ) dans différentes machines, et les laisser accéder aux même données via `NFS` est généralement une **mauvaise idée** ! Le problème est que `NFS` va devenir un frein au niveau de la vitesse. Il n'est pas destiné à une telle utilisation. Et de plus, vous devrez trouver une solution pour vous assurer que deux des `mysqld` n'interfèrent pas entre eux. Actuellement, il n'y a pas de plate-forme qui soit 100% fiable pour le verrouillage de fichiers (le démon `lockd` ), dans toutes les situations. Pourtant, il va y avoir un risque supplémentaire avec `NFS` ; cela rendra le travail encore plus compliqué pour le démon `lockd` . Alors, simplifiez-vous la vie, et oubliez cela. La solution efficace est d'avoir un ordinateur avec un système d'exploitation, qui gère efficacement les threads et a plusieurs processeurs.

Lorsque vous voulez vous connecter au serveur MySQL qui fonctionne avec un autre port que le port qui est compilé par défaut dans votre client, vous pouvez utiliser l'une des méthodes suivantes :

- Lancez le client avec l'option `--host 'hostname' --port=numero_port` pour vous connecter en TCP/IP, ou `[--host localhost] --socket=nom_fichier` pour vous connecter via une socket Unix.
- Dans vos programmes C ou Perl, vous pouvez indiquer le port ou la socket lors de la connexion au serveur MySQL.
- Si vous utilisez le code Perl avec le module `DBD::mysql` , vous pouvez lire les options des fichiers d'options MySQL. Fichier d'options `my.cnf` .

```
 $dsn = "DBI:mysql:test;mysql_read_default_group=client;
        mysql_read_default_file=/usr/local/mysql/data/my.cnf"
$dbh = DBI->connect($dsn, $user, $password);
```

- Modifiez les variables d'environnement `MYSQL_UNIX_PORT` et `MYSQL_TCP_PORT` pour qu'elles pointent sur la socket Unix et le port TCP/IP voulu avant de démarrer vos clients. Si vous utilisez une socket ou un port spécifique, il est recommandé de modifier ces variables dans votre fichier `.login` . Variables d'environnement MySQL .
- Specifiez la socket et le port TCP/IP par défaut dans le fichier `my.cnf` de votre dossier personnel. Fichier d'options `my.cnf` .

# 4.2 Règles de sécurité et droits d'accès au serveur MySQL

MySQL est pourvu d'un système avancé mais non standard de droits. Cette section décrit son fonctionnement.

## 4.2.1 Instructions générales de sécurité

Tous ceux qui utilisent MySQL avec un serveur connecté à l'internet doivent lire cette section, pour éviter les erreurs de sécurité les plus courantes.

En parlant de sécurité, nous mettons l'accent sur la nécessité de protéger la totalité du serveur (et non pas seulement le serveur MySQL), contre tous les types d'attaques possibles : cheval de troye, virus, dénis de services, écoute électronique. Nous ne couvrirons pas la totalité des aspects et des pannes ici.

MySQL utilise un système de sécurité basé sur les listes de contrôle d'accès (Access Control Lists; ACLs) pour toutes les connexions, requêtes et autres opérations qu'un utilisateur peut tenter d'exécuter. Il y a aussi le support des connexions chiffrées par mode SSL, entre le client et le serveur. De nombreux concepts sont présentés ici, et ne sont pas spécifiques à MySQL. Les mêmes concepts s'appliquent à de nombreuses applications.

Lorsque vous faîtes fonctionner MySQL, suivez ces règles le plus souvent possible :

- **Ne jamais donner à quiconque (hormis l'utilisateur mysql root) l'accès à la table `user` dans la base `mysql` !** Cette table est primordiale. **Le mot de passe chiffré stocké dans la table est le véritable mot de passe MySQL** . Quiconque connaît le mot de passe stocké dans la table `user` et a accès à la liste des hôtes d'un compte **peut facilement se connecter avec cet utilisateur** .
- Connaissez à fond le système de droits de MySQL. Les commandes `GRANT` et `REVOKE` sont utilisées pour contrôler les accès à MySQL. Ne donnez pas plus de droits que nécessaire. Ne donnez jamais des droits à tous les hôtes. Liste de points à vérifier :
    - Essayez `mysql -u root` . Si vous êtes capable de vous connecter avec succès au serveur sans qu'un mot de passe n'ai été demandé, vous avez un problème. Toute personne

peut se connecter au serveur MySQL en tant qu'utilisateur `root` avec tous les droits ! Passez en revue votre installation MySQL, en faisant attention aux mots de passe `root` .

- Utilisez la commande `SHOW GRANTS` et vérifiez qui a accès à quoi. Supprimez les droits qui ne sont pas nécessaires avec la commande `REVOKE` .

- Ne gardez par de mot de passe en clair dans vos tables. Lorsque votre ordinateur est infiltré, l'intrus pourra alors obtenir la liste complète des mots de passe, et les utiliser. Au lieu de cela, utilisez les fonctions `MD5()` , `SHA1()` ou une autre fonction de hash.
- Ne choisissez pas vos mots de passe dans un dictionnaire. Il y a des programmes spéciaux qui sont capable de les casser. Même des mots de passe comme ``xfish98'' sont très faibles. Bien meilleur est ``duag98'' qui contient aussi le mot ``fish'' mais tapé avec une touche à gauche sur le clavier standard QWERTY. Une autre méthode est d'utiliser ``Pfspeb'' qui est un mot créé à partir des premières lettres de la phrase ``Papa fume sa pipe en bois''. C'est un mot de passe facile à retenir, difficile à deviner pour une personne qui ne le connaît pas.
- Investissez dans un pare−feu. Cela vous protège de presque 50 % de tous les types de trous de sécurité des logiciels. Placez MySQL derrière un pare−feu, ou dans une zone démilitarisée (DMZ).Liste de points à vérifier :
  - Essayez de scanner vos ports depuis Internet, en utilisant des outils comme `nmap` . MySQL utilise le port 3306 par défaut. Ce port doit être inaccessible depuis des hôtes indus. Un autre moyen simple de tester si votre port MySQL est accessible ou pas, est d'essayer la commande suivante depuis une machine distante, où `server_host` est l'hôte de votre serveur MySQL :

    ```
    shell> telnet server_host 3306
    ```

    Si vous obtenez une connexion et des caractères étranges, c'est que le port est ouvert, et qu'il devrait être fermé par votre routeur ou votre pare−feu, à moins que vous n'ayez une bonne raison de le garder ouvert. Si `telnet` attend ou que la connexion est refusée, c'est que tout est bon : le port est bloqué.
- Ne faites jamais aucune confiance aux données des utilisateurs. Ils peuvent essayer de nombreux trucs pour vous envoyer du code spécial ou faire passer des séquences de caractères protégés via des formulaires web, ou n'importe quel outil que votre application utilise. Soyez certain que votre application reste sécurisée même si un utilisateur obtient du code comme `` ; DROP DATABASE mysql; ''. Ceci est un exemple extrême, mais de grosse fuites de données ou des pertes peuvent survenir si un pirate utilise des techniques similaires, et que vous n'êtes pas préparé pour.

  Vérifiez aussi toutes les données numériques. Une erreur commune est de protéger uniquement les chaînes. Certains pensent que si une base contient des données publiques, elle ne doit pas être protégée. C'est totalement faux. Au minimum, un attaque en dénis de service peut être pratiquée sur ces bases. Le plus simple moyen pour vous protéger contre ce type d'attaque est d'utiliser des apostrophes autour des constantes numériques : `SELECT * FROM table WHERE ID='234'` plutôt que `SELECT * FROM table WHERE ID=234` . MySQL convertit automatiquement la chaîne en nombre et supprime tous les caractères non numériques avant insertion.

  Point à vérifier :

  - Toutes les applications web :
    - Essayez d'entrer des guillemets `'''` et `'"'` dans tous vos formulaires web. Si vous obtenez une erreur MySQL, alors étudiez le problème immédiatement.
    - Essayez de modifier toute URL dynamique en ajoutant les codes `%22` ( `'"'` ), `%23` ( `'#'` ) et `%27` ( `'''` ) dedans.
    - Essayez de modifier les types de données des URL dynamiques, de numérique en chaîne, grâce aux exemples ci−dessus. Votre application doit être sécurisée contre ce type d'attaques.

◊ Essayez d'entrer des caractères comme des espaces ou des symboles spéciaux au lieu des nombres, dans les champs numériques. Votre application doit les supprimer avant de le passer à MySQL, ou votre application doit générer une erreur. Passer des valeurs non vérifiées à MySQL est très dangereux.
◊ Vérifiez la taille des données avant de les passer à MySQL.
◊ Donnez un autre nom d'utilisateur et un autre mot de passe à votre application que votre nom d'utilisateur d'administration. Ne donnez pas à votre application plus de droits que nécessaire.

♦ Utilisateurs de PHP :
◊ Etudiez la fonction `addslashes()` . Depuis PHP 4.0.3, la fonction `mysql_escape_string()` est disponible avec la même fonctionnalité que son alter ego en API MySQL C.

♦ Utilisateurs de l'API MySQL C :
◊ Vérifiez les appels à `mysql_real_escape_string()` .

♦ Utilisateurs de MySQL++ :
◊ Vérifiez les options `escape` et `quote` des requêtes.

♦ Utilisateurs Perl DBI :
◊ Vérifiez la méthode `quote()` ou utilisez des variables liées.

♦ Utilisateurs de Java JDBC :
◊ Utilisez l'objet `PreparedStatement` et les variables liées.

- Ne transmettez aucunes données non chiffrées via Internet. Ces données peuvent être accessibles à quiconque a le temps et la capacité d'intercepter des données et de les utiliser pour son propre usage. Au lieu de cela, utilisez un protocole chiffré tel que SSL ou SSH. MySQL supporte les connexions SSL en interne, depuis la version 4.0.0. Le forward de port de SSH peut être utilisé pour créer un tunnel chiffré et compressé pour les communications.
- Apprenez à utiliser les utilitaires `tcpdump` et `strings` . Pour la plupart des cas, vous pouvez vérifier si les flux de données MySQL sont chiffrés en utilisant une commande telle que celle ci :

```
shell> tcpdump -l -i eth0 -w - src or dst port 3306 | strings
```

(Cela fonctionne sous Linux et devrait aussi fonctionner sous les autres systèmes avec de légères adaptations). Attention : si vous ne voyez pas de données, cela ne signifie pas que vous avez des données chiffrées. Si vous avez besoin de forte sécurité, vous devriez consulter un expert de la sécurité.

## 4.2.2 Comment protéger MySQL contre les pirates

Lorsque vous vous connectez à MySQL, vous devriez avoir besoin d'un mot de passe. Ce mot de passe n'est pas transmis en texte clair sur le réseau, mais comme l'algorithme de chiffrement n'est pas très fort, quelques efforts permettront à un pirate d'obtenir votre mot de passe s'il est capable de surveiller le trafic entre le client et le serveur. Si la connexion entre le client et le serveur utilise des réseaux non fiables, il est alors recommandé d'utiliser un tunnel SSH.

Toutes les autres informations sont transférées comme du texte clair, et quiconque surveille la connexion pourra les lire. Si vous souhaitez relever ce niveau de sécurité, il est recommandé d'utiliser le protocole compressé (avec les versions de MySQL 3.22 et plus récentes), pour compliquer considérablement le problème. Pour rendre la communication encore plus sûre, vous pouvez aussi utiliser `ssh` . Vous trouverez une version `Open Source` du client `ssh` sur le site http://www.openssh.org/ , et une version commerciale du client `ssh` sur le site de http://www.ssh.com/ . Avec eux, vous pouvez mettre en place une connexion TCP/IP chiffrée entre le serveur et le client MySQL.

Si vous utilisez MySQL 4.0, vous pouvez aussi utiliser le support OpenSSL interne. Utilisation des connexions sécurisées .

Pour rendre le système MySQL encore plus sûr, nous vous recommandons de suivre les suggestions suivantes :

- Utilisez des mots de passe pour tous les utilisateurs MySQL. N'oubliez pas que tout le monde peut se connecter avec un nom d'utilisateur quelconque, simplement avec l'option `mysql -u autre_utilisateur nom_de_base` , si `autre_utilisateur` n'a pas de mot de passe. C'est un comportement classique pour les applications client/serveur que le client spécifie son nom d'utilisateur. Vous pouvez modifier les mots de passe de tous les utilisateurs en modifiant le script `mysql_install_db` avant de l'exécuter, ou vous pouvez modifier seulement le mot de passe du `root` MySQL comme ceci :
```
 shell> mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('nouveau_mot_de_passe')
    ->             WHERE user='root';
mysql> FLUSH PRIVILEGES;
```
- N'exécutez jamais le démon MySQL avec l'utilisateur Unix `root` . C'est très dangereux, car tout personne ayant le droit de `FILE` pour créer des fichiers au nom du `root` (par exemple, `~root/.bashrc` ). Pour éviter cela, `mysqld` refusera de s'exécuter au nom de `root` à moins que soit précisé l'option `--user=root` . `mysqld` peut être exécuté avec un utilisateur ordinaire sans droits particuliers. Vous pouvez aussi créer un utilisateur Unix `mysql` pour rendre cela encore plus sûr. Si vous exécutez `mysqld` sous un autre utilisateur Unix, vous n'avez pas à changer le mot de passe `root` dans la table `user` , car les noms d'utilisateurs MySQL n'ont rien à voir avec les noms d'utilisateurs Unix. Pour démarrer `mysqld` sous un autre nom d'utilisateur Unix, ajoutez la ligne `user` , qui spécifie le nom de l'utilisateur, dans le fichier d'options de `[mysqld] /etc/my.cnf` ou dans le fichier `my.cnf` présent dans le dossier de données du serveur. Par exemple :
```
 [mysqld]
user=mysql
```
  Cette ligne va forcer le serveur à démarrer en tant qu'utilisateur `mysql` , même si vous démarrez le serveur manuellement ou avec les scripts `safe_mysqld` , ou `mysql.server` . Pour plus de détails, voyez Comment faire tourner MySQL en tant qu'utilisateur normal .
- N'acceptez pas les liens symboliques pour les tables (cette fonctionnalité peut être désactivée avec l'option `--skip-symlink` option). Il est généralement important si vous exécutez `mysqld` en tant que root, car tout le monde aurait accès en écriture au dossier de données de MySQL, et pourrait alors effacer des fichiers systèmes. Utiliser les liens symboliques pour les tables .
- Vérifiez que l'utilisateur Unix qui exécute `mysqld` est le seul utilisateur avec les droits de lecture et écriture dans le dossier de base de données.
- Ne donnez pas le droit de `PROCESS` à tous les utilisateurs. La liste fournie par `mysqladmin processlist` affiche le texte des requêtes actuellement exécutées, ce qui permet à toute personne pouvant exécuter cette commande de lire des valeurs qui seraient en clair, comme : `UPDATE user SET password=PASSWORD('not_secure')` .

  `mysqld` réserve une connexion supplémentaire pour les utilisateurs qui ont le droit de `PROCESS` , afin que le `root` MySQL puisse toujours se connecter et vérifier que tout fonctionne bien, même s'il ne reste plus de connexions libres pour les autres utilisateurs.
- Ne donnez pas le droit de `FILE` à tous les utilisateurs. Tout utilisateur qui possède ce droit peut écrire un fichier n'importe où sur le serveur, avec les droits hérités du démon `mysqld` ! Pour rendre cela plus sécuritaire, tous les fichiers générés par `SELECT ... INTO OUTFILE` sont lisibles par tous, mais personne ne peut les modifier.

  Le droit de `FILE` peut aussi être utilisé pour lire n'importe quel fichier accessible en lecture au démon qui fait tourner MySQL. Il devient donc possible, suivant les configurations, d'utiliser la commande `LOAD DATA` sur le fichier `/etc/passwd` pour tout mettre en table, et ensuite le relire avec la commande `SELECT` .
- Si vous ne faites pas confiance à votre DNS, vos pouvez simplement utiliser des adresses IP au lieu des noms d'hôtes. Dans ce cas, soyez très prudents lors de la création de droits qui utilisent des caractères joker.

- Si vous voulez restreindre le nombre de connexions d'un utilisateur, vous pouvez le faire en utilisant la variable `max_user_connections` de `mysqld` .

## 4.2.3 Options de démarrage qui concernent la sécurité

Les options suivantes de `mysqld` affectent la sécurité :

*−−local−infile[=(0|1)]*
: If one uses `−−local−infile=0` then one can't use `LOAD DATA LOCAL INFILE` .

*−−safe−show−database*
: Avec cette option, la commande `SHOW DATABASES` ne retourne que les bases pour lesquelles l'utilisateur courant a des droits. Depuis la verison 4.0.2, cette option est abandonnée, et ne sert plus à rien (elle est activée par défaut), car désormais, il y a le droit de `SHOW DATABASES` . Syntaxe de `GRANT` et `REVOKE` .

*−−safe−user−create*
: Si cette option est activée, tout utilisateur ne peut créer d'autres utilisateurs avec les droits de `GRANT` , s'il ne dispose pas des droits d'insertion dans la table `mysql.user` . Si vous voulez donner un accès à un utilisateur pour qu'il puisse créer des utilisateurs avec les droits dont il dispose, vous pouvez lui donner les droits suivants :

```
mysql> GRANT INSERT(user) ON mysql.user TO 'user'&#64;'hostname';
```

: Cela va s'assurer que l'utilisateur ne peut par modifier une colonne directement, mais qu'il peut exécuter la commande `GRANT` sur d'autres utilisateurs.

*−−skip−grant−tables*
: Cette option force le serveur à ne pas utiliser les tables de droits. Cette option donne donc tous les droits à tout le monde sur le serveur ! Vous pouvez forcer un serveur en fonctionnement à reprendre les tables de droits en exécutant la commande `mysqladmin flush−privileges` ou `mysqladmin reload` .)

*−−skip−name−resolve*
: Les noms d'hôtes ne sont pas résolus. Toutes les valeurs de la colonne `Host` dans les tables de droits doivent être des adresses IP, ou bien `localhost` .

*−−skip−networking*
: Ne pas accepter les connexions TCP/IP venant du réseau. Toutes les connexions au serveur `mysqld` doivent être faîtes avec les sockets Unix. Cette option n'existe pas pour les versions antérieures à la 3.23.27, avec les MIT−pthread, car les sockets Unix n'étaient pas supportés par les MIT−pthreads à cette époque.

*−−skip−show−database*
: Ne pas autoriser la commande `SHOW DATABASES` , à moins que l'utilisateur n'ait les droits de `SHOW DATABASES` . Depuis la version 4.0.2, vous n'avez plus besoin de cette option, car les accès sont désormais donnés spécifiquement avec le droit `SHOW DATABASES` .

## 4.2.4 Problèmes de sécurité avec LOAD DATA LOCAL

Depuis MySQL 3.23.49 et MySQL 4.0.2, nous avons ajouté de nouvelles options pour traiter les problèmes de sécurité liés à `LOAD DATA LOCAL` .

Il existe deux problèmes particuliers pour le support de cette commande :

Comme la lecture du fichier est réalisée depuis le serveur, il est possible théoriquement de créer un serveur MySQL patché qui pourrait lire n'importe quel fichier sur la machine cliente, qui serait accessible à l'utilisateur.

Dans un environnement web, où les clients se connectent depuis un serveur web, un utilisateur peut se servir de la commande `LOAD DATA LOCAL` pour lire les fichiers qui sont sur le serveur web, et auquel ce dernier a accès (en supposant qu'un utilisateur peut exécuter n'importe quelle commande sur le serveur).

Il y a deux protections distinctes pour ces problèmes :

Si vous ne configurez pas MySQL avec l'option `--enable-local-infile` , alors `LOAD DATA LOCAL` sera désactivé par tous les clients, à moins que l'option `mysql_options(... MYSQL_OPT_LOCAL_INFILE, 0)` soit activée dans le client. `mysql_options()` .

Pour le client en ligne de commande `mysql` , `LOAD DATA LOCAL` peut être activé en spécifiant l'option `--local-infile[=1]` , ou désactivé avec `--local-infile=0` .

Par défaut, tous les clients MySQL et les librairies sont compilés avec `--enable-local-infile` , pour être compatible avec MySQL 3.23.48 plus ancien.

Vous pouvez désactiver toutes les commandes `LOAD DATA LOCAL` du serveur MySQL en démarrant `mysqld` avec `--local-infile=0` .

Au cas où `LOAD DATA LOCAL INFILE` est désactivé sur le serveur ou le client, vous obtiendrez le message d'erreur (1148) :

```
The used command is not allowed with this MySQL version
```

## 4.2.5 Rôle du système de privilèges

La fonction première du système de privilèges de MySQL est d'authentifier les utilisateurs se connectant à partir d'un hôte donné, et de leur associer des privilèges sur une base de données comme `SELECT` , `INSERT` , `UPDATE` et `DELETE` .

Les fonctionnalités additionnelles permettent d'avoir un utilisateur anonyme et de contrôler les privilèges pour les fonctions spécifiques à MySQL comme `LOAD DATA INFILE` et les opérations administratives.

## 4.2.6 Comment fonctionne le système de droits

Le système de droits de MySQL s'assure que les utilisateurs font exactement ce qu'ils sont supposés pouvoir faire dans la base. Lorsque vous vous connectez au serveur, vous identité est déterminée par **l'hôte d'où vous vous connectez** et **le nom d'utilisateur que vous spécifiez** . Le système donne les droits en fonction de votre identité et de **ce que vous voulez faire** .

MySQL considère votre nom d'hôte et d'utilisateur pour vous identifier, car il n'y pas que peu de raisons de supposer que le même nom d'utilisateur appartient à la même personne, quelque soit son point de connexion sur Internet. Par exemple, l'utilisateur `joe` qui se connecte depuis `office.com` n'est pas forcément la même personne que `joe` qui se connecte depuis `elsewhere.com` . MySQL gère cela en vous aidant à distinguer les différents utilisateurs et hôtes qui ont le même nom : vous pourriez donner des droits à `joe` lorsqu'il utilise sa connexion depuis `office.com` , et un autre jeu de droits lorsqu'il se connecte depuis `elsewhere.com` .

Le contrôle d'accès de MySQL se fait en deux étapes :

- Etape 1 : Le serveur vérifie que vous êtes autorisé à vous connecter.

- Etape 2 : En supposant que vous pouvez vous connecter, le serveur vérifie chaque requête que vous soumettez, pour vérifier si vous avez les droits suffisants pour l'exécuter. Par exemple, si vous sélectionnez des droits dans une table, ou effacez une table, le serveur s'assure que vous avez les droits de `SELECT` pour cette table, ou les droits de `DROP` , respectivement.

Le serveur utilise les tables `user` , `db` et `host` dans la base `mysql` durant les deux étapes. Les champs de cette table sont les suivants :

| **Nom de la table** | user | db | host |
|---|---|---|---|
| **Identifiant** | Host | Host | Host |
| | User | Db | Db |
| | Password | User | |
| **Champs de droits** | Select_priv | Select_priv | Select_priv |
| | Insert_priv | Insert_priv | Insert_priv |
| | Update_priv | Update_priv | Update_priv |
| | Delete_priv | Delete_priv | Delete_priv |
| | Index_priv | Index_priv | Index_priv |
| | Alter_priv | Alter_priv | Alter_priv |
| | Create_priv | Create_priv | Create_priv |
| | Drop_priv | Drop_priv | Drop_priv |
| | Grant_priv | Grant_priv | Grant_priv |
| | References_priv | | |
| | Reload_priv | | |
| | Shutdown_priv | | |
| | Process_priv | | |
| | File_priv | | |

Lors de la seconde étape du contrôle d'accès (vérification de la requête), le serveur peut, suivant la requête, consulter aussi les tables `tables_priv` et `columns_priv` . Les champs de ces tables sont :

| **Nom de la table** | tables_priv | columns_priv |
|---|---|---|
| **Identifiant** | Host | Host |
| | Db | Db |
| | User | User |
| | Table_name | Table_name |
| | | Column_name |
| **Champs de droits** | Table_priv | Column_priv |
| | Column_priv | |
| **Autre champs** | Timestamp | Timestamp |
| | Grantor | |

Chaque table de droit contient des champs d'identification et des champs de droits.

Les champs d'identification déterminent quels utilisateurs correspondent à cette ligne dans la table. Par exemple, une ligne dans la table `user` avec les valeurs dans les colonnes `Host` et `User` de `'thomas.loc.gov'` et `'bob'` servira à identifier les connexions qui sont faites par l'utilisateur `bob` depuis l'hôte `thomas.loc.gov` . De même, une ligne dans la table `db` avec les valeurs des colonnes `Host` , `User` et `Db` de `'thomas.loc.gov'` , `'bob'` et `'reports'` sera utilisée lorsque l'utilisateur `bob` se connecte depuis l'hôte `thomas.loc.gov` pour accéder à la base `reports` . Les tables `tables_priv` et `columns_priv` contiennent en plus des champs indiquant les tables et combinaisons tables et colonnes auxquelles les lignes s'appliquent.

Pour les contrôles d'accès, les comparaisons de nom d'hôte avec la colonne `Host` sont insensibles à la casse. Les colonnes `User` , `Password` , `Db` et `Table_name` sont sensibles à la casse. Les valeurs de la colonne `Column_name` sont insensibles à la casse pour les versions de MySQL 3.22.12 et plus récent.

Les champs de droits indiquent si le droit est donné, c'est à dire si l'opération indiquée peut être exécuté. Le serveur combine les informations dans différentes tables pour former une description complète de l'utilisateur. Les règles utilisées sont décrites dans Contrôle d'accès, étape 2 : vérification des requêtes .

Les champs d'identification sont des chaînes, déclarées comme suit. La valeur par défaut de chacun des champs est la chaîne vide.

| Nom du champs | Type | Notes |
|---|---|---|
| Host | CHAR(60) | |
| User | CHAR(16) | |
| Password | CHAR(16) | |
| Db | CHAR(64) | ( CHAR(60) pour les tables tables_priv et columns_priv ) |
| Table_name | CHAR(60) | |
| Column_name | CHAR(60) | |

Dans les tables `user` , `db` et `host` , tous les champs de droits sont déclarés avec le type `ENUM('N','Y')` : il peuvent prendre tous les valeurs de `'N'` (non) ou `'Y'` (oui, YES), et la valeur par défaut est `'N'` .

Dans les tables `tables_priv` et `columns_priv` , les champs de droits sont déclarés comme des champs de type `SET` :

| Nom de la table | Nom du champs | Valeurs possibles |
|---|---|---|
| tables_priv | Table_priv | 'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter' |
| tables_priv | Column_priv | 'Select', 'Insert', 'Update', 'References' |
| columns_priv | Column_priv | 'Select', 'Insert', 'Update', 'References' |

En bref, le serveur utilise les tables de droits comme ceci :

- La table `user` détermine si le serveur accepte ou rejette la connexion. Pour les connexions acceptées, tous les privilèges donnés dans la table `user` indiquent des privilèges globaux. Ces droits d'appliquent à **toutes** les bases du serveur.
- Les tables `db` et `host` sont utilisées conjointement :
  - Les champs d'identification de la table `db` déterminent quels utilisateurs peuvent accéder à quelles bases, depuis quel hôte. Les champs de droits indiquent alors les opérations permises.
  - La table `host` est utilisée comme extension de la table `db` lorsque vous voulez qu'une ligne de la table `db` s'applique à plusieurs hôtes. Par exemple, si vous voulez qu'un utilisateur soit capable d'utiliser une base depuis plusieurs hôtes dans votre réseau, laissez la colonne `Host` vide dans la table `db` , Ce mécanisme est décrit en détails dans Contrôle d'accès, étape 2 : vérification des requêtes .
- Les tables `tables_priv` et `columns_priv` sont similaires à la table `db` , mais sont plus atomiques : elle s'appliquent au niveau des tables et des colonnes, plutôt qu'au niveau des bases.

Notez que les droits d'administration tels que ( `RELOAD` , `SHUTDOWN` , etc...) ne sont spécifiés que dans la table `user` . En effet, ces opérations sont des opérations au niveau serveur, et ne sont pas liées à une base de données, ce qui fait qu'il n'y a pas de raison de les lier avec les autres tables. En fait `user` doit être consulté pour déterminer les autorisations d'administration.Le droit de `FILE` est spécifié par la table `user` . Ce n'est pas un droit d'administration, mais votre capacité à lire ou écrire des fichiers sur le serveur hôte et dépendant de la base à laquelle vous accédez.

Le serveur `mysqld` lit le contenu des tables de droits une fois, au démarrage. Lorsqu'il y a des modifications dans les tables, elles prennent effet tel qu'indiqué dans Quand les changements de privilèges prennent−ils effet ? .

Lorsque vous modifiez le contenu des tables de droits, c'est une bonne idée que de s'assurer que vous avez bien configuré les droits qui vous intéressent. Pour vous aider dans votre diagnostique, voyez Causes possibles de l'erreur `Access denied` . Pour des conseils sur les aspects sécurité, voyez Comment protéger MySQL contre les pirates .

Un outil de diagnostique pratique est le script `mysqlaccess` , que Yves Carlier a fourni à la distribution MySQL. Appelez `mysqlaccess` avec l'option the `--help` pour comprendre comment il fonctionne. Notez que `mysqlaccess` ne vérifie les accès que pour les tables `user` , `db` et `host` . Il n'utilise pas les tables de droit de

niveau table ou colonne.

## 4.2.7 Droits fournis par MySQL

Les droits des utilisateurs sont stockés dans les tables `user` , `db` , `host` , `tables_priv` et `columns_priv` de la base `mysql` (c'est−à−dire, la base nommée `mysql` ). Le serveur MySQL lit ces tables au démarrage, et dans les circonstances indiquées dans la section Quand les changements de privilèges prennent−ils effet ? .

Les noms utilisés dans ce manuel font référence aux droits fournis par MySQL version 4.0.2, tel que présentés dans la table ci−dessous, avec le nom de la colonne associée au droit, dans la table de droits, et dans le contexte d'application :

| Droit | Colonne | Contexte |
|---|---|---|
| ALTER | Alter_priv | tables |
| DELETE | Delete_priv | tables |
| INDEX | Index_priv | tables |
| INSERT | Insert_priv | tables |
| SELECT | Select_priv | tables |
| UPDATE | Update_priv | tables |
| CREATE | Create_priv | bases de données, tables ou index |
| DROP | Drop_priv | bases de données ou tables |
| GRANT | Grant_priv | bases de données ou tables |
| REFERENCES | References_priv | bases de données ou tables |
| CREATE TEMPORARY TABLES | Create_tmp_table_priv | administration du serveur |
| EXECUTE | Execute_priv | administration du serveur |
| FILE | File_priv | accès aux fichiers du serveur |
| LOCK TABLES | Lock_tables_priv | administration du serveur |
| PROCESS | Process_priv | administration du serveur |
| RELOAD | Reload_priv | administration du serveur |
| REPLICATION CLIENT | Repl_client_priv | administration du serveur |
| REPLICATION SLAVE | Repl_slave_priv | administration du serveur |
| SHOW DATABASES | Show_db_priv | administration du serveur |
| SHUTDOWN | Shutdown_priv | administration du serveur |
| SUPER | Super_priv | administration du serveur |

Les droits de `SELECT` , `INSERT` , `UPDATE` et `DELETE` vous permettent de faire des opérations sur les lignes qui existent, dans une table existante d'une base.

La commande `SELECT` requiert le droit de `SELECT` uniquement si des lignes sont lues dans une une table. Vous pouvez exéctuer une commande `SELECT` même sans aucun droit d'accès à une base de données dans le serveur. Par exemple, vous pourriez utiliser le client `mysql` comme une simple calculatrice :

```
 mysql> SELECT 1+1;
mysql> SELECT PI()*2;
```

Le droit de `INDEX` vous donne le droit de créer et détruire des index de table.Le droit de `ALTER` vous donne le droit de modifier une table avec la commande `ALTER TABLE` .

Les droits de `CREATE` et `DROP` vous permettent de créer de nouvelles tables et bases de données, et de les supprimer.

Notez que si vous donnez le droit de `DROP` pour la base de données `mysql` à un utilisateur, cet utilisateur pourra détruire la base qui contient les droits d'accès du serveur !

Le droit de `GRANT` vous permet de donner les droits que vous possédez à d'autres utilisateurs.

Le droit de `FILE` vous donne la possibilité de lire et écrire des fichiers sur le serveur avec les commandes `LOAD DATA INFILE` et `SELECT ... INTO OUTFILE` . Tout utilisateur qui possède ce droit peut donc lire ou

écrire dans n'importe quel fichier à l'intérieur duquel le serveur MySQL peut lire ou écrire.

Les autres droits sont utilisés pour les opérations administratives qui sont exécutées par l'utilitaire `mysqladmin` . La table ci−dessous montre quelle commande est associée à `mysqladmin` avec un de ces droits :

| Droit | Commande autorisée |
|---|---|
| RELOAD | reload , refresh , flush−privileges , flush−hosts , flush−logs et flush−tables |
| SHUTDOWN | shutdown |
| PROCESS | processlist |
| SUPER | kill |

La commande `reload` indique au serveur de relire les tables de droits. La commande `refresh` vide les tables de la mémoire, écrit les données et ferme le fichier de log. `flush−privileges` est un synonyme de `reload` . Les autres commandes `flush−*` effectuent des fonctions similaires à la commande `refresh` mais sont plus limitées dans leur application, et sont préférables dans certains contextes. Par exemple, si vous souhaitez simplement vider les tampons dans le fichier de log, utilisez `flush−logs` , qui est un meilleur choix que `refresh` .

La commande `shutdown` éteint le serveur.

La commande `processlist` affiche les informations sur les threads qui s'exécutent sur le serveur. La commande `kill` termine un des threads du serveur. Vous pouvez toujours afficher et terminer vos propres threads, mais vous aurez besoin des droits de `PROCESS` pour afficher les threads, et le droit de `SUPER` pour terminer ceux qui ont été démarrés par d'autres utilisateurs. Syntaxe de `KILL` .

C'est une bonne idée en général, de ne donner les droits de Grant qu'aux utilisateurs qui en ont besoin, et vous devriez être particulièrement vigilant pour donner certains droits :

- Le droit de `GRANT` permet aux utilisateurs de donner leurs droits à d'autres utilisateurs. Deux utilisateurs avec des droits différents et celui de `GRANT` pourront combiner leurs droits respectifs pour gagner un autre niveau d'utilisation du serveur.
- Le droit de `ALTER` peut être utilisé pour tromper le système en renommant les tables.
- Le droit de `FILE` peut servir à lire des fichiers accessibles à tous sur le serveur, et les placer dans une base de données. Le contenu pourra alors être lu et manipulé avec `SELECT` . Cela inclus le contenu de toutes les bases actuellement hébergées sur le serveur !
- Le droit de `SHUTDOWN` peut conduire au dénis de service, en arrêtant le serveur.
- Le droit de `PROCESS` permet de voir en texte clair les commandes qui s'exécutent actuellement, et notamment les changements de mot de passe.
- Les droits sur la base de données `mysql` peuvent être utilisés pour changer des mots de passe ou des droits dans la table des droits (Les mots de passe sont stockés chiffrés, ce qui évite que les intrus ne les lisent). S'ils accèdent à un mot de passe dans la table `mysql.user` , ils pourront l'utiliser pour se connecter au serveur avec cet utilisateur (avec des droits suffisants, le même utilisateur pourra alors remplacer un mot de passe par un autre).

Il y a des choses qui ne sont pas possibles avec le système de droits de MySQL :

- Vous ne pouvez pas explicitement interdire l'accès à un utilisateur spécifique. C'est à dire, vous ne pouvez pas explicitement décrire un utilisateur et lui refuser la connexion.
- Vous ne pouvez pas spécifier qu'un utilisateur a les droits de créer et de supprimer des tables dans une base, mais qu'il n'a pas les droits pour créer et supprimer cette base.

## 4.2.8 Se connecter au serveur MySQL

Les clients MySQL requièrent généralement que vous spécifiez les paramètres de connexion pour vous connecter au serveur MySQL : l'hôte que vous voulez utiliser, votre nom d'utilisateur et votre mot de passe. Par exemple, le client `mysql` peut être démarré comme ceci (les arguments optionnels sont entre crochets `'['` et `']'` ) :

```
shell> mysql [-h nom_d_hote] [-u nom_d_utilisateur] [-pvotre_mot_de_passe]
```

Les formes alternatives des options `-h` , `-u` , and `-p` sont `--host=host_name` , `--user=user_name` et `--password=your_pass` . Notez qu'il n'y a **aucun espace** entre l'option `-p` ou `--password=` et le mot de passe qui le suit.

**Note** : spécifier un mot de passe en ligne de commande n'est pas sécuritaire ! Tout utilisateur de votre serveur peut découvrir votre mot de passe en tapant la commande : `ps auxww` . Fichier d'options `my.cnf` .

`mysql` utilise des valeurs par défaut pour chacun des paramètes qui manquent en ligne de commande :

- Le nom d'hôte par défaut est `localhost` .
- Le nom d'utilisateur par défaut est votre nom d'utilisateur de système Unix.
- Aucun mot de passe n'est transmis si `-p` manque.

Par exemple, pour un utilisateur Unix `joe` , les commandes suivantes sont équivalentes :

```
shell> mysql -h localhost -u joe
shell> mysql -h localhost
shell> mysql -u joe
shell> mysql
```

Les autres clients MySQL se comportent de manière similaire.

Sous Unix, vous pouvez spécifier différentes valeurs par défaut qui seront utilisées lorsque vous établierez la connexion, de manière à ce que vous n'ayez pas à entrer ces informations en ligne de commande lorsque vous invoquez un programme client. Cela peut se faire de plusieurs façons :

- Vous pouvez spécifier les informations de connexion dans la section `[client]` du fichier de configuration `.my.cnf` de votre dossier personnel. La section qui vous interesse ressemble à ceci :
  ```
  [client]
  host=nom_d_hote
  user=nom_d'utilisateur
  password=votre_mot_de_passe
  ```
  Fichier d'options `my.cnf` .
- Vous pouvez spécifier les paramètres de connexion avec les variables d'environnement. L'hôte peut être spécifié à `mysql` avec la variable `MYSQL_HOST` . L'utilisateur MySQL peut être spécifié avec la variable `USER` (uniquement pour Windows). Le mot de passe peut être spécifié avec `MYSQL_PWD` : mais ceci est peu sécuritaire. Voir la prochaine section Variables d'environnement MySQL .

## 4.2.9 Contrôle d'accès, étape 1 : Vérification de la connexion

Lorsque vous tentez de vous connecter au serveur MySQL, le serveur accepte ou rejette la connexion en fonction de votre identité et du mot de passe que vous fournissez. Si le mot de passe ne correspond pas à celui qui est en base, le serveur vous interdit complètement l'accès. Sinon, le serveur accepte votre connexion et passe à l'étape 2, et la gestion de commandes.

Votre identité est basée sur trois informations :

- L'hôte depuis lequel vous vous connectez
- Votre nom d'utilisateur MySQL
- Votre mot de passe

La vérification d'identité est réalisée avec les trois colonnes de la table `user` ( `Host` , `User` et `Password` ). Le serveur accepte la connexion uniquement si une entrée dans la table `user` correspond à votre hôte, et que vous fournissez le mot de passe qui correspond.Les valeurs de la table `user` peuvent être paramétrées comme ceci :

- Une valeur de la colonne `Host` peut être un nom d'hôte, une adresse IP numérique, ou encore `'localhost'` , qui représente l'hôte local.
- Vous pouvez utiliser les caractères jokers `'%'` et `'_'` dans le champ `Host` .
- Une valeur de `'%'` dans la colonne `Host` remplace n'importe quelle autre valeur.
- Une valeur vide pour la colonne `Host` indique que les droits doivent être gérés avec les entrées de la table `host` qui correspond à l'hôte se connectant. Vous trouverez plus d'informations à ce sujet dans le prochain chapitre.
- Depuis MySQL version 3.23, les valeurs de `Host` spécifiées sous la forme d'IP numériques peuvent être complétées avec le masque de réseau qui indique combien de bits d'adresse sont utilisés. Par exemple :

  ```
  mysql> GRANT ALL PRIVILEGES ON db.*
     -> TO david&#64;'192.58.197.0/255.255.255.0';
  ```

  Cela permet à toute personne se connectant depuis une adresse IP qui satisfait la contrainte suivante :

  ```
  user_ip & netmask = host_ip.
  ```

  Dans l'exemple ci−dessus, toutes les IP dans l'intervalle 192.58.197.0 − 192.58.197.255 pourront se connecter au serveur MySQL.
- Les caractères joker ne sont pas autorisés dans le champ `User` , mais vous pouvez spécifier une valeur vide qui remplacera tous les noms. Si l'entrée de la table `user` , qui correspond à une connexion entrante, a un nom d'utilisateur vide, l'utilisateur est alors considéré comme anonyme (c'est l'utilisateur sans nom) et non pas le nom d'utilisateur fourni. Cela signifie qu'un utilisateur fournissant un nom d'utilisateur vide sera utilisé pour identifier ses droits dans les prochaines étapes.
- Le champ `Password` peut être vide. Cela ne signifie pas que n'importe quel mot de passe est valable, mais que l'utilisateur peut se connecter sans fournir de mot de passe.

Les valeurs non vides du champ `Password` représentent des valeurs du mot de passe chiffrées. MySQL ne stocke pas les mots de passe en clair, à la vue de tous. Au contraire, le mot de passe fourni pas l'utilisateur qui tente de se connecter est chiffré (avec la fonction `PASSWORD()` ). Le mot de passe ainsi chiffré est alors utilisé entre le client et le serveur pour vérifier s'il est valable. Cela évite que des mots de passe en clair circulent entre le client et le serveur, sur la connexion. Notez que du point de vue de MySQL, le mot de passe chiffré est le vrai mot de passe, ce qui fait que vous ne devez en aucun cas le donner à un tiers. En particulier, ne donnez pas accès en lecture aux utilisateurs normaux aux tables d'administration dans la base `mysql` ! A partir de sa version 4.1, MySQL utilise un mécanisme différent pour les logins, mots de passes qui est sécurisé même si les paquets TCP/IP sont sniffés et/ou que la base de données mysql est capturée.Les exemples ci−dessous illustrent comment différentes variantes de `Host` et `User` dans la table `user` s'appliquent aux connexion entrantes :

| `Host` **value** | `User` **value** | **Connexions autorisées** |
|---|---|---|
| 'thomas.loc.gov' | 'fred' | fred , se connectant depuis thomas.loc.gov |
| 'thomas.loc.gov' | '' | N'importe quel utilisateur, se connectant depuis thomas.loc.gov |
| '%' | 'fred' | fred , se connectant depuis n'importe quel hôte |
| '%' | '' | N'importe quel utilisateur, se connectant depuis n'importe quel hôte |
| '%.loc.gov' | 'fred' | fred , se connectant depuis n'importe quel hôte dans le domaine loc.gov |
| 'x.y.%' | 'fred' | fred , se connectant depuis x.y.net , x.y.com , x.y.edu , etc. (Ceci n'est probablement pas très utilisé) |

| | | |
|---|---|---|
| '144.155.166.177' | 'fred' | fred , se connectant depuis l'hôte d'IP 144.155.166.177 |
| '144.155.166.%' | 'fred' | fred , se connectant depuis un hôte d'IP dans la classe C 144.155.166 |
| '144.155.166.0/255.255.255.0' | 'fred' | Identique à l'exemple précédent |

Comme vous pouvez utiliser des caractères jokers dans les adresses IP de la colonne `Host` (par exemple, `'144.155.166.%'` pour identifier tout un sous−réseau), il est possible d'exploiter cette fonctionnalité en nommant un hôte `144.155.166.kekpart.com` . Pour contrer de telles tentatives, MySQL interdit les caractères jokers avec les noms d'hôtes qui commencent par des chiffres ou des points. Par exemple, si vous avez un nom d'hôte tel que `1.2.foo.com` , il ne sera jamais trouvé dans la colonne `Host` des tables de droits. Seule une adresse IP numérique peut être comparée avec un masque à caractère joker.

Une connexion entrante peut être identifiée par plusieurs entrées dans la table `user` . Par exemple, une connexion depuis `thomas.loc.gov` par `fred` sera identifiée de plusieurs façons différentes dans l'exemple ci−dessus. Comment le serveur va−t−il choisir si il y a plusieurs solutions ? Le serveur résoud cette question en triant les utilisateurs dans la colonne `user` après le démarrage, et il recherchera dans cette liste triée lorsqu'un utilisateur tente de se connecter. La première ligne qui satisfait les critères sera alors utilisée.

Le tri de la table `user` suit les règles suivantes. Supposons que votre table `user` ressemble à ceci :

```
 +-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| %         | root     | ...
| %         | jeffrey  | ...
| localhost | root     | ...
| localhost |          | ...
+-----------+----------+-
```

Lorsque le serveur lit cette table, il ordonne les lignes depuis les valeurs les plus spécialisées de la colonne `Host` jusqu'aux plus générales ( `'%'` dans la colonne `Host` signifie ``tous les hôtes'' et elle est la moins spécifique). Les entrées identiques dans la colonne `Host` sont ordonnées en fonction de la spécificité des valeurs de la colonne `User` (une entrée vide dans la colonne `User` signifie ``n'importe quel utilisateur'' et est spécifique). Le résultat de ce tri donne quelque chose comme ceci :

```
 +-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| localhost | root     | ...
| localhost |          | ...
| %         | jeffrey  | ...
| %         | root     | ...
+-----------+----------+-
```

Lorsqu'une connexion est en cours de mise en place, le serveur regarde dans cette liste, et utilisera la première entrée trouvée. Pour une connexion depuis l'hôte `localhost` avec le nom d'utilisateur `jeffrey` , les entrées `'localhost'` dans la colonne `Host` sont trouvées en premier. Parmi celles−la, la ligne avec un utilisateur vide satisfait les deux contraintes sur le nom et l'hôte. `'%'/'jeffrey'` pourrait avoir fonctionné, mais comme ce n'est pas le premier rencontré, il n'est pas utilisé.

Voici un autre exemple. Supposons que la table `user` ressemble à ceci :

```
 +----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
| %              | jeffrey  | ...
| thomas.loc.gov |          | ...
+----------------+----------+-
```

La table triée ressemble à ceci :

```
 +----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
| thomas.loc.gov |          | ...
| %              | jeffrey  | ...
+----------------+----------+-
```

Une connexion depuis l'hôte `thomas.loc.gov` avec `jeffrey` satisfait les conditions de la première ligne, tandis qu'une connexion depuis `whitehouse.gov` avec `jeffrey` satisfait la seconde ligne.Une erreur

commune est de penser que pour un utilisateur donné, toutes les entrées qui utilisent explicitement ce nom seront utilisées en premier lorsque la connexion est en cours d'établissement. Ceci est tout simplement faux. L'exemple précédent illustre cette situation, car la connexion depuis l'hôte `thomas.loc.gov` avec `jeffrey` est la première ligne qui est trouvée, alors que la ligne contenant `'jeffrey'` dans la colonne `User` est ignorée, car il n'y a pas de nom d'utilisateur.

Si vous avez des problèmes lors de la connexion, listez le contenu de la table `user` et triez−la à la main pour savoir quel est la première ligne qui est trouvée et utilisée.

Si la connexion s'est bien passée, mais que vos privilèges vous semblent faux, utilisez la fonction `CURRENT_USER()` (ajoutée en version 4.0.6) pour voir quel combinaison d'utilisateur/hôte votre connexion a utilisé. Fonctions diverses .

## 4.2.10 Contrôle d'accès, étape 2 : Vérification de la requête

Une fois que vous avez établi la connexion, le serveur passe à l'étape 2. Pour chaque requête qui est fournie avec la connexion, le serveur vérifie si vous avez les droits suffisants pour exécuter une commande, en fonction du type de commande. C'est à ce moment que les colonnes de droits des tables d'administration entrent en scène. Ces droits peuvent provenir de la table `user` , `db` , `host` , `tables_priv` ou `columns_priv` . Les tables d'administration sont manipulées avec les commandes `GRANT` et `REVOKE` . Syntaxe de `GRANT` et `REVOKE` . (Vous pouvez aussi vous reporter à la section Comment le système de droits fonctionne qui liste les champs présents dans chaque table d'administration).

La table d'administration `user` donne les droits aux utilisateurs au niveau global, c'est à dire que ces droits s'appliquent quelle que soit la base de données courante. Par exemple, si la table `user` vous donne le droit d'effacement , `DELETE` , vous pouvez effacer des données dans n'importe quelle base de ce serveur. En d'autres termes, les droits stockés dans la table `user` sont des droits de super utilisateur. Il est recommandé de ne donner des droits via la table `user` uniquement aux super utilisateurs, ou aux administrateurs de bases. Pour les autres utilisateurs, il vaut mieux laisser les droits dans la table `user` à `'N'` et donner des droits au niveau des bases uniquement, avec les tables `db` et `host` .

Les tables `db` et `host` donnent des droits au niveau des bases. Les droits peuvent être spécifiés dans ces tables comme ceci :

- Les caractères `'%'` et `'_'` peuvent être utilisés dans la colonne `Host` et `Db` des deux tables. Si vous souhaitez utiliser le caractère `'_'` comme nom de base, utiliser la séquence `'\_'` dans la commande `GRANT` .
- La valeur `'%'` dans la colonne `Host` de la table `db` signifie ``tous les hôtes''. Une valeur vide dans la colonne `Host` de la table `db` signifie ``consulte la table `host` pour plus de détails''.
- La valeur `'%'` ou vide dans la colonne `Host` de la table `host` signifie ``tous les hôtes''.
- La valeur `'%'` ou vide dans la colonne `Db` des deux tables signifie ``toutes les bases de données''.
- Un utilisateur vide dans la colonne `User` de l'une des deux tables identifie l'utilisateur anonyme.

Les tables `db` et `host` sont lues et triées par le serveur au démarrage (en même temps que la table `user` . La table `db` est triée suivant les valeurs des colonnes `Host` , `Db` et `User` , et la table `host` est triée en fonction des valeurs des colonnes `Host` et `Db` . Comme pour la table `user` , le tri place les entrées les plus spécifiques au début, et les plus générales à la fin. Lorsque le serveur recherche une ligne, il utilise la première qu'il trouve.

Les tables `tables_priv` et `columns_priv` spécifient les droits au niveau des tables et des colonnes. Les

valeurs des droits dans ces tables peuvent être spécifiés avec les caractères spéciaux suivants :

- Les caractères `'%'` et `'_'` peuvent être utilisés dans la colonne `Host` des deux tables.
- La valeur `'%'` dans la colonne `Host` des deux tables signifie ``tous les hôtes''.
- Les colonnes `Db` , `Table_name` et `Column_name` ne peuvent pas contenir de valeur vide ou de caractères jokers, dans les deux tables.

Les tables `tables_priv` et `columns_priv` sont triées en fonction des colonnes `Host` , `Db` et `User` . Ce tri est similaire à celui du tri de la table `db` , même si le tri est bien plus simple, car seul le champ `Host` peut contenir des caractères jokers.

Le processus de vérification est décrit ci−dessous. Si vous êtes familier avec le code source de contrôle d'accès, vous noterez que la description diffère légèrement de l'algorithme utilisé. La description est équivalente à ce que fait en réalité le code. La différence permet une meilleure approche pédagogique.

Pour les requêtes d'administration comme `SHUTDOWN` , `RELOAD` , etc., le serveur vérifie uniquement l'entrée dans la table `user` , car c'est la seule table qui spécifie des droits d'administration. Le droit est donné si la ligne utilisée dans la connexion courante dans la table `user` donne le droit, et sinon, ce droit est interdit. Par exemple, si vous souhaitez exécuter la commande `mysqladmin shutdown` mais que votre ligne dans la table `user` ne vous en donne pas le droit ( `SHUTDOWN` ), vous n'aurez pas le droit sans même vérifier les tables `db` ou `host` : ces tables ne contiennent pas de colonne `Shutdown_priv` , ce qui évite qu'on en ait besoin.

Pour les requêtes exploitant une base de données, comme `INSERT` , `UPDATE` , etc., le serveur vérifie d'abord les droits globaux de l'utilisateur (droits de super utilisateur), en regardant dans la table `user` . Si la ligne utilisée dans cette table donne droit à cette opération, le droit est donné. Si les droits globaux dans `user` sont insuffisants, le serveur déterminera les droits spécifiques à la base avec les tables `db` et `host` :

- Le serveur recherche dans la table `db` des informations en se basant sur les colonnes `Host` , `Db` et `User` . Les champs `Host` et `User` sont comparés avec les valeurs de l'hôte et de l'utilisateur qui sont connectés. Le champ `Db` est comparé avec le nom de la base de données que l'utilisateur souhaite utiliser. S'il n'existe pas de ligne qui corresponde à `Host` et `User` , l'accès est interdit.
- S'il existe une ligne dans la table `db` et que la valeur de la colonne `Host` n'est pas vide, cette ligne définit les droits de l'utilisateur.
- Si dans la ligne de la table `db` , la colonne `Host` est vide, cela signifie que la table `host` spécifie quels hôtes doivent être autorisés dans la base. Dans ce cas, une autre recherche est faite dans la table `host` pour trouver une ligne avec les colonnes `Host` et `Db` . Si aucune ligne de la table `host` n'est trouvée, l'accès est interdit. S'il y a une ligne, les droits de l'utilisateur sont calculés comme l'intersection ( **NON PAS** l'union !) des droits dans les tables `db` et `host` , c'est−à−dire que les droits doivent être marqués `'Y'` dans les deux tables (de cette façon, vous pouvez donner des droits généraux dans la table `db` puis les restreindre sélectivement en fonction des hôtes, en utilisant la table `host` .

Après avoir déterminé les droits spécifiques à l'utilisateur pour une base grâce aux tables `db` et `host` , le serveur les ajoute aux droits globaux, donnés par la table `user` . Si le résultat autorise la commande demandée, l'accès est donné. Sinon, le serveur vérifie les droits au niveau de la table et de la colonne dans les tables `tables_priv` et `columns_priv` , et les ajoute aux droits déjà acquis. Les droits sont alors donnés ou révoqués en fonction de ces résultats.

Exprimée en termes booléens, la description précédente du calcul des droits peut être résumé comme ceci :

```
 droits globaux
OR (droits de base AND droits d'hôte)
OR droits de table
OR droits de colonne
```

Il n'est peut−être pas évident pourquoi, si les droits globaux issus de la table `user` sont initialement insuffisants pour l'opération demandée, le serveur ajoute ces droits à ceux de base, table ou colonne ? La raison est que la requête peut demander l'application de plusieurs droits. Par exemple, si vous exécutez une commande `INSERT ... SELECT` , vous aurez besoin des droits de `INSERT` et de `SELECT` . Vos droits peuvent être tels que la table `user` donne un droit, mais que la table `db` en donne un autre. Dans ce cas, vous aurez les droits nécessaires pour faire une opération, mais le serveur ne peut le déduire d'une seule table : les droits de plusieurs tables doivent être combinés pour arriver à la bonne conclusion.

La table `host` sert à gérer une liste d'hôtes reconnus et sécuritaires.

Chez TcX, la table `host` contient une liste de toutes les machines du réseau local. Ces machines reçoivent tous les droits.

Vous pouvez aussi utiliser la table `host` pour spécifier les hôtes qui **ne sont pas** sécuritaires. Supposons que la machine `public.votre.domaine` t est placée dans une zone publique que vous considérez comme peu sûre. Vous pouvez autoriser l'accès de toutes les machines, hormis celle−ci, grâce à la table `host` configurée comme ceci :

```
 +−−−−−−−−−−−−−−−−−−−+−−−−+−
| Host                | Db | ...
+−−−−−−−−−−−−−−−−−−−+−−−−+−
| public.your.domain | %  | ... (tous les droits à 'N')
| %.your.domain      | %  | ... (tous les droits à 'Y')
+−−−−−−−−−−−−−−−−−−−+−−−−+−
```

Naturellement, vous devriez toujours tester vos requêtes dans la table de droits, en utilisant l'utilitaire `mysqlaccess` pour vous assurer que vous disposez des droits nécessaires pour réaliser cette opération.

## 4.2.11 Causes des erreurs Access denied

Si vous rencontrez des erreurs `Access denied` quand vous essayez de vous connecter au serveur MySQL, la liste suivante indique quelques actions à entreprendre pour corriger le problème :

- Après l'installation de MySQL, avez vous exécuté le script `mysql_install_db` pour initialiser le contenu de la table des droits ? Si tel n'est pas le cas, faites le. Mise en place des privilèges initiaux de MySQL . Testez les privilèges initiaux en exécutant la commande suivante :
  ```
  shell> mysql −u root test
  ```
  Le serveur devrait vous laisser vous connecter sans erreurs. Vous devez aussi vous assurer que vous avez un fichier `user.MYD` dans le dossier des bases de données MySQL. Ordinairement, il s'agit de `CHEMIN/var/mysql/user.MYD` , où `CHEMIN` est le chemin vers le dossier racine de l'installation MySQL.
- Après une installation toute fraîche, vous devez vous connecter au serveur et créer les utilisateurs en réglant leurs permissions d'accès :
  ```
  shell> mysql −u root mysql
  ```
  Le serveur devrait vous laisser vous connecter car l'utilisateur `root` de MySQL n'a pas de mot de passe initial. Ceci est aussi une faille de sécurité, et donc, vous devez choisir un mot de passe pour l'utilisateur `root` en même tant que les autres utilisateurs MySQL.Si vous essayez de vous connecter en `root` et que vous rencontrez cette erreur :
  ```
  Access denied for user: '&#64;unknown' to database mysql
  ```
  cela signifie que vous n'avez pas d'entrée dans la table `user` avec une valeur `'root'` de la colonne `User` et que `mysqld` cne peut trouver de nom d'hôte à votre client. Dans ce cas, vous

devez redémarrer le serveur avec l'option `--skip-grant-tables` et éditer le fichier `/etc/hosts` ou `\windows\hosts` pour ajouter une entrée pour votre hôte.

- Si vous obtenez une erreur qui ressemble à celle−ci :

```
shell> mysqladmin -u root -pxxxx ver
Access denied for user: 'root&#64;localhost' (Using password: YES)
```

Cela signifie que vous utilisez un mot de passe erroné. Mise en place d'un mot de passe .Si vous avez oublié le mot de passe root, vous pouvez redémarrer `mysqld` avec `--skip-grant-tables` pour changer le mot de passe. Comment modifier un mot de passe Root oublié .

Si vous obtenez l'erreur suivante même sans spécifier de mot de passe, cela signifie que vous avez un mauvais mot de passe dans un fichier `my.ini` . Fichier d'options `my.cnf` . Vous pouvez ne pas utiliser les fichiers d'options en utilisant l'option `--no-defaults` , comme suit :

```
shell> mysqladmin --no-defaults -u root ver
```

- Si vous mettez à jour une installation MySQL existante à partir d'une version plus ancienne que 3.22.11 vers la version 3.22.11 ou plus à jour, avez−vous exécuté le script `mysql_fix_privilege_tables` ? Si ce n'est pas le cas, faites le. La structure des tables de droits a changé avec la version 3.22.11 de MySQL lorsque la commande `GRANT` est devenue fonctionnelle.
- Si vos droits semblent avoir changé au milieu d'une session, il se peut qu'un super−utilisateur les ait changé. Recharger les tables de droits affecte les nouvelles connexions, mais affecte aussi les anciennes comme indiqué dans Quand les changements de privilèges prennent−ils effet ? .
- Si vous n'arrivez pas à faire fonctionner votre mot de passe, souvenez−vous que vous devez utiliser la fonction `PASSWORD()` si vous le changez avec les commandes `INSERT` , `UPDATE` , ou `SET PASSWORD` . L'utilisation de la fonction `PASSWORD()` n'est pas nécessaire si vous spécifiez le mot de passe en utilisant la commande `GRANT ... INDENTIFIED BY` ou la commande `mysqladmin password` . Mise en place d'un mot de passe .
- `localhost` est un synonyme de votre nom d'hôte local, et est aussi l'hôte par défaut auquel le client essaye de se connecter si vous n'en spécifiez pas un explicitement. Toutefois, les connexions à `localhost` ne fonctionnent pas si vous utilisez une version antérieure à la 3.23.27 qui utilise les MIT−pthreads (les connexions à `localhost` sont réalisées en utilisant les sockets Unix, qui n'étaient pas supportés par les MIT−pthreads à ce moment). Pour contourner ce problème sur de tels systèmes, vous devez utiliser l'option `--host` pour nommer l'hôte du serveur explicitement. Cela créera une connexion TCP/IP vers le serveur `mysqld` . Dans ce cas, vous devez avoir votre vrai nom d'hôte dans les entrées de la table `user` du serveur hôte. (Cela est vrai même si vous utilisez un programme client sur la même machine que le serveur.)
- Si vous obtenez une erreur `Access denied` lorsque vous essayez de vous connecter à la base de données avec `mysql -u nom_utilisateur nom_base` , vous pouvez avoir un problème dans la table `user` . Vérifiez le en vous exécutant `mysql -u root mysql` et entrant la commande SQL suivante :

```
mysql> SELECT * FROM user;
```

Le résultat devrait comprendre une entrée avec les colonnes `Host` et `User` correspondate au nom d'hôte de votre ordinateur et votre nom d'utilisateur MySQL.

- Le message d'erreur `Access denied` vous dira en tant que qui vous essayez de vous identifier, l'hôte à partir duquel vous voulez le faire, et si vous utilisez ou pas un mot de passe. Normalement, vous devez avoir une entrée dans la table `user` qui correspondent au nom d'hôte et nom d'utilisateur donnés dans le message d'erreur. Par exemple, si vous obtenez une erreur qui contient `Using password: NO` , cela signifie que vous avez essayé de vous connecter sans mot de passe.
- Si vous obtenez l'erreur suivante en essayant de vous connecter à partir d'un hôte différenet de celui sur lequel est placé le serveur, c'est qu'il n'y a pas d'enregistrement dans la table `user` qui correspond à cet hôte :

```
Host ... is not allowed to connect to this MySQL server
```

Vous pouvez contourner le problème en utilisant l'outil en ligne de commande `mysql` (sur l'hôte du serveur !) pour ajouter un enregistrement aux tables `user` , `db` , ou `host` pour la combinaison

utilisateur / nom d'hôte avec laquelle vous essayez de vous connecter puis exécuter `mysqladmin flush-privileges` . Si vous n'avez pas la version 3.22 de MySQL et que vous ne connaissez ni l'IP ni le nom d'hôte à partir duquel vous essayez de vous connecter, vous devez créer une entrée avec `'%'` dans la colonne `Host` dans la table `user` et redémarrer `mysqld` avec l'option `--log` sur la machine serveur. Après avoir essayé à nouveau de vous connecter à partir de la machine cliente, les informations contenues dans le log de MySQL vous apprendront comment vous vous êtes vraiment connectés. (Remplacez alors l'entrée de la table `user` contenant `'%'` avec le nom d'hôte qui apparait dans le log. Sinon, vous aurez un système non-sécurisé.)

Une autre raison pour cette erreur sous Linux est que vous utilisez une version binaire de MySQL qui est compilée avec une version de glibc différente de celle que vous utilisez. Dans ce cas, vous devez soit mettre à jour votre système d'exploitation/glibc, soit télécharger les sources de MySQL et les compiler vous-même. Un RPM de sources est normalement facile à compiler et installer, cela ne devrait donc pas vous poser de gros problèmes.

- Si vous obtenez une erreur où le nom d'hôte est absent ou que celui-ci est une adresse IP alors que vous avez bien entré le nom d'hôte :

```
 shell> mysqladmin -u root -pxxxx -h some-hostname ver
Access denied for user: 'root&#64;' (Using password: YES)
```

  Cela signifie que MySQL a rencontré des erreurs lors de la résolution de l'IP du nom d'hôte. Dans ce cas, vous pouvez exécuter `mysqladmin flush-hosts` pour vider le cache interne des DNS. Comment MySQL utilise les DNS .Les autres solutions sont :
  - Essayez de trouver le problème avec votre serveur DNS et corrigez le.
  - Spécifiez les IP à la place des noms d'hôtes dans les tables de droits de MySQL.
  - Démarrez `mysqld` avec `--skip-name-resolve` .
  - Démarrez `mysqld` avec `--skip-host-cache` .
  - Connectez vous à `localhost` si vous utilisez le serveur et le client sur la même machine.
  - Placez le nom de la machine cliente dans `/etc/hosts` .
- Si `mysql -u root test` fonctionne mais que `mysql -h votre_hote -u root test` provoque une erreur `Access denied` , il se peut que vous ayez entré de mauvaises informations pour votre nom d'hôte dans la table `user` . Un problème commun ici est que la valeur `Host` dans la table `user` spécifie un nom d'hôte non-qualifié, mais que vos routines système de résolution de noms retournent un nom de domaine pleinement qualifié (ou vice-versa). Par exemple, si vous avez une entrée avec l'hôte `'tcx'` dans la table `user` , mais que vos DNS disent à MySQL que votre nom d'hôte est `'tcx.subnet.se'` , l'entrée ne fonctionnera pas. Essayez d'ajouter une entrée dans la table `user` qui contient votre adresse IP en tant que valeur de la colonne `Host` . (Une alternative est d'ajouter une entrée dans la table `user` avec une valeur de `Host` qui contient un caractère spécial, par exemple, `'tcx.%'` . Toutefois, l'utilisation des noms d'hôtes se terminant par `'%'` est **non-sécurisé** et **n** 'est **pas** recommendé !)
- Si `mysql -u utilisateur test` fonctionne mais que `mysql -u utilisateur autre_base` ne fonctionne pas, vous n'avez pas d'entrée pour `autre_base` listée dans la table `db` .
- Si `mysql -u utilisateur nom_base` fonctionne à partir du serveur, mais que `mysql -h nom_hote -u utilisateur nom_base` ne fonctionne pas à partir d'une autre machine, cette machine n'est pas listée dans la table `user` ou `db` .
- Si vous n'arrivez pas à trouver pourquoi vous obtenez l'erreur `Access denied` , effacez toutes les entrées de la table `user` dont la valeur du champ `Host` contiennent des caractères spéciaux (entrées contenant `'%'` ou `'_'` ). Une erreur commune est d'insérer une nouvelle entrée avec `Host` = `'%'` et `User` = `'un utilisateur'` , en pensant que cela vous permettra de spécifier `localhost` pour vous connecter à partir de la même machine. La raison pour laquelle cela ne fonctionnera pas est que les droits par défaut incluent une entrée avec `Host` = `'localhost'` et `User` = `''` . Puisque cette entrée possède une valeur de `Host` égale à `'localhost'` , qui est plus spécifique que `'%'` , elle est utilisée de préférence à la nouvelle entrée lors de la connexion à partir de `localhost` ! La procédure correcte est d'insérer une seconde entrée avec `Host` = `'localhost'` et `User` = `'un_utilisateur'` , ou de supprimer l'entrée avec `Host` = `'localhost'` et `User` = `''` .
- Si vous avez l'erreur suivante, vous avez peut-être un problème avec la table `db` ou `host` :

```
Access to database denied
```

Si l'entrée sélectionnée dans la table `db` possède un champ `Host` vide, assurez−vous qu'il y a au moins une entrée correspondante dans la table `host` spécifiant les hôtes auxquels l'entrée dans la table `db` s'applique.Si vous obtenez l'erreur lors de l'utilisation des commandes SQL `SELECT ... INTO OUTFILE` ou `LOAD DATA INFILE` , votre entrée dans la table `user` ne possède probablement pas le privilège `FILE` .

- Souvenez−vous que le programme client utilise les paramètres de connexion spécifiés dans les fichiers de configuration ou des les variables d'environnement. Variables d'environnement MySQL . Si un client semble envoyer les mauvais paramètres de connexion par défaut lorsque vous ne les spécifiez pas en ligne de commande, vérifiez votre environnement et le fichier `.my.cnf` dans votre répertoire personnel. Vous devrez aussi vérifier les fichiers système de configuration de MySQL, bien qu'il est fort peu probable que les paramètres de connexion du client y soient spécifiés. Fichier d'options `my.cnf` . Si vous obtenez une erreur `Access denied` quand vous démarrer un client sans aucune option, assurez−vous de ne pas avoir spécifié un vieux mot de passe dans l'un de vos fichiers d'options ! Fichier d'options `my.cnf` .
- Si vous apportez des modifications au tables de droits directement (en utilisant une requête `INSERT` ou `UPDATE` ) et que vos changements semblent ignorés, souvenez vous que vous devez exécuter une requête `FLUSH PRIVILEGES` ou la commande `mysqladmin flush−privileges` pour demander au serveur de lire à nouveau les tables de droits. Sinon, vos changements ne seront pris en compte qu'au prochain démarrage du serveur. Souvenez−vous qu'après avoir choisi le mot de passe `root` avec une commande `UPDATE` , vous n'aurez pas à le spécifier avant de recharger les privilèges, car le serveur ne sait pas que vous l'avez modifié !
- Si vous avez des problèmes d'accès avec un programme Perl, PHP, Python, ou ODBC, essayez de vous connecter au serveur avec `mysql −u utilisateur nom_base` ou `mysql −u utilisateur −pvotre_passe nom_base` . Si vous pouvez vous connecter en utilisant le client `mysql` , c'est que le problème vient de votre programme et non des droits MySQL. (Notez qu'il n'y a pas d'espace entre `−p` et le mot de passe; vous pouvez aussi utiliser la syntaxe `−−password=votre_passe` pour spécifier le mot de passe. Si vous utilisez l'option `−p` toute seule, MySQL vous demandera le mot de passe.)
- Pour les tests, démarrez le démon `mysqld` avec l'option `−−skip−grant−tables` . Vous pourrez alors changer les tables de droits MySQL puis utiliser le script `mysqlaccess` pour vérifier si vos changements ont l'effet désiré. Lorsque vous êtes satisfait de vos modifications, exécutez `mysqladmin flush−privileges` pour dire au serveur `mysqld` de commencer à utiliser les nouvelles tables de droits. **Note** : Recharger les tables de droits écrase l'option `−−skip−grant−tables` . Cela vous permet de dire au serveur de commencer à prendre en considération les droits sans avoir à le couper et le redémarrer.
- Si rien ne fonctionne, démarrez le démon `mysqld` avec l'option de débogage (par exemple, `−−debug=d,general,query` ). Cela affichera l'hôte et les informations de l'utilisateur pour chaque tentative de connexion. Les informations à propos de chaque commande exécutée seront aussi affichées. Créer des fichiers de traçage .
- Si vous avez d'autres problèmes avec les tables de droits de MySQL et que vous sentez que vous devez envoyer le problème à la liste de diffusion, fournissez toujours le contenu de vos tables de droits. Vous pouvez obtenir les données avec la commande `mysqldump mysql` . Comme toujours, postez votre problème à l'aide du script `mysqlbug` . Comment rapporter des bogues ou problèmes . Dans certains cas, vous aurez besoin de redémarrer `mysqld` avec `−−skip−grant−tables` pour pouvoir exécuter `mysqldump` .

## 4.3 Gestion des comptes utilisateurs de MySQL

## 4.3.1 Syntaxe de GRANT et REVOKE

```
GRANT priv_type [(liste_colonnes)] [, priv_type [(liste_colonnes)] ...]
   ON &#123;nom_de_table | * | *.* | nom_base.*&#125;
   TO nom_utilisateur [IDENTIFIED BY [PASSWORD] 'password']
       [, nom_utilisateur [IDENTIFIED BY 'password'] ...]
   [REQUIRE
       NONE |
       [&#123;SSL| X509&#125;]
   [CIPHER cipher [AND]]
   [ISSUER issuer [AND]]
   [SUBJECT subject]]
   [WITH [GRANT OPTION | MAX_QUERIES_PER_HOUR # |
                         MAX_UPDATES_PER_HOUR # |
                         MAX_CONNECTIONS_PER_HOUR #]]

REVOKE priv_type [(liste_colonnes)] [, priv_type [(liste_colonnes)] ...]
   ON &#123;nom_de_table | * | *.* | nom_base.*&#125;
   FROM nom_utilisateur [, nom_utilisateur ...]
```

`GRANT` est disponible depuis MySQL version 3.22.11. Pour les versions plus anciennes de MySQL, la commande `GRANT` ne fait rien.

Les commandes `GRANT` et `REVOKE` permettent aux administrateurs système de créer des utilisateurs, de leur donner ou de leur retirer des droits, sur 4 niveaux :

*Niveau global*
: Les droits globaux s'appliquent à toutes les bases de données d'un serveur. Ces droits sont stockés dans la table `mysql.user` . `REVOKE ALL ON *.*` retirera seulement les privilèges globaux.

*Niveau base de données*
: Les droits de niveau de base de données s'appliquent à toutes les tables d'une base de données. Ces droits sont stockés dans les tables `mysql.db` et `mysql.host` . `REVOKE ALL ON db.*` retirera seulement les privilèges de base de données.

*Niveau table*
: Les droits de table s'appliquent à toutes les colonnes d'une table. Ces droits sont stockés dans la table `mysql.tables_priv` . `REVOKE ALL ON db.table` retirera seulement les privilèges de table.

*Niveau colonne*
: Les droits de niveau de colonnes s'appliquent à des colonnes dans une table. Ces droits sont stockés dans la table `mysql.columns_priv` . Quand vous utilisez `REVOKE` vous devez spécifier les mêmes colonnes qui s'étaient vues accorder des privilèges.

Si vous donnez des droits à un utilisateur qui n'existe pas, vous créerez cet utilisateur. Pour voir des illustrations du fonctionnement de la commande `GRANT` , voyez Ajouter de nouveaux utilisateurs à MySQL .

Pour les commandes `GRANT` et `REVOKE` , la clause `priv_type` peut être spécifiée par les constantes suivantes :

| | |
|---|---|
| ALL [PRIVILEGES] | Tous les droits sauf WITH GRANT OPTION . |
| ALTER | Autorise l'utilisation de ALTER TABLE . |
| CREATE | Autorise l'utilisation de CREATE TABLE . |
| CREATE TEMPORARY TABLES | Autorise l'utilisation de CREATE TEMPORARY TABLE . |
| DELETE | Autorise l'utilisation de DELETE . |
| DROP | Autorise l'utilisation de DROP TABLE . |
| EXECUTE | Autorise l'utilisateur à exécuter des procédures stockées (pour MySQL 5.0). |
| FILE | Autorise l'utilisation de SELECT ... INTO OUTFILE et LOAD DATA INFILE . |
| INDEX | Autorise l'utilisation de CREATE INDEX et DROP INDEX . |
| INSERT | Autorise l'utilisation de INSERT . |
| LOCK TABLES | Autorise l'utilisation de LOCK TABLES sur les tables pour lesquelles l'utilisateur a les droits de SELECT . |
| PROCESS | Autorise l'utilisation de SHOW FULL PROCESSLIST . |
| REFERENCES | Réservé pour le futur. |

| | |
|---|---|
| RELOAD | Autorise l'utilisation de FLUSH . |
| REPLICATION CLIENT | Donne le droit à l'utilisateur de savoir où sont les maîtres et esclaves. |
| REPLICATION SLAVE | Nécessaire pour les esclaves de réplication (pour lire les historiques binaires du maître). |
| SELECT | Autorise l'utilisation de SELECT . |
| SHOW DATABASES | SHOW DATABASES affiche toutes les bases de données. |
| SHUTDOWN | Autorise l'utilisation de mysqladmin shutdown . |
| SUPER | Autorise une connexion unique même si max_connections est atteind, et l'exécution des commandes CHANGE MASTER , KILL thread , mysqladmin debug , PURGE MASTER LOGS et SET GLOBAL . |
| UPDATE | Autorise l'utilisation de UPDATE . |
| USAGE | Synonyme de ``pas de droits''. |
| GRANT OPTION | Synonyme pour WITH GRANT OPTION |

USAGE peut être utilisé lorsque vous voulez créer un utilisateur sans aucun droit.

Les droits de CREATE TEMPORARY TABLES , EXECUTE , LOCK TABLES , REPLICATION ... , SHOW DATABASES et SUPER sont nouveaux en version 4.0.2. Pour utiliser ces droits après mise à jour en 4.0.2, vous devez exécuter le script mysql_fix_privilege_tables .

Dans les anciennes versions de MySQL, le droit de PROCESS donnait les mêmes droits que le nouveau droit SUPER .

Pour retirer le droit de GRANT à un utilisateur, utilisez les mêmes valeurs de priv_type avec GRANT OPTION :

```
mysql> REVOKE GRANT OPTION ON ... FROM ...;
```

Les valeurs de priv_type que vous pouvez spécifier pour une table sont SELECT , INSERT , UPDATE , DELETE , CREATE , DROP , GRANT OPTION , INDEX et ALTER .Les seules valeurs de priv_type que vous pouvez spécifier pour une colonne, lorsque vous utilisez la clause liste_colonnes ), sont SELECT , INSERT et UPDATE .

Vous pouvez donner des droits globaux en utilisant la syntaxe ON *.* . Vous pouvez donner des droits de base en utilisant la syntaxe ON nom_base.* . Si vous spécifiez ON * et que vous avez une base de données qui est déjà sélectionnée, vous allez donner des droits pour la base de données courante. **Attention** : si vous spécifiez ON * et que vous **n'avez pas** de base courante, vous allez affecter les droits au niveau du serveur !

**Notez bien** : les caractères joker '_' and '%' sont autorisés lors de la spécification de noms dans la commande GRANT . Cela signifie que si vous voulez utiliser par exemple le caractère littéral '_' comme nom de base, vous devez le spécifier sous la forme '_' dans la commande GRANT , pour éviter à l'utilisateur d'accéder à d'autres bases, dont le nom pourrait correspondre au masque d'expression régulière ainsi créé. Utilisez plutôt GRANT ... ON `foo\_bar`.* TO ... .

Afin de permettre l'identification des utilisateurs depuis des hôtes arbitraires, MySQL supporte la spécification du nom d'utilisateur nom_utilisateur sous la forme user@host . Si vous voulez spécifier un nom d'utilisateur user qui contient des caractères spéciaux tels que '-' , ou une chaîne d'hôte host qui contient des caractères joker (comme '%' ), vous pouvez placer le nom de l'utilisateur ou de l'hôte entre guillemets (par exemple, 'test-utilisateur'@'test-nomdhote' ).

Vous pouvez spécifier des caractères jokers dans le nom d'hôte. Par exemple, user@'%.loc.gov' fait correspondre l'utilisateur user de n'importe quel hôte du domaine loc.gov , et user@'144.155.166.%' fait correspondre l'utilisateur user à n'importe quelle adresse de la classe C 144.155.166 .

La forme simple de user est synonyme de user@"%" .

MySQL ne supporte pas de caractères joker dans les noms d'utilisateur. Les utilisateurs anonymes sont définis par l'insertion de ligne avec User='' dans la table mysql.user , ou en créant un utilisateur avec un nom vide, grâce à la commande GRANT .

**Note** : si vous autorisez des utilisateurs anonymes à se connecter à votre serveur, vous devriez aussi donner ces droits à tous les utilisateurs locaux user@localhost car sinon, la ligne dans la table

`mysql.user` sera utilisée lorsque l'utilisateur se connectera au serveur MySQL depuis la machine locale !

Vous pouvez vérifier si cela s'applique à vous en exécutant la requête suivante :

```
mysql> SELECT Host,User FROM mysql.user WHERE User='';
```

Actuellement, la commande `GRANT` supporte uniquement les noms d'hôte, colonne, table et bases de données d'au plus 60 caractères. Un nom d'utilisateur peut être d'au plus 16 caractères.Les droits pour les tables et colonnes sont combinés par OU logique, avec les quatre niveaux de droits. Par exemple, si la table `mysql.user` spécifie qu'un utilisateur a un droit global de `SELECT` , ce droit ne pourra pas être annulé au niveau base, table ou colonne.

Les droits d'une colonne sont calculés comme ceci :

```
 droit global
OR (droit de base de données ET droit d'hôte)
OR droit de table
OR droit de colonne
```

Dans la plupart des cas, vous donnez des droits à un utilisateur en utilisant un seul des niveaux de droits ci−dessus, ce qui fait que la vie n'est pas aussi compliquée. Le détails de la procédure de vérification des droits et disponible dans Sécurité générale et accès à MySQL .

Si vous donnez des droits à une paire utilisateur/hôte qui n'existe pas dans la table `mysql.user` , une ligne sera créée et restera disponible jusqu'à son effacement avec la commande `DELETE` . En d'autre termes, `GRANT` crée une ligne dans la table `user` , mais `REVOKE` ne la supprime pas. Vous devez le faire explicitement avec la commande `DELETE` .

Avec MySQL version 3.22.12 ou plus récent, si un nouvel utilisateur est créé, ou si vous avez les droits de GRANT globaux, le mot de passe sera configuré avec le mot de passe spécifié avec la clause `IDENTIFIED BY` , si elle est fournie. Si l'utilisateur a déjà un mot de passe, il sera remplacé par ce nouveau.

Si vous ne voulez pas faire passer le mot de passe en texte clair, vous pouvez immédiatement utiliser l'option `PASSWORD` suivi du mot de passe déjà chiffré avec la fonction `PASSWORD()` ou l'API C `make_scrambled_password(char *to, const char *password)` .

**Attention** : si vous créez un nouvel utilisateur, mais que vous ne spécifiez pas la clause `IDENTIFIED BY` , l'utilisateur n'aura pas de mot de passe. Ce n'est pas sûr.

Les mots de passe peuvent aussi être modifiés par la commande `SET PASSWORD` . Syntaxe de `SET` .

Si vous donnez les droits de base, une ligne sera ajoutée dans la table `mysql.db` . Lorsque les droits sur cette base seront supprimés avec la commande `REVOKE` , cette ligne disparaîtra.

Si un utilisateur n'a pas de droits sur une table, elle ne sera pas affichée lorsqu'il requiert la liste des tables avec la commande `SHOW TABLES` .

La clause `WITH GRANT OPTION` donne à l'utilisateur le droit de donner les droits qu'il possède à d'autres utilisateurs. La plus grande prudence est recommandée pour cette commande, car il permettra à terme à deux utilisateurs de combiner les droits dont ils disposent.

`MAX_QUERIES_PER_HOUR #` , `MAX_UPDATES_PER_HOUR #` et `MAX_CONNECTIONS_PER_HOUR #` sont nouveaux en MySQL 4.0.2. Ces deux options limitent le nombre de requêtes et de modifications qu'un utilisateur peut réclamer dans une heure. Si `#` vaut 0 (valeur par défaut), alors cela signifie qu'il n'y a pas de limitations pour cet utilisateur. Limiter les ressources utilisateur . Note: pour spécifier l'une de ces options pour un

utilisateur existant sans ajouter d'autres privilèges additionels, utilisez `GRANT USAGE ... WITH MAX_...` .

Vous ne pouvez pas donner à un autre utilisateur un droit que vous ne possédez pas vous−même. Le droit de `GRANT` vous donne le droit de diffuser les droits dont vous disposez déjà.

Soyez bien conscient que lorsque vous donnerez le droit de `GRANT` à un niveau particulier, tous les droits qu'un utilisateur possède déjà (où qui lui seront donnés dans le futur) seront alors diffusables à d'autres individus. Supposons que vous donnez les droits de `INSERT` à un utilisateur, pour une table. Si vous ajoutez le droit de `WITH GRANT OPTION` , cet utilisateur peut donner le droit de `SELECT` mais aussi celui d' `INSERT` . Si vous donnez aussi le droit de `UPDATE` , il pourra alors diffuser les droits de `INSERT` , `SELECT` et `UPDATE` .

Il est recommandé de ne pas donner les droits de `ALTER` à un utilisateur normal. Si vous le faites, les utilisateurs pourront alors contourner le système de droits en renommant des tables !

Notez que si vous utilisez des droits de niveau table ou colonne même pour un utilisateur, le serveur vérifiera alors ces droits pour tous les utilisateurs, et cela ralentira MySQL un peu.

Lorsque `mysqld` démarre, tous les droits sont stockés en mémoire. Les droits de bases, tables et colonnes prennent aussitôt effet, et les droits des utilisateurs prendront effet dès leur prochaine configuration. Les modifications sur les tables de droits que vous effectuez avec les commandes `GRANT` et `REVOKE` sont prises en compte immédiatement par le serveur. Si vous modifiez manuellement les tables (avec `INSERT` , `UPDATE` , etc...), vous devez exécuter la commande `FLUSH PRIVILEGES` , ou la commande en ligne `mysqladmin flush−privileges` pour indiquer au serveur qu'il doit recharger les droits. Quand les changements de privilèges prennent−ils effet ? .

Les différences notables entre l'ANSI SQL et MySQL pour la commande `GRANT` sont :

- Les droits MySQL sont donnés pour une combinaison nom d'utilisateur + nom d'hôte, et non pas pour un nom d'hôte seulement.
- L'ANSI SQL n'a pas de droits globaux ou de niveau base de données, et l'ANSI SQL ne supporte pas tous les types de droits que MySQL supporte. MySQL ne supporte pas le droit ANSI SQL de `TRIGGER` ou `UNDER` .
- Les droits ANSI SQL sont structurés de manière hiérarchique. Si vous supprimez un utilisateur, tous les droits donnés à cet utilisateur seront supprimés. Avec MySQL, les droits ne sont pas automatiquement supprimés, et vous devez les supprimer manuellement, si besoin.
- Avec MySQL, si vous avez le droit de `INSERT` sur uniquement quelques colonnes de la table, vous pourrez exécuter des insertions. Les colonnes pour lesquelles vous n'avez pas de droit prendront alors leur valeur par défaut. L'ANSI SQL vous impose d'avoir les droits d' `INSERT` sur toutes les colonnes.
- Lorsque vous détruisez une table avec ANSI SQL, tous les droits liés à la table sont supprimés. Si vous supprimez un droit en ANSI SQL, tous les droits qui étaient basés sur ce droit sont supprimés. Avec MySQL, les droits peuvent être abandonnés explicitement avec la commande `REVOKE` , ou en manipulant les tables de droits de MySQL.

Pour une description de l'utilisation de `REQUIRE` , voyez Utilisation des connexions sécurisées .

## 4.3.2 Nom d'utilisateurs MySQL et mots de passe

Il y a de nombreuses différences entre les utilisations des noms et mots de passe sous MySQL, et celles qui sont faites sous Unix ou Windows :

- Les noms d'utilisateurs, tels qu'utilisés pour le processus d'identification sous MySQL, n'ont rien à voir avec les noms d'utilisateurs Unix ou Windows. La plupart des clients utilisent par défaut leur mot de passe Unix, mais c'est surtout parce que c'est pratique. Les programmes clients permettent d'utiliser des noms d'utilisateurs différents avec les options `-u` et `--user` . Cela signifie que vous ne pouvez pas rendre une base de données sécuritaire sans donner de mots de passe à tous les clients. Tout le monde peut essayer de se connecter au serveur sous n'importe quel nom, et il sera possible de se connecter si un nom d'utilisateur n'a pas de mot de passe.
- Les noms d'utilisateurs MySQL peuvent avoir jusqu'à 16 caractères ; les noms d'utilisateurs Unix sont généralement limités à 8 caractères.
- Les mots de passe MySQL n'ont aucun rapport avec le passeport Unix. Il n'y a pas nécessairement de connexion entre le mot de passe que vous utilisez pour vous connecter sur la machine Unix et celui que vous utilisez pour accéder au serveur MySQL.
- MySQL chiffre les mots de passe avec un algorithme différent de celui qui est utilisé par Unix. Reportez−vous aux descriptions des fonctions `PASSWORD()` et `ENCRYPT()` dans Fonctions diverses . Notez que même si le mot de passe est enregistré 'brouillé', connaître votre mot de passe 'brouillé' est suffisant pour se connecter au serveur MySQL.

Les utilisateurs MySQL et leurs droits sont généralement créés par la commande `GRANT` . Syntaxe de `GRANT` et `REVOKE` .Lorsque vous vous connectez à un serveur MySQL avec un client en ligne de commande, vous devez spécifier le mot de passe avec l'option `--password=mot-de-passe` . Connexion au serveur MySQL .

```
mysql --user=monty --password=devine nom_base
```

Si vous voulez que le client vous demande le mot de passe à la volée, vous devriez utiliser l'option `--password` sans argument :

```
mysql --user=monty --password nom_base
```

ou la version courte :

```
mysql -u monty -p nom_base
```

Notez que dans le dernier exemple, le mot de passe n'est **pas** 'nom_base'.Si vous voulez utiliser l'option `-p` pour fournir un mot de passe, vous pouvez aussi essayer ceci :

```
mysql -u monty -pdevine nom_base
```

Sur certains systèmes, l'appel système que MySQL utiliser pour demander un mot de passe va automatiquement couper le mot de passe à 8 caractères. En interne, MySQL n'a pas de limite à la taille du mot de passe.

## 4.3.3 Quand les modifications de privilèges prennent−ils effets ?

Lorsque `mysqld` est lancé, toutes les tables de droits sont lues, et sont utilisées.

Les modifications aux tables de droits que vous faites avec `GRANT` , `REVOKE` et `SET PASSWORD` sont immédiatement prises en compte par le serveur.

Si vous modifiez les tables de droits manuellement (avec `INSERT` , `UPDATE` , etc...), vous devez exécuter la commande `FLUSH PRIVILEGES` ou la commande `mysqladmin flush-privileges` , ou encore `mysqladmin reload` pour dire au serveur de relire les tables de droits. Sinon, vos modifications n'auront **aucun effet** jusqu'au redémarrge du serveur. Si vous modifiez les tables de droits manuellement, mais que vous oubliez de recharger les droits, vous vous demanderez sûrement pourquoi vos modifications n'ont pas d'effet.

Lorsque le serveur remarque que les tables de droits ont été modifiées, les connexions existantes avec les clients sont modifiées comme ceci :

- Les droits de table et colonnes prennent effet à la prochaine requête du client.

- Les droits de bases prennent effet à la prochaine commande `USE nom_de_base` .
- Les droits globaux et les modifications de droits prennent effets lors de la prochaine connexion.

## 4.3.4 Création des premiers droits MySQL

Après avoir installé MySQL, vous devez configurer les premiers droits du serveur en exécutant le script `scripts/mysql_install_db` . Vue d'ensemble rapide de l'installation . Le script `mysql_install_db` démarre le serveur `mysqld` et initialise les tables de droits, avec les paramètres suivants :

- L'utilisateur MySQL `root` est créé en tant qu'administrateur avec tous les droits. Les connexions doivent se faire depuis l'hôte local. **Note** : Le mot de passe de l'utilisateur initial `root` est vide, ce qui permet à n'importe qui de se connecter en tant que `root` **sans mot de passe** , pour profiter de tous les droits.
- Un utilisateur anonyme est créé, qui peut faire ce qu'il veut avec toutes les tables dans la base de données `'test'` ou commençant par `'test_'` . Les connexions doivent être faites depuis l'hôte local. Cela signifie qu'un utilisateur local peu se connecter sans mot de passe et être traité comme un utilisateur anonyme.
- Les autres droits sont réservés. Par exemple, un utilisateur normal ne peut pas utiliser les commandes `mysqladmin shutdown` ou `mysqladmin processlist` .

**Note** : les droits par défaut sont différents sous Unix. Faire fonctionner MySQL sur Windows .

Comme votre installation de base n'est pas du tout sécurisée, la première action que vous devez prendre est de spécifier un mot de passe à votre utilisateur `root` . Vous pouvez faire cela comme ceci (notez que vous aurez à spécifier le mot de passe en utilisant la fonction `PASSWORD()` ) :

```
 shell> mysql −u root mysql
mysql> SET PASSWORD FOR root&#64;localhost=PASSWORD('new_password');
```

Si vous savez ce que vous faites, vous pouvez aussi directement manipuler les tables de droits :

```
 shell> mysql −u root mysql
mysql> UPDATE user SET Password=PASSWORD('new_password')
    −>     WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

Une autre méthode pour modifier de mot de passe est d'utiliser la commande en ligne `mysqladmin` :

```
 shell> mysqladmin −u root password new_password
```

Seuls les utilisateurs ayant des droits d'accès en lecture et écriture dans la base de données `mysql` peuvent modifier les mots de passe des autres utilisateurs. Tous les utilisateurs normaux, à l'exception des anonymes, peuvent uniquement modifier leur mot de passe personnel, avec l'une des commandes ci−dessus ou avec la commande `SET PASSWORD=PASSWORD('new password')` .

Notez que si vous modifiez le mot de passe dans la table `user` en utilisant la première méthode, vous devez indiquer au serveur qu'il doit relire les droits avec la commande `FLUSH PRIVILEGES` ), sinon ces modifications ne seront pas prises en compte.

Une fois que le mot de passe `root` a été modifié, vous devrez le fournir pour vous connecter au serveur en tant que `root` .

Vous pouvez aussi laisser le mot de passe du `root` vide, afin de faire des connexions facilement lors de la configuration initiale du serveur. Mais n'oubliez pas de le spécifier lors de la mise en production.

Voyez le script `scripts/mysql_install_db` pour savoir comment il configure les droits par défaut. Vous

pouvez l'utiliser comme base pour ajouter des droits à d'autres utilisateurs.

Si vous voulez que les droits initiaux soient différents, vous pouvez aussi modifier le script `mysql_install_db` avant de l'exécuter.

Pour recréer complètement les tables de droits, effacez tous les fichiers `.frm` , `.MYI` et `.MYD` dans le dossier contenant les tables de la base `mysql` . (C'est le dossier `mysql` dans le dossier de données, qui est indiqué par la commande `mysqld --help` .) Puis, exécutez le script `mysql_install_db` , après l'avoir éventuellement modifié avec les droits que vous souhaitez.

**Note** : pour les versions de MySQL antérieures à la version 3.22.10, il ne faut pas effacer les fichiers `.frm` . Si vous le fait accidentellement, vous devez les copier depuis la distribution MySQL avant d'exécuter le script `mysql_install_db` .

## 4.3.5 Ajouter de nouveaux utilisateurs à MySQL

Vous pouvez ajouter des utilisateurs de deux façons différentes : en utilisant la commande `GRANT` ou en manipulant la table des droits de MySQL directement. La méthode préférée consiste à utiliser la commande `GRANT` , car elle est plus concise et qu'il y a moins de risques d'erreur. Syntaxe de `GRANT` et `REVOKE` .

Il y a aussi beaucoup de programmes contribués comme `phpmyadmin` qui peuvent être utilisés pour créer et administrer les utilisateurs.

Les exemples suivants montrent comment utiliser le client `mysql` pour créer de nouveaux utilisateurs. Ces exemples supposent que les privilèges sont attribués en accord avec les valeurs par défaut discutées dans la section précédente. Cela signifie que pour effectuer des changements, vous devez être sur la même machine où `mysqld` tourne, vous devez vous connecter en tant qu'utilisateur MySQL `root` , et l'utilisateur `root` doit avoir le droit `INSERT` sur la base `mysql` et le droit d'administration `RELOAD` . Si vous avez changé le mot de passe de l'utilisateur `root` , vous devez le spécifier dans les commandes `mysql` ci−dessous.

Vous pouvez ajouter de nouveaux utilisateurs en utilisant des commandes `GRANT` :

```
 shell> mysql --user=root mysql
mysql> GRANT ALL PRIVILEGES ON *.* TO monty&#64;localhost
    ->     IDENTIFIED BY 'un_mot_de_passe' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO monty&#64;"%"
    ->     IDENTIFIED BY 'un_mot_de_passe' WITH GRANT OPTION;
mysql> GRANT RELOAD,PROCESS ON *.* TO admin&#64;localhost;
mysql> GRANT USAGE ON *.* TO dummy&#64;localhost;
```

Ces commandes `GRANT` ajoutent trois nouveaux utilisateurs :

*monty*
: Un super−utilisateur qui peut se connecter au serveur d'où il veut, mais qui doit utiliser le mot de passe `'un_mot_de_passe'` pour le faire. Notez que nous devons exécuter une commande `GRANT` pour `monty@localhost` et `monty@"%"` . Si nous n'ajoutons pas l'entrée avec `localhost` , l'entrée concernant l'utilisateur anonyme pour `localhost` qui est créée par `mysql_install_db` prendra précédence lors de la connexion à partir de l'hôte local, car elle a une entrée plus spécifique pour la valeur du champ `Host` et de plus, elle vient en premier dans l'ordre de tri de la table `user` .

*admin*
: Un utilisateur qui peut se connecter depuis `localhost` sans mot de passe et qui a les droits administratifs `RELOAD` et `PROCESS` . Cela permet à cet utilisateur d'exécuter les commandes `mysqladmin reload` , `mysqladmin refresh` , et `mysqladmin flush−*` , ainsi que `mysqladmin processlist` . Aucun droit lié aux bases de données n'est donné. (Ils peuvent l'êetre plus

tard en utilisant d'autres instructions `GRANT` .)

*dummy*

Un utilisateur qui peut se connecter sans mot de passe, mais seulement à partir de l'hôte local. Les droits globaux sont tous à `'N'` −le type de droit `USAGE` vous permet de créer un utilisateur démuni de privilèges. Il est supposé que vous lui assignerez les droits spécifiques aux bases de données plus tard.

Vous pouvez ajouter les mêmes droits d'accès aux utilisateurs en utilisant directement des requêtes `INSERT` puis en demandant au serveur de recharger les tables de droits :

```
 shell> mysql --user=root mysql
mysql> INSERT INTO user VALUES('localhost','monty',PASSWORD('un_mot_de_passe'),
    ->          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user VALUES('%','monty',PASSWORD('un_mot_de_passe'),
    ->          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user SET Host='localhost',User='admin',
    ->           Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host,User,Password)
    ->                  VALUES('localhost','dummy','');
mysql> FLUSH PRIVILEGES;
```

Selon votre version de MySQL, vous pouvez avoir un nombre différent de valeurs `'Y'` plus haut (les versions antérieures à la 3.22.11 possèdent moins de colonnes de privilèges). Pour l'utilisateur `admin` , la syntaxe d' `INSERT` étendue la plus lisible disponible depuis la version 3.22.11 est utilisée.

Notez que pour ajouter un super−utilisateur, vous avez juste besoin de créer une entrée dans la table `user` avec tous les champs de droits à `'Y'` . Aucune entrée n'est requise dans les tables `db` et `host` .

Les colonnes de privilèges de la table `user` n'étaient pas renseignées explicitement dans la dernière requête `INSERT` (pour l'utilisateur `dummy` ), ses colonnes prennent donc la valeur par défaut, `'N'` . C'est la même chose que ce que fait `GRANT USAGE` .

L'exemple suivant ajoute un utilisateur `custom` qui peut se connecter à partir des hôtes `localhost` , `server.domain` , et `whitehouse.gov` . Il ne pourra accéder à la base de données `bankaccount` qu'à partir de `localhost` , à la base `expenses` qu'à partir de `whitehouse.gov` , et à la base `customer` à partir des trois hôtes. Il utilisera le mot de passe `stupid` pour les trois hôtes.

Pour configurer les privilèges de cet utilisateur en utilisant des commandes `GRANT` , exécutez ce qui suit :

```
 shell> mysql --user=root mysql
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON bankaccount.*
    ->     TO custom&#64;localhost
    ->     IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON expenses.*
    ->     TO custom&#64;whitehouse.gov
    ->     IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON customer.*
    ->     TO custom&#64;'%'
    ->     IDENTIFIED BY 'stupid';
```

La raison pour laquelle nous faisons deux insertions pour l'utilisateur 'custom' est que nous voulons lui donner accès à MySQL à partir de la machine locale avec les sockets Unix et à partir de la machine distante 'whitehouse.gov' via TCP/IP.Pour régler les permissions d'accès en modifiant directement les tables de droits, exécutez ces commandes (notez l'appel à `FLUSH PRIVILEGES` à la fin) :

```
 shell> mysql --user=root mysql
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('localhost','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('server.domain','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('whitehouse.gov','custom',PASSWORD('stupid'));
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES
    -> ('localhost','bankaccount','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
```

```
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES
    -> ('whitehouse.gov','expenses','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES('%','customer','custom','Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;
```

Les trois premières requêtes `INSERT` ajoute les entrées dans la table `user` qui permettent l'iutilisateur `custom` à se connecter à partir de plusieurs hôtes avec le mot de passe donné, mais ne lui donnent aucun droit (tous les droits sont mis à la valeur par défaut qui est `'N'` ). Les trois requêtes `INSERT` suivantes ajoutent des entrées dans la table `db` qui autorisent `custom` à utiliser les bases de données `bankaccount` , `expenses` , et `customer` , mais seulement s'il y accède à partir de l'hôte spécifié. Comme d'habitude, lorsqueles tables de droits sont modifiées directement, on doit demander au serveur des les recharger (avec `FLUSH PRIVILEGES` ) pour que les changements soient pris en compte.Si vous voulez donner un accès spécifique à un utilisateur à partir de n'importe quelle machine d'un domaine donné, vous pouvez utiliser la commande `GRANT` qui suit :

```
 mysql> GRANT ...
    ->     ON *.*
    ->     TO monutilisateur&#64;"%.mondomaine.com"
    ->     IDENTIFIED BY 'monmotdepasse';
```

Pour faire la même chose en modifiant directement la table de droits, faites :

```
 mysql> INSERT INTO user VALUES ('%.mondomaine.com', 'monutilisateur',
    ->             PASSWORD('monmotdepasse'),...);
mysql> FLUSH PRIVILEGES;
```

## 4.3.6 Limiter les ressources utilisateurs

Depuis MySQL 4.0.2, il est possible de limiter certaines ressources accessibles à un utilisateur possible.

Jusqu'à présent, la seule méthode possible pour limiter l'utilisation des ressources serveurs MySQL était de configurer la variable de démarrage `max_user_connections` avec une valeur non nulle. Mais cette méthode est strictement globale, et ne permet pas de faire des aménagements personnalisés, comme cela pourrait arriver aux fournisseurs de services internet.

Par conséquent, un système de gestion des ressources a été introduit, au niveau des utilisateurs :

- Nombre de requête par heure : Toutes les commandes qu'un utilisateur peut exécuter.
- Nombre de modifications par heure : Toute commande qui implique la modification d'une table ou d'une base.
- Nombre de connexion réalisées par heure : Le nombre de nouvelles connexions par heure.

Un utilisateur dans le contexte ci−dessus est une ligne dans la table `user` , qui est identifié de manière unique par les colonnes `user` et `host` .

Par défaut, les utilisateurs ne sont pas limités dans l'utilisation des ressources ci−dessus, à moins que des limites ne leur soient imposées. Ces limites peuvent être configurées **uniquement** via la commande `GRANT (*.*)` , avec cette syntaxe :

```
 GRANT ... WITH MAX_QUERIES_PER_HOUR N1
               MAX_UPDATES_PER_HOUR N2
               MAX_CONNECTIONS_PER_HOUR N3;
```

Il est possible de spécifier toute combinaison de clauses ci−dessus. `N1` , `N2` et `N3` sont des entiers et représentent le compte par heure.

Si l'utilisateur atteint l'une des limites ci−dessus dans une heure, sa connexion sera interrompue ou refusée, et un message d'erreur approprié sera émis.

Les valeurs courantes d'utilisation d'un utilisateur peuvent être remises à zéro avec la commande `GRANT` , et n'importe laquelle des clauses ci−dessus, y compris une commande `GRANT` avec les valeurs courantes.

De plus, les valeurs courantes de tous les utilisateurs peuvent être remises à zéro si les droits sont rechargés (dans le serveur, ou avec la commande `mysqladmin reload` ) ou si la commande SQL `FLUSH USER_RESOURCES` est exécutée.

Cette fonctionnalité est activée aussitôt qu'une limite est donnée à un utilisateur, avec la commande `GRANT` .

Comme pré−requis à l'activation de cette fonctionnalité, la table `user` de la base `mysql` doit contenir les colonnes additionnelles, définies dans le script de création `mysql_install_db` et `mysql_install_db.sh` , du dossier `scripts` .

## 4.3.7 Configurer les mots de passe

Dans la plupart des cas, vous devez utiliser la commande `GRANT` pour configurer les utilisateurs et leur mot de passe. Les explications suivantes s'adressent donc aux utilisateurs avancés. Syntaxe de `GRANT` et `REVOKE` .

Les exemples des sections précédentes illustrent un principe important : lorsque vous stockez un mot de passe important avec les commandes `INSERT` ou `UPDATE` , vous devez utiliser la fonction `PASSWORD()` pour le chiffrer. Ceci est dû au fait que la table `user` stocke les mots de passe sous une forme chiffrée, et non pas en texte clair. Si vous oubliez cela, vous risquez de configurer vos mots de passe comme ceci :

```
 shell> mysql −u root mysql
mysql> INSERT INTO user (Host,User,Password)
    −> VALUES('%','jeffrey','biscuit');
mysql> FLUSH PRIVILEGES;
```

Le résultat est que la valeur `'biscuit'` est stockée dans la colonne de mot de passe de la table `user` . Lorsque l'utilisateur `jeffrey` tente de se connecter au serveur avec ce mot de passe, le client `mysql` chiffre ce mot de passe avec la fonction `PASSWORD()` , génère un vecteur d'identification basé sur la version **chiffrée** du mot de passe et un nombre aléatoire, obtenu du serveur, puis envoie le tout au serveur. Le serveur utilise la valeur du mot de passe `password` dans la table `user` (ce qui **n'est pas la valeur chiffrée** de `'biscuit'` ) pour effectuer la recherche et comparer les résultats. La comparaison échoue et le serveur rejette la connexion :

```
 shell> mysql −u jeffrey −pbiscuit test
Access denied
```

Les mots de passe doivent donc être chiffrés lors de leur insertion dans la table `user` . Aussi les commandes `INSERT` doivent être spécifiées comme ceci :

```
 mysql> INSERT INTO user (Host,User,Password)
    −> VALUES('%','jeffrey',PASSWORD('biscuit'));
```

Vous devez aussi utiliser la fonction `PASSWORD()` lorsque vous utilisez la commande `SET PASSWORD` :

```
 mysql> SET PASSWORD FOR jeffrey&#64;"%" = PASSWORD('biscuit');
```

Si vous modifiez les mots de passe en utilisant la commande `GRANT ... IDENTIFIED BY` ou la commande `mysqladmin password` , la fonction `PASSWORD()` n'est pas nécessaire. Ces commandes assureront le chiffrement de votre mot de passe pour vous, ce qui vous permet de spécifier le mot de passe de `'biscuit'` comme ceci :

```
 mysql> GRANT USAGE ON *.* TO jeffrey&#64;"%" IDENTIFIED BY 'biscuit';
```

or

```
 shell> mysqladmin −u jeffrey password biscuit
```

**Note** : `PASSWORD()` n'effectue pas le chiffrement du mot de passe de la même façon qu'Unix. Il ne faut pas supposer que si vos mots de passe Unix et MySQL sont les mêmes, la fonction `PASSWORD()` retournera la même valeur que celle qui est stockée dans le fichier de mots de passe d'UNIX. Noms

d'utilisateurs MySQL et mots de passe .

## 4.3.8 Garder vos mots de passe en lieu sûr

Il est recommandé de ne pas placer votre mot de passe là où il risque d'être découvert par d'autres personnes. Les méthode que vous utiliserez pour spécifier votre mot de passe lors de la connexion avec le client sont listées ici, avec les risques liés à chaque méthode :

- Ne donnez jamais l'accès à un utilisateur normal, à la table `mysql.user` . En connaissant le mot de passe chiffré d'un utilisateur lui permettrait de se connecter sous cet utilisateur. Les mots de passe sont chiffrés pour que personne ne puisse lire le mot de passe en clair (ce qui arriverait si vous utilisiez le même mot de passe dans une autre application).
- Utilisez l'option `-pyour_pass` ou `--password=your_pass` en ligne de commande. C'est très pratique mais très peu sûr, car votre mot de passe devient visible aux programmes qui effectuent des bilans de l'état du serveur, comme la commande `ps` , qui peut être invoquée par d'autres utilisateurs pour lister les commandes qui sont exécutées : les clients MySQL remplacent la valeur de cet argument par une série de zéro durant l'initialisation, mais il reste tout de même une courte période où ce mot de passe est lisible en clair.
- Utilisez l'option `-p` ou `--password` (sans la valeur du `mot de passe` ). Dans ce cas, le programme client va solliciter la saisie du mot de passe depuis le terminal :
  ```
  shell> mysql -u user_name -p
  Enter password: ********
  ```
  Les caractères `'*'` représentent votre mot de passe.

  Cette méthode est bien plus sûre pour saisir votre mot de passe qu'en le spécifiant directement en ligne de commande, car il n'est pas visible des autres utilisateurs. Cependant, cette méthode n'est possible qu'avec les programmes que vous utilisez en mode interactif. Si vous voulez invoquer le client depuis un script qui s'exécute de manière non interactive, il n'y aura pas d'opportunité pour saisir ce mot de passe dans le terminal. Sur certains systèmes, vous pourriez même voir la première ligne de votre script lue et interprétée comme votre mot de passe, incorrectement.
- Stockez votre mot de passe dans le fichier de configuration. Par exemple, vous pouvez lister votre mot de passe dans la section `[client]` du fichier `.my.cnf` dans votre dossier personnel :
  ```
  [client]
  password=mot_de_passe
  ```
  Si vous stockez ce mot de passe dans le fichier `.my.cnf` , le fichier ne doit pas être lisible par le groupe ou par les autres utilisateurs, ou encore accessible en écriture : seul le propriétaire de ce fichier doit avoir ces droits. Assurez-vous les droits d'accès au fichiers sont `400` ou `600` . Fichier d'options `my.cnf` .
- Vous pouvez stocker votre mot de passe dans la variable d'environnement `MYSQL_PWD` , mais cette méthode doit être considérée comme extrêmement peu sûre, et doit être évitée autant que possible. Certaines versions de la commande en ligne `ps` incluent une option pour afficher les variables d'environnement des processus : votre mot de passe sera alors facilement accessible, et en texte clair, si vous configurez la commande `MYSQL_PWD` . Même sur les systèmes sans une telle version de la commande `ps` , il est peu recommandé de supposer que les variables d'environnement sont inaccessibles par une méthode quelconque. Variables d'environnement MySQL .

En conclusion, la méthode la plus sûre est encore de laisser le client vous demander le mot de passe, ou de le spécifier dans le fichier de configuration `.my.cnf` .

## 4.3.9 Utilisation des connexions sécurisées

### 4.3.9.1 Introduction aux connexions sécurisées

Disponible depuis la version 4.0.0, MySQL supporte les connexions sécurisées. Pour comprendre comment MySQL utilise SSL, il est nécessaire de comprendre les concepts SSL et X509 de base. Ceux qui les connaissent, peuvent aisément sauter ce chapitre.

Par défaut, MySQL utilise une connexion en clair entre le client et le serveur. Cela signifie qu'une personne peut surveiller votre trafic, et lire les données échangées. Cette personne pourrait aussi modifier les données qui transitent entre le client et le serveur. Parfois, vous aurez besoin d'échanger des informations sur un réseau public, mais en sécurisant ces informations. Dans ce cas, utiliser une connexion sans protection est inacceptable.

SSL est un protocole qui utilise différents algorithmes de chiffrement pour s'assurer que les données qui transitent par un réseau public peuvent être considérées comme fiables. Ce protocole dispose de méthodes pour s'assurer que les données n'ont pas été modifiées, ce que soit par une altération, une perte ou une répétition des données. SSL inclut aussi des algorithmes pour reconnaître et fournit des outils de vérifications d'identitée, pris en charge par le standard X509.

Le chiffrement est une méthode pour rendre des données illisibles. En fait, les pratiques actuelles requièrent d'autres éléments de sécurité issus des algorithmes de chiffrement. Ils doivent savoir résister à de nombreux types d'attaque, comme la modification de l'ordre des messages ou les répétitions inopinées.

X509 est un standard qui rend possible l'identification d'une personne sur l'internet. Il est particulièrement utilisé pour les applications e−commerce. En termes simples, il doit y avoir une entreprise (appelée l'``autorité de certification'') qui assigne un certificat électronique à toute personne qui en a besoin. Ces certificats utilisent un chiffrement assymétrique qui exploitent deux clés de chiffrement, une clé publique et une clé privée. Le propriétaire d'un certificat peut prouver son identité en montrant son certificat à l'autre partie. Un certificat est constitué de la clé publique du propriétaire. Toute donnée qui est chiffrée avec cette clé publique doit être déchiffrée avec la clé secrète correspondante, qui est détenue par le propriétaire du certificat.

MySQL n'utilise pas les connexions chiffrées par défaut, car cela ralentit considérablement le protocole de communication. Toute fonctionnalité supplémentaire requiert du travail supplémentaire de la part du serveur, et chiffrer des données est une tâche particulièrement coûteuse, qui peut ralentir considérablement les tâches principales de MySQL. Par défaut, MySQL est paramétré pour être aussi rapide que possible.

Si vous avez besoin de plus d'informations sur SSL, X509 ou le chiffrement, utilisez votre moteur de recherche préféré sur Internet, et utilisez ces mots clés pour avoir plus de détails.

### 4.3.9.2 Pré requis aux connexions sécurisées

Pour faire fonctionner les connexions sécurisées avec MySQL, vous devez disposer de ceci :

- Installation de la librairie d'OpenSSL. Nous avons testé MySQL avec OpenSSL 0.9.6. http://www.openssl.org/ .
- Configuration de MySQL avec les options `--with-vio --with-openssl` .
- Si vous utilisez une ancienne version de MySQL, vous devez modifier votre table `mysql.user` , en lui ajoutant les colonnes reliées aux notions SSL. Vous pouvez faire cela en exécutant le

script `mysql_fix_privilege_tables.sh` . Ceci est nécessaire si vos tables de droits proviennent d'un version de MySQL antérieure à la version 4.0.0.

- Vous pouvez vérifier que vous posséder un serveur `mysqld` qui supporte OpenSSL en examinant le résultat de la commande `SHOW VARIABLES LIKE 'have_openssl'` : elle doit retourner `YES` .

### 4.3.9.3 Configurer les certificats SSL pour MySQL

Voici un exemple de configuration de certificats SSL pour MySQL :

```
 DIR=`pwd`/openssl
PRIV=$DIR/private

 mkdir $DIR $PRIV $DIR/newcerts
cp /usr/share/ssl/openssl.cnf $DIR
replace ./demoCA $DIR -- $DIR/openssl.cnf

 # Créez les dossiers nécessaires : $database, $serial et $new_certs_dir
 # optionnel

 touch $DIR/index.txt
echo "01" > $DIR/serial

 #
# Génération du cerificat d'autorité (CA)
#

 openssl req -new -x509 -keyout $PRIV/cakey.pem -out $DIR/cacert.pem \
    -config $DIR/openssl.cnf

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ................++++++
# .........++++++
# writing new private key to '/home/monty/openssl/private/cakey.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be incorporated
# into your certificate request.
# What you are about to enter is what is called a Distinguished Name or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL admin
# Email Address []:

 #
# Création des clé et requêtes serveur
#
openssl req -new -keyout $DIR/server-key.pem -out \
    $DIR/server-req.pem -days 3600 -config $DIR/openssl.cnf

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ..++++++
# ..........++++++
# writing new private key to '/home/monty/openssl/server-key.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be incorporated
# into your certificate request.
# What you are about to enter is what is called a Distinguished Name or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL server
# Email Address []:
#
 # Please enter the following 'extra' attributes
# to be sent with your certificate request
```

```
# A challenge password []:
# An optional company name []:

 #
# Supprimez la passe-phrase de la clé (optionnel)
#

 openssl rsa -in $DIR/server-key.pem -out $DIR/server-key.pem

 #
# Signez le certificat serveur
#
openssl ca  -policy policy_anything -out $DIR/server-cert.pem \
    -config $DIR/openssl.cnf -infiles $DIR/server-req.pem

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Enter PEM pass phrase:
# Check that the request matches the signature
# Signature ok
# The Subjects Distinguished Name is as follows
# countryName           :PRINTABLE:'FI'
# organizationName      :PRINTABLE:'MySQL AB'
# commonName            :PRINTABLE:'MySQL admin'
# Certificate is to be certified until Sep 13 14:22:46 2003 GMT (365 days)
# Sign the certificate? [y/n]:y
#
 #
 # 1 out of 1 certificate requests certified, commit? [y/n]y
# Write out database with 1 new entries
# Data Base Updated

 #
# Créez les clé et requêtes client
#
openssl req -new -keyout $DIR/client-key.pem -out \
    $DIR/client-req.pem -days 3600 -config $DIR/openssl.cnf

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ....................................++++++
# ..............................................++++++
# writing new private key to '/home/monty/openssl/client-key.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be incorporated
# into your certificate request.
# What you are about to enter is what is called a Distinguished Name or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL user
# Email Address []:
#
 # Please enter the following 'extra' attributes
# to be sent with your certificate request
# A challenge password []:
# An optional company name []:

 #
# Supprimez la passe-phrase de la clé (optionnel)
#
openssl rsa -in $DIR/client-key.pem -out $DIR/client-key.pem

 #
# Signez le cerficat client
#

 openssl ca  -policy policy_anything -out $DIR/client-cert.pem \
    -config $DIR/openssl.cnf -infiles $DIR/client-req.pem

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Enter PEM pass phrase:
# Check that the request matches the signature
# Signature ok
# The Subjects Distinguished Name is as follows
# countryName           :PRINTABLE:'FI'
# organizationName      :PRINTABLE:'MySQL AB'
# commonName            :PRINTABLE:'MySQL user'
# Certificate is to be certified until Sep 13 16:45:17 2003 GMT (365 days)
# Sign the certificate? [y/n]:y
#
 #
```

```
 # 1 out of 1 certificate requests certified, commit? [y/n]y
# Write out database with 1 new entries
# Data Base Updated

 #
# Créez le fichier my.cnf que vous pourrez utiliser pour tester les différents certificats
#

 cnf=""
cnf="$cnf [client]"
cnf="$cnf ssl-ca=$DIR/cacert.pem"
cnf="$cnf ssl-cert=$DIR/client-cert.pem"
cnf="$cnf ssl-key=$DIR/client-key.pem"
cnf="$cnf [mysqld]"
cnf="$cnf ssl-ca=$DIR/cacert.pem"
cnf="$cnf ssl-cert=$DIR/server-cert.pem"
cnf="$cnf ssl-key=$DIR/server-key.pem"
echo $cnf | replace " " '
' > $DIR/my.cnf

 #
# To test MySQL

 mysqld --defaults-file=$DIR/my.cnf &

 mysql --defaults-file=$DIR/my.cnf
```

Vous pouvez aussi tester votre configuration en modificant le fichier `my.cnf` ci−dessuss pour le faire pointer sur les certificats de démonstration dans le dossier mysql−source−distribution/SSL.

### 4.3.9.4 Options de GRANT

MySQL peut vérifier les certificats X509 en plus de la combinaisons habituelle de nom d'utilisateur et mot de passe. Toutes les options habituelles sont toujours nécessaires (nom d'utilisateur, masque d'adresse IP, nom de base de données, nom de table).

Voici différentes possibilités pour limiter les connexions :

- Sans aucune option SSL ou X509, toutes les connexions chiffrées ou non chiffrées sont autorisées si le nom d'utilisateur et le mot de passe sont valides.
- L'option `REQUIRE SSL` requiert que les connexions soient chiffrées avec SSL. Notez que cette option peut être omise si il n'y a pas de ligne ACL qui autorise une connexion sans SSL.
```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret" REQUIRE SSL;
```
- `REQUIRE X509` impose au client d'avoir un certificat valide, mais le certificat lui même est de peu d'importance. La seule restriction est qu'il doit être possible de vérifier la signature avec une des autorités de certification.
```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret" REQUIRE X509;
```
- `REQUIRE ISSUER "issuer"` restreint les tentatives de connexion : le client doit se présenter avec un certificat X509 valide, émis par l' autorité de certification `"issuer"` . Utiliser un certificat X509 implique obligatoirement des chiffrements, donc l'option `SSL` est sous entendue.
```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE ISSUER "C=FI, ST=Some-State, L=Helsinki,
    "> O=MySQL Finland AB, CN=Tonu Samuel/Email=tonu&#64;mysql.com";
```
- `REQUIRE SUBJECT "subject"` impose au client d'avoir un certificat X509 valide, avec le sujet `"subject"` . Si le client présente un certificat valide, mais que le `"subject"` est différent, la connexion est refusée.
```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE SUBJECT "C=EE, ST=Some-State, L=Tallinn,
    "> O=MySQL demo client certificate,
    "> CN=Tonu Samuel/Email=tonu&#64;mysql.com";
```
- `REQUIRE CIPHER "cipher"` est utilisé pour s'assurer que les chiffrements sont suffisamment robuste, et que la bonne longueur de clé est utilisée. SSL lui même peut être faible si des algorithmes sont utilisés avec des clés courtes. En utilisant cette option, il est possible d'imposer la méthode de chiffrement avec la connexion.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
   -> IDENTIFIED BY "goodsecret"
   -> REQUIRE CIPHER "EDH-RSA-DES-CBC3-SHA";
```

Les options `SUBJECT` , `ISSUER` et `CIPHER` peuvent être combinées avec la clause `REQUIRE` comme ceci :

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
   -> IDENTIFIED BY "goodsecret"
   -> REQUIRE SUBJECT "C=EE, ST=Some-State, L=Tallinn,
   "> O=MySQL demo client certificate,
   "> CN=Tonu Samuel/Email=tonu&#64;mysql.com"
   -> AND ISSUER "C=FI, ST=Some-State, L=Helsinki,
   "> O=MySQL Finland AB, CN=Tonu Samuel/Email=tonu&#64;mysql.com"
   -> AND CIPHER "EDH-RSA-DES-CBC3-SHA";
```

Depuis MySQL 4.0.4, le mot clé `AND` est optionnel, entre les options `REQUIRE` .L'ordre de ces options n'a pas d'importance, mais aucune option ne peut être spécifiée deux fois.

# 4.4 Prévention des désastres et restauration

## 4.4.1 Sauvegardes de base de données

Comme les tables MySQL sont stockées sous forme de fichiers, il est facile d'en faire une sauvegarde. Pour avoir une sauvegarde consistante, faites un `LOCK TABLES` sur les tables concernées suivi d'un `FLUSH TABLES` pour celles-ci. Syntaxe des `LOCK TABLES/UNLOCK TABLES` . Syntaxe de `FLUSH` . Vous n'avez besoin que d'un verrou en lecture; cela permet aux autre threads de continuer à effectuer des requêtes sur les tables dont vous faites la copie des fichiers dans le dossier des bases de données. `FLUSH TABLE` est requise pour s'assurer que toutes les pages d'index actifs soient écrits sur le disque avant de commencer la sauvegarde.

Si vous voulez faire une sauvegarde d'une table avec SQL, vous pouvez utiliser `SELECT INTO OUTFILE` ou `BACKUP TABLE` . Syntaxe des `SELECT` . Syntaxe de `BACKUP TABLE` .

Une autre façon de sauvegarder une base de données est d'utiliser l'utilitaire `mysqldump` ou le `script mysqlhotcopy` . `mysqldump` , exporter les structures de tables et les données . `mysqlhotcopy` , copier les bases et tables MySQL .

- Effectuez une sauvegarde complète de votre base de données :

```
shell> mysqldump --tab=/chemin/vers/un/dossier --opt --all

ou

shell> mysqlhotcopy base /chemin/vers/un/dossier
```

  Vous pouvez aussi copier tout simplement tous les fichiers de tables (les fichiers `*.frm` , `*.MYD` , et `*.MYI` ) du moment que le serveur ne met rien à jour. Le script `mysqlhotcopy` utilise cette méthode.
- Arrêtez `mysqld` si il est en marche, puis démarrez le avec l'option `--log-update[=nom_fichier]` . Le log des mises à jour . Le ou les fichiers de log fournissent les informations dont vous avez besoin pour répliquer les modifications de la base de données qui sont subséquents au moment où vous avez exécuté `mysqldump` .

Si vous avez besoin de restaurer quelque chose, essayez d'abord de recouvrer vos tables avec `REPAIR TABLE` ou `myisamchk -r` en premier. Cela devrait fonctionner dans 99.9% des cas. Si `myisamchk` ne réussi pas, essayez la procédure suivante (cela ne fonctionnera que si vous avez démarré MySQL avec `--log-update` , Le log des mises à jour ) :

- Restorez le backup original de `mysqldump` .
- Exécutez la commande suivante pour remettre en marche les mises à jours dans le log binaire :

```
shell> mysqlbinlog hostname-bin.[0-9]* | mysql
```

  Si vous utilisez le journal des mises à jour vous pouvez utiliser :

```
shell> ls -1 -t -r hostname.[0-9]* | xargs cat | mysql
```

`ls` est utilisée pour avoir tous les fichiers de mise à jour dans le bon ordre.

Vous pouvez aussi faire des sauvegardes sélectives avec `SELECT * INTO OUTFILE 'nom_fichier' FROM nom_de_table` et restaurez avec `LOAD DATA INFILE 'nom_fichier' REPLACE ...` Pour éviter les lignes dupliquées, vous aurez besoin d'une `PRIMARY KEY` ou une clef `UNIQUE` dans la table. Le mot clef `REPLACE` fait que les anciens enregistrements sont remplacés par les nouveaux lorsque l'un d'eux duplique un ancien sur une valeur de clef unique.

Si vous obtenez des problèmes de performances sur votre système, vous pouvez les contourner en mettant en place une réplication et faisant les copies sur l'esclave au lieu du maître. Introduction .

Si vous utilisez un système de fichiers Veritas , vous pourrez faire :

- A partir d'un client (ou de Perl), exécutez : `FLUSH TABLES WITH READ LOCK` .
- A partir d'un autre shell, exécutez : `mount vxfs snapshot` .
- Depuis le premier client, exécutez : `UNLOCK TABLES` .
- Copiez les fichiers à partir de snapshot.
- Démontez snapshot.

## 4.4.2 Syntaxe de BACKUP TABLE

```
BACKUP TABLE nom_de_table[,nom_de_table...] TO '/chemin/vers/le/dossier/de/sauvegardes'
```

Cette commande copie le nombre minimal de fichiers de table dont en a besoin pour la restaurer vers le dossier de sauvegardes après avoir rafraîchit les changements dans le disque. Cela ne fonctione actuellement que pour les tables au format `MyISAM` . Pour les tables `MyISAM` , elle ne copie que les fichiers `.frm` (définition) et `.MYD` (données), le fichier d'indes pouvant, lui, être reconstruit à partir des deux autres.

Avant d'utiliser cette commande, merci de lire Sauvegardes de base de données .

Pendant la sauvegarde, un verrou de lecture est posé sur chaque table, une par une, lors de leur copie. Si vous voulez sauvegarder une image instantanée de plusieurs table, vous devez d'abord exécuter un `LOCK TABLES` obtenant un verrou de lecture pour chaque table concernée.

La commande retourne une table avec les colonnes suivantes :

| **Colonne** | **Valeur** |
|---|---|
| Table | Nom de la table |
| Op | Toujours ``backup'' |
| Msg_type | status , error , info ou encore warning . |
| Msg_text | Le message. |

Notez que `BACKUP TABLE` n'est disponible en MySQL que depuis la version 3.23.25.

## 4.4.3 Syntaxe de RESTORE TABLE

```
RESTORE TABLE nom_de_table[,nom_de_table...] FROM '/chemin/vers/le/dossier/de/sauvegardes'
```

Restaure la ou les tables à partir d'une sauvegarde effectuée avec `BACKUP TABLE` . Les tables existantes ne seront pas écrasées et dans ce cas là, vous obtiendrez une erreur. La restauration prendra plus de temps que la sauvegarde à cause de la reconstruction du fichier d'index. Plus vous avez de

clefs, plus la restauration sera longue. Just as BACKUP TABLE , RESTORE TABLE currently works only for MyISAM tables.

Cette commande retourne un tableau avec les colonnes suivantes :

| Colonne | Valeur |
|---|---|
| Table | Nom de la table |
| Op | Toujours ``restore'' |
| Msg_type | status , error , info ou encore warning . |
| Msg_text | Le message. |

## 4.4.4 Syntaxe de CHECK TABLE

```
CHECK TABLE tbl_name[,tbl_name...] [option [option...]]

option = QUICK | FAST | MEDIUM | EXTENDED | CHANGED
```

CHECK TABLE ne fonctionne qu'avec les tables MyISAM et InnoDB . Avec les tables MyISAM , c'est l'équivalent de la commande myisamchk −m table_name sur la table.Par défaut, l'option MEDIUM est utilisée.

Cette commande vérifie l'intégrité des tables. Pour les tables MyISAM , des statistiques importantes sont mises à jour. La commande retourne les infomrations suivantes sur la table dans les colonnes suivantes :

| Colonne | Valeur |
|---|---|
| Table | Nom de la table. |
| Op | Toujours ``check''. |
| Msg_type | Un des statut status , error , info ou warning . |
| Msg_text | Le message. |

Notez que vous pouvez obtenir de nombreuses lignes d'informations pour chaque table. La dernière ligne sera du type Msg_type status et doit être normalement OK . Si vous n'obtenez pas de statut OK ou Not checked , il vous faudra exécuter une réparation de la table. Utilisation de myisamchk pour maintenir les tables et recouvrir les données . Not checked signifie que la table a indiqué qu'il n'y a pas de vérification à faire.

Les différents types de vérifications sont les suivants :

| Type | Signification |
|---|---|
| QUICK | N'analyse pas les lignes pour vérifier les liens erronés. |
| FAST | Ne vérifie que les tables qui n'ont pas été correctement fermées. |
| CHANGED | Ne vérifie que les tables qui ont changées depuis la dernière vérification, ou bien qui n'ont pas été correctement fermées. |
| MEDIUM | Analyse les lignes pour s'assurer que les liens effacés sont corrects. Cette option calcule aussi la somme de contrôle des lignes, et la vérifie avec la somme de contrôle des clés. |
| EXTENDED | Fait une vérification complète des liens pour chaque ligne. Cela vérifie que la table est totalement cohérente, mais cela peut prendre beaucoup de temps. |

Pour les tables à format de dynamique de type MyISAM , une vérifiation de table sera toujours démarrée avec une option de niveau MEDIUM . Pour les tables à format de ligne statique, nous évitons les niveaux de QUICK et FAST car les lignes sont rarement corrompues.

Vous pouvez combiner les options de vérification comme ceci :

```
CHECK TABLE test_table FAST QUICK;
```

L'exemple ci−dessus va simplement faire une vérification de la table, pour s'assurer qu'elle a été correctement fermée.

**Note** : dans certains cas, CHECK TABLE va modifier la table! Cela arrive si la table a été marquée comme 'corrupted' et 'not closed properly' mais CHECK TABLE n'a trouvé aucun problème dans la table. Dans ce

cas, `CHECK TABLE` va marquer la table comme correcte.

Si une table est corrompue, il est probable que les problèmes sont dans les fichiers d'index et non pas dans les données. Tous les types de vérifications présentés ci−dessus vérifient les index soigneusement, et ils devraient trouver la plupart des erreurs.

Si vous voulez simplement vérifier une table que vous supposez correcte, vous pouvez n'utiliser aucune option, ou l'option `QUICK` . Cette dernière peut aussi être utilisée si vous êtes pressé, et que vous pouvez prendre le risque minime que `QUICK` ne trouve pas d'erreur dans votre fichier. Dans la plupart des cas, MySQL doit trouver toutes les erreurs de données, pour un usage normal. Si cela arrive, alors la table est marquée comme 'corrupted', auquel cas, la table ne pourra pas être utilisée tant qu'elle n'a pas été réparée).

`FAST` et `CHANGED` sont surtout destinées à être utilisées depuis un script : par exemple, il peut être exécuté depuis une tâche `cron` , si vous voulez vérifier la table de temps en temps. Dans la plupart des cas, l'option `FAST` doit être préférée à `CHANGED` : le seul cas ou vous pourriez préferez `CHANGED` est lorsque vous soupsonnez avoir trouvé un bogue dans les tables `MyISAM` .

`EXTENDED` ne doit être utilisé qu'après une vérification normale, et que vous obtenez toujours des erreurs étranges lorsque MySQL essaie de modifier une ligne ou trouve une ligne avec clé (ce qui est très rare, si une vérification a réussie).

Certains problèmes rapportés par la commande `CHECK TABLE` , ne peuvent être corrigés automatiquement :

- `Found row where the auto_increment column has the value 0` .Cela signifie que vous avez dans votre table une ligne qui contient la valeur 0 alors qu'elle est de type `AUTO_INCREMENT` . (Il est possible de créer une ligne où la colonne `AUTO_INCREMENT` vaut 0 en spécifiant explicitement la valeur 0 dans la colonne avec la commande `UPDATE` .

  Ce n'est pas une erreur en soit, mais cela peut poser des problèmes si vous décidez de sauver cette table dans un fichier texte, et de la restaurer, ou encore d'appliquer la commande `ALTER TABLE` sur la table. Dans ce cas, la colonne `AUTO_INCREMENT` va changer automatiquement de valeur, en suivant les règles des colonnes de type `AUTO_INCREMENT` , qui vont causer un problème de clé doublon.

  Pour se débarasser de cette alerte, vous devez utiliser une commande `UPDATE` surla table, pour mettre une valeur différente de 0 dans cette colonne.

## 4.4.5 Syntaxe de REPAIR TABLE

```
REPAIR TABLE tbl_name[,tbl_name...] [QUICK] [EXTENDED] [USE_FRM]
```

`REPAIR TABLE` ne fonctionne que les tables de type `MyISAM` , et c'est l'équivalent de la commande en ligne `myisamchk −r table_name` .Normalement, vous n'avez pas à exécuter cette commande, mais si une catastrophe vous frappe, vous êtes presque assurés de retrouver vos données dans les tables MyISAM, avec la commande `REPAIR TABLE` . Si vos tables sont souvent corrompues, vous devrie toutefois rechercher la cause de ce problème! Que faire si MySQL crashe constamment . Problèmes avec les tables `MyISAM` .

`REPAIR TABLE` répare autant que possible les tables corrompues. La commande retourne la table suivante :

| Colonne | Valeur |
|---|---|
| Table | Nom de la table |
| Op | Toujours ``repair'' |
| Msg_type | Un des statut status , error , info ou warning . |
| Msg_text | Le message. |

Notez que vous pourriez obtenir de nombreux messages pour chaque table. La dernière ligne doit être du format `Msg_type status` et doit être normalement `OK` . Si vous n'obtenez pas `OK` , vous devez essayer de réparer votre table avec la commande `myisamchk -o` , car `REPAIR TABLE` de supporte pas encore toutes les options de `myisamchk` . Dans un futur proche, nous allons rendre cette commande encore plus souple.

Si l'option `QUICK` est fournie, alors MySQL va essayer de ne réparer que le fichier d'index.

Si vous utilisez l'option `EXTENDED` , alors MySQL va essayer de créer l'index ligne par ligne, au lieu de créer un index à la fois, par tri. C'est une méthode qui peut s'avérer plus efficace que de trier sur des clés de taille fixe, si vous avez des clés `CHAR` longues qui se compressent bien. Ce type de réparation est l'équivalent de `myisamchk --safe-recover` .

Depuis `MySQL` 4.0.2, il existe le mode `USE_FRM` pour `REPAIR` . Utilisez-le si le fichier `.MYI` manque, ou si son entête est corrompu. Avec ce mode, MySQL va recréer la table, en utilisant les informations dans le fichier `.frm` . Ce type de réparation ne peut pas être fait avec `myisamchk` .

## 4.4.6 Utilisation de myisamchk pour la maintenance des tables et leur recouvrement

A partir de la version 3.23.13 de MySQL, vous pouvez vérifier vos tables MyISAM avec la commande `CHECK TABLE` . Syntaxe de `CHECK TABLE` . Vous pouvez les réparer avec la commande `REPAIR TABLE` . Syntaxe de `REPAIR TABLE` .

Pour vérifier/réparer les tables MyISAM ( `.MYI` et `.MYD` ) vous devez utiliser l'utilitaire `myisamchk` . Pour vérifier/réparer les tables ISAM ( `.ISM` et `.ISD` ) vous devez utiliser l'utilitaire `isamchk` . Types de tables MySQL .

Dans ce qui suit, nous allons parler de `myisamchk` , mais tout s'applique aussi à l'ancien `isamchk` .

Vous pouvez utiliser l'utilitaire `myisamchk` pour obtenir des informations à propos des tables de la base de données, les vérifier et réparer, ou les optimiser. La section suivante décrit comment invoquer `myisamchk` (incluant une description de ses options), comment mettre en place un planificateur de maintenance, et comment utiliser les différentes fonctionnalités de `myisamchk` .

Vous pouvez, dans la plupart des cas, utiliser la commande `OPTIMIZE TABLES` pour optimiser et réparer les tables, mais ce n'est ni aussi rapide, ni aussi sûr (en cas d'erreurs fatales réelles) que `myisamchk` . D'un autre côté, `OPTIMIZE TABLE` est plus facile à utiliser et vous n'avez pas besoin de vous occuper de libérer les tables. Syntaxe de `OPTIMIZE TABLE` .

Même si la réparation de `myisamchk` est assez sécurisée, il est toujours bon de créer une sauvegarde **avant** d'effectuer les réparations (ou quoi que ce soit qui puisse entraîner de grosses modifications sur la table)

### 4.4.6.1 Syntaxe de l'utilitaire myisamchk

`myisamchk` s'exécute avec une commande de la forme :

```
shell> myisamchk [options] tbl_name
```

Les `options` spécifient ce que vous voulez que `myisamchk` fasse. Elles sont décrites dans ce chapitre. Vous pouvez aussi obtenir une liste d'options en invoquant le programme avec `myisamchk --help` . Sans option, `myisamchk` va simplement vérifier les tables. Pour obtenir plus d'information ou pour demander à `myisamchk` de prendre des mesures correctives, il faut ajouter l'une des options listées ici. `tbl_name` est la table que vous voulez réparer ou vérifier. Si vous exécutez `myisamchk` autre part que dans le dossier de données, vous devez spécifier le chemin jusqu'au fichier, car sinon, `myisamchk` n'aura aucune idée d'où chercher les données dans votre base. En fait, `myisamchk` ne se préoccupe pas du fait que le fichier que vous utilisez est dans le dossier de base ou pas : vous pouvez copier le fichier à réparer dans un autre dossier, et y faire les opérations d'entretien.

Vous pouvez spécifier plusieurs noms de tables à `myisamchk` si vous le voulez. Vous pouvez aussi spécifier un nom sous la forme d'un fichier d'index (avec l'option `.MYI` ), qui vous permettra de spécifier toutes les tables dans un dossier en utilisant le schéma `*.MYI` . Par exemple, si vous êtes dans le dossier de données, vous pouvez spécifier toutes les tables dans le dossier comme ceci :

```
shell> myisamchk *.MYI
```

Si vous n'êtes pas dans le dossier de données, et que vous souhaitez vérifier toutes les tables, vous devez ajouter le chemin jusqu'au dossier :

```
shell> myisamchk /path/to/database_dir/*.MYI
```

Vous pouvez même vérifier toutes les tables de toutes les bases avec le chemin suivant :

```
shell> myisamchk /path/to/datadir/*/*.MYI
```

La méthode recommandée pour vérifier rapidement toutes les tables est :

```
 myisamchk --silent --fast /path/to/datadir/*/*.MYI
isamchk --silent /path/to/datadir/*/*.ISM
```

Si vous voulez vérifier toutes les tables et réparer celles qui sont corrompues, vous pouvez utiliser la ligne suivante :

```
 myisamchk --silent --force --fast --update-state -O key_buffer=64M \
          -O sort_buffer=64M -O read_buffer=1M -O write_buffer=1M \
          /path/to/datadir/*/*.MYI
isamchk --silent --force -O key_buffer=64M -O sort_buffer=64M \
        -O read_buffer=1M -O write_buffer=1M /path/to/datadir/*/*.ISM
```

La commande ci−dessus suppose que vous avez plus de 64 Mo de libres.

Notez que si vous obtenez une erreur comme celle−ci :

```
myisamchk: warning: 1 clients is using or hasn't closed the table properly
```

Cela signifie que vous essayez de vérifier une table qui a été modifiée par un autre programme (comme le serveur `mysqld` ) qui n'a pas encore refermé le fichier de table, ou que le fichier n'a pas été correctement refermé.

Si `mysqld` fonctionne, vous devez forcer la fermeture correcte des fichiers de tables avec la commande `FLUSH TABLES` , et vous assurer que personne n'utilise les tables durant vos opérations avec `myisamchk` . En MySQL version 3.23, la meilleure méthode pour éviter ce problème est d'utiliser la commande `CHECK TABLE` au lieu de `myisamchk` pour vérifier les tables.

### 4.4.6.2 Options générales de myisamchk

`myisamchk` supporte les options suivantes :

*−# or −−debug=debug_options*
: Affiche le log de débogage. La chaîne `debug_options` vaut souvent : `'d:t:o,filename'` .

*−? or −−help*
: Affiche le message d'aide, et termine le programme.

*−O var=option, −−set−variable var=option*
: Spécifie la valeur d'une variable. Notez bien que `--set-variable` est obsolète depuis MySQL 4.0, il suffit désormais d'utiliser `--var=option` . Les variables disponibles et leur valeurs par défaut sont accessibles avec la commande `myisamchk --help` :

| Variable | Valeur |
|---|---|
| key_buffer_size | 523264 |
| read_buffer_size | 262136 |
| write_buffer_size | 262136 |
| sort_buffer_size | 2097144 |
| sort_key_blocks | 16 |
| decode_bits | 9 |

`sort_buffer_size` est utilisée pour la réparation de clés, en les triant, ce qui est le cas par défaut avec l'option `--recover` .

`key_buffer_size` est utilisée lorsque vous vérifiez une table avec l'option `--extended-check` ou lorsque les clés sont réparées par insertion des clés ligne par ligne (comme lors des insertions normales). La réparation via le buffer de clé est utilisée dans les situations suivantes :

Si vous utilisez l'option `--safe-recover` .
Si les fichiers temporaires où il faut trier les clés, seront deux fois plus grands que lors de la création direct du fichier de clé. C'est souvent le cas lorsque vous avez de grandes colonnes de type `CHAR` , `VARCHAR` ou `TEXT` car le tri doit stocker la valeur totale de la colonne durant l'opération. Si vous avez beaucoup d'espace disque temporaire, et que vous pouvez forcer `myisamchk` à réparer avec la méthode de tri, vous pouvez utiliser l'option `--sort-recover` .

La réparation avec la méthode du buffer de clé prend bien moins d'espace disque que d'utiliser le tri, mais c'est aussi bien plus lent.

Si vous voulez une réparation plus rapide, donnez à la variable ci−dessus la valeur du quart de votre mémoire disponible. Vous pouvez donner de grandes valeurs aux deux variables, car seul un des deux tampons décrits sera utilisé.

*−s or −−silent*
: Mode silencieux. N'affiche que les erreurs. Vous pouvez utiliser deux fois cette option ( `-ss` ) pour rendre `myisamchk` très silencieux.

*−v or −−verbose*
: Mode détaillé. Affiche beaucoup de détails sur les opérations en cours. Vous pouvez l'utiliser avec les options `-d` et `-e` . Utilisez `-v` plusieurs fois pour avoir encore plus d'informations ( `-vv` , `-vvv` )!

*−V or −−version*
: Affiche la version de `myisamchk` et s'arrête.

*−w or, −−wait*
: Au lieu d'afficher une erreur si la table est verrouillée, attend jusqu'à ce que la table soit libre avant de continuer. Notez que si vous utilisez `mysqld` avec l'option `--skip-external-locking` , la table ne pourra être verrouillée que par une autre commande `myisamchk` .

### 4.4.6.3 Options de vérifications pour myisamchk

*−c ou −−check*
: Vérifie les erreurs d'une table. Ceci est l'opération par défaut de `myisamchk` si vous ne lui donnez aucune autre option.

*−e ou −−extend−check*
: Vérifie la table minutieusement (ce qui est un peu lent si vous avez des index). Cette option ne doit être utilisée que pour les cas extrèmes. Normalement, `myisamchk` ou `myisamchk −−medium−check` devrait, dans la plupart des cas, être capable de trouver s'il y a des erreurs dans la table.

  Si vous utilisez `−−extended−check` et que vous avez beaucoup de mémoire, vous devez augmenter de beaucoup la valeur de `key_buffer_size` !

*−F ou −−fast*
: Ne vérifie que les tables qui n'ont pas été fermées proprement.

*−C ou −−check−only−changed*
: Ne vérifie que les tables qui ont changé depuis la dernière vérification.

*−f ou −−force*
: Redémarrez `myisamchk` avec `−r` (répare) sur la table, si `myisamchk` trouve une erreur dans la table.

*−i ou −−information*
: Affiche des statistiques à propos de la table vérifiée.

*−m ou −−medium−check*
: Plus rapide que extended−check, mais ne trouve que 99.99% des erreurs. Devrait, cependant, être bon pour la plupart des cas.

*−U ou −−update−state*
: Enregistre dans le fichier `.MYI` lorsque la table à été vérifiée ou a crashé. Cela devrait être utilisé pour tirer tous les avantages de l'option `−−check−only−changed` , mais vous ne devez pas utiliser cette option si le serveur `mysqld` utilise cette table et que vous utilisez `mysqld` avec `−−skip−external−locking` .

*−T ou −−read−only*
: Ne marque pas la table comme vérifiée. C'est pratique si vous utilisez `myisamchk` pour vérifier une table issue d'une autre application qui n'utilise pas les verrous. (comme `mysqld −−skip−external−locking` ).

### 4.4.6.4 Options de réparation de myisamchk

Les options suivantes sont utilisées avec `myisamchk` et l'option de réparation `−r` ou `−o` :

*−D # ou −−data−file−length=#*
: Taille maximale du fichier de données (lors de la recréation du fichier de données, et qu'il est 'complet').

*−e ou −−extend−check*
: Essaie de retrouver toutes les lignes possibles du fichier de données. Normalement, cette option va aussi découvrir beaucoup de ligne erronées. N'utilisez pas cette option si vous n'êtes pas totalement desespéré.

*−f ou −−force*
: Ecrase les anciens fichiers temporaires ( `table_name.TMD` ) au lieu d'annuler.

*−k # ou keys−used=#*
: Si vous utilisez les tables ISAM, indique au gestionnaire de table ISAM qu'il doit uniquement modifier les `#` premiers index. Si vous utilisez le gestionnaire de table `MyISAM` , cette option indique quelles clés utiliser, et chaque bit binaire représente une clé (la première clé est le bit 0). Cela permet de réaliser des insertions plus rapides.

Les index désactivés pourront être réactivés avec l'option `myisamchk -r` .

*-l ou --no-symlinks*
: Ne pas suivre les lignes symboliques. Normalement, `myisamchk` répare les tables qu'un lien symbolique représente. Cette option n'existe pas en MySQL 4.0, car MySQL 4.0 ne va pas supprimer les liens symboliques durant la réparation.

*-r ou --recover*
: Peut réparer presque tout, sauf les clés uniques qui ne le sont plus (ce qui est extrêmement rare avec les tables ISAM/MyISAM). Si vous voulez restaurer un table, c'est l'option à utiliser en premier. Si myisamchk indique que la table ne peut pas être corrigée avec l'option `-r` , vous pouvez alors passer à l'option `-o` . Notez que dans le cas rarissime où `-r` , le fichier de données est toujours intact. Si vous avez beaucoup de mémoire, vous pouvez augmenter la taille du buffer `sort_buffer_size` !

*-o ou --safe-recover*
: Utilise une ancienne méthode de restauration (lit toutes les lignes dans l'ordre, et modifie l'arbre d'index conformément pour les lignes trouvées). C'est une méthode qui est beaucoup plus lente que l'option `-r` , mais elle est capable de traiter certaines situations exceptionnelles que `-r` ne pourrait pas traiter. Cette méthode utilise aussi moins d'espace disque que `-r` . Normalement, vous devriez commencer à réparer avec l'option `-r` , et uniquement sur l'échec de cette option, passer à `-o` .Si vous avez beaucoup de mémoire, vous devriez augmenter la taille du buffer de clé ! `key_buffer_size` !

*-n ou --sort-recover*
: Force `myisamchk` à utiliser le tri pour résoudre les clés, même si le fichier temporaire doit être énorme.

*--character-sets-dir=...*
: Dossier qui contient les jeux de caractères.

*--set-character-set=name*
: Change le jeu de caractères utilisé par l'index.

*-t ou --tmpdir=path*
: Chemin pour stocker les fichiers temporaires. Si cette option n'est pas fournie, `myisamchk` va utiliser la variable d'environnement `TMPDIR` pour cela.

*-q ou --quick*
: Réparation rapide, sans modifier le fichier de données. Il est possible d'ajouter l'option `-q` pour forcer `myisamchk` à modifier le fichier original en cas de clés doublons.

*-u ou --unpack*
: Décompresse des données compressées avec myisampack.

### 4.4.6.5 Autres options de myisamchk

Les autres actions que `myisamchk` peut réaliser, en dehors de vérifier et réparer une table sont :

*-a or --analyze*
: Analyser la distribution des clés. Cela améliore les performances des jointures en permettant à l'optimiseur de jointure de mieux choisir l'ordre d'utilisation des clés. `myisamchk --describe --verbose table_name'` ou `SHOW KEYS` dans MySQL.

*-d or --description*
: Affiche des informations sur la table.

*-A or --set-auto-increment[=value]*
: Force `AUTO_INCREMENT` à commencer avec une valeur supérieure. Si aucune valeur n'est fournie, la prochaien valeur de la colonne `AUTO_INCREMENT` sera la plus grande valeur de la colonne +1.

*-S or --sort-index*
: Trie les blocs de l'arbre d'index dans l'ordre haut-bas. Cela va optimiser les

recherches, et les scans de tables par clés.

*−R or −−sort−records=#*

Trie les lignes en fonction de l'index. Cela rassemble vos données, et peut accélérer les lectures de lignes par intervalle avec `SELECT` et `ORDER BY` sur cet index (ce tri peut être TRES lent la première fois). Pour connaitre les numéros d'index de tables, utilisez la commande `SHOW INDEX` , qui affiche les index dans le même ordre que `myisamchk` ne les voit. Les index sont numérotés à partir de 1.

### 4.4.6.6 Utilisation de la mémoire par myisamchk

L'espace mémoire est très important quand vous utilisez `myisamchk` . `myisamchk` n'utilise pas plus de mémoire que ce que vous spécifiez avec les options `−O` . Si vous pensez utiliser `myisamchk` sur des fichiers très grands, vous devez d'abord décider la quantité de mémoire que vous souhaitez utiliser. Avec des valeurs plus grandes, vous pouvez accélérer `myisamchk` . Par exemple, si vous avez plus de 32 Mo de RAM, vous pourriez utiliser les options suivantes (en plus des autres options que vous pourriez spécifier) :

```
shell> myisamchk −O sort=16M −O key=16M −O read=1M −O write=1M ...
```

Utiliser `−O sort=16M` sera probablement suffisant pour la plupart des cas.Soyez conscient que `myisamchk` utilise des fichiers temporaires dans le dossier `TMPDIR` . Si `TMPDIR` est un fichier en mémoire, vous pourriez facilement rencontrer des erreurs de mémoire. Si cela arrive, choisissez une autre valeur pour `TMPDIR` , avec plus d'espace disque, et redémarrez `myisamchk` .

Lors de la réparation, `myisamchk` va aussi avoir besoin d'espace disque :

- Doublez la taille du fichier de données (l'original plus une copie). Cet espace n'est pas nécessaire si vous faites des réparations de type `−−quick` , car dans ce cas, seul le fichier d'index sera recréé. Cet espace est nécessaire sur le même disque que l'original !
- De l'espace pour le nouveau fichier d'index qui remplacera l'ancien. L'ancien fichier d'index est réduit dès le démarrage, ce qui vous permet généralement d'ignorer cet espace. Cet espace est nécessaire sur le même disque que l'original !
- Lorsque vous utilisez les options `−−recover` ou `−−sort−recover` (mais pas lorsque vous utilisez `−−safe−recover` ), vous aurez besoin d'espace pour le buffer de tri : `(plus_grande_cle + taille_du_pointeur_de_ligne)*nombre_de_lignes * 2` . Vous pouvez vérifier la taille des clés et la taille du pointeur de ligne avec la commande `myisamchk −dv table` . Cet espace est alloué sur le disque temporaire (spécifié par `TMPDIR` par `−−tmpdir=#` ).

Si vous avez des problèmes avec l'espace disque durant la réparation, vous pouvez utiliser l'option `−−safe−recover` au lieu de `−−recover` .

### 4.4.6.7 Utiliser myisamchk pour restaurer une table

Si vous utilisez `mysqld` avec l'option `−−skip−external−locking` (qui est la configuration par défaut pour certains systèmes, comme Linux), vous ne pouvez pas utiliser `myisamchk` pour vérifier une table, lorsque `mysqld` utilise aussi la table. Si vous pouvez être sûr que personne n'utilise cette table via `mysqld` lorsque vous utilisez `myisamchk` , vous n'aurez qu'à utiliser la commande `mysqladmin flush−tables` avant de commencer à vérifier les tables. Si vous ne pouvez pas garantir cette condition, vous devez alors éteindre le serveur `mysqld` pour vérifier les tables. Si vous exécutez `myisamchk` alors que `mysqld` modifie la table, vous pourriez obtenir un diagnostic de corruption de la table, alors que ce n'est pas le cas.

Si vous n'utilisez pas l'option `−−skip−external−locking` , vous pouvez vous servir de `myisamchk` pour vérifier les tables à tout moment. Pendant que vous le faites, les autres clients qui tentent de modifier la table devront attendre que `myisamchk` ait fini.

Si vous utilisez `myisamchk` pour réparer ou optimiser les tables, vous **devez** toujours vous assurer que `mysqld` n'utilise pas cette table (ce qui s'applique aussi si vous utilisez `--skip-external-locking` ). Si vous n'éteignez pas le serveur `mysqld` , vous devez au moins utiliser `mysqladmin flush-tables` avant de lancer `myisamchk` . Vos tables **peuvent être corrompues** si le serveur et `myisamchk` travaillent dans une même table simultanément.

Ce chapitre décrit comment vérifier et gérer les corruptions de données dans les bases MySQL. Si vos tables sont fréquemment corrompues, vous devriez commencer par en rechercher la raison ! Que faire si MySQL crashe constamment .

La section sur les tables `MyISAM` contient différentes raisons pour lesquelles une table peut être corrompue. Problèmes avec les tables `MyISAM` .

Lorsque vous effectuez une restauration de table, il est important que chaque table `tbl_name` dans une base corresponde aux trois fichiers dans le dossier de base, du dossier de données :

| Fichier | Utilisation |
|---|---|
| tbl_name.frm | Définition de la table |
| tbl_name.MYD | Fichier de données |
| tbl_name.MYI | Fichier d'index |

Chacun de ces trois fichiers est sujet à des corruptions diverses, mais les problèmes surviennent généralement dans les fichiers de données ou d'index.

`myisamchk` fonctionne en créant une copie du fichier `.MYD` (les données), ligne par ligne. Il termine sa réparation en supprimant l'ancien fichier `.MYD` et en renommant le nouveau à la place de l'ancien. Si vous utilisez l'option `--quick` , `myisamchk` ne crée pas de fichier temporaire `.MYD` mais suppose plutôt que le fichier `.MYD` est correct et il génère simplement un nouveau fichier d'index sans toucher au fichier `.MYD` . C'est une méthode sécuritaire, car `myisamchk` va automatiquement détecter si le fichier `.MYD` est corrompu, et annulera alors la réparation si c'est le cas. Vous pouvez aussi ajouter deux options `--quick` à `myisamchk` . Dans ce cas, `myisamchk` ne s'interrompt pas sur certaines erreurs (comme des clés doublons), et essaie de résoudre ce problème en modifiant le fichier `.MYD` . Normalement, l'utilisation de deux options `--quick` n'est utile que si vous n'avez pas trop d'espace disque pour réaliser la réparation. Dans ce cas, vous devez au moins faire une copie de sauvegarde avant d'utiliser `myisamchk` .

### 4.4.6.8 Comment vérifier la cohérence d'une table

Pour vérifier les tables de type MyISAM, utilisez les commandes suivantes :

*myisamchk nom_de_table*
: Cette commande trouvera 99.99% de toutes les erreurs. Ce qu'elle ne peut pas découvrir comme erreurs, sont celles qui impliquent **uniquement** le fichier de données (ce qui est très inhabituel). Si vous voulez vérifier une table, vous devriez utiliser l'utilitaire `myisamchk` sans les options ou avec les options `-s` ou `--silent` .

*myisamchk -m nom_de_table*
: Cette commande trouvera 99.999% de toutes les erreurs. Elle vérifie toutes les entrées dans le fichier d'index, puis lit toutes les lignes. Elle calcule une somme de contrôle pour toutes les clés et les lignes, et vérifie que les deux se correspondent dans l'arbre d'index.

*myisamchk -e nom_de_table*
: Cette commande fait une vérification complète et exhaustive de toutes les données ( `-e` signifie ``extended check''). Elle fait une lecture de contrôle de chaque ligne, pour vérifier qu'elle correspond bien aux index. Cette commande va prendre un long moment sur les grosses tables. `myisamchk` va normalement s'arrêter dès qu'il trouve une

erreur. Si vous voulez obtenir plus d'information sur cette erreur, vous pouvez utiliser l'option `--verbose` (ou `-v` ). Cela fera que `myisamchk` va continuer à travailler et accumuler jusqu'à 20 erreurs. En utilisation normale, l'utilisation de cet utilitaire sans options est suffisante.

*myisamchk −e −i nom_de_table*

Comme les commandes précédentes, mais l'option `-i` indique à `myisamchk` qu'il doit afficher des informations statistiques.

### 4.4.6.9 Comment réparer des tables

Dans la section présente, nous allons uniquement parler de l'utilitaire `myisamchk` sur les tables `MyISAM` (extensions `.MYI` et `.MYD` ). Si vous utilisez les tables `ISAM` (extensions `.ISM` et `.ISD` ), vous devriez vous servir de `isamchk` à la place.

Depuis MySQL version 3.23.14, vous pouvez réparer les tables MyISAM avec la commande SQL `REPAIR TABLE` . Syntaxe de `REPAIR TABLE` .

Les symptômes de corruption de tables sont des requêtes qui s'interrompent inopinément :

- `tbl_name.frm` locked against change : `tbl_name.frm` est verrouillée en écriture
- Can't find file `tbl_name.MYI` (Errcode: ###) : Impossible de trouver le fichier `tbl_name.MYI` (Errcode: ###)
- Unexpected end of file : Fin de fichier inattendue
- Record file is crashed : Fichier de données crashé
- Got error ### from table handler : Reception de l'erreur ### de la part du gestionnaire de tablePour obtenir plus d'informations sur l'erreur, vous pouvez exécuter la commande `perror ###` . Voici les erreurs les plus courantes :

```
 shell> perror 126 127 132 134 135 136 141 144 145
126 = Index file is crashed / Wrong file format : le fichier d'index est corrompu / le format du fichier est incorrect.
127 = Record-file is crashed : le fichier de données est corrompu.
132 = Old database file / ce fichier provient d'une vieille base de données.
134 = Record was already deleted (or record file crashed) / La ligne était déjà effacée.
135 = No more room in record file / Plus de place dans le fichier de données.
136 = No more room in index file / Plus de place dans le fichier d'index.
141 = Duplicate unique key or constraint on write or update / Doublon pour une clé unique trouvé durant la lecture ou l'écriture.
144 = Table is crashed and last repair failed / la table est corrompue et la dernière réparation a échoué.
145 = Table was marked as crashed and should be repaired / La table a été marquée comme corrompue et doit être réparée.
```

  Notez que l'erreur 135, "no more room in record file", n'est pas une erreur qui sera facile à corriger. Dans ce cas, vous devez utiliser la commande suivante :

```
 ALTER TABLE table MAX_ROWS=xxx AVG_ROW_LENGTH=yyy;
```

Dans d'autres cas, vous devrez réparer vos tables. `myisamchk` peut généralement détecter et corriger la plupart des erreurs.

Le processus de réparation se déroule en 4 étapes décrites ici. Avant de vous lancer, vous devriez vous placer dans le dossier de données et vérifier les permissions des fichiers de données. Assurez−vous qu'ils sont bien lisibles par l'utilisateur Unix que MySQL utilise (et à vous aussi, car vous aurez besoin d'accéder à ces fichiers durant la vérification. Si vous devez corriger ces fichiers, vous aurez aussi besoin des droits d'écriture.

Si vous utilisez MySQL version 3.23.16 et plus récent, vous pouvez (et vous devriez) utiliser les commandes `CHECK` et `REPAIR` pour réparer vos tables `MyISAM` . Syntaxe de `CHECK TABLE` . Syntaxe de `REPAIR TABLE` .

La section du manuel sur l'entretien des tables inclut la présentation des options des utilitaires `isamchk` / `myisamchk` . Utilisation de `myisamchk` pour maintenir les tables et recouvrir les données .

La section suivante est destinée aux cas où les commandes ci−dessus ont échoué ou que vous voulez exploiter les fonctionnalités avancées que `isamchk` / `myisamchk` proposent.

Si vous allez réparer une table en ligne de commande, il est recommandé d'arrêter le serveur `mysqld` . Notez que lorsque vous exécutez une commande `mysqladmin shutdown` sur un serveur distant, le serveur `mysqld` sera encore opérationnel pendant un instant après que `mysqladmin` ait terminé, jusqu'à ce que toutes les requêtes et toutes les clés aient été écrites sur le disque.

**Etape 1 : Vérifier vos tables**

Exécutez la commande `myisamchk *.MYI` ou `myisamchk -e *.MYI` si vous avez plus de temps. Utilisez `-s` (silencieux) pour supprimer les informations peu pertinentes.

Si le serveur `mysqld` a terminé, vous devriez utiliser l'option −−update pour indiquer à `myisamchk` d'enregistrer la vérification des tables ('checked').

Vous n'aurez à réparer que les tables pour lesquelles `myisamchk` vous annonce une erreur. Pour de telles tables, passez à l'étape 2.

Si vous obtenez des erreurs étranges lors de la vérification, (comme, l'erreur `out of memory` ), ou si `myisamchk` crashe, passez à l'étape 3.

**Etape 2 : réparation simple et facile**

Note : Si vous voulez réparer très rapidement, vous devriez ajouter `-O sort_buffer=# -O key_buffer=#` (où # vaut environ le quart de la mémoire du serveur), à toutes les commandes `isamchk/myisamchk` .

Premièrement, essayez `myisamchk -r -q tbl_name` ( `-r -q` signifie ``mode de réparation rapide''). Cette commande va tenter de réparer le fichier d'index sans toucher au fichier de données. Si le fichier de données contient toutess les données qu'il est sensé contenir, et que les points d'ancrage pour les effacements sont corrects, cette commande doit réussir, et la table sera alors réparée. Passez alors à la table suivante. Sinon, suivez la procédure suivante :

- Faites une copie de sauvegarde de votre fichier de données.
- Utilisez la commande `myisamchk -r tbl_name` ( `-r` signifie ``mode de réparation''). Cette commande va supprimer les lignes invalides et effacer ces lignes du fichier de données, puis reconstruire le fichier d'index.
- Si l'instruction précédente a échoué, utilisez `myisamchk --safe-recover tbl_name` . Le mode restauration sécuritaire utilise une vieille méthode de réparation qui peut gérer certains cas rares, mais elle est bien plus lente.

Si vous obtenez des erreurs étranges lors de la répaaration (comme des erreurs de type `out of memory` ), ou si `myisamchk` crashe, passez à l'étape 3.

**Etape 3 : Réparations difficiles**

Nous ne devriez atteindre cette étape que si les 16 premiers ko du fichier d'index sont détruits, ou qu'il contient des données erronées, ou si le fichier d'index manque. Dans ce cas, il est nécessaire de créer un nouveau fichier d'index. Faites ceci :

- Déplacez le fichier de données dans une archive sûre.
- Utilisez le fichier description de la table pour créer de nouveaux fichiers de données et d'index vides.

```
 shell> mysql db_name
mysql> SET AUTOCOMMIT=1;
mysql> TRUNCATE TABLE table_name;
mysql> quit
```

  Si votre version SQL ne dispose pas de `TRUNCATE TABLE` , utilisez la commande `DELETE FROM table_name` .

- Copiez l'ancien fichier de données à la place du nouveau fichier de données (ne faites pas un simple déplacement de fichier. Utilisez une copie, au cas où un problème surviendrait).

Retournez à l'étape 2. `myisamchk -r -q` doit alors fonctionner (et ceci ne doit pas être une boucle infinie).

Depuis `MySQL` 4.0.2, vous pouvez aussi utiliser `REPAIR ... USE_FRM` qui effectue toute cette opération automatiquement.

**Etape 4 : Réparation très difficiles**

Vous ne devriez atteindre cette étape que si votre fichier de description a aussi crashé. Cela ne devrait jamais arriver, car le fichier de description n'est jamais modifié une fois que la table est créée.

- Restaurez le fichier de description avec une sauvegarde, et retournez à l'étape 3. Vous pouvez aussi restaurer le fichier d'index et retourner à l'étape 2. Dans ce dernier cas, vous pouvez démarrer avec l'option `myisamchk -r` .
- Si vous n'avez pas de sauvegarde, mais que vous savez exactement comment la table a été créée, vous pouvez créer une telle table dans une autre base. Supprimez alors le nouveau fichier de données, puis déplacez les fichiers de description et d'index dans votre base de données crashée. Cela vous donnera un nouveau fichier d'index et de description. Retournez à l'étape 2 et essayez de reconstruire le fichier d'index.

### 4.4.6.10 Optimisation de tables

Pour réorganiser les lignes fragmentées et éliminer l'espace perdu par les effacements et les modifications de lignes, vous pouvez exécuter l'utilitaire `myisamchk` en mode de restauration :

```
shell> myisamchk -r tbl_name
```

Vous pouvez optimiser une table de la même façon que vous le faîtes avec la commande SQL `OPTIMIZE TABLE` . `OPTIMIZE TABLE` effectue une réparation de la table, et une analyse des index, puis trie l'arbre d'index pour accélérer les recherches de clés. L'utilisation de la commande réduit aussi les interférences entre le serveur et l'utilitaire car c'est le serveur lui−même qui fait le travail. Syntaxe de `OPTIMIZE TABLE` .

`myisamchk` dispose aussi d'un grand nombre d'options que vous pouvez utiliser pour améliorer les performances de la table :

- `-S` , `--sort-index`
- `-R index_num` , `--sort-records=index_num`
- `-a` , `--analyze`

Pour une description complète de ces options, Syntaxe d'invocation de `myisamchk` .

## 4.4.7 Mettre en place un régime d'entretien de MySQL

Depuis MySQL version 3.23.13, vous pouvez vérifier les tables de type MyISAM avec la commande `CHECK TABLE` . Syntaxe de `CHECK TABLE` . Vous pouvez aussi réparer les tables avec la commande `REPAIR TABLE` . Syntaxe de `REPAIR TABLE` .

C'est une bonne idée que d'effectuer des vérifications des tables régulièrement, plutôt que

d'attendre qu'un problème survienne. Pour faire ces vérifications, vous pouvez utiliser la commande `myisamchk -s` . L'option `-s` (raccourci pour `--silent` ) fait que `myisamchk` s'exécute en mode silencieux, et n'affiche que les messages d'erreurs.

C'est aussi une bonne idée que de vérifier les tables lorsque le serveur démarre. Par exemple, à chaque fois qu'une machine redémarre au milieu d'une modification de table, vous devrez faire une vérification de toutes les tables qui pourraient être affectées : c'est une ``table supposéement corrompue''. Vous pouvez ajouter un test à `safe_mysqld` pour qu'il exécute `myisamchk` , afin de vérifier toutes les tables qui ont été modifiées dans les 24 dernières heures, si il reste un vieux fichier `.pid` (identifiant de processus) après un redémarrage : le fichier `.pid` est créé par le serveur `mysqld` lorsqu'il démarre, et il est supprimé lorsque le serveur s'arrête dans des conditions normales. La présence d'un fichier `.pid` au démarrage indique que le serveur s'est arrêté anormalement.

Un test encore meilleur serait de vérifier toutes les tables dont la date de modification est plus récente que celle du fichier `.pid` .

Vous devriez aussi vérifiez vos tables régulièrement durant les opérations normales. Chez MySQL AB, nous utilisons une tâche en `cron` pour vérifier toutes nos tables importantes au moins une fois par semaine, avec une ligne comme celle-ci dans le fichier `crontab` :

```
35 0 * * 0 /path/to/myisamchk --fast --silent /path/to/datadir/*/*.MYI
```

Cela nous affiche les informations sur les tables qui ont été corrompues, de façon à ce que nous puissions les examiner et les réparer.

Comme nous n'avons jamais eu de table qui se soit corrompue inopinément (des tables qui se corrompent pour d'autres raisons que des problèmes matériels) depuis quelques années (ce qui est véridique), une fois par semaine est un bon rythme pour nous.

Nous recommandons que vous commenciez par exécuter la commande `myisamchk -s` chaque nuit, sur toutes les tables qui ont été modifiées dans les 24 dernières heures, jusqu'à ce que vous preniez confiance en MySQL.

Normalement, vous n'avez pas à maintenir autant les tables MySQL. Si vous changez les tables avec un format de ligne dynamique (les tables avec des colonnes `VARCHAR` , `BLOB` ou `TEXT` ) ou que vous avez des tables avec de nombreuses lignes effacées, vous pouvez envisager de faire des défragmentations du fichier, pour récupérer cet espace. Une fois par mois est un bon rythme.

Vous pouvez faire cela avec la commande SQL `OPTIMIZE TABLE` sur les tables en question, ou bien, si vous avez éteint le serveur `mysqld` , faites :

```
 isamchk -r --silent --sort-index -O sort_buffer_size=16M */*.ISM
myisamchk -r --silent --sort-index  -O sort_buffer_size=16M */*.MYI
```

## 4.4.8 Obtenir des informations sur une table

Pour obtenir la description d'une table ou des statistiques à son sujet, utiliser les commandes affichées ici. Nous allons expliquer certains de leurs détails ultérieurement.

- myisamchk -d nom_de_table Exécute `myisamchk` en ``mode description'' pour produire une description de votre table. Si vous démarrez le serveur MySQL en utilisant l'option `--skip-external-locking` , `myisamchk` va rapporter une erreur si la table est modifiée durant l'exécution de la commande. Cependant, comme `myisamchk` ne modifie pas les tables, durant le mode description, il n'y a pas de risque de perte de données.

- myisamchk −d −v nom_de_table Pour produire plus d'informations durant l'exécution de `myisamchk`, ajoutez l'option `−v` pour indiquer qu'elle doit fonctionner en mode détaillé.
- myisamchk −eis nom_de_table Affiche les informations les plus importantes pour une table. C'est une commande lente, car elle doit lire toute la table.
- myisamchk −eiv nom_de_table C'est l'équivalent de `−eis`, mais qui vous indique ce qui se passe.

Exemple d'affichage résultant de `myisamchk −d` :

```
 MyISAM file:     company.MYI
Record format:   Fixed length
Data records:    1403698  Deleted blocks:         0
Recordlength:    226

 table description:
Key Start Len Index   Type
1   2     8   unique  double
2   15    10  multip. text packed stripped
3   219   8   multip. double
4   63    10  multip. text packed stripped
5   167   2   multip. unsigned short
6   177   4   multip. unsigned long
7   155   4   multip. text
8   138   4   multip. unsigned long
9   177   4   multip. unsigned long
    193   1           text
```

Exemple d'affichage résultant de `myisamchk −d −v` :

```
 MyISAM file:         company
Record format:       Fixed length
File−version:        1
Creation time:       1999−10−30 12:12:51
Recover time:        1999−10−31 19:13:01
Status:              checked
Data records:           1403698  Deleted blocks:              0
Datafile parts:         1403698  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     3
Max datafile length: 3791650815  Max keyfile length: 4294967294
Recordlength:               226

 table description:
Key Start Len Index   Type                   Rec/key     Root Blocksize
1   2     8   unique  double                       1 15845376      1024
2   15    10  multip. text packed stripped         2 25062400      1024
3   219   8   multip. double                      73 40907776      1024
4   63    10  multip. text packed stripped         5 48097280      1024
5   167   2   multip. unsigned short            4840 55200768      1024
6   177   4   multip. unsigned long             1346 65145856      1024
7   155   4   multip. text                      4995 75090944      1024
8   138   4   multip. unsigned long               87 85036032      1024
9   177   4   multip. unsigned long              178 96481280      1024
    193   1           text
```

Exemple d'affichage résultant de `myisamchk −eis` :

```
 Checking MyISAM file: company
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:  98%  Packed:   17%

 Records:          1403698    M.recordlength:     226
Packed:             0%
Recordspace used:     100%   Empty space:          0%
Blocks/Record:   1.00
Record blocks:    1403698    Delete blocks:        0
Recorddata:     317235748    Deleted data:         0
Lost space:             0    Linkdata:             0

 User time 1626.51, System time 232.36
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 627, Swaps 0
Blocks in 0 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 639, Involuntary context switches 28966
```

Exemple d'affichage résultant de `myisamchk −eiv` :

```
 Checking MyISAM file: company
Data records: 1403698   Deleted blocks:       0
− check file−size
− check delete−chain
block_size 1024:
```

```
index  1:
index  2:
index  3:
index  4:
index  5:
index  6:
index  7:
index  8:
index  9:
No recordlinks
- check index reference
- check data record references index: 1
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 2
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
- check data record references index: 3
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
- check data record references index: 5
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 6
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 7
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 8
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 9
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:   9%  Packed:   17%

 - check records and index references
[LOTS OF ROW NUMBERS DELETED]

 Records:          1403698    M.recordlength:     226   Packed:             0%
Recordspace used:     100%   Empty space:          0%  Blocks/Record:   1.00
Record blocks:    1403698    Delete blocks:        0
Recorddata:     317235748    Deleted data:         0
Lost space:             0    Linkdata:             0

 User time 1639.63, System time 251.61
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 10580, Swaps 0
Blocks in 4 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 10604, Involuntary context switches 122798
```

Voici les tailles des fichiers de données et d'index utilisés dans les tables précédentes :

```
 -rw-rw-r--   1 monty    tcx     317235748 Jan 12 17:30 company.MYD
-rw-rw-r--   1 davida   tcx      96482304 Jan 12 18:35 company.MYM
```

Des détails sur les types d'informations retournés par `myisamchk` sont listés ici. Le ``keyfile'' est le fichier d'index. ``Record'' et ``row'' sont synonymes de ligne :

- ISAM file Nom du fichier d'index ISAM.
- Isam−version Version du format ISAM. Actuellement, c'est toujours 2.
- Creation time Date de création du fichier de données.
- Recover time Date de dernière reconstruction du fichier de données ou d'index.
- Data records Combien de lignes sont stockées dans la table.
- Deleted blocks Combien de blocs effacés occupent toujours de l'espace. Vous pouvez optimiser la table pour récupérer cet espace. Optimisation de table .
- Data file: Parts Pour les tables au format de ligne dynamique, ceci indique combien de blocs de données sont présents. Pour une table optimisée sans lignes fragmentées, la valeur doit être égale à `Data records` .
- Deleted data Combien d'octets de données effacées et non réutilisées sont présents dans la table. Vous pouvez optimiser la table pour récupérer cet espace. Optimisation de table .
- Data file pointer La taille du pointeur de fichier de données, en octets. C'est généralement 2, 3, 4, ou 5 octets. La plupart des tables peuvent se gérer avec 2 octets, mais ceci ne peut être contrôlé par MySQL actuellement. Pour les tables à format de ligne fixe, c'est une adresse de ligne. Pour les tables dynamiques, c'est une adresse d'octet.
- Keyfile pointer La taille du pointeur de fichier d'index, en octets. C'est généralement 1, 2 ou 3 octets. La plupart des tables supportent 2 octets, mais cela est calculé automatiquement par MySQL. C'est toujours une adresse de bloc.
- Max datafile length Taille maximale du fichier de données ( `.MYD` file), en octets.
- Max keyfile length Taille maximale du fichier d'index ( `.MYI` file), en octets.
- Recordlength Taille occupée par chaque ligne, en octets.

- Record format Le format utilisé pour stocker les lignes de la table. Les exemples ci−dessus utilisaient `Fixed length` . Les autres valeurs possibles sont `Compressed` et `Packed` .
- table description Une liste de toutes les clés de la table. Pour chaque clé, des informations de bas niveau sont présentées :
    - Key Le numéro d'index.
    - Start Où, dans la ligne, l'index débute.
    - Len Taille de cette partie d'index. Pour les nombres compactés, c'est toujours la taille maximale de la colonne. Pour les chaînes, c'est plus petit que la taille maximale de la colonne index, car vous pouvez indexer un préfixe de la chaîne.
    - Index `unique` et `multip.` (multiple). Indique si une valeur peut exister plusieurs fois dans cet index.
    - Type De quel type de données cet index est. C'est un type de données ISAM avec les options `packed` , `stripped` ou `empty` .
    - Root Adresse du premier bloc d'index.
    - Blocksize La taille de chaque bloc d'index. Par défaut, c'est 1024, mais cette valeur peut être modifiée lors de la compilation.
    - Rec/key C'est une valeur statistique, utilisée par l'optimisateur. Il indique combien de lignes sont disponibles par valeur de cette clé. Une clé unique aura toujours une valeur de 1. Cela peut être modifié une fois que la table est chargée (ou modifiée de façon majeure), avec la commande `myisamchk −a` . Si ce n'est pas mis à jour, une valeur par défaut de 30 est utilisée.
- Dans le premier exemple ci−dessus, la neuvième clé est une clé multi−partie, avec deux parties.
- Keyblocks used Quel pourcentage des blocs de clé est utilisé. Comme les tables utilisées dans les exemples ont tout juste été réorganisées avec `myisamchk` , ces valeurs sont très grandes (très proches du maximum théorique).
- Packed MySQL essaie de compacter les clés ayant un préfixe commun. Cela ne peut être utilisé que pour les colonnes de type `CHAR` / `VARCHAR` / `DECIMAL` . Pour les longues chaînes comme des noms, cette technique va significativement réduire l'espace utilisé. Dans le troisième exemple ci−dessus, la quatrième clé fait 10 caractères de long et a une réduction de 60 % dans l'espace utilisé effectivement.
- Max levels La profondeur du B−tree. Les grandes tables avec de longues clés peuvent obtenir de grandes valeurs.
- Records Combien de lignes sont enregistrées dans la table.
- M.recordlength La taille moyenne d'une ligne. Pour les tables avec un format de ligne statique, c'est la taille de chaque ligne.
- Packed MySQL efface les espaces à la fin des chaînes. `Packed` indique le pourcentage d'économie d'espace réalisé.
- Recordspace used Quel est le pourcentage d'utilisation du fichier de données.
- Empty space Quel est le pourcentage d'utilisation du fichier d'index.
- Blocks/Record Le nombre moyen de blocs par enregistrements (c'est à dire, de combien de liens une ligne fragmentées est constituée). C'est toujours 1.0 pour les tables à format de ligne statique. Cette valeur doit être aussi proche que possible de 1.0. Si elle grossit trop, vous pouvez réorganiser la table avec `myisamchk` . Optimisation de table .
- Recordblocks Combien de blocs sont utilisés. Pour les tables à format de ligne fixe, c'est le même nombre que le nombre de lignes.
- Deleteblocks Combien de blocs (liens) sont effacés.
- Recorddata Combien d'octets sont utilisés dans le fichier.
- Deleted data Combien d'octets dans le fichier de données sont effacés (inutilisés).
- Lost space Si une ligne est modifiée, et réduite en taille, de l'espace est perdu. Ce chiffre est la somme de ces espaces perdus, en octets.
- Linkdata Lorsque le format de ligne dynamique est utilisé, les fragments de lignes sont liés avec des pointeurs de (4 à 7 octets chacun). `Linkdata` est la somme du stockage utilisé par ces pointeurs.

Si une table a été compressée avec `myisampack` , `myisamchk -d` affiche des informations supplémentaires à propos de chaque colonne. Voir `myisampack` , le générateur de tables MySQL compressées en lecture seule , pour un exemple de ces informations, et une description de leur signification.

# 4.5 Référence de langage d'administration de la base de données

## 4.5.1 Syntaxe de OPTIMIZE TABLE

```
OPTIMIZE TABLE tbl_name[,tbl_name]...
```

`OPTIMIZE TABLE` doit être utilisée si une grande partie de la base a été effacée, ou si vous avez fait de nombreuses modificaitons dans une table à format de ligne dynamique (des tables qui ont des colonnes de type `VARCHAR` , `BLOB` et `TEXT` ). Les lignes effacées sont conservées dans une liste, et les prochaines opérations d' `INSERT` réutilisent les vieilles positions de lignes. Vous pouvez vous servir de la commande `OPTIMIZE TABLE` pour récupérer l'espace utilisé et défragmenter le fichier de données.Pour le moment, `OPTIMIZE TABLE` fonctionne uniquement avec les tables de type `MyISAM` et `BDB` . Pour les tables `BDB` , `OPTIMIZE TABLE` est actuellement l'équivalent de `ANALYZE TABLE` . Syntaxe de `ANALYZE TABLE` .

Vous pouvez vous arranger pour que `OPTIMIZE TABLE` fonctionne sur d'autres types de tables, en démarrant `mysqld` avec `--skip-new` ou `--safe-mode` , mais dans ce cas, `OPTIMIZE TABLE` est simplement l'équivalent de `ALTER TABLE` .

`OPTIMIZE TABLE` fonctionne comme ceci :

- Si la table contient des lignes effacées ou des lignes fragmentées, la table est compactée.
- Si les pages d'index ne sont pas triées, `OPTIMIZE TABLE` les trie. Si les statistiques ne sont pas à jour (et que la table n'a pas pu effectuer de réparation en triant l'index), elles sont mises à jour.

`OPTIMIZE TABLE` pour les tables `MyISAM` est l'équivalent de la commande `myisamchk --quick --check-only-changed --sort-index --analyze` sur la table.

Notez que la table est verrouilée durant la commande `OPTIMIZE TABLE` .

## 4.5.2 Syntaxe de ANALYZE TABLE

```
ANALYZE TABLE nom_de_table[,nom_de_table...]
```

Cette commande analyse et stocke la clé de distribution de la table. Durant l'analyse, la table est verrouilée en lecture. Cette commande fonctionne avec les tables `MyISAM` et `BDB` .

C'est l'équivalent de la commande en ligne `myisamchk -a` .

MySQL utilise les clés de distribution pour décider dans quel ordre les tables doivent être rassemblées lors des jointures qui ne s'effectuent pas sur une constante.

La commande retourne une table avec les colonnes suivantes :

| Colonne | Valeur |
|---|---|

| | |
|---|---|
| Table | Nom de la table |
| Op | ``analyze'' (toujours) |
| Msg_type | Un des status , error , info ou warning . |
| Msg_text | Le message. |

Vous pouvez vérifier la clé de distribution stockée avec la commande SHOW INDEX . Lire des informations sur les bases, colonnes, tables ou index .

Si la table n'a pas changé depuis la dernière commande ANALYZE TABLE , elle ne sera pas analysée à nouveau.

## 4.5.3 Syntaxe de FLUSH

```
FLUSH flush_option [,flush_option] ...
```

Vous devez utiliser la commande FLUSH si vous voulez effacer certains caches internes de MySQL. Pour exécuter FLUSH , vous devez avoir le droit RELOAD .

flush_option peut être l'une des suivantes :

| Option | Description |
|---|---|
| HOSTS | Vide le cache des hôtes. Vous devez vider ce cache si certaines des adresses IP de vos clients changent, ou si vous obtenez des erreurs du type Host ... is blocked . Lorsque plus de max_connect_errors erreurs successives surviennent pour un hôte, lors des connexions au serveur MySQL, MySQL suppose qu'il y a un problème, et interdit l'accès à l'hôte. Erreur Host '...' is blocked . Vous pouvez démarrer mysqld avec -O max_connect_errors=999999999 pour éviter ce message. |
| DES_KEY_FILE | Recharge les clés DES depuis le fichier de stockage spécifié par --des-key-file lors du démarrage du serveur. |
| LOGS | Ferme et réouvre tous les fichiers de log. Si vous avez spécifié un fichier de log de mise à jour, ou un fichier de log binaire sans extension, le numéro d'extension du fichier de log sera incrémenté d'une unité. Si vous avez utilisé une extension dans le nom du fichier, MySQL va fermer et réouvrir le même fichier. Le log des mises à jour. Ceci est la même chose que d'envoyer le signal SIGHUP au serveur mysqld . |
| PRIVILEGES | Recharge les privilèges des tables de droits dans la base mysql . |
| QUERY CACHE | Défragmente le cache des requêtes pour mieux en utiliser la mémoire. Cette commande n'effacera aucune requête du cache, à la différence de RESET QUERY CACHE . |
| TABLES | Ferme toutes les tables ouvertes, et force les tables utilisées à se refermer. Cela vide aussi le cache de requêtes. |
| [TABLE \| TABLES] nom_de_table [,nom_de_table...] | Vide du cache uniquement les tables nommées. |
| TABLES WITH READ LOCK | Ferme toutes les tables ouvertes, et verrouille en lecture toute les tables et bases, jusqu'à ce que vous exécutiez une commande UNLOCK TABLES . C'est très pratique pour générer des sauvegardes, si vous avez un système de fichiers comme Veritas, qui peut prendre des photos du système. |
| STATUS | Remet la plupart des variables de statut à zéro. A n'utiliser que pour corriger une requête. |
| USER_RESOURCES | Remet toutes les ressources à zéro. Cela va autoriser de nouveau les utilisateurs qui ont été bloqués. Limiter les ressources utilisateur. |

Vous pouvez aussi accéder à toutes les commandes décrites plus haut en les donnant en arguments à mysqladmin (exemple : flush-hosts , flush-logs , reload , ou encore flush-tables ).

Reportez-vous aussi à la commande RESET avec la réplication. Syntaxe de RESET .

## 4.5.4 Syntaxe de la commande RESET

```
RESET reset_option [,reset_option] ...
```

La commande RESET sert à remettre à zéro des données. C'est aussi une version plus puissante de la commande FLUSH . Syntaxe de FLUSH .

Pour exécuter la commande RESET , vous devez avoir les droits RELOAD .

| Option | Description |
|---|---|
| MASTER | Efface tous les logs binaires listés dans le fichier d'index, et l'index binlog est vidé. Dans les version antérieures à la version 3.23.26, cette commande s'appelait FLUSH MASTER (Master) |

| | |
|---|---|
| SLAVE | Annule la position de réplication de l'esclave dans les historiques du maître. Dans les version antérieures à la version 3.23.26, cette commande s'appelait FLUSH SLAVE (Slave) |
| QUERY CACHE | Supprime tous les résultats de requêtes du cache de requête. |

## 4.5.5 Syntaxe de KILL

```
KILL thread_id
```

Chaque connexion à `mysqld` utilise un thread unique. Vous pouvez voir les threads en cours d'exécution en utilisant la commande `SHOW PROCESSLIST` et en terminer un avec la commande `KILL thread_id` .Si vous avez le droit `PROCESS` , vous pouvez voir tous les threads. Si vous avez le droit `SUPER` , vous pouvez terminer tout les threads. Sinon, vous ne pouvez terminer que vos propres threads.

Vous pouvez aussi utiliser les commandes `mysqladmin processlist` et `mysqladmin kill` pour examiner et terminer les threads.

**Note** : Vous ne pouvez actuellement pas utiliser `KILL` avec la librairie du serveur embarqué, car celui−ci utilise les threads de l'application hôte, il ne crée pas ses propres threads.

Quand vous exécutez un `KILL` , un thread spécifique est crée pour ce thread.

Dans la plupart des cas, la terminaison du thread pourra prendre un certain temps vu que le thread de terminaison est invoqué à intervalles spécifiques.

- Pour les boucles de `SELECT` , `ORDER BY` et `GROUP BY` , le thread de terminaison est vérifié après avoir lu un enregistrement. S'il est activé, la requête est abandonnée.
- Lors d'un `ALTER TABLE` le thread de terminaison est vérifié avant la lecture de chacune des colonnes de la table d'origine. S'il est activé, la commande est abandonnée et la table temporaire effacée.
- Lors d'un `UPDATE` ou d'un `DELETE` , le thread de terminaison est vérifié après chaque lecture de bloc et chaque mise à jour ou suppression de ligne. S'il est activé, la requête est abandonnée. Notez que si vous utilisez les transactions, les modifications ne seront pas perdues !
- `GET_LOCK()` stoppera avec `NULL` .
- Un thread `INSERT DELAYED` videra rapidement toutes les lignes en mémoire et se terminera.
- Si le thread est dans le gestionnaire des verrous de tables (état : `Locked` ), le verrou sur la table sera vite enlevé.
- Si le thread est en attente de libération d'espace disque lors d'un appel à `write` , l'opération est avortée avec un message d'erreur indiquant que le disque est plein.

## 4.5.6 Syntaxe de SHOW

```
    SHOW DATABASES [LIKE wild]
ou SHOW [OPEN] TABLES [FROM nom_base] [LIKE wild]
ou SHOW [FULL] COLUMNS FROM nom_de_table [FROM nom_base] [LIKE wild]
ou SHOW INDEX FROM nom_de_table [FROM nom_base]
ou SHOW TABLE STATUS [FROM nom_base] [LIKE wild]
ou SHOW STATUS [LIKE wild]
ou SHOW VARIABLES [LIKE wild]
ou SHOW LOGS
ou SHOW [FULL] PROCESSLIST
ou SHOW GRANTS FOR utilisateur
ou SHOW CREATE TABLE nom_de_table
ou SHOW MASTER STATUS
ou SHOW MASTER LOGS
ou SHOW SLAVE STATUS
ou SHOW WARNINGS [LIMIT #]
ou SHOW ERRORS [LIMIT #]
ou SHOW TABLE TYPES
```

`SHOW` fournit des informations sur les bases de données, les colonnes, les tables, ou l'état du serveur. Si la clause `LIKE wild` est utilisée, la chaîne `wild` peut être une chaîne qui utilise les caractères SQL joker `'%'` et `'_'` .

### 4.5.6.1 Obtenir des informations sur les bases, tables, colonnes et index

Vous pouvez utiliser la syntaxe alternative `db_name.tbl_name` à la syntaxe `tbl_name FROM db_name` . Ces deux commandes sont équivalentes :

```
 mysql> SHOW INDEX FROM mytable FROM mydb;
mysql> SHOW INDEX FROM mydb.mytable;
```

`SHOW DATABASES` liste les bases que le serveur MySQL héberge. Vous pouvez aussi obtenir cette liste en utilisant la commande `mysqlshow` . Depuis la version 4.0.2, vous ne verrez que les bases pour lesquelles vous avez des droits, à moins que vous n'ayez le droit global de `SHOW DATABASES` .

`SHOW TABLES` liste les tables d'une base de données. Vous pouvez aussi accéder à cette liste avec la commande `mysqlshow db_name` .

**Note** : si un utilisateur n'a aucun droit sur une table, la table ne sera pas affichée dans le résultat de `SHOW TABLES` et `mysqlshow db_name` .

`SHOW OPEN TABLES` liste les tables qui sont actuellement ouvertes dans le cache de table. Comment MySQL ouvre et ferme les tables . Le champs `Comment` indique combien de fois la table a été mise en cache ( `cached` ) et utilisée ( `in_use` ).

`SHOW COLUMNS` liste les colonnes d'une table données. Si vous spécifiez l'option `FULL` , vous obtiendrez aussi les droits dont vous disposez sur chaque colonne. Si les types de colonnes sont différents de ce à quoi vous vous attendiez après votre requête `CREATE TABLE` , notez que MySQL change parfois les types de colonnes. Silent column changes .

La requête `DESCRIBE` fournit des informations similaires à `SHOW COLUMNS` . Syntaxe de `DESCRIBE` (Obtenir des informations sur les colonnes) .

`SHOW FIELDS` est un synonyme de `SHOW COLUMNS` , et `SHOW KEYS` est un synonyme de `SHOW INDEX` . Vous pouvez aussi lister les colonnes d'une table ou ses index avec les commandes en ligne `mysqlshow db_name tbl_name` et `mysqlshow -k db_name tbl_name` .

`SHOW INDEX` retourne le détails sur les index, dans un format qui ressemble vaguement à l'affichage obtenu par la commande `SQLStatistics` de ODBC. Les colonnes suivantes sont disponibles :

| Colonne | Signification |
|---|---|
| Table | Nom de la table. |
| Non_unique | 0 si l'index ne peut pas contenir de doublons. |
| Key_name | Nom de l'index. |
| Seq_in_index | Numéro de séquence de la colonne dans l'index, commençant à 1. |
| Column_name | Nom de la colonne. |
| Collation | Comment la colonne est placée dans l'index. Avec MySQL, les valeurs peuvent être 'A' (Ascending) ou NULL (Non trié). |
| Cardinality | Nombre de valeurs uniques dans l'index. Ceci est mis à jour en exécutant la commande isamchk -a . |
| Sub_part | Numbre de caractères indexés si la colonne est partiellement indexée. NULL siginifie que la clée entière est indexée. |
| Null | Contient 'YES' si la colonne peut contenir la valeur NULL . |
| Index_type | La méthode d'indexation utilisée. |
| Comment | Différentes remarques. Actuellement, il indique si l'index est FULLTEXT ou non (MySQL < 4.0.2). |

Notez que la cardinalité ( `Cardinality` ) est comptée grâce à des statistiques stockées sous forme d'entier. Ce n'est peut être pas très précis pour les petites tables.

Les colonnes `Null` et `Index_type` ont été ajoutées en MySQL 4.0.2.

### 4.5.6.2 SHOW TABLE STATUS

```
SHOW TABLE STATUS [FROM db_name] [LIKE wild]
```

`SHOW TABLE STATUS` (nouveau en version 3.23) fonctionne comme `SHOW STATUS` , mais fournit des informations sur les tables. Vous pouvez aussi obtenir ces informations en utilisant la commande en ligne `mysqlshow --status db_name` . Les données suivantes sont retournées :

| Colonne | Signification |
|---|---|
| Name | Nom de la table. |
| Type | Type de table. Types de tables MySQL. |
| Row_format | Le format de stockage de ligne (Fixed, Dynamic ou Compressed). |
| Rows | Nombre de lignes. |
| Avg_row_length | Taille moyenne d'une ligne. |
| Data_length | Taille du fichier de données. |
| Max_data_length | Taille maximale du fichier de données. |
| Index_length | Taille du fichier d'index. |
| Data_free | Nombre d'octets alloués mais non utilisés. |
| Auto_increment | Prochaine valeur d'autoincrement. |
| Create_time | Date de création de la table. |
| Update_time | Date de dernière modification de la table. |
| Check_time | Date de dernier entretien de la table. |
| Create_options | Options supplémentaires utilisées avec CREATE TABLE . |
| Comment | Le commentaire utilisé lors de la création de la table (ou des informations sur pourquoi MySQL n'a pu accéder aux informations de la table). |

Les tables `InnoDB` indiqueront l'espace disque libre dans le commentaire de table.

### 4.5.6.3 Syntaxe de SHOW STATUS

`SHOW STATUS` affiche des informations sur le statut du serveur (comme par exemple, `mysqladmin extended-status` ). L'affichage ressemble à ce qui est affiché ci-dessous, mais les valeurs différeront sûrement de votre propre serveur.

```
 +--------------------------+------------+
| Variable_name            | Value      |
+--------------------------+------------+
| Aborted_clients          | 0          |
| Aborted_connects         | 0          |
| Bytes_received           | 155372598  |
| Bytes_sent               | 1176560426 |
| Connections              | 30023      |
| Created_tmp_disk_tables  | 0          |
| Created_tmp_tables       | 8340       |
| Created_tmp_files        | 60         |
| Delayed_insert_threads   | 0          |
| Delayed_writes           | 0          |
| Delayed_errors           | 0          |
| Flush_commands           | 1          |
| Handler_delete           | 462604     |
| Handler_read_first       | 105881     |
| Handler_read_key         | 27820558   |
| Handler_read_next        | 390681754  |
| Handler_read_prev        | 6022500    |
| Handler_read_rnd         | 30546748   |
| Handler_read_rnd_next    | 246216530  |
| Handler_update           | 16945404   |
| Handler_write            | 60356676   |
| Key_blocks_used          | 14955      |
| Key_read_requests        | 96854827   |
| Key_reads                | 162040     |
| Key_write_requests       | 7589728    |
| Key_writes               | 3813196    |
| Max_used_connections     | 0          |
| Not_flushed_key_blocks   | 0          |
| Not_flushed_delayed_rows | 0          |
| Open_tables              | 1          |
| Open_files               | 2          |
| Open_streams             | 0          |
| Opened_tables            | 44600      |
| Questions                | 2026873    |
```

```
| Select_full_join         | 0          |
| Select_full_range_join   | 0          |
| Select_range             | 99646      |
| Select_range_check       | 0          |
| Select_scan              | 30802      |
| Slave_running            | OFF        |
| Slave_open_temp_tables   | 0          |
| Slow_launch_threads      | 0          |
| Slow_queries             | 0          |
| Sort_merge_passes        | 30         |
| Sort_range               | 500        |
| Sort_rows                | 30296250   |
| Sort_scan                | 4650       |
| Table_locks_immediate    | 1920382    |
| Table_locks_waited       | 0          |
| Threads_cached           | 0          |
| Threads_created          | 30022      |
| Threads_connected        | 1          |
| Threads_running          | 1          |
| Uptime                   | 80380      |
+--------------------------+------------+
```

Les variables de statut listées ci−dessus ont la signification suivante :

| Variable | Signification |
|---|---|
| Aborted_clients | Nombre de connexions annulées parce que le client est mort sans se déconnecter correctement. Erreurs de communication / connexion annulée. |
| Aborted_connects | Nombre de tentatives de connexions au serveur MySQL qui ont échouées. Erreurs de communication / connexion annulée. |
| Bytes_received | Nombre d'octets reçu de tous les clients. |
| Bytes_sent | Nombre d'octets envoyés à tous les clients. |
| Com_xxx | Nombre d'exécution de chaque commande. |
| Connections | Nombre de tentatives de connexions au serveur MySQL. |
| Created_tmp_disk_tables | Nombre de tables temporaires implicites créées sur le disque lors d'exécutions de commandes. |
| Created_tmp_tables | Nombre de tables temporaires implicites créées en mémoire lors d'exécutions de commandes. |
| Created_tmp_files | Combien de fichiers temporaires mysqld a créé. |
| Delayed_insert_threads | Nombre de gestionnaires d'insertion retardées sont en cours d'utilisation. |
| Delayed_writes | Nombre de lignes écrites avec INSERT DELAYED . |
| Delayed_errors | Nombre de lignes écrites avec INSERT DELAYED pour lesquelles des erreurs sont survenues (probablement une erreur de doublons ( duplicate key )). |
| Flush_commands | Nombre de commandes FLUSH . |
| Handler_commit | Nombre de commandes internes COMMIT . |
| Handler_delete | Nombre de fois qu'une ligne a été effacées dans une table. |
| Handler_read_first | Nombre de fois que la première ligne a été lue dans un index. Si ce chiffre est haut, c'est que le serveur fait de nombreuses recherches par analyse complète de la table, par exemple SELECT col1 FROM foo , en supposant que col1 est indexé. |
| Handler_read_key | Nombre de requête de lecture de ligne basées sur une clé. Si ce chiffre est grand, c'est une bonne indication de l'indexation correcte de vos tables. |
| Handler_read_next | Nombre de requête de lecture de la ligne suivante en ordre. Cela sera augmenté si vous listez une colonne avec une contrainte d'intervalle. Cette valeur sera aussi incrémentée si vous effectuez un scan d'index. |
| Handler_read_prev | Nombre de requête de lecture de la clé précédente, dans l'ordre. C'est souvent utilisé pour optimiser les clauses ORDER BY ... DESC . |
| Handler_read_rnd | Nombre de lecture d'une ligne basée sur une position fixe. Ce chiffre sera grand si vous effectuez de nombreuses requêtes qui réclament le tri du résultat. |
| Handler_read_rnd_next | Nombre de requêtes de lecture de la prochaine ligne dans le fichier de données. Ce chiffre sera grand si vous faîtes de nombreux scans de tables. Généralement, cela indique que vos requêtes ne sont pas écrites pour profiter des index que vous avez mis en place. |
| Handler_rollback | Nombre de commandes internes ROLLBACK . |
| Handler_update | Nombre de requête de modification d'une ligne dans une table. |
| Handler_write | Nombre de requête pour insérer une ligne dans une table. |
| Key_blocks_used | The Nombre de blocs utilisés dans un cache de clé. |
| Key_read_requests | The Nombre de requêtes de lecture d'un bloc de clé dans le cache. |
| Key_reads | Nombre de lecture physique d'un bloc de clé sur le disque. |
| Key_write_requests | Nombre de requêtes d'écriture d'un bloc de clé dans le cache. |
| Key_writes | Nombre d'écriture physiques de bloc de clé sur le disque. |
| Max_used_connections | Nombre maximum de connexions utilisées simultanément. |
| Not_flushed_key_blocks | Nombre de blocs de clés dans le cache de clés, qui ont été modifiées, mais pas encore écrites sur le disque. |
| Not_flushed_delayed_rows | Nombre de lignes en attente d'écriture dans les listes INSERT DELAY . |
| Open_tables | Nombre de tables ouvertes. |
| Open_files | Nombre de fichiers ouverts. |
| Open_streams | Nombre de flôts ouverts (utilisés généralement pour les logs). |
| Opened_tables | Nombre de tables qui ont été ouvertes. |
| Rpl_status | Statut de la réplication sans erreur (réservé pour utilisation ultérieure). |
| Select_full_join | Nombre de jointures sans clé (si cette variable vaut 0, vous devriez vérifier soigneusement les index de vos tables). |
| Select_full_range_join | Nombre de jointures où une recherche d'intervalle a été utilisée. |
| Select_range | Nombre de jointures où une rercherche d'intervalle a été utilisée sur la première table. (Ce n'est généralement pas important, même si cette valeur est importante). |

| | |
|---|---|
| Select_scan | Nombre de jointures où la première table a été totalement scannée. |
| Select_range_check | Nombre de jointures sans clé, où l'utilisation de clé a été vérifiée après chaque ligne (si cette variable vaut 0, vous devriez vérifier soigneusement les index de vos tables). |
| Questions | Nombre de requêtes envoyées au serveur. |
| Slave_open_temp_tables | Nombre de tables temporaires actuellement utilisée par le thread esclave. |
| Slave_running | Cette variable vaut ON si ce serveur est un esclave connecté au maître. |
| Slow_launch_threads | Nombre de threads qui ont pris plus de slow_launch_time secondes pour être créés. |
| Slow_queries | Nombre de requêtes qui ont pris plus de long_query_time pour s'exécuter. Le log des requêtes lentes. |
| Sort_merge_passes | Nombre de passes que l'algorithme de tri a du faire. Si cette valeur est grande, vous devriez vérifier la taille de sort_buffer . |
| Sort_range | Nombre de tris qui ont été fait sur des intervalles. |
| Sort_rows | Nombre de lignes triées. |
| Sort_scan | Nombre de tris qui ont été fait en scannant la table. |
| ssl_xxx | Variables utilisées par SSL; Réservée pour utilisation ultérieure. |
| Table_locks_immediate | Nombre de fois que la table a reçu immédiatement un verrou. Disponible depuis 3.23.33. |
| Table_locks_waited | Nombre de fois qu'une table n'a pu recevoir de verrou immédiatement, et qu'il a fallu attendre. Si ce chiffre est haut, vous avez des problèmes de performance, et vous devriez optimiser vos requêtes, couper vos tables en deux, ou utiliser la réplication. Disponible depuis la version 3.23.33. |
| Threads_cached | Nombre de threads dans le cache de thread. |
| Threads_connected | Nombre de connexions actuellement ouvertes. |
| Threads_created | Nombre de threads créés pour gérer les connexions. |
| Threads_running | Nombre de threads qui ne dorment pas. |
| Uptime | Durée de vie du serveur, en secondes depuis le redémarrage. |

Quelques commentaires sur les variables ci−dessus :

- Si `Opened_tables` est grand, alors votre `table_cache` est probablement trop petit.
- Si `Key_reads` est grand, alors `key_buffer_size` est probablement trop petit. Le taux d'utilisation du cache peut être calculé avec le ratio `Key_reads` / `Key_read_requests` .
- Si `Handler_read_rnd` est grand, vous avez probablement trop de requêtes qui analysent toute la table, ou vous avez des jointures qui n'exploitent pas d'index.
- Si `Threads_created` est grand, vous devriez augmenter la valeur de la variable `thread_cache_size` . Le taux d'utilisation du cache peut être calculé avec `Threads_created` / `Connections` .
- Si `Created_tmp_disk_tables` est grand, vous devriez augmenter la valeur de la variable `tmp_table_size` pour obtenir des tables temporaires en mémoire, et non plus sur le disque.

### 4.5.6.4 Syntaxe de SHOW VARIABLES

```
SHOW [GLOBAL | SESSION] VARIABLES [LIKE wild]
```

`SHOW VARIABLES` affiche les valeurs des variables systèmes de MySQL. Vous pouvez aussi obtenir ces informations avec la commande `mysqladmin variables` . Si les valeurs par défaut ne vous conviennent pas, vous pouvez modifier la plupart de ces variables, en ligne de commande, lorsque `mysqld` est lancé. Options en ligne de commande de `mysqld` .Les options `GLOBAL` et `SESSION` sont nouvelles depuis MySQL 4.0.3. Avec `GLOBAL` vous allez lister les variables qui sont utilisées pour les nouvelles connexions à MySQL. Avec `SESSION` , vous allez lister les valeurs qui ont actuellement cours pour la connexion courante. Si vous n'utilisez ni l'une, ni l'autre des options, `SESSION` sera utilisée.

Vous pouvez changer la plupart des options avec la commande `SET` . Syntaxe de `SET` .

Le résultat de cette commande ressemble à la table ci−dessous, même si les nombres et formats diffèrent un peu :

```
 +--------------------------------+-----------------------------+
| Variable_name                   | Value                        |
+--------------------------------+-----------------------------|
| back_log                        | 50                           |
| basedir                         | /usr/local/mysql             |
| bdb_cache_size                  | 8388572                      |
| bdb_log_buffer_size             | 32768                        |
| bdb_home                        | /usr/local/mysql             |
| bdb_max_lock                    | 10000                        |
| bdb_logdir                      |                              |
| bdb_shared_data                 | OFF                          |
| bdb_tmpdir                      | /tmp/                        |
| bdb_version                     | Sleepycat Software: ...     |
| binlog_cache_size               | 32768                        |
```

```
| bulk_insert_buffer_size         | 8388608                      |
| character_set                   | latin1                       |
| character_sets                  | latin1 big5 czech euc_kr     |
| concurrent_insert               | ON                           |
| connect_timeout                 | 5                            |
| convert_character_set           |                              |
| datadir                         | /usr/local/mysql/data/       |
| delay_key_write                 | ON                           |
| delayed_insert_limit            | 100                          |
| delayed_insert_timeout          | 300                          |
| delayed_queue_size              | 1000                         |
| flush                           | OFF                          |
| flush_time                      | 0                            |
| ft_min_word_len                 | 4                            |
| ft_max_word_len                 | 254                          |
| ft_max_word_len_for_sort        | 20                           |
| ft_boolean_syntax               | + -><()~*:""&|               |
| have_bdb                        | YES                          |
| have_innodb                     | YES                          |
| have_isam                       | YES                          |
| have_raid                       | NO                           |
| have_symlink                    | DISABLED                     |
| have_openssl                    | YES                          |
| have_query_cache                | YES                          |
| init_file                       |                              |
| innodb_additional_mem_pool_size | 1048576                      |
| innodb_buffer_pool_size         | 8388608                      |
| innodb_data_file_path           | ibdata1:10M:autoextend       |
| innodb_data_home_dir            |                              |
| innodb_file_io_threads          | 4                            |
| innodb_force_recovery           | 0                            |
| innodb_thread_concurrency       | 8                            |
| innodb_flush_log_at_trx_commit  | 0                            |
| innodb_fast_shutdown            | ON                           |
| innodb_flush_method             |                              |
| innodb_lock_wait_timeout        | 50                           |
| innodb_log_arch_dir             |                              |
| innodb_log_archive              | OFF                          |
| innodb_log_buffer_size          | 1048576                      |
| innodb_log_file_size            | 5242880                      |
| innodb_log_files_in_group       | 2                            |
| innodb_log_group_home_dir       | ./                           |
| innodb_mirrored_log_groups      | 1                            |
| interactive_timeout             | 28800                        |
| join_buffer_size                | 131072                       |
| key_buffer_size                 | 16773120                     |
| language                        | /usr/local/mysql/share/...   |
| large_files_support             | ON                           |
| local_infile                    | ON                           |
| locked_in_memory                | OFF                          |
| log                             | OFF                          |
| log_update                      | OFF                          |
| log_bin                         | OFF                          |
| log_slave_updates               | OFF                          |
| log_slow_queries                | OFF                          |
| log_warnings                    | OFF                          |
| long_query_time                 | 10                           |
| low_priority_updates            | OFF                          |
| lower_case_table_names          | OFF                          |
| max_allowed_packet              | 1047552                      |
| max_binlog_cache_size           | 4294967295                   |
| max_binlog_size                 | 1073741824                   |
| max_connections                 | 100                          |
| max_connect_errors              | 10                           |
| max_delayed_threads             | 20                           |
| max_heap_table_size             | 16777216                     |
| max_join_size                   | 4294967295                   |
| max_sort_length                 | 1024                         |
| max_user_connections            | 0                            |
| max_tmp_tables                  | 32                           |
| max_write_lock_count            | 4294967295                   |
| myisam_max_extra_sort_file_size | 268435456                    |
| myisam_max_sort_file_size       | 2147483647                   |
| myisam_recover_options          | force                        |
| myisam_sort_buffer_size         | 8388608                      |
| net_buffer_length               | 16384                        |
| net_read_timeout                | 30                           |
| net_retry_count                 | 10                           |
| net_write_timeout               | 60                           |
| open_files_limit                | 0                            |
| pid_file                        | /usr/local/mysql/name.pid    |
| port                            | 3306                         |
| protocol_version                | 10                           |
| read_buffer_size                | 131072                       |
| read_rnd_buffer_size            | 262144                       |
| rpl_recovery_rank               | 0                            |
| query_cache_limit               | 1048576                      |
| query_cache_size                | 0                            |
| query_cache_type                | ON                           |
| safe_show_database              | OFF                          |
| server_id                       | 0                            |
| slave_net_timeout               | 3600                         |
```

```
| skip_external_locking          | ON                          |
| skip_networking                | OFF                         |
| skip_show_database             | OFF                         |
| slow_launch_time               | 2                           |
| socket                         | /tmp/mysql.sock             |
| sort_buffer_size               | 2097116                     |
| sql_mode                       | 0                           |
| table_cache                    | 64                          |
| table_type                     | MYISAM                      |
| thread_cache_size              | 3                           |
| thread_stack                   | 131072                      |
| tx_isolation                   | READ-COMMITTED              |
| timezone                       | EEST                        |
| tmp_table_size                 | 33554432                    |
| tmpdir                         | /tmp/                       |
| version                        | 4.0.4-beta                  |
| wait_timeout                   | 28800                       |
+--------------------------------+-----------------------------+
```

Tous les options sont décrites ici. Les tailles de buffer, les longueurs et piles sont données en octets. Vous pouvez spécifier ces valeurs avec le suffixe `'K'` ou `'M'` pour indiquer respectivement kilo−octets ou méga−octets. Par `16M` représente 16 mégaoctets. Le suffixe peut être en majuscule ou en minuscule. `16M` et `16m` sont équivalents :

- `ansi_mode` . Vaut `ON` si `mysqld` a été démarré en mode `--ansi` . Exécuter MySQL en mode ANSI .
- `back_log` Le nombre de connexions sortantes que MySQL peut supporter. Cette valeur entre en jeu lorsque le thread principal MySQL reçoit de **très** nombreuses requêtes de connexions en très peu de temps. MySQL prend un peu de temps (même si c'est très peu de temps), pour vérifier la connexion et démarrer un nouveau thread. La valeur de `back_log` indique combien de requête seront mises en attente durant ce temps. Vous devrez augmenter ce nombre si vous voulez mettre en attente plus de requêtes durant une courte période de temps.En d'autres termes, cette valeur est la taille de la queue d'attente pour les connexions TCP/IP entrantes. Votre système d'exploitation a ses propres limites pour ce type de queue. La page du manuel Unix `listen(2)` doit contenir plus de détails. Vérifiez la documentation de votre OS pour connaître la valeur maximale de votre système. Si vous donne une valeur à `back_log` qui est plus grande que celle que votre système supporte, cela restera sans effet.
- `basedir` La valeur de l'option `--basedir` .
- `bdb_cache_size` Le buffer qui est alloué pour mettre en cache des lignes et des index pour les tables `BDB` . Si vous n'utilisez pas la tables `BDB` , vous devriez démarrer `mysqld` avec l'option `--skip-bdb` pour ne pas gaspiller de mémoire.
- `bdb_log_buffer_size` Le buffer qui est alloué pour mettre en cache des lignes et des index pour les tables `BDB` . Si vous n'utilisez pas la tables `BDB` , vous devriez démarrer `mysqld` avec l'option `--skip-bdb` pour ne pas gaspiller de mémoire.
- `bdb_home` La valeur de l'option `--bdb-home` .
- `bdb_max_lock` Le nombre maximum de verrous (par défaut 10 000) que vous pouvez activer simultanément dans une table `BDB` . Vous devriez augmenter cette valeur si vous obtenez des erreurs du type `bdb: Lock table is out of available locks` ou `Got error 12 from ...` lorsque vous avez de longues transactions ou que `mysqld` doit examiner de nombreuses lignes pour calculer la requête.
- `bdb_logdir` La valeur de l'option `--bdb-logdir` .
- `bdb_shared_data` Vaut `ON` si vous utilisez l'option `--bdb-shared-data` .
- `bdb_tmpdir` La valeur de l'option `--bdb-tmpdir` .
- `binlog_cache_size` . La taille du cache qui contient les requêtes SQL destinées au log binaire, durant une transaction. Si vous utilisez souvent de longues transactions multi−requêtes, vous devriez augmenter cette valeur pour améliorer les performances. Syntaxe des `BEGIN/COMMIT/ROLLBACK` .
- `bulk_insert_buffer_size` (était `myisam_bulk_insert_tree_size` ) `MyISAM` utilise une cache hiérarchisé pour les insertions de masses (c'est à dire `INSERT ... SELECT` , `INSERT ... VALUES (...), (...), ...` , et `LOAD DATA INFILE` ). Cette variable limite la taille du cache en octets, par threads. Utiliser la valeur de 0 va désactiver cette optimisation. **Note** : ce cache est uniquement utilisé lorsque vous ajoutez des données dans une table non−vide. Par défaut, cette option vaut 8 Mo.
- `character_set` Le jeu de caractères par défaut.
- `character_sets` Les jeux de caractères supportés.

- concurrent_inserts Si cette option vaut ON , MySQL va vous permettre de réaliser des commandes INSERT sur les tables MyISAM en même temps que d'autres commandes SELECT seront exécutées. Vous pouvez désactiver cette option en démarrant mysqld avec l'option --safe or --skip-new .
- connect_timeout Le nombre de secondes d'attente d'un paquet de connexion avant de conclure avec une erreur Bad handshake .
- datadir La valeur de l'option --datadir .
- delay_key_write Les options pour les tables MyISAM . Elles peuvent prendre l'une des valeurs suivantes :

| | |
|---|---|
| OFF | Toutes les commandes CREATE TABLE ... DELAYED_KEY_WRITES sont ignorés. |
| ON | (par défaut) MySQL va honorer l'option DELAY_KEY_WRITE de CREATE TABLE . |
| ALL | Toutes les nouvelles tables ouvertes sont traitées comme si elles étaient créées avec l'option DELAY_KEY_WRITE . |

  Si DELAY_KEY_WRITE est activé, cela signifie que le buffer de clé des tables ayant cette option ne seront pas écrit sur le disque dès la fin de la modification de la table, mais attendrons que la table soit écrite. Cela accélère notablement les écritures des modifications, mais il faut penser à ajouter une vérification automatique des tables au démarrage avec myisamchk --fast --force .
- delayed_insert_limit Après avoir inséré delayed_insert_limit lignes, le gestionnaire de INSERT DELAYED va vérifiez si il n'y a pas de commande SELECT en attente. Si c'est le cas, il va autoriser ces commandes avant de continuer.
- delayed_insert_timeout Combien de temps le thread INSERT DELAYED doit attendre les commandes INSERT avant de s'achever.
- delayed_queue_size Quelle taille de file (en lignes) doit être allouée pour gérer les commandes INSERT DELAYED . Si la file se remplit, tous les clients qui émettent des commandes INSERT DELAYED devront attendre un peu de place avant de pouvoir continuer.
- flush Cette option vaut ON si vous avez démarré MySQL avec l'option --flush .
- flush_time Si cette option a une valeur non nulle, toutes les flush_time secondes, toutes les tables seront fermées (pour libérez des ressources et synchroniser les index sur le disque). Nous ne recommandons cette option que sur les systèmes Windows 9x/Me, ou les systèmes qui ont très peu de ressource.
- ft_min_word_len La taille minimale d'un mot inclus dans un index FULLTEXT . **Note : les index FULLTEXT doivent être reconstruit après avoir modifié cette variable.** (Cette option est nouvelle en MySQL 4.0.)
- ft_max_word_len La taille maximale d'un mot inclus dans un index FULLTEXT . **Note : les index FULLTEXT doivent être reconstruit après avoir modifié cette variable.** (Cette option est nouvelle en MySQL 4.0.)
- ft_max_word_len_for_sort La taille maximale d'un mot dans un index FULLTEXT , pour qu'il soit inclus dans la méthode de recréation rapide de l'index, avec la commande REPAIR , CREATE INDEX , ou ALTER TABLE . Les mots longs sont insérés par une méthode plus lente. La règle idéale est la suivante : avec ft_max_word_len_for_sort qui croit, **MySQL** va créer des fichiers temporaires plus grand, et donc, ralentir le processus, à cause des accès disques, et cela va mettre moins de clés dans un bloc de tri (encore une fois, limitant la vitesse). Lorsque ft_max_word_len_for_sort est plus petit, **MySQL** va insérer un grand nombre de mot avec la méthode lente, mais les mots courts seront insérés très vite.
- ft_boolean_syntax Liste des opérateurs supportés par MATCH ... AGAINST(... IN BOOLEAN MODE) . Recherche full−text avec MySQL .
- have_innodb YES si mysqld supporte les tables InnoDB. DISABLED si --skip-innodb est utilisée.
- have_bdb YES si mysqld supporte les tables Berkeley. DISABLED si --skip-bdb est utilisée.
- have_raid YES si mysqld supporte l'option RAID .
- have_openssl YES si mysqld supporte le chiffrement SSL entre le client et le serveur.
- init_file Le nom du fichier spécifié avec l'option --init-file lorsque vous démarrez le serveur. C'est un fichier qui contient les requêtes SQL que vous voulez voir exécutées dès le démarrage.
- interactive_timeout Le nombre de secondes durant lequel le serveur attend une activité de la part de la connexion avant de la fermée. Un client interactif est un client qui utilise l'option CLIENT_INTERACTIVE avec mysql_real_connect() . Voir aussi wait_timeout .

- `join_buffer_size` La taille du buffer qui est utilisée pour les jointures complètes (les jointures qui n'utilisent pas d'index). Ce buffer est alloué une fois pour chaque jointure entre deux tables. Augmentez cette valeur si vous voulez obtenir des jointures plus rapides, lorsque l'ajout d'index n'est pas possible. Normalement, le mieux est d'ajouter de bons index.
- `key_buffer_size` Les blocs d'index sont mis en buffer et partagés par tous les threads. `key_buffer_size` est la taille du buffer utilisé.

  Augmentez cette valeur pour obtenir une meilleure gestion des index (pour les lectures et écritures multiples), autant que vous le pouvez : 64Mo sur une machine de 256Mo est une valeur répandue. Toutefois, si vous utilisez une valeur trop grande (par exemple, plus de 50% de votre mémoire totale), votre système risque de commencer à utiliser sa mémoire swap, et devenir très lent. N'oubliez pas que MySQL ne met pas en cache les données lues, et il faut laisser le système d'exploitation respirer.

  Vous pouvez vérifier les performances du buffer de clés avec la commande `show status` et en examinant les variables `Key_read_requests` , `Key_reads` , `Key_write_requests` et `Key_writes` . Le ratio `Key_reads/Key_read_request` doit être normalement < 0.01. Le ratio `Key_write/Key_write_requests` est généralement proche de 1 si vous utilisez principalement des modifications et effacement, mais il peut être bien plus petit si vous faites des modifications qui affectent de nombreuses lignes en même temps, ou si vous utilisez l'option `DELAY_KEY_WRITE` . Syntaxe de `SHOW` .

  Pour obtenir encore plus de vitesse lors de l'écriture de plusieurs lignes en même temps, utilisez `LOCK TABLES` . Syntaxe des `LOCK TABLES/UNLOCK TABLES` .
- `language` La langue utilisée pour les message d'erreurs.
- `large_file_support` Si `mysqld` a été compilé avec le support des grands fichiers.
- `locked_in_memory` Si `mysqld` a été verrouillé en mémoire avec `--memlock`
- `log` Si le log de toutes les requêtes est désactivé.
- `log_update` Si le log de modification est activé.
- `log_bin` Si le log binaire est activé.
- `log_slave_updates` Si les modifications des esclaves doivent être enregistrées.
- `long_query_time` Si une requête prend plus de
- `long_query_time` secondes, le compteur de requêtes lentes `Slow_queries` sera incrémenté. Si vous utilisez l'option `--log-slow-queries` , ces requêtes seront enregistrées dans un historique de requêtes lentes. Cette durée est mesurée en temps réel, et non pas en temps processus, ce qui fait que les requêtes qui seraient juste sous la limite avec un système légèrement chargé, pourrait être au dessus avec le même système, mais chargé. Le log des requêtes lentes .
- `lower_case_table_names` Si cette option vaut 1, les noms de tables sont stockées en minuscules sur le disque, et les comparaisons de nom de tables seront insensibles à la casse. Depuis la version 4.0.2, cette option s'applique aussi aux noms de bases. Sensibilité à la casse pour les noms .
- `max_allowed_packet` La taille maximale d'un paquet. Le buffer de message est initialisé avec `net_buffer_length` octets, mais peut grandir jusqu'à `max_allowed_packet` octets lorsque nécessaire. Cette valeur est par défaut petit, pour intercepter les gros paquets, probablement erronés. Vous devez augmenter cette valeur si vous utilisez de grandes colonnes `BLOB` . Cette valeur doit être aussi grande que le plus grand `BLOB` que vous utiliserez. Le protocole limite actuellement `max_allowed_packet` à 16Mo en MySQL 3.23 et 1Go en MySQL 4.0.
- `max_binlog_cache_size` Si une transaction multi-requête requiert plus que cette quantité de mémoire, vous obtiendrez une erreur `"Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage"` .
- `max_binlog_size` Disponible depuis la version 3.23.33. Si vous écrivez dans le log binaire (de réplication) et que cela dépasse la taille de `max_binlog_size` , une erreur sera indiquée. Vous ne pouvez pas donner à `max_binlog_size` une valeur inférieure à 1024 octets, ou plus grande que 1 Go.

- `max_connections` Le nombre maximal de clients simultanés accepté. En augmentant cette valeur, vous augmentez le nombre de pointeur de fichier que requiert `mysqld` . Voyez plus bas les commentaires sur les pointeurs de fichiers. Erreur `Too many connections` .
- `max_connect_errors` Si il y a plus que `max_connect_errors` connexion interrompues depuis un même hôte, cet hôte sera bloqué dans ses prochaines tentatives de connexions. Vous pouvez débloquer un hôte avec la commande `FLUSH HOSTS` .
- `max_delayed_threads` Ne pas lancer plus que `max_delayed_threads` threads pour gérer les insertions `INSERT DELAYED` . Si vous essayez d'insérer des données dans une nouvelle table alors que tous les gestionnaires `INSERT DELAYED` sont utilisés, la ligne sera insérée comme si l'option `DELAYED` n'avait pas été spécifiée.
- `max_heap_table_size` Ne pas autoriser la création de tables de type HEAP plus grande que `max_heap_table_size` .
- `max_join_size` Les jointures qui liront probablement plus de `max_join_size` lignes, retourneront une erreur. Utilisez cette valeur si vos utilisateurs font des jointures avec de mauvaises clauses `WHERE` , qui prennent trop de temps, et retournent des millions de lignes.
- `max_sort_length` Le nombre d'octets à utiliser lors du tri des colonnes de type `BLOB` et `TEXT` . Seuls les `max_sort_length` octets de chaque valeur seront utilisés pour le tri. Le reste est ignoré.
- `max_user_connections` Le nombre maximum de connexions actives pour un utilisateur particulier (0 = pas de limite).
- `max_tmp_tables` (Cette option ne fait actuellement rien du tout). Le nombre maximum de tables temporaires qu'un client peut garder ouvertes en même temps.
- `max_write_lock_count` Après `max_write_lock_count` pose de verrou en écriture, autorise quelques verrous en lecture.
- `myisam_recover_options` La valeur de l'option `--myisam-recover` .
- `myisam_sort_buffer_size` Le buffer qui est alloués lors du tri d'index avec la commande `REPAIR` ou lors de la création d'index avec `CREATE INDEX` ou `ALTER TABLE` .
- `myisam_max_extra_sort_file_size` . Si un fichier temporaire utilisé pour la création rapide d'index devient plus grand que le cache de clé spécifié ici, la méthode utilisant le cache de clé sera utilisé. C'est principalement utilisé pour forcer les index longs de grandes tables à utiliser une méthode plus lente pour écrire l'index. **Notez** que ce paramètre est spécifié en mégaoctets avant la version 4.0.3 et en octets depuis cette version.
- `myisam_max_sort_file_size` La taille maximale du fichier temporaire que MySQL est autorisé à utiliser durant la recréation des fichiers d'index (avec `REPAIR` , `ALTER TABLE` ou `LOAD DATA INFILE` ). Si la taille du fichier dépasse `myisam_max_sort_file_size` , l'index sera créé avec un cache de clé (plus lent). **Notez** que ce paramètre est spécifié en mégaoctets avant la version 4.0.3 et en octets depuis.
- `net_buffer_length` Le buffer de communication est remis à zéro entre deux requêtes. Cela ne devrait pas être modifié, mais si vous avez très peu de mémoire, vous pouvez le remettre à la taille présumée de la requête (c'est à dire, la taille de requête envoyé par le client. Si la requête dépasse cette taille, le buffer est automatiquement agrandi jusqu'à `max_allowed_packet` octets).
- `net_read_timeout` Nombre de secondes d'attente des dernières données, avant d'annuler la lecture. Notez que lorsque nous n'attendons pas de données d'une connexion, le délai d'expiration est donné par `write_timeout` . Voir aussi `slave_net_timeout` .
- `net_retry_count` Si une lecture sur une port de communication est interrompu, `net_retry_count` tentatives sont faites avant d'abandonner. Cette valeur doit être particulièrement grande pour `FreeBSD` car les interruptions internes sont envoyés à tous les threads.
- `net_write_timeout` Nombre de secondes d'attente pour qu'un bloc soit envoyé à une connexion, avant d'annuler l'écriture.
- `open_files_limit` Si `open_files_limit` ne vaut pas 0, alors `mysqld` va utiliser cette valeur pour réserver des pointeurs de fichiers à utiliser avec `setrlimit()` . Si cette valeur est 0, alors `mysqld` va réserver `max_connections*5` ou `max_connections + table_cache*2` (le plus grand des deux) pointeurs de fichiers. Vous devriez augmenter cette valeur si `mysqld` vous donne des erreurs du type `'Too many open files'` .
- `pid_file` La valeur de l'option `--pid-file` .
- `port` La valeur de l'option `--port` .

- `protocol_version` La version du protocole utilisé par le serveur MySQL.
- `read_buffer_size` (was `record_buffer` ) Chaque thread qui doit faire une recherche séquentielle alloue une buffer de cette taille pour chaque table qu'il scanne. Si vous faîtes de nombreuses recherches séquentielles, vous devriez augmenter cette valeur.
- `record_rnd_buffer_size` Lors de la lecture de lignes qui sont placées dans un ordre trié, les lignes sont lues via ce buffer pour éviter les recherches sur le disque. `record_rnd_buffer_size` peut améliorer grandement les performances de la clause `ORDER BY` si cette valeur est grande. Comme c'est une variable qui est spécifique pour les threads, il ne faut pas trop l'augmenter, mais simplement la modifier lorsque de grandes requêtes sont exécutées.
- `query_cache_limit` Ne met pas en cache les résultats qui sont plus grands que `query_cache_limit` . Par défaut, 1 Mo.
- `query_cache_size` La mémoire allouée pour stocker les résultats des vieilles requêtes. Si `query_cache_size` vaut 0, le cache de requête est désactivé (par défaut).
- `query_cache_type` `query_cache_type` peut prendre les valeurs numériques suivantes :

| Valeur | Alias | Commentaire |
|---|---|---|
| 0 | OFF | Ne met pas en cache les résultats. |
| 1 | ON , | Met en cache tous les résultats exceptés les requêtes SELECT SQL_NO_CACHE ... . |
| 2 | DEMAND | Met en cache uniquement les requêtes SELECT SQL_CACHE ... . |

- `safe_show_database` Ne montre pas les bases pour lesquelles un utilisateur n'a pas des droits de bases ou de tables. Cela peut améliorer considérablement la sécurité si vous craignez de voir les utilisateurs découvrir ce que les autres ont mis en place. Voir aussi `skip_show_database` .
- `server_id` La valeur de l'option `--server-id` .
- `skip_locking` `skip_locking` vaut OFF si `mysqld` utilise le verrouillage externe.
- `skip_networking` `skip_networking` vaut ON si seules les connexions locales (via socket) sont autorisées.
- `skip_show_database` `skip_show_database` empêche les utilisateurs d'exécuter des commandes `SHOW DATABASES` si ils n'ont pas les droits de `PROCESS` . Cela peut améliorer la sécurité si vous craignez de voir les utilisateurs découvrir ce que les autres ont mis en place. Voir aussi `safe_show_database` .
- `slave_net_timeout` Nombre de secondes d'attente de données en lecture ou écriture sur une connexion maître / esclave avant d'annuler.
- `slow_launch_time` Si la création du thread prend plus de `slow_launch_time` secondes, le compteur de threads lents `Slow_launch_threads` sera incrémenté.
- `socket` La socket Unix utilisé par le serveur.
- `sort_buffer` Chaque thread qui doit faire un tri alloue un buffer de cette taille. Augmentez cette taille pour accélérer les clauses `ORDER BY` ou `GROUP BY` . Où MySQL conserve les fichiers temporaires .
- `table_cache` Le nombre de tables ouvertes pour tous les threads réunis. En augmentant cette valeur, vous augmentez le nombre de pointeurs de fichiers que `mysqld` utilise. Vous pouvez vérifier si vous avez besoin de plus de cache de tables en étudiant la valeur de la variable `Opened_tables` . Syntaxe de `SHOW` . Si cette variable est grande, est que vous ne faites pas souvent de commande `FLUSH TABLES` (qui force les tables à se recharger), vous devrez alors augmenter cette valeur.Pour plus d'informations sur le cache de table, voyez Comment MySQL ouvre et ferme les tables .
- `table_type` Le type de table par défaut.
- `thread_cache_size` Combien de threads nous allons conserver en cache pour réutilisation. Lorsqu'un client se déconnecte, les threads du client sont mis en cache s'il n'y en a pas déjà `thread_cache_size` de conservé. Tous les nouveaux threads sont d'abord prélevé dans le cache, et uniquement lorsque le cache est vide, un nouveau thread est créé. Cette variable peut vous permettre d'améliorer les performances si vous avez de nombreuses connexions. Normalement, `thread_cache_size` ne donne pas d'amélioration notable si vous avez une bonne implémentation des threads. En examinant la différence entre les variables de statut `Connections` et `Threads_created` vous pouvez voir comment votre système de cache de threads est efficace.
- `thread_concurrency` Sous Solaris, `mysqld` va appeler `thr_setconcurrency()` avec cette valeur. `thr_setconcurrency()` permet à l'application de donner au système de threads une indication sur le

nombre de threads qui seront exécutés en même temps.

- `thread_stack` La taille de la pile pour chaque thread. De nombreuses limites détectées par `crash-me` sont dépendantes de cette valeur. La valeur par défaut est suffisamment grande pour des opérations normales. La suite de tests comparatifs de MySQL .
- `timezone` Le fuseau horaire du serveur.
- `tmp_table_size` Si une table temporaire en mémoire excède cette taille, MySQL va automatiquement la convertir en une table `MyISAM` sur le disque. Augmentez la valeur de `tmp_table_size` si vous faites un usage intensif de la clause `GROUP BY` et que vous avez beaucoup de mémoire.
- `tmpdir` Le dossier utilisé pour les fichiers temporaires et les tables temporaires.
- `version` Le numéro de version du serveur.
- `wait_timeout` Le nombre de secondes d'attente du serveur sur une connexion non interactive avant de la refermer.Lors du démarrage du thread, `SESSION.WAIT_TIMEOUT` est initialisé avec `GLOBAL.WAIT_TIMEOUT` ou `GLOBAL.INTERACTIVE_TIMEOUT` , suivant le type de client (tel que défini par l'option de connexion `CLIENT_INTERACTIVE` ). Voir aussi `interactive_timeout` .

La section du manuel qui est consacrée à l'optimisation de MySQL contient des informations sur l'utilisation des variables ci−dessus. Choix des paramètres du serveur .

### 4.5.6.5 Syntaxe de SHOW LOGS

La commande `SHOW LOGS` affiche les informations d'état de vos fichiers de logs. Actuellement, elle n'affiche que les informations pour les fichiers de log des tables Berkeley DB.

- `File` affiche le chemin complet jusqu'au fichier de log.
- `Type` affiche le type de fichier de log ( `BDB` pour les tables de types Berkeley DB)
- `Status` affiche le status du fichier de log ( `FREE` si le fichier peut être supprimé, ou `IN USE` si le fichier est utilisé par une transaction en cours)

### 4.5.6.6 Syntaxe de SHOW PROCESSLIST

`SHOW [FULL] PROCESSLIST` affiche la liste de processus qui sont en cours d'exécution. Vous pouvez aussi obtenir ces informations avec la commande en ligne `mysqladmin processlist` . Si vous avez les droits de `SUPER` , vous pourrez aussi voir les autres threads. Sinon, vous ne pourrez voir que les votre. Syntaxe de `KILL` . Si vous n'utilisez pas l'option `FULL` , seuls les 100 premiers caractères de chaque requête seront affichés.

Cette commande est très pratique si vous obtenez trop d'erreurs 'too many connections' et que vous voulez savoir ce qui se passe. MySQL réserve une connexion supplémentaire pour un client ayant les droits de `SUPER` , de façon à ce qu'il y ait toujours la possibilité de se connecter et de vérifier le système (en supposant que vous ne donnez pas ce droit à tous vos utilisateurs).

Certains états sont souvent disponible dans le résultat de `mysqladmin processlist`

- `Checking table` Le thread fait une vérification (automatique) de la table.
- `Closing tables` Le thread est en train d'écrire les données modifiées sur le disque, et il va fermer les tables. Cela doit être une opération très rapide. Si ce n'est pas le cas, vous devriez vérifier si vous n'avez pas un disque plein, ou que le disque est sous haute charge.
- `Connect Out` Connexion d'un esclave sur le maître.
- `Copying to tmp table on disk` Le résultat temporaire était plus grand que `tmp_table_size` et le thread passe d'une table en mémoire à une table sur disque.
- `Creating tmp table` Le thread est en train de créer une table temporaire pour contenir le résultat

d'une requête.

- `deleting from main table` Lors de l'exécution de la première partie d'une requête d'effacement multi−table, et que MySQL n'a commencé à effacer que dans la première table.
- `deleting from reference tables` Lors de l'exécution de la deuxième partie d'une requête d'effacement multi−table, et que MySQL a commencé à effacer dans les autres tables.
- `Flushing tables` Le thread exécute la commande `FLUSH TABLES` et il attend que tous les threads ferme leur tables.
- `Killed` Quelqu'un a envoyé une commande `KILL` et le thread s'annuler la prochaine fois qu'il vérifie l'option de kill. Cette option est vérifiée dans chaque boucle majeure de MySQL, mais dans certains cas, il peut lui prendre un court instant avant de s'arrêter. Si le thread est verrouillé par un autre thread, l'arrêt va prendre effet aussitôt que l'autre thread lève son verrou.
- `Sending data` Le thread traite des lignes pour une commande `SELECT` et il envoie les données au client.
- `Sorting for group` Le thread est en train de faire un tri pour satisfaire une clause `GROUP BY` .
- `Sorting for order` Le thread est en train de faire un tri pour satisfaire une clause `ORDER BY` .
- `Opening tables` Cela signifie simplement que le thread essaie d'ouvrir une table. Ce doit être une opération très rapide, à moins que quelque chose ne retarde l'ouverture. Par exemple, une commande `ALTER TABLE` ou `LOCK TABLE` peut empêcher l'ouverture de table, jusqu'à l'achèvement de la commande.
- `Removing duplicates` La requête utilisait `SELECT DISTINCT` de telle manière que MySQL ne pouvait pas optimiser les lignes distinctes au début du traitement. A cause de cela, MySQL doit effectuer une opération de plus pour supprimer toutes les lignes en doubles, avant d'envoyer les lignes au client.
- `Reopen table` Le thread a reçu un verrou pour une table, mais a noté après l'avoir reçu que la structure de la table a changé. Il a libéré le verrou, fermé la table, et maintenant il essaie de la réouvrir.
- `Repair by sorting` Le thread répare la table en utilisant la méthode de tri pour créer l'index.
- `Repair with keycache` Le thread répare la table en utilisant la méthode de création des clés à partir du cache de clé. C'est bien plus lent que `la réparation par tri` .
- `Searching rows for update` Le thread effectue une première phase pour trouver toutes les lignes qui satisfont les critères avant de les modifier. Cela doit être fait si `UPDATE` modifie l'index qui sera utilisé pour trouver les lignes.
- `Sleeping` Le thread attend que le client envoie une nouvelle commande.
- `System lock` Le thread attend le verrou externe pour la table. Si vous n'utilisez pas de serveurs MySQL multiples qui exploitent les mêmes tables, vous pouvez désactiver les verrous systèmes avec l'option `−−skip−external−locking` .
- `Upgrading lock` Le gestionnaire de `INSERT DELAYED` essaie d'obtenir un verrou pour insérer des lignes.
- `Updating` Le thread recherche des lignes pour les modifier.
- `User Lock` Le thread attend un `GET_LOCK()` .
- `Waiting for tables` Le thread a reçu l'annonce que la structure de table a été modifiée, et il doit réouvrir la table pour obtenir une nouvelle structure. Pour être capable de réouvrir la table, il doit attendre que les autres threads aient fermé la table en question.Cette annonce survient lorsqu'un autre autre thread a été utilisé avec la commande `FLUSH TABLES` ou une des commandes suivantes, appliquées à la table en question : `FLUSH TABLES table_name` , `ALTER TABLE` , `RENAME TABLE` , `REPAIR TABLE` , `ANALYZE TABLE` ou `OPTIMIZE TABLE` .
- `waiting for handler insert` Le gestionnaire de `INSERT DELAYED` a traité toutes insertions, et en attend de nouvelles.

La plupart des états sont des opérations très rapides. Si le thread s'attarde dans un de ces états pour plusieurs secondes, il doit y avoir un problème qui mérite d'être étudié.Il existe encore d'autres états qui ne sont pas mentionné ci−dessus, mais la majorité sont utilisés pour trouver des bogues dans `mysqld` .

### 4.5.6.7 SHOW GRANTS

`SHOW GRANTS FOR user` affiche la commande nécessaire pour donner les même droits qu'un utilisateur existant.

```
 mysql> SHOW GRANTS FOR root&#64;localhost;
+-----------------------------------------------------------------------+
| Grants for root&#64;localhost                                          |
+-----------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'&#64;'localhost' WITH GRANT OPTION |
+-----------------------------------------------------------------------+
```

### 4.5.6.8 Syntaxe de SHOW CREATE TABLE

Affiche la commande `CREATE TABLE` nécessaire pour créer une table donnée.

```
 mysql> SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE t (
  id int(11) default NULL auto_increment,
  s char(60) default NULL,
  PRIMARY KEY (id)
) TYPE=MyISAM
```

`SHOW CREATE TABLE` va protéger le nom de la table et des colonnes selon l'option `SQL_QUOTE_SHOW_CREATE` . Syntaxe de `SET` .

### 4.5.6.9 SHOW WARNINGS | ERRORS

```
 SHOW WARNINGS [LIMIT #]
SHOW ERRORS [LIMIT #]
```

Cette commande est implementée en MySQL 4.1.0.Cette commande affiche les erreurs, alertes et notes qui ont été générées par la dernière commande. Les erreurs et alertes sont remises à zéro pour chaque nouvelle commande qui utilisent une table.

Le serveur MySQL retourne le nombre total d'alertes et d'erreurs que vous avez obtenu lors de la dernière commande. Ils sont disponibles avec la fonction `mysql_warning_count()` .

Jusqu'à `max_error_count` messages peuvent être stockés (variable globale et spécique aux threads).

Vous povez lire le nombre d'erreurs dans `@error_count` et le nombre d'alertes dans `@warning_count` .

`SHOW WARNINGS` affiche aussi toutes les erreurs, alertes et notes de la dernière commande, alors que `SHOW ERRORS` ne montre que les erreurs.

```
 mysql> DROP TABLE IF EXISTS no_such_table;
mysql> SHOW WARNINGS;

 +-------+------+-------------------------------+
| Level | Code | Message                       |
+-------+------+-------------------------------+
| Note  | 1051 | Unknown table 'no_such_table' |
+-------+------+-------------------------------+
```

### 4.5.6.10 SHOW TABLE TYPES

```
 SHOW TABLE TYPES
```

Cette commande est implementée en MySQL 4.1.0.

`SHOW TABLE TYPES` affiche les informations de statut concernant les types de tables. Ceci est particulièrement utilise pour vérifier si un type de table est supporté, ou pour savoir quel est le type de table par défaut.

```
mysql> SHOW TABLE TYPES;

+--------+---------+-------------------------------------------------------------+
| Type   | Support | Comment                                                     |
+--------+---------+-------------------------------------------------------------+
| MyISAM | DEFAULT | Type par défaut depuis 3.23, à hautes performances          |
| HEAP   | YES     | Basé sur des hashs, stockés en mémoire, temporaires         |
| MERGE  | YES     | Collection de tables MyISAM identiques                      |
| ISAM   | YES     | Type de table obsolète. Remplacé par MyISAM                 |
| InnoDB | YES     | Supporte les transactions, le verrouillage de ligne, et     |
|        |         | les clés étrangères.                                        |
| BDB    | NO      | Supporte les transactions et le verrouillage de pages       |
+--------+---------+-------------------------------------------------------------+
6 rows in set (0.00 sec)
```

L'option 'Support' `DEFAULT` indique si ce type de table est supporté, et dans ce cas, c'est le type par défaut. Si le serveur est démarré avec l'option `--default-table-type=InnoDB` , alors le support InnoDB contiendra la valeur `DEFAULT` .

### 4.5.6.11 SHOW PRIVILEGES

```
SHOW PRIVILEGES
```

Cette commande est implementée en MySQL 4.1.0. `SHOW PRIVILEGES` affiche la liste des droits que le serveur MySQL supporte.

```
mysql> show privileges;
+------------+-------------------------+-------------------------------------------------------+
| Privilege  | Context                 | Comment                                               |
+------------+-------------------------+-------------------------------------------------------+
| Select     | Tables                  | Lire des lignes d'une table                           |
| Insert     | Tables                  | Insérer des lignes dans une table                     |
| Update     | Tables                  | Modifier les lignes existantes                        |
| Delete     | Tables                  | Effacer des lignes existantes                         |
| Index      | Tables                  | Créer ou effacer des indexs                           |
| Alter      | Tables                  | Modifier la structure d'une table                     |
| Create     | Databases,Tables,Indexes | Créer une nouvelle base ou table                      |
| Drop       | Databases,Tables        | Effacer une base ou table                             |
| Grant      | Databases,Tables        | Donner à d'autres les droits courants                 |
| References | Databases,Tables        | Avoir des références sur les tables                   |
| Reload     | Server Admin            | Rafraîchir les droits, tables et logs                 |
| Shutdown   | Server Admin            | Eteindre le serveurver                                |
| Process    | Server Admin            | Voir la version texte des requêtes en cours           |
| File       | File access on server   | Lire et écrire des fichiers sur le serveur            |
+------------+-------------------------+-------------------------------------------------------+
14 rows in set (0.00 sec)
```

# 4.6 Localisation de MySQL et utilisation internationale

## 4.6.1 Le jeu de caractères utilisé pour les données et le stockage

Par défaut, MySQL utilise le jeu de caractères ISO-8859-1 (Latin1) avec tri en accord au Suédois/Finnois. C'est le jeu de caractère le mieux adapté pour les USA et l'Europe de l'ouest.

Tous les binaires standards MySQL sont compilés avec `--with-extra-charsets=complex` . Cela ajoutera du code à tous les programmes standards pour qu'ils puissent gérer `latin1` et tous les jeux de caractères multi-octets compris dans le binaire. Les autres jeux de caractères seront chargés à partir d'un fichier de définition de jeu si besoin.

Le jeu de caractères détermine quels caractères sont autorisés dans les noms et comment s'effectuent les tris dans les clauses `ORDER BY` et `GROUP BY` de la commande `SELECT` .

Vous pouvez changer le jeu de caractères avec l'option de démarrage du serveur `--default-character-set` . Les jeux de caractères disponibles dépendent des options `--with-charset=charset` et `--with-extra-charsets=list-of-charset | complex | all` de `configure` , et des fichiers de configuration de jeux de caractères situés dans `SHAREDIR/charsets/Index` . Les options habituelles de `configure` .

Si vous changez le jeu de caractères lors de l'utilisation de MySQL (ce qui pourra aussi changer l'ordre de tri), vous devez exécuter `myisamchk -r -q --set-character-set=charset` sur toutes les tables. Sinon, vos index pourront ne pas être ordonnés correctement.

Lorsqu'un client se connecte à un serveur MySQL, le serveur envoie le jeu de caractères utilisé par défaut au client. Le client changera de jeu de caractères pour cette connexion.

Vous devez utiliser `mysql_real_escape_string()` pour protéger les chaînes pour une requête SQL. `mysql_real_escape_string()` est identique à l'ancienne fonction `mysql_escape_string()` , excepté qu'elle prend le gestionnaire de connexion `MYSQL` en tant que premier paramètre.

Si le client est compilé avec d'autres chemins que ceux où le serveur est installé et que la personne qui a configuré MySQL n'a pas inclut tous les jeux de caractères dans le binaire MySQL, vous devez indiquer au client où il peut trouver les jeux de caractères additionnels dont il aura besoin si le serveur utilise un autre jeu de caractères que le client.

On peut le spécifier en plaçant dans un fichier d'options MySQL :

```
 [client]
character-sets-dir=/usr/local/mysql/share/mysql/charsets
```

où le chemin pointe vers le répertoire où les jeux de caractères dynamiques de MySQL sont stockés.

On peut forcer le client à utiliser un jeu de caractères spécifique en précisant :

```
 [client]
default-character-set=character-set-name
```

mais on n'en a normalement jamais besoin.

#### 4.6.1.1 Jeu de caractères allemand

Pour obtenir l'ordre de tri Allemand, vous devez démarrer `mysqld` avec `--default-character-set=latin1_de` . Cela vous donnera les caractéristiques différentes.

Lors du tri et de la comparaison des chaînes, les remplacements suivants sont faits dans la chaîne avant d'effectuer la comparaison :

```
 ä  ->  ae
ö  ->  oe
ü  ->  ue
ß  ->  ss
```

Tous les caractères accentués sont convertis en leur majuscule non-accentuée. Toutes les lettres sont transformées en majuscules.

Lors de la comparaison des chaînes de caractères avec `LIKE` le conversion un -> deux caractères n'est pas effectuée. Toutes les lettres sont transformées en majuscules. Les accents sont supprimés de toutes les lettres, à l'exception de : `Ü` , `ü` , `Ö` , `ö` , `Ä` et `ä` .

## 4.6.2 Langue des messages d'erreurs

`mysqld` peut émettre des messages d'erreurs dans les langues suivantes : Tchèque, Danois, Néerlandais, Anglais (par défaut), Estonien, Français, Allemand, Grec, Hongrois, Italien, Japonais, Coréen, Norvégien, Norwegian−ny, Polonais, Portugais, Roumain, Russe, Slovaque, Espagnol et Suédois.

Pour démarrer `mysqld` avec une langue particulière, utilisez soit l'option `−−language=lang` , soit `−L lang` . Par exemple :

```
shell> mysqld −−language=french
```

ou :

```
shell> mysqld −−language=/usr/local/share/french
```

Notez que tout les noms de langue sont spécifiés en minuscule.Les fichiers de langue sont situés (par défaut) dans `Array` .

Pour modifier le fichier de messages d'erreurs, vous devez éditer le fichier `errmsg.txt` et exécuter la commande suivante pour générer le fichier `errmsg.sys` :

```
shell> comp_err errmsg.txt errmsg.sys
```

Si vous changez de version de MySQL, pensez à modifier le nouveau fichier `errmsg.txt` .

## 4.6.3 Ajouter un nouveau jeu de caractères

Pour ajouter un autre jeu de caractères à MySQL, utilisez la procédure suivante.

Décidez s'il s'agit d'un jeu simple ou complexe. Si le jeu de caractères n'a pas besoin d'utiliser des routines d'assemblage de chaînes spéciales pour le tri et n'a pas besoin du support des jeux de caractères multi−octets, il est simple. S'il a besoin de l'une de ces deux fonctionnalités, il est complexe.

Par exemple, `latin1` et `danish` sont des jeux de caractères simples tandis que `big5` et `czech` sont complexes.

Dans la section suivante, nous supposons que vous nommez votre jeu de caractères `MONJEU` .

Pour un jeu de caractères simple, effectuez ce qui suit :

- Ajoutez MONJEU à la fin du fichier `sql/share/charsets/Index` Assignez−lui un nombre unique.
- Créez le fichier `sql/share/charsets/MONJEU.conf` . (Vous pouvez vous inspirer de `sql/share/charsets/latin1.conf` .)La syntaxe pour le fichier est très simple :
    - Les commentaires commencent avec le caractère '#' et se terminent à la fin de la ligne.
    - Les mots sont séparés par un nombre changeant d'espaces blancs.
    - Lors de la définition d'un jeu de caractères, chaque mot doit être un nombre au format hexadécimal.
    - Le tableau `ctype` prends les 257 premiers mots. Les tableaux `to_lower[]` , `to_upper[]` et `sort_order[]` prennent chacun 256 mots après cela.

  Les tableaux de définition des caractères .
- Ajoutez le nom du jeu de caractères aux listes `CHARSETS_AVAILABLE` et `COMPILED_CHARSETS` dans `configure.in` .
- Reconfigurez, recompilez et testez.

Pour un jeu de caractères complexe faites ce qui suit :

- Créez le fichier `strings/ctype−MONJEU.c` dans la distribution des sources MySQL.
- Ajoutez MONJEU à la fin du fichier `sql/share/charsets/Index` . Assignez−lui un nombre unique.
- Regardez un des fichiers `ctype−*.c` existant pour voir ce qui doit être défini, par exemple, `strings/ctype−big5.c` . Notez que les tableaux dans votre fichier doivent avoir des noms tels que `ctype_MONJEU` , `to_lower_MONJEU` , etc. Cela correspond aux tableaux dans les jeux de caractères simples. Les tableaux de définition des caractères . Pour un jeu de caractère complexe
- Au début du fichier, placez un commentaire spécial comme celui−ci :

  ```
  /*
  * This comment is parsed by configure to create ctype.c,
  * so don't change it unless you know what you are doing.
  *
  * .configure. number_MONJEU=MYNUMBER
  * .configure. strxfrm_multiply_MONJEU=N
  * .configure. mbmaxlen_MONJEU=N
  */
  ```

  Le programme `configure` utilise ce commentaire pour inclure automatiquement le jeu de caractères dans la librairie MySQL.

  Les lignes strxfrm_multiply et mbmaxlen seront expliquées dans les sections suivantes. Ne les incluez que si vous avez besoin des fonctions d'assemblage des chaînes ou des fonctions de jeu de caractères multi−octets, respectivement.
- Vous devez alors créer les fonctions suivantes :
  - `my_strncoll_MONJEU()`
  - `my_strcoll_MONJEU()`
  - `my_strxfrm_MONJEU()`
  - `my_like_range_MONJEU()`

  Support d'assemblage de chaînes .
- Ajoutez le nom du jeu de caractères aux listes `CHARSETS_AVAILABLE` et `COMPILED_CHARSETS` dans `configure.in` .
- Reconfigurez, recompilez et testez.

Le fichier `sql/share/charsets/README` fournit plus d'instructions.

Si vous voulez qu'un jeu de caractères soit ajouté dans la distribution MySQL, envoyez un patch à internals@lists.mysql.com .

## 4.6.4 Le tableau de définition des caractères

`to_lower[]` et `to_upper[]` sont de simples tableaux qui contiennent les caractères minuscules et majuscules correspondant à chaque membre du jeu de caractère. Par exemple :

```
 to_lower['A'] doit contenir 'a'
to_upper['a'] doit contenir 'A'
```

`sort_order[]` est une carte indiquant comment les caractères doivent être ordonnés pour les comparaisons et les tris. Pour beaucoup de jeux de caractères, c'est la même chose que `to_upper[]` (ce qui signifie que le tri sera insensible à la casse). MySQL triera les caractères en se basant sur la valeur de `sort_order[caractère]` . Pour des règles de tri plus compliquées, voyez la discussion suivante sur l'assemblage des chaînes. Support d'assemblage de chaînes .

`ctype[]` est un tableau de valeurs de bit, avec un élément par caractère. (Notez que `to_lower[]` , `to_upper[]` , et `sort_order[]` sont indexés par la valeur du caractère, mais que `ctype[]` est indexé par la valeur du caractère + 1. C'est une vielle habitude pour pouvoir gérer `EOF` .)

Vous pouvez trouver les définitions de bitmask suivantes dans `m_ctype.h` :

```
 #define _U      01      /* Majuscule */
#define _L      02      /* Minuscule */
#define _N      04      /* Numérique (nombre) */
```

```
#define _S      010     /* Caractère d'espacement */
#define _P      020     /* Ponctuation */
#define _C      040     /* Caractère de contrôle */
#define _B      0100    /* Blanc */
#define _X      0200    /* nombre hexadecimal */
```

L'entrée `ctype[]` de chaque caractère doit être l'union des valeurs de masque de bits qui décrivent le caractère. Par exemple, `'A'` est un caractère majuscule ( `_U` ) autant qu'une valeur hexadécimale ( `_X` ), et donc `ctype['A'+1]` doit contenir la valeur :

```
 _U + _X = 01 + 0200 = 0201
```

## 4.6.5 Support d'assemblage des chaînes

Si les règles de tri de votre langue sont trop complexes pour être gérées par le simple tableau `sort_order[]` , vous devez utiliser les fonctions d'assemblage de chaînes.

Jusqu'à présent, la meilleure documentation traitant de ce sujet est présente dans les jeux de caractères implémentés eux−mêmes. Regardez les jeux de caractères `big5` , `czech` , `gbk` , `sjis` , et `tis160` pour des exemples.

Vous devez spécifier la valeur de `strxfrm_multiply_MYSET=N` dans le commentaire spécial au début du fichier. `N` doit être le rationnel maximal vers lequel la chaîne pourra croître durant `my_strxfrm_MYSET` (cela doit être un entier positif).

## 4.6.6 Support des caractères multi−octets

Si vous voulez ajouter le support de jeu de caractères incluant des caractères multi−octets, vous devez utiliser les fonctions de caractères multi−octets.

Jusqu'à présent, la meilleure documentation traitant de ce sujet est présente dans les jeux de caractères implémentés eux−mêmes. Regardez les jeux de caractères `euc_kr` , `gb2312` , `gbk` , `sjis` , et `ujis` pour des exemples. Ils sont implémentés dans les fichiers `ctype−'charset'.c` dans le dossier `strings` .

Vous devez spécifier la valeur de `mbmaxlen_MYSET=N` dans le commentaire spécial en haut du fichier source. `N` doit être la taille en octet du caractère le plus large dans le jeu.

## 4.6.7 Problèmes avec les jeux de caractères

Si vous essayez d'utiliser un jeu de caractères qui n'est pas compilé dans votre exécutable, vous pouvez rencontrer différents problèmes :

- Votre programme a un chemin faux en ce qui concerne l'endroit où sont stockés les jeux de caractères. (Par défaut `/usr/local/mysql/share/mysql/charsets` ). Cela peut être réparé en utilisant l'option `−−character−sets−dir` du programme en question.
- Le jeu de caractères est un jeu de caractères multi−octets qui ne peut être chargé dynamiquement. Dans ce cas, vous devez recompiler le programme en incluant le support du jeu de caractère.
- Le jeu de caractères est un jeu de caractères dynamique, mais vous n'avez pas de fichier de configuration lui étant associé. Dans ce cas, vous devez installer le fichier de configuration du jeu de caractères à partir d'une nouvelle distribution MySQL.
- Votre fichier `Index` ne contient pas le nom du jeu de caractères.

  ```
   ERROR 1105: File '/usr/local/share/mysql/charsets/?.conf' not found
  (Errcode: 2)
  ```

  Dans ce cas, vous devez soit obtenir un nouveau fichier `Index` ou ajouter à la main le nom du jeu de caractères manquant.

Pour les tables `MyISAM` , vous pouvez vérifier le nom du jeu de caractères et son nombre associé d'une table avec `myisamchk -dvv nom_de_table` .

# 4.7 Scripts serveur MySQL et utilitaires

## 4.7.1 Présentation des scripts serveurs et des utilitaires

Tous les programmes MySQL prennent des options différentes. Toutefois, tous les programmes MySQL disposent de l'option `--help` qui vous aidera a connaitre la liste complète des différentes options. Essayez par exemple `mysql --help` .

Vous pouvez modifier toutes les valeurs par défaut des programmes en les plaçant dans le fichier de configuration. Fichier d'options `my.cnf` .

Voici la liste des programmes coté serveur de MySQL :

*myisamchk*
: Un utilitaire pour décrire, vérifier, optimiser et réparer les tables MySQL. Comme `myisamchk` a de très nombreuses fonctions, ce programme dispose de son propre chapitre dans la documentation. Administration de la base .

*make_binary_distribution*
: Prépare une version exécutable d'un MySQL compilé. Il doivent être envoyés par FTP à `/pub/mysql/Incoming` sur `support.mysql.com` pour en faire profiter les autres utilisateurs MySQL.

*mysqlbug*
: Le script de rapport de bogue. Ce script doit toujours être utilisé lorsque vous devez envoyez un rapport de bogues aux listes de diffusion de MySQL.

*mysqld*
: Le démon SQL. Il doit toujours être en fonctionnement.

*mysql_install_db*
: Crée les tables de droits MySQL, avec les droits par défaut. Il est généralement exécuté une fois, lors de la première installation de MySQL.

## 4.7.2 safe_mysqld , le script père de mysqld

`safe_mysqld` est la méthode recommandée pour démarrer un démon `mysqld` sous Unix. `safe_mysqld` ajoute des fonctionnalités de sécurité telles que le redémarrage automatique lorsqu'une erreur survient et l'enregistrement d'informations d'exécution dans un fichier de log.

Si vous n'utilisez pas `--mysqld=#` ou `--mysqld-version=#` `safe_mysqld` va utiliser un programme appelé `mysqld-max` s'il existe. Sinon, `safe_mysqld` va démarrer le démon `mysqld` . Cela rend très facile le test de `mysqld-max` au lieu de `mysqld` ; copiez simplement `mysqld-max` à coté de `mysqld` et il sera utilisé.

Normalement, vous ne devriez jamais éditer le script `safe_mysqld` , mais plutôt utiliser les options de `safe_mysqld` dans la section `[safe_mysqld]` du fichier `my.cnf` . `safe_mysqld` va lire toutes les options des sections `[mysqld]` , `[server]` et `[safe_mysqld]` , dans le fichier d'options. Fichier d'options `my.cnf` .

Notez que toutes les options de ligne de commande passées à `safe_mysqld` sont transmises à `mysqld` . Si vous voulez utiliser une option de `safe_mysqld` que `mysqld` ne supporte pas, vous devez la spécifier dans le fichier d'options.

La plupart des options de `safe_mysqld` sont les mêmes que celles de `mysqld` . Options en ligne de commande de `mysqld` .

`safe_mysqld` supporte les options suivantes :

*−−basedir=path*
*−−core−file−size=#*
Taille du fichier core que `mysqld` doit être capable de créer. Il est passé à `ulimit −c` .
*−−datadir=path*
*−−defaults−extra−file=path*
*−−defaults−file=path*
*−−err−log=path*
*−−ledir=path*
Chemin de `mysqld`
*−−log=path*
*−−mysqld=mysqld−version*
Nom de la version de `mysqld` dans le dossier `ledir` que vous voulez démarrer.
*−−mysqld−version=version*
Similaire à `−−mysqld=` mais vous ne donnez que le suffixe de `mysqld` . Par exemple, si vous utilisez `−−mysqld−version=max` , `safe_mysqld` va démarrer la version `ledir/mysqld−max` . Si l'argument de `−−mysqld−version` est vide, `ledir/mysqld` sera utilisé.
*−−no−defaults*
*−−open−files−limit=#*
Nombre de fichiers que `mysqld` doit être capable d'ouvrir. Passé à `ulimit −n` . Notez que vous devez démarrer `safe_mysqld` en tant que root pour que cette option fonctionne correctement !
*−−pid−file=path*
*−−port=#*
*−−socket=path*
*−−timezone=#*
Configure le fuseau horaire (le `TZ` ).
*−−user=#*

Le script `safe_mysqld` a été écrit pour qu'il soit capable de démarrer le serveur qui a été installé à partir des sources ou de la version binaire, même si l'installation de MySQL est légèrement exotique. `safe_mysqld` suppose que les conditions suivantes sont remplies :

- Le serveur et les bases de données sont placées dans un dossier relativement au dossier d'où `safe_mysqld` est appelé. `safe_mysqld` cherche dans les sous dossiers `bin` et `data` (pour les distributions binaires) et, `libexec` et `var` (pour les distributions sources). Cette condition doit être remplie si vous exécutez `safe_mysqld` depuis votre dossier d'installation MySQL (par exemple, `/usr/local/mysql` pour une distribution binaire).
- Si le serveur et les bases de données ne peuvent être trouvées dans le dossier de travail, `safe_mysqld` essaie de les trouver en utilisant leurs chemins absolus. Les chemin typiquement étudiés sont `/usr/local/libexec` et `/usr/local/var` . Les chemins réels sont déterminés lorsque la distribution est compilée, et `safe_mysqld` a alors aussi été généré. Ils doivent être corrects si MySQL a été installé dans un dossier standard.

Comme `safe_mysqld` essaie de trouver le serveur et les bases dans un dossier situé dans le dossier de travail, vous pouvez installer la version binaire de MySQL n'importe où, du moment que vous démarrez le script `safe_mysqld` dans le dossier d'installation de MySQL :

```
 shell> cd mysql_installation_directory
shell> bin/safe_mysqld &
```

Si safe_mysqld échoue, même si il est appelé depuis le dossier d'installation, vous pouvez le modifier pour qu'il reconnaisse le chemin que vous utilisez jusqu'à mysqld . Notez que si vous faites évoluer votre installation de MySQL, votre version de safe_mysqld sera écrasée, et vous devrez la rééditer.

## 4.7.3 mysqld_multi , un programme pour gérer plusieurs serveurs MySQL

mysqld_multi sert à gérer plusieurs serveurs mysqld qui utilisent différentes sockets Unix et ports TCP/IP.

Le programme va rechercher les groupes nommés [mysqld#] dans le fichier my.cnf (ou le fichier appelé --config-file=... ), où # peut être n'importe quel nombre positif, supérieur ou égal à 1. Ce nombre est appelé le numéro de groupe d'options. Les numéros de groupe permettent de distinguer un groupe d'options d'un autre, et sont utilisés comme argument du script mysqld_multi pour spécifier quel serveur vous voulez démarrer, arrêter ou examiner. Les options listées dans ces groupes doivent être les mêmes que celle que vous utiliseriez dans une section dédiée au démon [mysqld] . Voyez, par exemple, Démarrer et arrêter MySQL automatiquement . Cependant, pour mysqld_multi , vous devez vous assurer que chaque groupe contient des valeurs pour les options telles que port, socket, etc., qui seront utilisées par chaque processus mysqld .

mysqld_multi est utilisé ave la syntaxe suivante :

```
 Usage: mysqld_multi [OPTIONS] &#123;start|stop|report&#125; [GNR,GNR,GNR...]
or     mysqld_multi [OPTIONS] &#123;start|stop|report&#125; [GNR-GNR,GNR,GNR-GNR,...]
```

Chaque GNR représente un numéro de groupe d'options. Vous pouvez démarrer, arrêter ou examiner n'importe quel numéro de groupe d'options, ou même plusieurs d'entre eux en même temps. Pour un exemple de comment configurer le fichier d'options, utilisez cette commande :

```
shell> mysqld_multi --example
```

Les valeurs de numéro de groupe d'options peuvent être une liste de valeurs séparées par une virgule ou un tiret. Dans ce dernier cas, toutes les numéro de groupe d'options situés entre les deux numéros seront alors affectés. Sans numéro de groupe d'options spécifié, tous les numéros de groupes du fichier d'options sont affectés. Notez que vous ne devez pas avoir d'espace dans la liste des numéros de groupe d'options. Tout ce qui est placé au-delà de l'espace sera ignoré.

mysqld_multi supporte les options suivantes :

*--config-file=...*
: Un fichier de configuration alternatif. Note : cela ne va pas modifier les options de ce programme ( [mysqld_multi] ), mais uniquement les groupes [mysqld#] . Sans cette option, tout sera lu dans le fichier d'options traditionnel my.cnf .

*--example*
: Affiche un exemple de fichier de configuration.

*--help*
: Affiche l'aide et quitte.

*--log=...*
: Fichier de log. Le chemin complet et le nom du fichier sont nécessaires. Note : si le fichier existe déjà, les prochaines données seront ajoutées au fichier.

*--mysqladmin=...*
: L'exécutable mysqladmin à utiliser lors de l'arrêt du serveur.

*--mysqld=...*
: L'exécutable mysqld à utiliser. Notez que vous pouvez donner cette option à safe_mysqld . Ces options sont passées à mysqld . Assurez-vous que vous avez bien mysqld dans votre variable d'environnement PATH ou corrigez safe_mysqld .

*--no-log*

Affiche les données d'historique à l'écran plutôt que dans le fichier de log. Par défaut, le fichier de log est activé.

*−−password=...*
: Le mot de passe de l'utilisateur `mysqladmin` .

*−−tcp−ip*
: Connexion au serveur MySQL via le port TCP/IP au lieu de la socket Unix. Cela affecte l'arrêt et le rapport. Si le fichier de socket manque, le serveur peut continuer de tourner, mais il n'est plus accessible que par port TCP/IP. Par défaut, les connexions sont faites avec les sockets Unix.

*−−user=...*
: L'utilisateur MySQL pour `mysqladmin` .

*−−version*
: Affiche le numéro de version et quitte.

Quelques notes pour `mysqld_multi` :

- Assurez−vous que l'utilisateur MySQL, qui stoppe les services `mysqld` (e.g en utilisant la commande `mysqladmin` ), a les mêmes nom d'utilisateur et mot de passe pour tous les dossiers de données utilisés. Et assurez−vous que cet utilisateur a bien les droits de `SHUTDOWN` ! Si vous avez de nombreux dossiers de données et de nombreuses bases `mysql` avec différents mots de passe pour le serveur `root` MySQL, vous souhaiterez peut être créer un utilisateur commun `multi_admin` à chaque base, avec le même mot de passe (voir ci−dessous). Voici comment faire :
```
 shell> mysql −u root −S /tmp/mysql.sock −proot_password −e
"GRANT SHUTDOWN ON *.* TO multi_admin&#64;localhost IDENTIFIED BY 'multipass'"
```
Comment le système de droits fonctionne . Vous devrez utiliser la même commande pour chaque serveur `mysqld` qui fonctionne : changez simplement la socket, `−S=...` ).
- `pid−file` est très important, si vous utilisez `safe_mysqld` pour démarrer `mysqld` (e.g., `−−mysqld=safe_mysqld` ). Chaque `mysqld` doit avoir son propre fichier `pid−file` . L'avantage d'utiliser `safe_mysqld` au lieu de `mysqld` est que `safe_mysqld` ``surveille'' tous les processus `mysqld` et les redémarrera si un processus `mysqld` s'arrête suite à la reception d'un signal `kill −9` , ou pour toute autre raison comme une erreur de segmentation (que MySQL ne devrait jamais faire, bien sûr !). Notez bien que le script `safe_mysqld` vous imposera peut être d'être démarré depuis un dossier spécial. Cela signifie que vous devrez probablement utiliser la commande shell `cd` jusqu'à un certain dossier avant de pouvoir exécuter `mysqld_multi` . Si vous avez des problèmes pour démarrer, voyez le script `safe_mysqld` . Vérifiez notamment ces lignes :
```
 ----------------------------------------------------------------------------
MY_PWD=`pwd` Check if we are starting this relative (for the binary
release) if test −d /data/mysql −a −f ./share/mysql/english/errmsg.sys
−a −x ./bin/mysqld
----------------------------------------------------------------------------
```
`safe_mysqld` , le script père de `mysqld` . Le test ci−dessus devrait fonctionner, ou bien vous rencontrerez probablement des problèmes.
- N'oubliez jamais les problèmes que peut soulever le démarrage de plusieurs démons `mysqld` avec le même dossier de données. Utilisez des dossiers de données séparés, à moins que vous ne **sachiez bien** ce que vous faites.
- Le fichier de socket et le port TCP/IP doivent être différents pour chaque `mysqld` .
- Le premier et le cinquième groupe `mysqld` on été intentionnellement ignorés dans le groupe d'exemple. Vous pouvez laisser des trous dans le fichier de configuration. Cela vous donnera plus de souplesse. L'ordre dans lequel les démons `mysqlds` sont démarrés ou arrétés dépend de l'ordre dans lequel ils apparaissent dans le fichier de configuration.
- Lorsque vous voulez faire référence à un groupe en utilisant le numéro de groupe d'options, utilisez simplement le numéro du groupe à la fin du nom du groupe. Par exemple, le numéro de groupe de `[mysqld17]` est le 17.
- Vous pouvez vouloir utiliser l'option `−−user` pour `mysqld` , mais afin de faire cela, vous devez exécuter le script `mysqld_multi` en tant que `root` Unix. Placer cette option dans le fichier de configuration ne changera rien : vous obtiendrez une alerte, si vous n'êtes pas le super

utilisateur, et les démons `mysqld` seront démarrés avec vos droits Unix. **Important** : assurez−vous bien que le fichier de données et le fichier de `pid−file` sont accessibles en lecture et écriture (et exécution pour le dernier) à l'utilisateur Unix qui lance les processus `mysqld` . **N'utilisez pas** le compte root Unix pour cela, à moins que vous ne **sachiez** ce que vous faîtes.

- **Très important** : assurez−vous de bien comprendre la signification des options que vous passez à `mysqld` s et **pourquoi vous avez besoin** de plusieurs processus `mysqld` . Démarrer plusieurs serveurs `mysqld` s dans le même dossier **ne vous donnera aucun** gain de performance dans un système threadé.

Faire fonctionner plusieurs serveurs MySQL sur la même machine .Voici un exemple de fichier de configuration fourni par `mysqld_multi` .

```
 # This file should probably be in your home dir (~/.my.cnf) or /etc/my.cnf
# Version 2.1 by Jani Tolonen

 [mysqld_multi]
mysqld     = /usr/local/bin/safe_mysqld
mysqladmin = /usr/local/bin/mysqladmin
user       = multi_admin
password   = multipass

 [mysqld2]
socket     = /tmp/mysql.sock2
port       = 3307
pid−file   = /usr/local/mysql/var2/hostname.pid2
datadir    = /usr/local/mysql/var2
language   = /usr/local/share/mysql/english
user       = john

 [mysqld3]
socket     = /tmp/mysql.sock3
port       = 3308
pid−file   = /usr/local/mysql/var3/hostname.pid3
datadir    = /usr/local/mysql/var3
language   = /usr/local/share/mysql/swedish
user       = monty

 [mysqld4]
socket     = /tmp/mysql.sock4
port       = 3309
pid−file   = /usr/local/mysql/var4/hostname.pid4
datadir    = /usr/local/mysql/var4
language   = /usr/local/share/mysql/estonia
user       = tonu

 [mysqld6]
socket     = /tmp/mysql.sock6
port       = 3311
pid−file   = /usr/local/mysql/var6/hostname.pid6
datadir    = /usr/local/mysql/var6
language   = /usr/local/share/mysql/japanese
user       = jani
```

Fichier d'options `my.cnf` .

## 4.7.4 myisampack , le générateur de tables MySQL compressées en lecture seule

`myisampack` sert à compresser des tables MyISAM et `pack_isam` sert à compresser les tables ISAM. Comme les tables ISAM sont obsolètes, nous ne traiterons que de `myisampack` , mais tout ce qui est dit au sujet de `myisampack` est aussi vrai pour `pack_isam` .

`myisampack` fonctionne en compressant séparément chaque colonne de la table. Les informations nécessaires à la décompression sont lues en mémoire lorsque la table est ouverte. Cela donne de bien meilleures performances lors de l'accès à des lignes individuelles, car nous n'avez qu'à décompresser exactement une des lignes, et non pas un bloc de disque entier. Généralement, `myisampack` compresse le fichier avec un gain de 40 à 70 %.

MySQL utilise la carte mémoire ( `mmap()` ) sur les tables compressées et utilise les outils classiques de lecture et écriture si `mmap()` ne fonctionne pas.Notez bien ceci :

- Après avoir compressé la table, celle−ci n'est plus accessible qu'en lecture. C'est souvent un état voulu (par exemple, pour être gravée sur un CD). De plus, autoriser les écritures dans une table compressée fait partie de notre liste de tâche, mais avec une très faible priorité.
- `myisampack` peut aussi compresser des colonnes `BLOB` ou `TEXT` . L'ancien `pack_isam` (pour les tables `ISAM` ) ne peut le faire.

`myisampack` est invoqué comme ceci :

```
shell> myisampack [options] filename ...
```

Chaque nom de fichier doit être le nom d'un fichier d'index ( `.MYI` ). Si vous n'êtes pas dans le dossier de données, vous devez spécifier le chemin complet jusqu'au fichier. Il est toléré que vous omettiez l'extension du fichier `.MYI` .

`myisampack` supporte les options suivantes :

*−b, −−backup*
: Fait une sauvegarde de la table sous le nom de `tbl_name.OLD` .

*−#, −−debug=debug_options*
: Affiche le log de débogage. La chaîne `debug_options` vaut souvent `'d:t:o,filename'` .

*−f, −−force*
: Force la compression de la table, même si elle grossit ou si le fichier temporaire existe déjà. `myisampack` crée un fichier temporaire appelé `tbl_name.TMD` lors de la compression. Si vous tuez `myisampack` , le fichier `.TMD` peut ne pas être effacé. Normalement, `myisampack` se termine avec une erreur s'il découvre que le fichier `tbl_name.TMD` existe. Avec `--force` , `myisampack` reprendra le travail.

*−?, −−help*
: Affiche le message d'aide et quitte.

*−j big_tbl_name, −−join=big_tbl_name*
: Rassemble toutes les tables indiquées dans la ligne de commande dans une seule table appelée `big_tbl_name` . Toutes les tables qui seront combinées **doivent** être identiques (mêmes noms de colonnes, mêmes types, mêmes index, etc.)

*−p #, −−packlength=#*
: Spécifie la taille de stockage de la longueur de ligne, en octets. Cette valeur doit être 1, 2, ou 3. ( `myisampack` stocke toutes les lignes avec des pointeurs de lignes de 1, 2 ou 3 octets. Dans les cas normaux, `myisampack` peut déterminer la taille correcte avant de compresser le fichier, mais il peut aussi se rendre compte durant le processus qu'une autre taille aurait été plus appropriée, ou plus courte. Dans ce cas, `myisampack` va imprimer une note pour que vous le sachiez lors de la prochaine compression du même fichier.

*−s, −−silent*
: Mode silencieux. Seules les erreurs seront affichées.

*−t, −−test*
: Ne compresse pas la table, mais teste juste la compression.

*−T dir_name, −−tmp_dir=dir_name*
: Utilise le dossier indiqué comme dossier pour les fichiers temporaires.

*−v, −−verbose*
: Mode détaillé. Toutes les informations sur la progression de la compression seront affichées.

*−V, −−version*
: Affiche la version et quitte.

*−w, −−wait*

Attend et reessaie, si la table était déjà en cours d'utilisation. Si le serveur `mysqld` a été démarré avec l'option `--skip-external-locking` , ce n'est pas une bonne idée d'appeler `myisampack` , car la table risque d'être modifiée durant la compression.

La séquence de commande illustre la session de compression :

```
 shell> ls -l station.*
-rw-rw-r--   1 monty    my         994128 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          53248 Apr 17 19:00 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

 shell> myisamchk -dvv station

 MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-02-02  3:06:43
Data records:              1192  Deleted blocks:              0
Datafile: Parts:           1192  Deleted data:                0
Datafile pointer (bytes):     2  Keyfile pointer (bytes):     2
Max datafile length:   54657023  Max keyfile length:   33554431
Recordlength:               834
Record format: Fixed length

 table description:
Key Start Len Index   Type                      Root  Blocksize    Rec/key
1   2     4   unique  unsigned long              1024       1024          1
2   32    30  multip. text                      10240       1024          1

 Field Start Length Type
1     1     1
2     2     4
3     6     4
4     10    1
5     11    20
6     31    1
7     32    30
8     62    35
9     97    35
10    132   35
11    167   4
12    171   16
13    187   35
14    222   4
15    226   16
16    242   20
17    262   20
18    282   20
19    302   30
20    332   4
21    336   4
22    340   1
23    341   8
24    349   8
25    357   8
26    365   2
27    367   2
28    369   4
29    373   4
30    377   1
31    378   2
32    380   8
33    388   4
34    392   4
35    396   4
36    400   4
37    404   1
38    405   4
39    409   4
40    413   4
41    417   4
42    421   4
43    425   4
44    429   20
45    449   30
46    479   1
47    480   1
48    481   79
49    560   79
50    639   79
51    718   79
52    797   8
53    805   1
54    806   1
55    807   20
56    827   4
57    831   4
```

```
 shell> myisampack station.MYI
Compressing station.MYI: (1192 records)
- Calculating statistics

 normal:     20  empty-space:      16  empty-zero:        12  empty-fill:  11
pre-space:   0  end-space:        12  table-lookups:      5  zero:         7
Original trees:  57  After join: 17
- Compressing file
87.14%

 shell> ls -l station.*
-rw-rw-r--   1 monty    my         127874 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          55296 Apr 17 19:04 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

 shell> myisamchk -dvv station

 MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-04-17 19:04:26
Data records:              1192  Deleted blocks:              0
Datafile: Parts:           1192  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     1
Max datafile length:   16777215  Max keyfile length:     131071
Recordlength:               834
Record format: Compressed

 table description:
Key Start Len Index   Type                        Root  Blocksize    Rec/key
1   2     4   unique  unsigned long             10240       1024          1
2   32    30  multip. text                      54272       1024          1

 Field Start Length Type                         Huff tree  Bits
1     1     1      constant                            1     0
2     2     4      zerofill(1)                         2     9
3     6     4      no zeros, zerofill(1)               2     9
4     10    1                                          3     9
5     11    20     table-lookup                        4     0
6     31    1                                          3     9
7     32    30     no endspace, not_always             5     9
8     62    35     no endspace, not_always, no empty   6     9
9     97    35     no empty                            7     9
10    132   35     no endspace, not_always, no empty   6     9
11    167   4      zerofill(1)                         2     9
12    171   16     no endspace, not_always, no empty   5     9
13    187   35     no endspace, not_always, no empty   6     9
14    222   4      zerofill(1)                         2     9
15    226   16     no endspace, not_always, no empty   5     9
16    242   20     no endspace, not_always             8     9
17    262   20     no endspace, no empty               8     9
18    282   20     no endspace, no empty               5     9
19    302   30     no endspace, no empty               6     9
20    332   4      always zero                         2     9
21    336   4      always zero                         2     9
22    340   1                                          3     9
23    341   8      table-lookup                        9     0
24    349   8      table-lookup                       10     0
25    357   8      always zero                         2     9
26    365   2                                          2     9
27    367   2      no zeros, zerofill(1)               2     9
28    369   4      no zeros, zerofill(1)               2     9
29    373   4      table-lookup                       11     0
30    377   1                                          3     9
31    378   2      no zeros, zerofill(1)               2     9
32    380   8      no zeros                            2     9
33    388   4      always zero                         2     9
34    392   4      table-lookup                       12     0
35    396   4      no zeros, zerofill(1)              13     9
36    400   4      no zeros, zerofill(1)               2     9
37    404   1                                          2     9
38    405   4      no zeros                            2     9
39    409   4      always zero                         2     9
40    413   4      no zeros                            2     9
41    417   4      always zero                         2     9
42    421   4      no zeros                            2     9
43    425   4      always zero                         2     9
44    429   20     no empty                            3     9
45    449   30     no empty                            3     9
46    479   1                                         14     4
47    480   1                                         14     4
48    481   79     no endspace, no empty              15     9
49    560   79     no empty                            2     9
50    639   79     no empty                            2     9
51    718   79     no endspace                        16     9
52    797   8      no empty                            2     9
53    805   1                                         17     1
54    806   1                                          3     9
55    807   20     no empty                            3     9
56    827   4      no zeros, zerofill(2)               2     9
57    831   4      no zeros, zerofill(1)               2     9
```

Les informations affichées par `myisampack` sont décrites ici :

*normal*
Le nombre de colonnes pour lesquelles aucune compression n'est utilisée.

*empty−space*
Le nombre de colonnes dont les valeurs ne contiennent que des octets : elles n'occuperont plus qu'un octet.

*empty−zero*
Le nombre de colonnes dont les valeurs ne contiennent que des zéros : elles n'occuperont plus qu'un octet.

*empty−fill*
Le nombre de colonnes de type entier qui n'occupent pas la totalité de l'espace de leur type. Elles seront réduites en taille (par exemple, une colonne de type `INTEGER` sera transformée en `MEDIUMINT` ).

*pre−space*
Le nombre de colonnes de nombres à virgule flottante qui ont des valeurs stockées avec des espaces initiaux. Dans ce cas, chaque valeur va contenir le nombre d'espace initiaux.

*end−space*
Le nombre de colonnes qui ont de nombreux espaces terminaux. Dans ce cas, chaque valeur va contenir un compte du nombre d'espaces terminaux.

*table−lookup*
La colonne n'a que quelques valeurs différentes, qui seront converties en une colonne de type `ENUM` avant une compression de type Huffman.

*zero*
Le nombre de colonnes pour lesquelles toutes les valeurs sont zéro.

*Original trees*
Le nombre initial d'arbres Huffman.

*After join*
Le nombre d'arbres Huffman distincts obtenus après avoir joint les arbres pour économiser de l'espace d'entête.

Après la compression d'une table, `myisamchk −dvv` affiche des informations supplémentaires pour chaque champ :

*Type*
Le type de fichier peut contenir les informations suivantes :

*constant*
Toutes les lignes ont la même valeur.

*no endspace*
Ne stocke pas les espaces finaux.

*no endspace, not_always*
Ne stocke pas les espaces finaux et ne compresse pas les espaces finaux pour toutes les valeurs.

*no endspace, no empty*
Ne stocke pas les espaces finaux. Ne stocke pas les valeurs vides.

*table−lookup*
La colonne a été convertie en `ENUM` .

*zerofill(n)*
Les `n` chiffres significatifs sont toujours 0, et n'ont pas été stockés.

*no zeros*
Ne stocke pas les zéros.

*always zero*
Les valeurs 0 sont stockées sur un octet.

*Huff tree*
L'arbre Huffman associé au champ.

*Bits*
Le nombre de bits utilisés par l'arbre Huffman.

Après la compression de pack_isam / myisampack vous devez exécuter la commande isamchk / myisamchk pour recréer l'index. A ce moment, vous pouvez aussi trier les blocs d'index et créer des statistiques nécessaires pour l'optimisateur MySQL :

```
 myisamchk -rq --analyze --sort-index table_name.MYI
isamchk   -rq --analyze --sort-index table_name.ISM
```

Après avoir installé la table compressée dans un dossier de données MysQL, vous devez exécutez la commande mysqladmin flush-tables pour forcer mysqld à utiliser cette nouvelle table.

Si vous voulez décompresser une table compressée, vous pouvez le faire avec l'option --unpack de la commande isamchk ou myisamchk .

## 4.7.5 mysqld-max , la version étendue du serveur mysqld

mysqld-max est le serveur MySQL ( mysqld ) configuré avec les options suivantes :

| Option | Comment |
|---|---|
| --with-server-suffix=-max | Ajoute un suffixe à la chaîne de version de mysqld |
| --with-innodb | Supporte les tables InnoDB . |
| --with-bdb | Supporte les tables Berkeley DB (BDB) |
| CFLAGS=-DUSE_SYMDIR | Support des liens symboliques sous Windows. |

Vous pouvez obtenir les exécutables MySQL-max à http://www.mysql.com/downloads/mysql-max-3.23.html .

La distribution Windows de MySQL inclut les deux programmes standard mysqld.exe et mysqld-max.exe . http://www.mysql.com/downloads/mysql-3.23.html . Installer MySQL sous Windows .

Notez que les tables InnoDB et Berkeley DB ne sont pas disponibles sur toutes les plate-formes, et que certaines versions Max ne supportent pas les deux. Vous pouvez vérifier quelles tables sont supportées avec la requête suivante :

```
 mysql> SHOW VARIABLES LIKE "have_%";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_bdb      | YES   |
| have_innodb   | NO    |
| have_isam     | YES   |
| have_raid     | NO    |
| have_openssl  | NO    |
+---------------+-------+
```

La signification des valeurs est :

| Valeur | Signification |
|---|---|
| YES | L'option est activatée et utilisable. |
| NO | MySQL n'a pas été compilé avec le support pour cette option. |
| DISABLED | L'option xxxx est désactivée car mysqld a été lancé avec --skip-xxxx ou parce que mysqld n'a pas été lancé avec les bonnes options. Dans ce cas, le fichier hostname.err devrait contenir la raison pour laquelle l'option est désactivée. |

**Note** : pour être capable de créer des tables InnoDB , vous **devez** éditer vos options de démarrage et inclure au moins l'option innodb_data_file_path . Options de démarrage InnoDB .

Pour avoir de meilleures performances avec les tables BDB, vous devriez aussi ajouter certaines options de configuration. Options de démarrage BDB .

safe_mysqld va automatiquement lancer tout programme mysqld avec le suffixe -max . Cela rend très facile les tests de mysqld , dans le cadre d'un installation existante. Il suffit d'exécuter le programme configure avec les options que vous souhaitez, et d'installer un nouveau programme mysqld sous le nom de mysqld-max dans le même dossier que votre ancien mysqld . safe_mysqld , le script père de mysqld .

Les paquets RPM `mysqld-max` utilisent la fonctionnalité de `safe_mysqld` mentionnée plus tôt. Il suffit d'installer le programme `mysqld-max` et `safe_mysqld` va automatiquement utiliser cet exécutable lorsque `safe_mysqld` sera redémarré.

La table suivante illustre les types de tables que MySQL−Max inclut :

| **Système** | BDB | InnoDB |
|---|---|---|
| AIX 4.3 | Non | Oui |
| HP−UX 11.0 | Non | Oui |
| Linux−Alpha | Non | Oui |
| Linux−Intel | Oui | Oui |
| Linux−IA64 | Non | Oui |
| Solaris−Intel | Non | Oui |
| Solaris−SPARC | Oui | Oui |
| Caldera (SCO) OSR5 | Oui | Oui |
| UnixWare | Oui | Oui |
| Windows/NT | Oui | Oui |

# 4.8 MySQL Scripts clients et utilitaires

## 4.8.1 Présentation des scripts serveurs et utilitaires

Tous les clients MySQL qui communiquent avec le serveur via la librairie `mysqlclient` utilisent les variables d'environnement suivantes :

| **Nom** | **Description** |
|---|---|
| MYSQL_UNIX_PORT | La socket par défaut; utilisé pour les connexions à localhost |
| MYSQL_TCP_PORT | Le port TCP/IP par défaut |
| MYSQL_PWD | Le mot de passe par défaut |
| MYSQL_DEBUG | Des options de débogage et de traçage |
| TMPDIR | Le dossier où les fichiers et tables temporaires sont écrits |

L'utilisation de `MYSQL_PWD` n'est pas une technique sûre. Connexion au serveur MySQL .

Le client `mysql` utilise le fichier indiqué dans la variable d'environnement `MYSQL_HISTFILE` pour sauver l'historique de la ligne de commande. La valeur par défaut pour la situation du fichier d'historique est `$HOME/.mysql_history` , où `$HOME` est la valeur de la variable d'environnement `HOME` . Variables d'environnement MySQL .

Tous les programmes MySQL utilisent différentes options. Toutefois, tous les programmes MySQL disposent de l'option `--help` pour vous donner une description complète de toutes les options du programme. Par exemple, essayez d'utiliser `mysql --help` .

Vous pouvez remplacer les valeurs par défaut des options de tous les clients standard en mettant d'autres valeurs dans le fichier de configuration. Fichier d'options `my.cnf` .

Voici un bref aperçu des programmes clients MySQL :

*msql2mysql*
: Un script shell qui convertit les programmes `mSQL` vers MySQL. Il ne gère pas toutes les situations, mais c'est une très bonne base de travail.

*mysqlaccess*
: Un script qui vérifie les droits d'accès du trio hôte, utilisateur et base de données.

*mysqladmin*
: Un utilitaire pour réaliser des opérations d'administration de la base, telles que les créations de bases, le rafraîchissement des tables de droits, l'écriture des tables sur

le disque et la réouverture des fichiers de log. à `mysqladmin` permet aussi de lire la version, les processus et les informations de statut du serveur. `mysqladmin` , administrer un serveur MySQL .

*mysqldump*
: Exporte une base de données MySQL dans un fichier sous la forme de requêtes SQL, ou de fichiers texte, avec la tabulation comme séparateur. Un freeware amélioré, d'après une idée originale de Igor Romanenko. `mysqldump` , exporter les structures de tables et les données .

*mysqlimport*
: Importe les fichiers textes dans les tables, en utilisant la commande `LOAD DATA INFILE` . `mysqlimport` , importer des données depuis des fichiers texte .

*mysqlshow*
: Affiche des informations sur les bases, tables, colonnes et index.

*replace*
: Un utilitaire qui est utilisé par `msql2mysql` , qui a d'autres applications générales. `replace` modifie des chaînes dans des fichiers, ou sur l'entrée standard. Utilise une machine d'états pour rechercher les plus grands chaînes en premier. Sert à échanger des chaînes. Par exemple, cette commande échange les chaînes `a` et `b` dans les fichiers spécifiés :

```
shell> replace a b b a -- file1 file2 ...
```

## 4.8.2 mysql , l'outil en ligne de commande

`mysql` est un simple script SQL (qui exploite GNU `readline` ). Il supporte une utilisation interactive et non−interactive. Lorsqu'il est utilisé interactivement, les résultats des requêtes sont présentés sous la forme d'une table au format ASCII. Lorsqu'il est utilisé non−interactivement, par exemple, comme filtre, le résultat est fourni au format de liste avec séparation par tabulation (le format d'affichage peut être modifié en utilisant les options de ligne de commande). Vous pouvez exécuter le script comme ceci :

```
shell> mysql database < script.sql > output.tab
```

Si vous avez des problèmes liés à des insuffisances de mémoire avec le client, utilisez l'option `--quick` ! Cela force `mysql` à utiliser `mysql_use_result()` plutôt que `mysql_store_result()` pour lire les résultats.Utiliser `mysql` est très simple. Il suffit de le démarrer comme ceci : `mysql database` ou `mysql --user=user_name --password=your_password database` . Tapez une commande SQL, puis terminez−la avec `';'` , `'\g'` ou `'\G'` , et finissez avec Entrée.

`mysql` supports the following options:

*−?, −−help*
: Affiche cette aide et quitte.

*−A, −−no−auto−rehash*
: Pas de rehachage automatique. Il faut utiliser la commande 'rehash' pour obtenir la complétion des noms de tables et champs. Cela accélère le démarrage du client.

*−−prompt=...*
: Modifie l'invite de commande de MySQL.

*−b, −−no−beep*
: Eteind le son d'erreur.

*−B, −−batch*
: Affiche les résultats avec une tabulation comme résultat, et chaque ligne avec une nouvelle ligne. N'utilise pas l'historique.

*−−character−sets−dir=...*
: Le dossier où les jeux de caractères sont créés.

*−C, −−compress*

Utilise la compression avec le protocole client−serveur.

*−#, −−debug[=...]*

Démarre le log de débogage. Par défaut, il vaut 'd:t:o,/tmp/mysql.trace'.

*−D, −−database=...*

La base de données à utiliser. C'est particulièrement pratique dans le fichier d'options `my.cnf` .

*−−default−character−set=...*

Configure le jeu de caractères par défaut.

*−e, −−execute=...*

Exécute une commande et quitte. Le résultat est au format de l'option −−batch

*−E, −−vertical*

Affiche le résultat d'une requête verticalement. Sans cette option, vous pouvez aussi obtenir ce format en terminant votre requête avec `\G` .

*−f, −−force*

Continue même si vous obtenez une erreur SQL.

*−g, −−no−named−commands*

Les commandes nommées sont désactivées. Utilisez la forme * uniquement, ou utilisez les commandes nommées au début d'une ligne se terminant par un point−virgule ( `';'` ). Depuis la version 10.9, le client démarre avec cette option **activée** par défaut. Avec l'option −g, le format long des commandes va continuer à fonctionner.

*−G, −−enable−named−commands*

Les commandes nommées sont **activées** . Le format long est autorisé, ainsi que les commandes courtes *.

*−i, −−ignore−space*

Ignore les espaces après les noms de fonctions.

*−h, −−host=...*

Connexion avec l'hôte indiqué.

*−H, −−html*

Produit un résultat au format HTML.

*−X, −−xml*

Produit un résultat au format XML.

*−L, −−skip−line−numbers*

N'écrit pas les numéros de lignes dans les erreurs. Très pratique lorsque vous voulez comparer des résultats qui incluent des messages d'erreurs.

*−−no−pager*

Désactive le système de page, et affiche directement dans la sortie standard. Voyez l'aide interactive (\h).

*−−no−tee*

Désactive le outfile. Voyez l'aide interactive (\h).

*−n, −−unbuffered*

Vide le buffer de requête après chaque requête.

*−N, −−skip−column−names*

N'écrit pas les noms de colonnes dans les résultats.

*−O, −−set−variable var=option*

Spécifie la valeur d'une variable. `−−help` liste toutes ces variables. Notez bien que `−−set−variable` est obsolète depuis MySQL 4.0, et qu'il suffit désormais d'utiliser `−−var=option` .

*−o, −−one−database*

Ne modifie que la base par défaut. C'est pratique pour éviter les modifications dans les autres bases dans le fichier de log.

`−−pager[=...]`

Type d'affichage. Par défaut, la variable d'environnement `ENV` vaut `PAGER` . Les paginateurs valides sont less, more, cat [> filename], etc. Voyez l'aide interactive (\h). Cette option n'est pas effective en mode batch. Les paginateurs ne fonctionnent

qu'avec Unix.

*−p[password], −−password[=...]*
: Le mot de passe utilisé lors de la connexion sur le serveur. S'il n'est pas donné en ligne de commande, il sera demandé interactivement. Notez que si vous utilisez la forme courte `-p` , vous ne devez pas laisser d'espace entre l'option et le mot de passe.

*−P −−port=...*
: Le numéro de port TCP/IP pour la connexion.

*−q, −−quick*
: Ne met pas en cache le résultat, et l'affiche ligne par ligne. C'est plus lent pour le serveur, si le résultat est interrompu. N'utilise pas le fichier d'historique.

*−r, −−raw*
: Ecrit les valeurs des colonnes sans les conversions de protections. Utilisé en mode `--batch`

*−s, −−silent*
: Mode très silencieux.

*−S −−socket=...*
: Le fichier de socket à utiliser pour la connexion.

*−t −−table*
: Affichage au format de table. C'est le mode par défaut pour le mode non−batch.

*−T, −−debug−info*
: Affiche des informations de débogage au moment de la fin du programme.

*−−tee=...*
: Ajoute tout dans le fichier de sortie. Voyez l'aide interactive (\h). Ne fonctionne pas en mode batch.

*−u, −−user=#*
: Nom d'utilisateur pour la connexion, si ce n'est pas l'utilisateur Unix courant.

*−U, −−safe−updates[=#], −−i−am−a−dummy[=#]*
: N'autorise que les commandes `UPDATE` et `DELETE` qui utilisent des clés. Voir plus bas pour des informations sur cette option. Vous pouvez annuler cette option si vous l'avez dans le fichier d'option `my.cnf` en utilisant la syntaxe `--safe-updates=0` .

*−v, −−verbose*
: Affichage plus détaillé (−v −v −v indique le format d'affichage de table).

*−V, −−version*
: Affiche la version et quitte.

*−w, −−wait*
: Attend et retente si la connexion s'interrompt, au lieu de quitter.

Vous pouvez aussi spécifier les variables suivantes avec l'option `-O` ou `--set-variable` ; notez bien que `--set-variable` est obsolète depuis MySQL 4.0, utilisez la nouvelle syntaxe `--var=option` :

| Nom de variable | Par défaut | Description |
|---|---|---|
| connect_timeout | 0 | Nombre de secondes avant que la connexion n'expire. |
| max_allowed_packet | 16777216 | Taille maximale du paquet de communication avec le serveur. |
| net_buffer_length | 16384 | Buffer pour les communications TCP/IP et socket. |
| select_limit | 1000 | Limite automatique pour les commandes SELECT avec l'option −−i−am−a−dummy |
| max_join_size | 1000000 | Limite automatique pour les commandes de jointure avec l'option −−i−am−a−dummy. |

Si vous tapez 'help' en ligne de commande, `mysql` va afficher les commandes qu'il supporte :

```
 mysql> help

 MySQL commands:
help    (\h)    Display this text.
?       (\h)    Synonym for `help'.
clear   (\c)    Clear command.
connect (\r)    Reconnect to the server.
                Optional arguments are db and host.
edit    (\e)    Edit command with $EDITOR.
ego     (\G)    Send command to mysql server,
                display result vertically.
exit    (\q)    Exit mysql. Same as quit.
```

```
go      (\g)    Send command to mysql server.
nopager (\n)    Disable pager, print to stdout.
notee   (\t)    Don't write into outfile.
pager   (\P)    Set PAGER [to_pager].
                Print the query results via PAGER.
print   (\p)    Print current command.
prompt  (\R)    Change your mysql prompt.
quit    (\q)    Quit mysql.
rehash  (\#)    Rebuild completion hash.
source  (\.)    Execute a SQL script file.
                Takes a file name as an argument.
status  (\s)    Get status information from the server.
tee     (\T)    Set outfile [to_outfile].
                Append everything into given outfile.
use     (\u)    Use another database.
                Takes database name as argument.
```

La commande `pager` ne fonctionne que sous Unix.

La commande `status` donne des détails sur la connexion et le serveur utilisés. Si vous fonctionnez en mode `--safe-updates` , `status` va aussi afficher les valeurs des variables de `mysql` qui affectent vos requêtes.

Une option de démarrage utile pour les débutants (introduit en MySQL version 3.23.11) est `--safe-updates` (ou `--i-am-a-dummy` pour les utilisateurs qui ont déjà utilisé une commande `DELETE FROM table_name` mais on oublié la clause `WHERE` . Lorsque vous utilisez cette option, `mysql` envoie la commande suivante au serveur MySQL lors de l'ouverture de la connexion :

```
 SET SQL_SAFE_UPDATES=1,SQL_SELECT_LIMIT=#select_limit#,
    SQL_MAX_JOIN_SIZE=#max_join_size#"
```

où `#select_limit#` et `#max_join_size#` sont des variables qui peuvent être configurées depuis `mysql` . Syntaxe de `SET` .L'effet de la commande ci-dessus est :

- Vous n'êtes pas autorisé à faire une commande `UPDATE` ou `DELETE` si vous n'avez pas de contrainte de clé dans la clause `WHERE` . Il est possible toutefois de forcer la commande `UPDATE/DELETE` en utilisant `LIMIT` :

  ```
  UPDATE table_name SET not_key_column=# WHERE not_key_column=# LIMIT 1;
  ```
- Tous les grands résultats sont automatiquement limités à `#select_limit#` lignes.
- Les commandes `SELECT` qui vont probablement devoir examiner plus de `#max_join_size` lignes seront annulées.

Quelques conseils pratiques sur le client `mysql` :Certaines données seront plus faciles à lire lorsqu'affichées verticalement, au lieu d'horizontalement. Par exemple, des textes longs qui incluent des nouvelles lignes. Il sera alors plus facile de les lire verticalement.

```
 mysql> SELECT * FROM mails WHERE LENGTH(txt) < 300 lIMIT 300,1\G
*************************** 1. row ***************************
  msg_nro: 3068
     date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Monty
    reply: monty&#64;no.spam.com
  mail_to: "Thimble Smith" <tim&#64;no.spam.com>
      sbj: UTF-8
      txt: >>>>> "Thimble" == Thimble Smith writes:

 Thimble> Hi.  I think this is a good idea.  Is anyone familiar with UTF-8
Thimble> or Unicode? Otherwise, I'll put this on my TODO list and see what
Thimble> happens.

 Yes, please do that.

 Regards,
Monty
     file: inbox-jani-1
     hash: 190402944
1 row in set (0.09 sec)
```

Pour les logs, vous pouvez utiliser l'option `tee` . Le `tee` peut être démarré avec l'option `--tee=...` , ou depuis la ligne de commande interactive avec la commande `tee` . Toutes les données affichées sur l'écran seront enregistrées dans un fichier spécifié. Cela peut être très utile pour pouvoir déboguer. Le `tee` peut être désactivé avec la commande en ligne `notee` . Exécuter `tee` plusieurs fois va redémarrer

l'enregistrement. Sans paramètre, la commande va ouvrir le dernier fichier utilisé. Notez que `tee` va écrire les données après chaque requête, avant que l'invite apparaisse, pour attendre la prochaine commande.Afficher les résultats en mode interactif avec les programmes Unix tels que less, more ou tout autre programme similaire est actuellement possible avec l'option `--pager[=...]` . Sans argument, le client `mysql` va rechercher la variable d'environnement PAGER et l'utiliser pour `pager` . `pager` peut être démarré en ligne de commande avec `pager` et désactivée avec `nopager` . La commande prend un argument optionnel et `pager` prendra alors cette valeur. La commande `pager` peut être appelée sans argument, mais cela impose que l'option `--pager` ait été utilisée, ou que le mode `pager` sera par défaut la sortie standard. `pager` fonctionne uniquement sous Unix, car il utilise la fonction popen(), qui n'existe pas sous Windows. Sous Windows, l'option `tee` peut être utilisée à la place, même si elle n'est pas aussi pratique que `pager` dans certaines situations.Quelques conseils sur `pager` :

- Vous pouvez l'utiliser pour écrire dans un fichier :
  ```
  mysql> pager cat > /tmp/log.txt
  ```
  et les résultats iront directement dans le fichier. Vous pouvez aussi passer des options pour le programme que vous voulez utiliser comme `pager` :
  ```
  mysql> pager less -n -i -S
  ```
- Dans la présentation ci−dessus, notez bien l'option '−S'. Vous pourriez la trouver très pratique lorsque vous étudiez des résultats : essayer l'option d'affichage horizontal (validez vos commandes avec '\g' ou ';') et vertical (terminez vos commandes avec '\G'). Parfois, un résultat très large est difficile à lire et avec l'option −S dirigée vers less, vous pouvez naviguer dans le résultat interactivement de droite à gauche, ce qui évite que les lignes les plus longues soient coupées. Cela rend la lecture du texte bien plus facile. Vous pouvez passer d'un mode à l'autre depuis la ligne de commande. Voyez l'aide pour plus de détails sur less.
- Vous pouvez combiner des méthodes très complexes pour gérer les résultats, comme, la possibilité d'envoyer les résultats vers deux fichiers dans deux dossiers différents, ou deux disques différents, tout en affichant les résultats à l'écran avec less :
  ```
  mysql> pager cat | tee /dr1/tmp/res.txt | \
  tee /dr2/tmp/res2.txt | less -n -i -S
  ```

Vous pouvez aussi combiner les deux fonctions ci−dessus : activer le `tee` , spécifier le `pager` 'less' et vous serez capable de naviguer dans les résultats avec le less Unix, tout en enregistrant tous les résultats dans un fichier. La différence entre le `tee` d'Unix utilisé avec le `pager` et le `tee` intégré du client `mysql` , est que le `tee` intégré fonctionne même si vous n'avez pas de `tee` Unix disponible. Le `tee` enregistre tout ce qui est affiché à l'écran, alors que le `tee` Unix utilisé avec `pager` n'en note pas autant. Enfin, le `tee` interactif est plus facile à activer et désactiver, lorsque vous souhaitez enregistrer un résultat dans un fichier, mais que vous voulez désactiver cette fonctionnalité à d'autres moments.

Depuis MySQL version 4.0.2, il est possible de modifier l'invite de commande de `mysql` .

Vous pouvez utiliser les options de prompt suivantes :

| Option | Description |
|---|---|
| \v | version de mysqld |
| \d | database en cours |
| \h | hôte MySQL |
| \p | port de connexion |
| \u | nom d'utilisateur |
| \U | Identifiant complet username@host |
| \\ | '\' |
| \n | nouvelle ligne |
| \t | tabulation |
| \ | espace |
| _ | espace |
| \R | heure 24h (0−23) |
| \r | heure 12h (1−12) |
| \m | minutes |
| \y | année sur deux chiffres |

| | |
|---|---|
| \Y | année sur quatre chiffres |
| \D | format de date complet |
| \s | secondes |
| \w | jour de la semaine en trois lettres (Mon, Tue, ...) |
| \P | am/pm |
| \o | mois au format numérique |
| \O | mois en trois lettres (Jan, Feb, ...) |
| \c | compteur du nombre de commande |

'\' suivi de n'importe quelle lettre représente la lettre littéralement.

Vous pouvez modifier l'invite de commande comme ceci :

*Environment Variable*

Vous pouvez utiliser la variable d'environnement MYSQL_PS1 , en lui donnant la chaîne d'invite. Par exemple :

```
shell> export MYSQL_PS1="(\u&#64;\h) [\d]> "
```

my.cnf
.my.cnf

Vous pouvez configurer l'invite de commandes dans le fichier d'options MySQL. dans le groupe mysql . Par exemple :

```
[mysql]
prompt=(\u&#64;\h) [\d]>\_
```

*Command Line*

Vous pouvez utiliser l'option de démarrage --prompt en ligne de commande avec mysql . Par exemple :

```
shell> mysql --prompt="(\u&#64;\h) [\d]> "

(user&#64;host) [database]>
```

*Interactively*

Vous pouvez aussi utiliser la commande prompt (ou \R ) depuis le client pour modifier interactivement l'invite de commande. Par exemple :

```
mysql> prompt (\u&#64;\h) [\d]>\_
PROMPT set to '(\u&#64;\h) [\d]>\_'
(user&#64;host) [database]>
(user&#64;host) [database]> prompt
Returning to default PROMPT of mysql>
mysql>
```

## 4.8.3 mysqladmin , administrer un serveur MySQL

Un utilitaire pour exécuter des commandes d'administration. La syntaxe est :

```
shell> mysqladmin [OPTIONS] command [command-option] command ...
```

Vous pouvez obtenir une liste des options supportées par votre version de mysqladmin avec la commande mysqladmin --help .Le mysqladmin actuel supporte les commandes suivantes :

*create databasename*
: Crée une nouvelle base.

*drop databasename*
: Efface une base et toutes ces tables.

*extended-status*
: Affiche un message de statut du serveur très complet.

*flush-hosts*
: Vide tous les hôtes mis en cache.

*flush-logs*
: Vide de la mémoire tous les logs.

*flush-tables*
: Vide de la mémoire toutes les tables.

*flush-privileges*
: Recharger les tables de droits (identique à la commande reload).

*kill id,id,...*
: Termine un thread MySQL.

*password*
: Spécifie un nouveau mot de passe. Modifie l'ancien mot de passe en un nouveau.

*ping*
: Vérifie si mysqld fonctionne ou pas.

*processlist*
: Affiche la liste des processus du serveur.

*reload*
: Recharge les tables de droits.

*refresh*
: Vide de la mémoire toutes les tables, puis ferme et réouvre les fichiers de logs.

*shutdown*
: Eteind le serveur.

*slave−start*
: Démarre l'esclave de réplication.

*slave−stop*
: Eteind l'esclave de réplication.

*status*
: Affiche le message de statut court du serveur.

*variables*
: Affiche les variable disponibles.

*version*
: Affiche la version du serveur.

Toutes les commandes peuvent être réduites à leur préfixe simple. Par exemple :

```
 shell> mysqladmin proc stat
+----+-------+-----------+----+-------------+------+-------+------+
| Id | User  | Host      | db | Command     | Time | State | Info |
+----+-------+-----------+----+-------------+------+-------+------+
| 6  | monty | localhost |    | Processlist | 0    |       |      |
+----+-------+-----------+----+-------------+------+-------+------+
Uptime: 10077  Threads: 1  Questions: 9  Slow queries: 0
Opens: 6 Flush tables: 1  Open tables: 2
Memory in use: 1092K  Max memory used: 1116K
```

La commande `mysqladmin status` liste les colonnes suivantes :

| Colonne | Description |
|---|---|
| Uptime | Nombre de secondes de vie du serveur MySQL. |
| Threads | Nombre de threads actifs (clients). |
| Questions | Nombre de questions reçu des clients depuis le démarrage de mysqld . |
| Slow queries | Nombre de requêtes qui ont pris plus de long_query_time secondse. Le log des requêtes lentes. |
| Opens | Combien de tables sont ouvertes par mysqld . |
| Flush tables | Nombre de commandes flush ... , refresh et reload . |
| Open tables | Nombre de tables qui sont ouvertes actuellement. |
| Memory in use | Mémoire allouée directement par mysqld (uniquement disponible si MySQL a été compilé avec l'option −−with−debug=full). |
| Max memory used | Maximum de mémoire allouée directement par mysqld (uniquement disponible si MySQL a été compilé avec l'option −−with−debug=full). |

Si vous exécutez une commande `mysqladmin shutdown` sur une socket (en d'autres termes, sur un serveur où `mysqld` fonctionne ), `mysqladmin` va attendre que le fichier `pid−file` de MySQL soit effacé pour s'assurer que le serveur `mysqld` a été correctement stoppé.

## 4.8.4 Utiliser mysqlcheck pour l'entretien et la réparation

Depuis MySQL version 3.23.38, vous pouvez utiliser un nouvel outil d'entretien et de réparation pour les tables `MyISAM` . La différence avec `myisamchk` est que `mysqlcheck` doit être utilisé lorsque le serveur `mysqld` fonctionne, alors que `myisamchk` doit être utilisé lorsque le serveur ne fonctionne pas. L'intérêt st que vous n'avez plus besoin d'interrompre le serveur pour vérifier ou réparer vos tables.

`mysqlcheck` utilise les commandes du serveur MySQL `CHECK` , `REPAIR` , `ANALYZE` et `OPTIMIZE` , d'une manière pratique pour l'utilisateur.

Il y a trois façons différentes d'utiliser `mysqlcheck` :

```
 shell> mysqlcheck [OPTIONS] database [tables]
shell> mysqlcheck [OPTIONS] --databases DB1 [DB2 DB3...]
shell> mysqlcheck [OPTIONS] --all-databases
```

Il peut aussi être utilisé comme `mysqldump` lorsqu'il faut choisir les bases et tables à traiter.

`mysqlcheck` dispose d'une fonctionnalité spéciale, comparé aux autres clients : le comportement par défaut, c'est à dire la vérification des tables, peut être modifiée en renommant le fichier binaire. Si vous voulez avoir un fichier qui réparer les tables par défaut, il suffit de copier `mysqlcheck` sur votre disque, et de l'appeler `mysqlrepair` , ou bien, de faire un lien symbolique sur l'exécutable et de l'appeler `mysqlrepair` . SI vous appelez `mysqlrepair` , il va réparer les tables par défaut.

Les noms que vous pouvez utiliser pour que `mysqlcheck` réparer automatiquement les tables sont :

```
 mysqlrepair:   L'option par défaut est -r
mysqlanalyze:  L'option par défaut est -a
mysqloptimize: L'option par défaut est -o
```

Les options disponibles pour `mysqlcheck` sont listées ici. Vérifiez que votre version les supporte avec la commande `mysqlcheck --help` .

*-A, --all-databases*
: Vérifie toutes les bases. C'est la même chose que --databases dans toutes les bases sélectionnées.

*-1, --all-in-1*
: Au lieu de faire une requête par table, exécute toutes les requêtes dans une requête, séparément pour chaque base. Les noms de tables seront séparés par une virgule.

*-a, --analyze*
: Analyse les tables indiquées.

*--auto-repair*
: Si une table vérifiées est corrompue, la corrige automatiquement. La réparation sera faite après la vérification de toutes les tables, si des tables corrompues ont été découvertes.

*-#, --debug=...*
: Affiche le log de debug. Souvent, c'est dans 'd:t:o,filename'

*--character-sets-dir=...*
: Dossier contenant le jeu de caractères

*-c, --check*
: Vérifie les tables en erreur

*-C, --check-only-changed*
: Vérifie uniquement les tables qui ont été modifiées depuis la dernière modification, ou qui n'ont pas été correctement fermées.

*--compress*
: Utiliser la compression du protocole client/serveur.

*-?, --help*
: Affiche ce message d'aide, et termine.

*-B, --databases*
: Pour tester plusieurs bases de données. Notez que la différence d'utilisation : dans ce cas, aucune table n'est précisé. Tous les arguments de noms sont considérés comme des noms de base.

*--default-character-set=...*
: Spécifie le jeu de caractères par défaut.

*-F, --fast*

Ne vérifie que les tables qui n'ont pas été correctement fermées.

*−f, −−force*
: Continue même si on rencontre une erreur SQL.

*−e, −−extended*
: Si vous utilisez cette option avec CHECK TABLE, elle va s'assurer que la table est totalement cohérente, mais prendre un très long temps.

Si vous utilisez cette option avec REPAIR TABLE, elle va réaliser une réparation exhaustive de la table, qui peut non seulement prendre un temps très long, mais produire de nombreuses lignes erronées.

*−h, −−host=...*
: Connexion à l'hôte.

*−m, −−medium−check*
: Plus rapide que la vérification complète, mais ne trouvera que 99.99 % de toutes les erreurs. Cela devrait être la bonne option pour la plupart des situations.

*−o, −−optimize*
: Optimise la table

*−p, −−password[=...]*
: Le mot de passe à utiliser lors de la connexion au serveur. Si aucun mot de passe n'est fourni, il sera demandé en ligne de commande.

*−P, −−port=...*
: Le numéro de port de la connexion.

`−−protocol=(TCP | SOCKET | PIPE | MEMORY)`
: Pour spécifier le protocole à utiliser pour la connexion. Nouveau en MySQL 4.1.

*−q, −−quick*
: Si vous utilisez cette option avec CHECK TABLE, elle va éviter que l'analyse ne scanne les lignes pour vérifier les mauvais liens. C'est la méthode d'analyse la plus rapide.

Si vous utilisez cette option avec REPAIR TABLE, elle va essayer de ne réparer que le fichier d'index. C'est la méthode la plus rapide pour la réparation.

*−r, −−repair*
: Peut corriger presque tout, sauf les problèmes de doublons pour les clés uniques.

*−s, −−silent*
: Affiche moins de messages d'erreurs.

*−S, −−socket=...*
: Nom du fichier de socket à utiliser pour la connexion.

*−−tables*
: Remplace l'option −−databases (−B).

*−u, −−user=#*
: Nom d'utilisateur pour la connexion, si ce n'est pas l'utilisateur courant.

*−v, −−verbose*
: Afficher des informations sur les différentes étapes.

*−V, −−version*
: Affiche les informations de version, et termine.

## 4.8.5 mysqldump , exporter les structures de tables et les données

Utilitaire qui permet d'exporter une base ou un groupe de bases vers un fichier texte, pour la sauvegarde ou le transfert entre deux serveurs (pas nécessairement entre serveurs MySQL). L'export contiendra les requêtes SQL nécessaires pour créer la table et la remplir.

Si vous faîtes une sauvegarde du serveur, vous devriez aussi utiliser la commande `mysqlhotcopy` . `mysqlhotcopy` , copier les bases et tables MySQL .

```
shell> mysqldump [OPTIONS] database [tables]
OR     mysqldump [OPTIONS] --databases [OPTIONS] DB1 [DB2 DB3...]
OR     mysqldump [OPTIONS] --all-databases [OPTIONS]
```

Si vous ne spécifiez pas de table, ou si vous utilisez l'option `--databases` ou `--all-databases` , la base de données complète sera exportée.

Vous pouvez obtenir une liste des options valides pour votre version de `mysqldump` avec la commande `mysqldump --help` .

Notez que si vous exécutez `mysqldump` sans l'option `--quick` ou `--opt` , `mysqldump` va charger la totalité du résultat en mémoire, avant de l'écrire. Cette option peut résoudre des problèmes de mémoire si vous exportez de grosses tables.

Notez que si vous utilisez une nouvelles copie du programme `mysqldump` , et que vous allez faire un export qui sera lu dans une vieille version de MySQL, vous ne devriez pas utiliser les options `--opt` et `-e` .

`mysqldump` supporte les options suivantes :

*--add-locks*
: Ajoute une commande `LOCK TABLES` avant l'export de table et une commande `UNLOCK TABLE` après(Pour accélérer les inserionts dans MySQL).

*--add-drop-table*
: Ajoute une commande `drop table` avant chaque requête de création de table.

*-A, --all-databases*
: Exporte toutes les tables. C'est l'équivalent de l'option `--databases` avec toutes les bases de données sélectionnées.

*-a, --all*
: Inclut toutes les options de créations de table spécifiques à MySQL.

*--allow-keywords*
: Permet la création de colonnes ayant des noms de mots réservés. Cela fonctionne en préfixant chaque nom de colonne avec le nom de la table.

*-c, --complete-insert*
: Utilise une commande complète d'insertion (avec les noms des colonnes).

*-C, --compress*
: Compresse toutes les informations entre le client et le serveur, les deux supporte la compression.

*-B, --databases*
: Pour exporter plusieurs bases de données. Notez la différence d'utilisation. Dans ce cas, aucune table n'est spécifié. Tous les arguments de noms sont considérés comme des noms de base. Une ligne `USE db_name;` sera ajoutée dans l'export avant chaque base de données.

*--delayed*
: Les insertions se font avec la commande `INSERT DELAYED` .

*-e, --extended-insert*
: Utilise la nouvelle syntaxe multi-ligne `INSERT` . (Cela donne des insertions plus courtes et plus efficaces).

*-#, --debug[=option_string]*
: Trace l'utilisation du programme (pour le débogage).

*--help*
: Affiche le message d'aide et quitte.

*--fields-terminated-by=...*

*--fields-enclosed-by=...*

*--fields-optionally-enclosed-by=...*

*--fields-escaped-by=...*

*−−lines−terminated−by=...*
Ces options sont utilisées avec l'option `-T` et ont la même signification que les clauses correspondantes de la commande `LOAD DATA INFILE` . Syntaxe des `LOAD DATA INFILE` .

*−F, −−flush−logs*
*f*crit tout le fichier de log du serveur avant de commencer l'export.

*−f, −−force,*
Continue même si une erreur SQL survient durant l'export.

*−h, −−host=..*
Exporte les données depuis le serveur MySQL vers l'hôte indiqué. L'hôte par défaut est `localhost` .

*−l, −−lock−tables.*
Verrouille toutes les tables avant de commencer l'export. Les tables sont verrouillées avec `READ LOCAL` pour permettre des insertions concurentes sur les tables `MyISAM` .Notez que lorsque vous exportes des tables de bases différentes, l'option `--lock-tables` va verrouiller chaque base séparément. Cette option ne vous garantira pas que vos tables seront logiquement cohérente entre les bases. Des tables de différentes bases pourraient être exportées dans des états très différents.

*−K, −−disable−keys*
`/*!40000 ALTER TABLE tb_name DISABLE KEYS */;` et `/*!40000 ALTER TABLE tb_name ENABLE KEYS */;` seront ajoutés dans le résultat. Cela rendra les chargements de données plus rapides sur les serveurs MySQL 4.0 car les index sont alors créés après l'insertion de toutes les données.

*−n, −−no−create−db*
`CREATE DATABASE /*!32312 IF NOT EXISTS*/ db_name;` ne sera pas ajouté dans l'export. Sinon, la ligne ci−dessus sera ajoutée, si l'une des options `--databases` ou `--all-databases` ont été activée.

*−t, −−no−create−info*
N'écrit pas les informations de création de table (la requête `CREATE TABLE` ).

*−d, −−no−data*
N'écrit aucune ligne d'informations sur la atble. C'est très pratique si vous voulez simplement exporter la structure de la table.

*−−opt*
Identique à `--quick --add-drop-table --add-locks --extended-insert --lock-tables` . Vous obtiendrez l'export le plus rapide à importer dans un serveur MySQL.

*−pyour_pass, −−password[=your_pass]*
Le mot de passe à utiliser lors de la connexion au serveur. Si vous spécifiez en omettant la partie `'=your_pass'` , `mysqldump` vous demandera le mot de passe en ligne de commande.

*−P port_num, −−port=port_num*
Le port TCP/IP à utiliser avec l'hôte. Ceci sert pour les connexions d'hôte à hôte, autre que `localhost` , pour lequel les sockets Unix seront utilisées).

*−q, −−quick*
Ne garde pas en buffer les requêtes, mais écrit immédiatement dans la sortie. Utilise `mysql_use_result()` pour cela.

*−Q, −−quote−names*
Protège les noms des tables et colonnes avec le caractère `'``'` .

*−r, −−result−file=...*
Ecrit directemetn dans le fichier indiqué. Cette option doit être utilisé sur MSDOS, car cela évite que la nouvelle ligne `'\n'` soient converties en `'\n\r'` (nouvelle ligne et retour chariot).

*−−single−transaction*
Cette option ajoute la commande SQL `BEGIN` avant d'exporter les données vers le serveur. C'est généralement pratique pour les tables `InnoDB` et le niveau d'isolation de transaction `READ_COMMITTED` , car ce mode va exporter l'état de la base au moment de la commande `BEGIN` sans bloquer les autres applications.Lorsque vous utilisez cette

option, pensez bien que seules les tables transactionnelles seront exportées dans un état cohérent, c'est à dire que les tables `MyISAM` ou `HEAP` qui seront exportées avec cette option, pourront changer d'état.

L'option `--single-transaction` a été ajoutée en version 4.0.2. Cette option est mutuellement exclusive avec l'option `--lock-tables` car `LOCK TABLES` va valider une transaction interne précédente.

*−S /path/to/socket, −−socket=/path/to/socket*
: Le fichier de socket à utiliser lors de la connexion à `localhost` (qui est l'hôte par défaut).

*−−tables*
: Remplace l'option −−databases (−B).

*−T, −−tab=path−to−some−directory*
: Crée un fichier `table_name.sql` , qui contient les commandes SQL CREATE, et un fichier `table_name.txt` , qui contient les données, pour chaque table. Le format du fichier `.txt` est celui qui est spécifié par les options `--fields-xxx` et `--lines--xxx` . **Note** : cette option ne fonctionne qui si `mysqldump` est exécuté sur la même machine que le démon `mysqld` , et que le nom d'utilisateur et le groupe de `mysqld` (normalement l'utilisateur `mysql` , et le groupe `mysql` ) doivent avoir des permission pour créer et écrire un fichier dans le dossier que vous spécifiez.

*−u user_name, −−user=user_name*
: Le nom d'utilisateur MySQL lors de la connexion à un serveur distant. La valeur par défaut est votre nom d'utilisateur Unix.

*−O var=option, −−set−variable var=option*
: Spécifie la valeur d'une variable. Les noms possibles des variables sont spécifiés ci−dessous. Notez bien que `--set-variable` est obsolète depuis MySQL 4.0, il vous suffit alors d'utiliser la syntaxe `--var=option` .

*−v, −−verbose*
: Mode détaillé. Affiche plus d'informations sur les faits et gestes du programme.

*−V, −−version*
: Affiche la version du programme et quitte.

*−w, −−where='where−condition'*
: Exporte uniquement les lignes sélectionnées. Notez que les guillemets sont obligatoires.

*−X, −−xml*
: Exporte la base au format XML.

*−x, −−first−slave*
: Verrouille toutes les tables dans les bases.

```
"--where=user='jimf'" "-wuserid>1" "-wuserid<1"
```

*−O net_buffer_length=#, where # < 16M*
: Lors de la création de commandes d'insertions multilignes, (comme avec l'option `--extended-insert` ou `--opt` ), `mysqldump` va créer des lignes jusqu'à la taille de `net_buffer_length` . Si vous augmentez cette valeur, vous devriez aussi vous assurer que la variable MySQL `max_allowed_packet` est plus grande que `net_buffer_length` .

L'usage normal de `mysqldump` est probablement de faire des sauvegardes de bases. Sauvegardes de base de données .

```
mysqldump --opt database > backup-file.sql
```

Vous pouvez importer les données dans la base MySQL avec :

```
mysql database < backup-file.sql
```

ou

```
mysql -e "source /patch-to-backup/backup-file.sql" database
```

Cependant, il est très pratique pour remplir un autre serveur MySQL avec des informations depuis une base :

```
mysqldump --opt database | mysql ---host=remote-host -C database
```

Il est possible d'exporter plusieurs bases de données en une seule commande :

```
mysqldump --databases database1 [database2 ...] > my_databases.sql
```

Si vous souhaitez exporter toutes les bases, vous pouvez utiliser :

```
mysqldump --all-databases > all_databases.sql
```

## 4.8.6 mysqlhotcopy , copier les bases et tables MySQL

`mysqlhotcopy` est un script Perl qui utilise `LOCK TABLES` , `FLUSH TABLES` et `cp` ou `scp` pour faire rapidement des sauvegardes de bases. C'est la méthode la plus rapide pour faire une sauvegarde de bas. C'est aussi le moyen le plus sûr pour copier des tables et bases, mais il ne peut fonctionner que sur la machine qui contient les fichiers de données.

```
mysqlhotcopy db_name [/path/to/new_directory]

mysqlhotcopy db_name_1 ... db_name_n /path/to/new_directory

mysqlhotcopy db_name./regex/
```

`mysqlhotcopy` supporte les options suivantes :

*-?, --help*
: Affiche un écran d'aide et quitte.

*-u, --user=#*
: Nom d'utilisateur pour la connexion au serveur.

*-p, --password=#*
: Mot de passe utilisé pour la connexion au serveur.

*-P, --port=#*
: Port utilisé pour la connexion au serveur.

*-S, --socket=#*
: Socket utilisé pour la connexion au serveur.

*--allowold*
: Ne pas annuler si la sauvegarde existe déjà (renomme la simplement en _old)

*--keepold*
: Ne pas effacer une sauvegarde précédente (celle qui a été renommée) lorsque c'est terminé.

*--noindices*
: Ne pas inclure les fichiers d'index complet dans la copie, pour faire des fichiers de sauvegarde plus petit et plus rapide. Les index peuvent toujours être reconstruits plus tard avec `myisamchk -rq.` .

*--method=#*
: Méthode de copie ( `cp` ou `scp` ).

*-q, --quiet*
: Mode silencieux. N'affiche que les erreurs.

*--debug*
: Active le débogage.

*-n, --dryrun*
: Rapporte les actions réalisées sans les faire.

*--regexp=#*
: Copie toutes les bases dont le nom vérifie un masque d'expression régulière.

*--suffix=#*
: Suffixe des noms des bases copiées.

*--checkpoint=#*
: Insère un point de contrôle dans la table spécifiée (base.table)

*--flushlog*
: Vide les logs sur le disque une fois que toutes les tables sont verrouillées.

*--tmpdir=#*
: Dossier temporaire (au lieu de /tmp).

Vous pouvez essayer `perldoc mysqlhotcopy` pour avoir plus de documentation sur `mysqlhotcopy` . `mysqlhotcopy` lit les options du groupe `[client]` et `[mysqlhotcopy]` dans le fichier d'options.

Pour être capable d'exécuter `mysqlhotcopy` , vous avez besoin des droits d'écriture dans le dossier de sauvegarde, et le droit de `SELECT` dans les tables que vous souhaitez copier, ainsi que les droits MySQL de `RELOAD` (pour utiliser la commande `FLUSH TABLES` ).

## 4.8.7 mysqlimport , importer des données depuis des fichiers texte

`mysqlimport` fournit une interface en ligne de commande à la commande SQL `LOAD DATA INFILE` . La plupart des options de `mysqlimport` correspondent directement aux mêmes options de `LOAD DATA INFILE` . Syntaxe des `LOAD DATA INFILE` .

`mysqlimport` est appelé comme ceci :

```
shell> mysqlimport [options] database textfile1 [textfile2 ...]
```

Pour chaque fichier texte indiqué dans la ligne de commande, `mysqlimport` supprime toute extension du nom du fichier, et utilise le résultat pour déterminer le nom de la table qui va recevoir le contenu du fichier. Par exemple, pour des fichiers appelés `patient.txt` , `patient.text` et `patient` seront tous importés dans la table `patient` .

`mysqlimport` supporte les options suivantes :

*−c, −−columns=...*
: Cette option prend une liste de noms de colonnes, séparés par des virgules. Ce champs est utilisé pour créer une commande `LOAD DATA INFILE` correcte, qui sera alors passée à MySQL. Syntaxe des `LOAD DATA INFILE` .

*−C, −−compress*
: Compresse toutes les informations entre le client et le serveur, si c'est possible.

*−#, −−debug[=option_string]*
: Trace l'utilisation du programme (pour le débogage).

*−d, −−delete*
: Vide la table avant d'importer le fichier texte.

*−−fields−terminated−by=...*
*−−fields−enclosed−by=...*
*−−fields−optionally−enclosed−by=...*
*−−fields−escaped−by=...*
*−−lines−terminated−by=...*
: Ces options ont la même signification que les clauses correspondantes de `LOAD DATA INFILE` . Syntaxe des `LOAD DATA INFILE` .

*−f, −−force*
: Ignore les erreurs. Par exemple, si une table n'existe pas pour un fichier texte, `mysqlimport` va continuer de traiter les autres fichiers. Sans `--force` , `mysqlimport` se termine dès qu'une erreur survient.

*−−help*
: Affiche le message d'aide et quitte.

*−h host_name, −−host=host_name*
: Importe les données sur le serveur MySQL, avec l'hôte spécifié. La valeur par défaut est `localhost` .

*−i, −−ignore*
: Voir la description de `--replace` .

*−l, −−lock−tables*

Verrouille **toutes** les tables en écriture avant de ne traiter les fichiers textes. Cela assure que toutes les tables sont synchronisée sur le serveur.

*−L, −−local*
: Lit le fichier d'entrée dans le client. Par défaut, les fichiers textes sont supposés être lus par le serveur, si vous vous connectez à `localhost` (qui l'hôte par défaut).

*−pyour_pass, −−password[=your_pass]*
: Le mot de passe à utiliser lors de la connexion au serveur. Si vous ne spécifiez pas la partie `'=your_pass'` , `mysqlimport` va vous demander le mot de passe en ligne.

*−P port_num, −−port=port_num*
: Le port TCP/IP utilisé avec l'hôte. Cela sert pour les connexions à des hôtes qui ne sont pas `localhost` , pour lequel la socket Unix est utilisée.

*−r, −−replace*
: Les options `−−replace` et `−−ignore` contrôles la gestion des lignes lues envers les lignes qui existent déjà sur le serveur. Si vous spécifiez l'option `−−replace` , les nouvelles lignes remplaceront les lignes existantes. Si vous spécifiez `−−ignore` , les lignes qui sont en double dans une table qui dispose d'une colonne de type unique. Si vous ne spécifiez pas ces options, une erreur surviendra lorsqu'une clé en double sera trouvée, et la lecture du reste du fichier sera annulé.

*−s, −−silent*
: Mode silencieux. N'affiche que les erreurs qui surviennent.

*−S /path/to/socket, −−socket=/path/to/socket*
: Le fichier de socket à utiliser lors de la connexion à `localhost` (qui est l'hôte par défaut).

*−u user_name, −−user=user_name*
: Le nom de l'utilisateur MySQL à utiliser lors de la connexion au serveur MySQL. La valeur par défaut est celui de votre utilisateur Unix.

*−v, −−verbose*
: Mode détaillé. Affiche bien plus d'informations sur les actions du programme.

*−V, −−version*
: Affiche la version et quitte.

Voici un exemple d'utilisation de `mysqlimport` :

```
 $ mysql −−version
mysql  Ver 9.33 Distrib 3.22.25, for pc−linux−gnu (i686)
$ uname −a
Linux xxx.com 2.2.5−15 #1 Mon Apr 19 22:21:09 EDT 1999 i586 unknown
$ mysql −e 'CREATE TABLE imptest(id INT, n VARCHAR(30))' test
$ ed
a
100     Max Sydow
101     Count Dracula
.
w imptest.txt
32
q
$ od −c imptest.txt
0000000   1   0   0  \t   M   a   x       S   y   d   o   w  \n   1   0
0000020   1  \t   C   o   u   n   t       D   r   a   c   u   l   a  \n
0000040
$ mysqlimport −−local test imptest.txt
test.imptest: Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
$ mysql −e 'SELECT * FROM imptest' test
+−−−−−−+−−−−−−−−−−−−−−−+
| id   | n             |
+−−−−−−+−−−−−−−−−−−−−−−+
|  100 | Max Sydow     |
|  101 | Count Dracula |
+−−−−−−+−−−−−−−−−−−−−−−+
```

## 4.8.8 Afficher les bases, tables et colonnes

`mysqlshow` peut être utilisé pour lister les bases qui existent, leurs tables et leurs colonnes.

Avec le programme `mysql` vous pouvez obtenir des informations avec la commande `SHOW` . Syntaxe de `SHOW` .

`mysqlshow` est utilisé comme ceci :

```
shell> mysqlshow [OPTIONS] [database [table [column]]]
```

- Si aucune base n'est indiquée, toutes les bases de données sont listées.
- Si aucune table n'est nommée, toutes les tables de la base sont affichées.
- Si aucune colonne n'est nommée, toutes les colonnees et leur type sont affichés.

Notez que dans les nouvelles versions de MySQL, vous ne verrez que les bases de données, tables et colonnes pour lesquelles vous avez des droits.

Si le dernier argument contient un caractère joker shell ou SQL ( `*` , `?` , `%` ou `_` ) alors seules les entités qui valident ce masque sont affichées. Si une base contient des caractères soulignés, ils doivent être protégés avec un anti−slash (certains shell Unix en demande même deux), afin de lister correctement les tables et les colonnes. Les '*' sont convertis en SQL '%' et '?' en SQL '_'. Cela peut causer des confusions lorsque vous essayez d'afficher des colonnes qui contiennent un souligné `_` , comme c'est le cas avec `mysqlshow` qui ne vous affiche que les colonnes qui vérifient le masque. Ceci est facilement corrigé en ajoutant un caractère `%` en plus dans la ligne de commande (comme argument séparé).

## 4.8.9 mysql_config lit les options de compilations du client MySQL

`mysql_config` vous indique des informations pratiques pour compiler votre client MySQL et le connecter au serveur.

`mysql_config` supporte les options suivantes :

*−−cflags*
: Options de compilations utilisées pour trouver les fichiers inclus.

*−−libs*
: Les librairies et options requises pour compiler avec la librairie client MySQL.

*−−socket*
: Le nom par défaut de la socket, défini lors de la configuration de MySQL.

*−−port*
: Le numéro de port par défaut, défini lors de la configuration de MySQL.

*−−version*
: Le numéro de version et la version de la distribution MySQL.

*−−libmysqld−libs*
: Les librairies et options requises pour compiler avec la librairie intégrée MySQL.

Si vous exécutez `mysql_config` sans aucune option, il va afficher toutes les options qu'il supporte, ainsi que la valeur de toutes les options :

```
 shell> mysql_config
sage: /usr/local/mysql/bin/mysql_config [OPTIONS]
Options:
        --cflags         [-I'/usr/local/mysql/include/mysql']
        --libs           [-L'/usr/local/mysql/lib/mysql' -lmysqlclient -lz -lcrypt -lnsl -lm -L/usr/lib -lssl -lcrypto]
        --socket         [/tmp/mysql.sock]
        --port           [3306]
        --version        [4.0.8-gamma]
        --libmysqld-libs [ -L'/usr/local/mysql/lib/mysql' -lmysqld -lpthread -lz -lcrypt -lnsl -lm  -lpthread  -lrt]
```

Vous pouvez utiliser le résultat, pour compiler un client MySQL comme ceci :

```
 CFG=/usr/local/mysql/bin/mysql_config
sh -c "gcc -o progname `$CFG --cflags` progname.c `$CFG --libs`"
```

### 4.8.10 perror , expliquer les codes d'erreurs

Pour la plupart des erreurs système, MySQL va, en plus d'un message interne, aussi afficher un code d'erreur, dans l'un des styles suivants : `message ... (errno: #)` or `message ... (Errcode: #)` .

Vous pouvez découvrir ce que ce code d'erreur signifie soit en examinant la documentation de votre système, soit en utilisant l'utilitaire `perror` .

`perror` affiche une description pour le code d'erreur, ou, pour une erreur du gestionnaire de tables MyISAM/ISAM.

`perror` est appelé comme ceci :

```
 shell> perror [OPTIONS] [ERRORCODE [ERRORCODE...]]

 Example:

 shell> perror 13 64
Error code  13:  Permission denied
Error code  64:  Machine is not on the network
```

Notez que les messages d'erreurs sont dépendants du système.

### 4.8.11 Comment exécuter des commandes SQL depuis un fichier texte

Le client `mysql` peut être utilisé en mode interactif comme ceci :

```
 shell> mysql database
```

Toutefois, il est aussi possible de rassembler les commandes SQL dans un fichier, et d'indiquer à `mysql` de lire les commandes dans ce fichier. Pour faire cela, créez un fichier texte `fichier_texte` qui contient les commandes SQL que vous souhaitez exécuter. Puis, exécutez ce fichier avec `mysql` comme ceci :

```
 shell> mysql database < fichier_texte
```

Vous pouvez aussi démarrer votre fichier texte avec la commande `USE nom_base` . Dans ce cas, il n'est pas néeccaire de spécifier le nom de la base de données dans la ligne de commande :

```
 shell> mysql < fichier_texte
```

Si vous avez déjà démarré le client `mysql` , vous pouvez exécuter un script SQL en utilisant la commande `source` :

```
 mysql> source nom_fichier;
```

Pour plus d'informations sur le mode batch, consultez Utiliser `mysql` en mode batch .

## 4.9 Les fichiers de log de MySQL

MySQL a plusieurs fichiers de log qui peuvent vous aider à savoir ce qui se passe à l'intérieur de `mysqld` :

| Fichier | Description |
|---|---|
| Le log d'erreurs | Problèmes rencontrés lors du démarrage, de l'exécution ou du stoppage de mysqld . |
| Le log isam | Garde une trace des changements liés au tables ISAM. Utilisé uniquement pour déboguer le code isam. |
| Le log de requêtes | Connexions établies et requêtes exécutées. |
| Le log de mises à jour | Désapprouvé : Enregistre toutes les commandes qui changent les données. |
| Le log binaire | Enregistre toutes les commandes qui changent quelque chose. Utilisé pour la replication. |
| Le log des requêtes lentes | Enregistre toutes les requêtes qui ont pris plus de long_query_time à s'exécuter ou celles qui n'ont pas utilisé d'index. |

Tous les fichiers de log peuvent être trouvés dans le dossier de données de `mysqld` . Vous pouvez forcer `mysqld` à réouvrir les fichiers de log (ou dans quelques cas à passer à un nouveau log) en exécutant `FLUSH LOGS` . Syntaxe de `FLUSH` .

## 4.9.1 Le log d'erreurs

`mysqld` affiche toutes les erreurs dans stderr, le script `safe_mysqld` les redirige vers un fichier nommé `'hostname'.err` . (Sur Windows, `mysqld` les écrit directement dans `\mysql\data\mysql.err` .)

Ce fichier contient des informations indiquant quand `mysqld` a été démarré et stoppé pour cause d'erreurs critiques rencontré lors de son fonctionnement. Si `mysqld` se termine inopinément et que `safe_mysqld` a besoin de redémarrer `mysqld` , `safe_mysqld` ajoutera une ligne `restarted mysqld` dans ce fichier. Ce log contient aussi un avertissement si `mysqld` trouve une table qui a besoin d'être automatiquement vérifiée ou réparée.

Sur quelques systèmes d'exploitation, le log d'erreurs contiendra un traçage de la pile mémoire relatif à l'endroit où `mysqld` s'est terminé. Il peut être utilisé pour trouver les raisons du crash de `mysqld` . Utilisation d'un traçage de la pile .

## 4.9.2 Le log général de requêtes

Si vous voulez savoir ce qui se passe à l'intérieur de `mysqld` , vous devez le démarrer avec `--log[=fichier]` . Cela aura pour effet d'écrire toutes les connexions et les requêtes dans le fichier de log (pas défaut nommé `'hostname'.log` ). Ce log peut être très utile quand vous suspectez une erreur dans un client et voulez savoir exactement ce que `mysqld` pense que le client lui a envoyé.

Les anciennes versions du script `mysql.server` (de MySQL 3.23.4 à 3.23.8) passent à `safe_mysqld` une option `--log` (active le log général de requêtes). Si vous avez besoin de meilleurs performances lorsque vous démarrez MySQL dans un environnement de production, vous pouvez supprimer l'option `--log` de `mysql.server` ou la changer en `--log-bin` . Le log binaire des mises à jour .

Les entrées dans ce log sont écrites en même temps que `mysqld` reçoit une question. Cela peut être différent de l'ordre dans lequel les commandes sont exécutées. Cela est contraire au log des mises à jour et du log binaire qui sont écrits après l'exécution de la requête, mais avant les verrouillages.

## 4.9.3 Le log de modification

**Note** : le log de modifications a été remplacé par le log binaire. Le log binaire des mises à jour . Avec ce nouveau log, vous pouvez faire tout ce que vous faisiez avec le log de modifications.

Lors l'option `--log-update[=file_name]` est utilisée au démarrage, `mysqld` écrit un fichier de log contenant toutes les commandes SQL qui modifie les données. Si aucun fichier n'est spécifié, il prendra la valeur par défaut du nom de l'hôte. Si un fichier est spécifié mais qu'aucun chemin n'est indiqué, le fichier sera écrit dans le dossier de données. Si le fichier `file_name` n'a pas d'extension, `mysqld` va créer un fichier de log avec ce nom : `file_name.###` , où `###` est un nombre qui s'incrémente à chaque vois que vous exécutez la commande `mysqladmin refresh` , `mysqladmin flush-logs` , `FLUSH LOGS` ou que vous redémarrez le serveur.

**Note** : pour que la technique ci-dessus fonctionne, vous ne devez pas créer de fichiers avec le nom du fichier de log + une extension, qui pourrait être considérée comme un nombre, dans le dossier qui contient les log de modifications.

Si vous utilisez les options `--log` ou `-l` , `mysqld` écrit un log général avec le nom de `hostname.log` , et les redémarrage ou les rafraîchissements de fichiers de logs ne généreront pas de nouveaux fichiers (même si le fichier lui même est bien refermé, puis ouvert). Dans ce cas, vous pouvez le copier sous Unix avec :

```
 mv hostname.log hostname-old.log
mysqladmin flush-logs
cp hostname-old.log to-backup-directory
rm hostname-old.log
```

Le fichier de log de modification est très utile, car il n'enregistre que les commandes qui modifient les données. Ce qui fait qu'une commande `UPDATE` ou `DELETE` avec une requête `WHERE` qui ne trouve pas de lignes ne seront pas écrites dans ce fichier. Il va même ignorer les requêtes `UPDATE` qui modifie une colonne avec la valeur déjà présente.

L'enregistrement dans le log de modification est fait juste après l'achèvement de la requête, mais avant la levée des verrous, et les validations. Cela garantit que la requête sera enregistrée.

Si vous voulez modifier une base grâce aux fichier de log de modification, vous pouvez utiliser la commande suivante (en supposant que vos fichiers de log de modification porte le nom de `file_name.###` ) :

```
 shell> ls -1 -t -r file_name.[0-9]* | xargs cat | mysql
```

`ls` est utilisé pour obtenir toute la liste des fichiers de logs du dossier.Ceci peut être utile si vous devez repartir d'un fichier de sauvegarde après un crash, et que vous souhaitez réexécuter les modifications qui ont eu lieu depuis la sauvegarde.

## 4.9.4 Le log binaire

Notre objectif est de remplacer l'utilisation du log de modifications par le log binaire de modifications. Nous vous recommandons donc de changer de format de log le plus tôt possible.

Le log binaire contient toutes les informations qui sont disponibles dans le log de modifications, dans un format bien plus efficace. Il contient aussi des informations sur le temps d'exécution de la requête. Il ne contient pas les requêtes qui n'ont pas modifié les données. Si vous voulez enregistrer toutes les requêtes, vous devez utiliser le fichier de log général. Le log général de requêtes .

Le log binaire est aussi utilisé lorsque de la réplication d'un maître par un esclave. Réplication avec MySQL .

Lorsque l'option de démarrage `--log-bin[=file_name]` est utilisée, `mysqld` écrit un fichier de log contenant toutes les commandes SQL qui modifient les données. Si aucun nom de fichier n'est donné, le nom de la machine hôte est utilisé, suivi de `-bin` . Si un nom est donné, mais qu'il ne contient pas de chemin, le fichier sera écrit dans le dossier de données.

Si vous fournissez une extension à `--log-bin=filename.extension` , l'extension sera automatiquement supprimée.

Au nom du fichier de log binaire, `mysqld` va ajouter une extension qui est un nombre automatiquement incrémenté chaque fois que vous exécutez `mysqladmin refresh` , `mysqladmin flush-logs` , `FLUSH LOGS` ou redémarrez le serveur. Un nouveau fichier de log sera automatiquement créé lorsque le fichier en cours atteint la taille de `max_binlog_size` . Vous pouvez effacer tous les fichiers de log inactifs avec la commande `RESET MASTER` . Syntaxe de `RESET` .

Vous pouvez utiliser les options suivantes avec `mysqld` pour modifier ce qui est enregistré dans le fichier de log :

| Option | Description |
|---|---|
| binlog-do-db=database_name | Indique au maître qu'il doit enregistrer les modifications des bases spécifiées, et ignorer les autres. (Example: binlog-do-db=some_database ) |
| binlog-ignore-db=database_name | Indique autre maître que les modifications des bases spécifiées ne doit pas être enregistrées. (Exemple : binlog-ignore-db=some_database ) |

Pour être capable de connaître les différents fichiers de logs qui ont été utilisés, `mysqld` va aussi créer un fichier d'index qui contient les noms des fichiers de log utilisé. Par défaut, ils ont le même nom que le fichier de log, avec l'extension `'.index'` . Vous pouvez modifier le nom du fichier d'index avec l'option `--log-bin-index=[filename]` .

Si vous utilisez la réplication, vous ne devez pas effacer les anciens log binaires jusqu'à ce que vous soyez sûrs que les esclaves n'en auront plus besoin. Une façon de faire cela est d'utiliser la commande `mysqladmin flush-logs` une fois par jour, et d'effacer les fichiers de log qui ont plus de trois jours. Vous pouvez les supprimer manuellement, ou utilisez de préférence la commande `PURGE MASTER LOGS TO` ( Commandes SQL liées à la réplication ) qui va aussi modifier le fichier de log binaires pour vous.

Vous pouvez examiner le fichier de log binaire avec la commande `mysqlbinlog` . Par exemple, vous pouvez mettre à jour le serveur MySQL depuis la ligne de commande comme ceci :

```
shell> mysqlbinlog log-file | mysql -h server_name
```

Vous pouvez aussi utiliser le programme `mysqlbinlog` pour lire le fichier de log binaire directement dans le serveur MySQL.

`mysqlbinlog --help` vous donnera plus d'informations sur comment utiliser ce programme.

Si vous utilisez `BEGIN [WORK]` ou `SET AUTOCOMMIT=0` , vous devez utiliser le fichier de log binaire pour les sauvegardes, plutôt que le vieux fichier de log de modifications.

L'enregistrement dans le fichier de log binaire est fait immédiatement après l'achèvement de la requête, mais avant la libération des verrous ou la validation de la requête. Cela garantit que les requêtes seront enregistrées dans l'ordre d'exécution.

Les modifications dans les tables non transactionnelles sont enregistrées dans le fichier de log binaire immédiatement après exécution. Pour les tables transactionnelles comme `BDB` ou `InnoDB` , toutes les modifications ( `UPDATE` , `DELETE` ou `INSERT` ) qui modifient les tables sont mises en cache jusqu'à ce qu'une commande `COMMIT` ne les envoie au serveur. A ce moment, `mysqld` écrit la totalité de la transaction dans le log binaire, avant d'appliquer la commande `COMMIT` . Tous les threads vont, au démarrage, allouer un buffer de la taille de `binlog_cache_size` octets pour enregistrer les requêtes. Si la requête est plus grande que ce buffer, le thread va ouvrir un fichier temporaire pour écrire la transaction. Le fichier temporaire sera supprimé dès que le thread se termine.

L'option `max_binlog_cache_size` (par défaut 4Go) peut être utilisé pour limiter la taille utilisée pour mettre en cache une transaction multi-requête. Si la transaction est plus grande que que cette taille, elle sera annulée.

Si vous utilisez les log de modification ou binaire, les insertions concurrentes seront converties en insertions normales lors de l'utilisation de `CREATE ... SELECT` ou `INSERT ... SELECT` . Cela garantit que vous pourrez recréer une copie exacte de la table en appliquant les même commandes sauvegardées.

## 4.9.5 Le log des requêtes lentes

Lorsqu'il est démarré avec l'option `--log-slow-queries[=file_name]` , `mysqld` va écrire dans un fichier les requêtes SQL qui vont mettre plus de `long_query_time` secondes à s'exécuter. Le temps d'acquisition d'un verrou n'est pas compté.

Les requêtes lentes sont enregistrées après l'achèvement de l'exécution de la requête, et libération du verrou. Cela peut être différent de l'ordre dans lequel les commandes sont exécutées.

Si aucun nom de fichier n'est donné, le fichier de log prendra par défaut le nom de la machine, suffixé avec `-slow.log` . Si un nom de fichier est donné, mais qu'il manque le chemin, le fichier sera écrit dans le dossier de données.

Le log de requêtes lentes peut être utilisé pour repérer les requêtes qui prennent longtemps à s'exécuter, et donc, qui sont candidates à l'optimisation. Avec un grand fichier de log, cela peut devenir difficile. Vous pouvez alors passer le fichier de log à `mysqldumpslow` pour obtenir un sommaire des requêtes dans ce fichier.

Si vous utilisez l'option `--log-long-format` alors les requêtes qui n'utilisent pas d'index sont aussi enregistrées. Options en ligne de commande de `mysqld` .

## 4.9.6 Entretien des fichiers de log

Le serveur MySQL peut créer un grand nombre de fichiers de logs différents, qui permettent de suivre ce qui se passe. Fichiers de log . Vous devez toutefois nettoyer régulièrement ces fichiers, pour être sûr que les logs ne prennent pas tout le disque de la machine.

Lorsque vous utilisez MySQL avec des fichiers de log, vous voudrez, de temps en temps, supprimer ou sauvegarder les fichiers, et demander à MySQL d'utiliser de nouveaux fichiers. Sauvegardes de base de données .

Sous une installation Linux ( `Redhat` ), vous pouvez utiliser le script `mysql-log-rotate` pour cela. Si vous avez installé MySQL depuis une distribution RPM, le script doit avoir été installé automatiquement. Notez que vous devez être prudent avec cette commande si vous utilisez les logs pour la réplication.

Sur d'autres systèmes, vous devez installer un court script par vous même, qui sera exécuté via le démon `cron` .

Vous pouvez forcer MySQL à utiliser de nouveaux fichiers de log en utilisant la commande `mysqladmin flush-logs` ou avec la commande SQL `FLUSH LOGS` . Si vous utilisez MySQL version 3.21, vous devez utiliser `mysqladmin refresh` .

Les commandes ci-dessus effectue les tâche suivantes :

- Si le log standard ( `--log` ) ou le log de requêtes lentes ( `--log-slow-queries` ) est utilisé, la commande ferme et réouvre le fichier de log ( `mysql.log` et `` `hostname`-slow.log `` par défaut).
- Si le log de modifications est utilisé ( `--log-update` ), la commande ferme le log de modification et ouvre un nouveau fichier, avec un nouveau numéro de séquence plus grand.

Si vous utilisez uniquement le log de modification, vous pour simplement vider les logs sur le disque, et sauver l'ancien fichier de modification dans une sauvegarde. Si vous utilisez le log normal, vous pouvez faire ceci :

```
shell> cd mysql-data-directory
shell> mv mysql.log mysql.old
shell> mysqladmin flush-logs
```

puis faire une sauvegarde du fichier, et le supprimer ( `mysql.old` ).

# 4.10 Réplication de MySQL

Cette section décrit les différentes fonctionnalités de la réplication MySQL. Elle sert de référence pour les options disponibles avec la réplication. Vous y trouverez une introduction à la réplication. Vers la fin, vous y trouverez les questions et problèmes les plus fréquents, avec leur solution.

Nous vous suggérons de visiter notre site web http://www.mysql.com/ souvent pour y lire les mises à jour de cette section. La réplication est constamment améliorée, et nous modifions souvent le manuel.

## 4.10.1 Introduction à la réplication

La réplication monodirectionnelle est à améliorer la solidité et la vitesse de votre application. Pour la solidité, vous pouvez avoir une copie sur un serveur de sauvegarde, qui prend le relais immédiatement si le maître rencontre des problèmes. L'amélioration de la rapidité est obtenue en envoyant les requêtes sans modification sur un serveur qui ne fait que de la lecture. Bien sur, cela ne fonctionnera que si les lectures dominent, mais c'est le cas le plus répandu.

Depuis la version 3.23.15, MySQL supporte la réplication mono directionnelle, en interne. Un serveur sert de maître, et les autres serveurs d'esclaves. Notez qu'un serveur peut servir de maître pour certains, et d'esclaves pour d'autres serveurs. Le maître va conserver un log binaire des modifications ( Le log binaire des mises à jour ) et un fichier d'index des modifications à prendre en compte. L'esclave, au moment de la connexion, informe le maître où il s'est arrêté, rattrape les modifications qui ont eu lieux depuis, puis se bloque en attente des prochaines modifications.

Notez que si vous faites de la réplication de base, toutes les modifications de cette base devraient être faites sur le maître !

Un autre avantage de la réplication est que vous pouvez entretenir des sauvegardes immédiates de vos données, depuis le maître vers l'esclave. Sauvegardes de base de données .

## 4.10.2 Présentation de l'implémentation de la réplication

La réplication MySQL est basée sur le fait que le serveur va garder la trace de toutes les évolutions de vos bases (modifications, effacements, etc.) dans un fichier de log binaire ( Le log binaire des mises à jour ) et les esclaves vont venir lire les requêtes du maître dans ce fichier de log, pour pouvoir exécuter les mêmes requêtes sur leurs copies.

Il est **très important** de comprendre que le fichier de log binaire est simplement un enregistrement des modifications depuis un point fixe dans le temps (le moment où vous activez le log binaire). Tous les esclaves que vous activez auront besoin de la copie des données qui existaient au moment du démarrage du log. Si vous démarrez vos esclaves sur sans qu'ils ne disposent des données identiques à celles du maître **au moment du démarrage du log binaire** , votre réplication va échouer.

Voyez dans la table suivante les indications de compatibilité entre les maîtres et les esclaves,

suivant les versions. Depuis la version 4.0, nous recommandons d'utiliser la même version pour les maîtres et esclaves.

| | | Maître | Maître | Maître | Maître |
|---|---|---|---|---|---|
| | | 3.23.33 et + | 4.0.0 | 4.0.1 | 4.0.3 et + |
| Esclave | 3.23.33 et + | oui | non | non | non |
| Esclave | 4.0.0 | non | oui | non | non |
| Esclave | 4.0.1 | oui | non | oui | non |
| Esclave | 4.0.3 et + | oui | non | non | oui |

**Note** : MySQL version 4.0.2 n'est pas recommandé pour la réplication.

Depuis la version 4.0.0, vous pouvez utiliser la commande `LOAD DATA FROM MASTER` pour configurer un esclave. Soyez bien conscient qu'actuellement, `LOAD DATA FROM MASTER` ne fonctionne que si toutes les tables du maître sont du type `MyISAM` , et qu'il est possible d'obtenir un verrou de lecture global, pour qu'aucune lecture ne se fasse durant le transfert des tables depuis le maître. Cette limitation est de nature temporaire, et elle est dûe au fait que nous n'avons pas encore programmé un système de sauvegarde des tables sans verrou. La limitation sera supprimée dans la future version 4.0 une fois que nous aurons programmé le système de sauvegarde, qui permettra à `LOAD DATA FROM MASTER` de fonctionner sans bloquer le maître.

Etant donné la limitation ci−dessus, nous vous recommandons actuellement d'utiliser la commande `LOAD DATA FROM MASTER` uniquement si le jeu de données du maître est petit, ou si un verrou prolongé sur le maître est acceptable. Suivant la vitesse de lecture de `LOAD DATA FROM MASTER` en fonction des systèmes, une règle de base indique que le transfert se fera au rythme de 1 Mo par seconde. Vous pourrez ainsi obtenir une estimation du temps qu'il vous faudra pour transférer les données, si le maître et l'esclave sont connectés sur un réseau de 100 MBit/s, avec des configurations à base de Pentium 700 MHz. Bien sur, votre cas particulier pourra varier en fonction de votre système : la règle ci−dessus vous donnera une première évaluation du temps à attendre.

Une fois que l'esclave est correctement configuré, et qu'il fonctionne, il va simplement se connecter au maître et attendre des requêtes de modifications. Si le maître est indisponible ou que l'esclave perd la connexion avec le maître, il va essayer de se reconnecter toutes les `master−connect−retry` secondes jusqu'à ce qu'il soit capable d'établir la communication, et de recommencer à appliquer les modifications.

Chaque esclave garde la trace du point où il en était rendu. Le serveur maître n'a pas de notions du nombre d'esclave qui se connectent, ou qui sont à jour à un moment donné.

La prochaine section explique le processus de réplication en détails.

## 4.10.3 Comment mettre en place la réplication

Voici les instructions pour mettre en place la réplication sur votre serveur MySQL. Nous supposons que vous voulez répliquer toutes vos bases, et que vous ne l'avez jamais configuré auparavant. Vous aurez besoin d'éteindre brièvement le serveur principal pour suivre toutes les instructions.

Si cette méthode n'est pas la plus simple pour configurer un esclave, ce n'est pas la seule. Par exemple, si vous avez déjà une sauvegarde des données du maître, et que le maître a déjà un identifiant de serveur, et le log binaire activé, vous pouvez configurer l'esclave sans éteindre le serveur et sans bloquer les mises à jours. Pour plus de détails, voyez FAQ de la réplication .

Si vous voulez devenir un véritable gourou de la réplication MySQL, nous vous suggérons de commencer par étudier, tester et expérimenter toutes les commandes mentionnées dans le chapitre Commandes SQL liées à la réplication . Vous devriez aussi vous familiariser avec les options de

démarrage de `my.cnf` , Options de réplication dans le fichier `my.cnf` .

- Assurez vous que vous avez une version récente de MySQL installée comme maître et comme esclave.Utilisez les versions 3.23.29 et plus récente. Les anciennes versions utilisaient un autre format de log binaire, et avaient des bogues qui ont été corrigés dans les nouvelles versions. Ne nous rapportez pas de bogues avant d'avoir vérifié si le problème est encore présent dans la dernière version de MySQL.
- Créez un utilisateur MySQL spécial pour la réplication sur le maître, avec les droits de `FILE` (dans les versions plus anciennes que la versions 4.0.2) ou le droit de `REPLICATION SLAVE` pour les nouvelles versions. Vous devez aussi lui donner les droits de connexion depuis tous les esclaves. Si l'utilisateur ne fait que de la réplication (ce qui est recommandé), vous n'avez pas à lui donner d'autres droits.Par exemple, pour créer un utilisateur appelé `repl` qui peut accéder au maître, vous pourriez utiliser une commande comme :
```
mysql> GRANT FILE ON *.* TO repl&#64;"%" IDENTIFIED BY '<password>';
```
- Eteignez le serveur MySQL maître.
```
mysqladmin -u root -p<password> shutdown
```
- Sauvegardez toutes les données de votre serveur maître.

  Le plus simple pour cela (sous Unix) et d'utiliser la commande `tar` pour produire une archive de votre dossier de données total. Le dossier de données dépend de votre installation.

```
tar -cvf /tmp/mysql-snapshot.tar /path/to/data-dir
```

  Les utilisateurs Windows peuvent utiliser le programme `WinZIP` ou un logiciel similaire pour créer une archive du dossier de données.
- Dans le fichier `my.cnf` du maître, ajouter les options `log-bin` et `server-id=unique number` à la section `[mysqld]` et redémarrez le serveur. Il est très important que l'identifiant des esclaves soient différents de celui du maître. Pensez à `server-id` comme à une valeur comparable à une adresse IP : elle identifie de manière unique un serveur dans la communauté des réplicateurs.
```
 [mysqld]
log-bin
server-id=1
```
- Redémarrez le serveur maître MySQL.
- Ajoutez les lignes de configuration suivantes dans le fichier `my.cnf` des esclaves :
```
 master-host=<nom d'hôte du maître>
master-user=<nom d'utilisateur de réplication>
master-password=<mot de passe de l'utilisateur de réplication>
master-port=<port TCP/IP du maître>
server-id=<un identifiant unique, entre 2 et 2^32-1>
```
  Remplacez les valeurs entre `<>` avec ce qui correspond à votre système.

  `server-id` doit être différent pour chaque serveur qui participe à la réplication. Si vous ne spécifiez pas d'identifiant `server-id` , il sera mis à 1 si vous n'avez pas défini de `master-host` , et sinon, il vaudra 2. Notez que dans le cas où vous omettez le `server-id` , le maître va refuser les connexion des esclaves, et les esclaves vont refuser de se connecter au maître. Par conséquent, omettre `server-id` n'est bon que pour faire des sauvegarde du log binaire.
- Copiez la sauvegarde des données dans vos esclaves. Assurez vous que les droits sur ces données sont corrects. L'utilisateur qui fait fonctionner MySQL doit avoir les droits d'écriture et de lecture sur ces fichiers, tout comme le maître l'avait.
- Redémarrez les esclaves

Après avoir suivi les instructions ci-dessus, les esclaves doivent se connecter au maître, et rattraper les modifications qui ont eu lieu depuis la sauvegarde des données.

Si vous avez oublié de spécifier un `server-id` pour un esclave, vous allez obtenir l'erreur suivante dans le fichier d'erreur :

```
 Warning: one should set server_id to a non-0 value if master_host is set.
The server will not act as a slave.
```

Si vous avez oublié de le faire pour le maître, les esclaves ne pourront pas se connecter avec le maître.Si un esclave n'est pas capable de faire la réplication pour une raison quelconque, vous allez trouvez le message d'erreur dans le fichier de log d'erreurs de l'esclave.

Une fois qu'un esclave fonctionne, vous allez trouver un fichier appelé `master.info` dans le même dossier que vos fichiers de logs. Le fichier `master.info` est utilisé par l'esclave pour savoir où il en est rendu dans le log binaire du maître. **Ne le supprimez pas** et ne l'éditez pas, à moins que vous ne sachiez vraiment ce que vous faites. Même dans ce cas, il est préférable d'utiliser la commande `CHANGE MASTER TO` .

## 4.10.4 Fonctionnalités de la réplication et problèmes connus

Voici une liste des fonctionnalités supportées par la réplication de MySQL :

- La réplication s'effectue correctement sur les valeurs `AUTO_INCREMENT` , `LAST_INSERT_ID()` et `TIMESTAMP` .
- `RAND()` dans les commandes `UPDATE` ne se réplique pas correctement. Utilisez `RAND(some_non_rand_expr)` si vous répliquez des `UPDATE` avec `RAND()` . Vous pourriez, par exemple, utiliser `UNIX_TIMESTAMP()` comme argument de `RAND()` .
- Vous devez utiliser le même jeu de caractères ( `--default-character-set` ) sur le maître et l'esclave. Si ce n'est pas le cas, vous obtiendrez des erreurs de clés doublons dans l'esclave, car une clé qui est indiquée comme unique sur le maître peut se révéler un doublon pour l'esclave.
- En version 3.23, `LOAD DATA INFILE` sera géré proprement, tant que le fichier réside toujours sur le maître au moment de la propagation de la réplication. Sinon, `LOAD LOCAL DATA INFILE` sera ignoré. En version 4.0, cette limitation a été levée (toutes les formes de `LOAD DATA INFILE` sont correctement répliquées).
- Les requêtes d' `UPDATE` qui utilisent des variables utilisateurs ne sont pas correctement répliquées (pour le moment).
- Les commandes `FLUSH` ne sont pas enregistrées dans le fichier de log car elle ne sont pas répliquées aux esclaves. Ce n'est pas normalement un problème, car `FLUSH` ne modifie pas les tables. Cela peut signifier que vous avez modifié des droits dans les tables MySQL directement sans la commande `GRANT` et que vous avez répliqué les droits de `mysql` sans pouvoir faire de commande `FLUSH PRIVILEGES` sur vos esclaves pour les prendre en compte.
- Les tables temporaires sont répliquées depuis la version 3.23.29, à l'exception des cas où vous éteignez le serveur esclave (et pas juste le thread esclave), que vous avez des tables temporaires ouvertes et qu'elles sont utilisées dans des modifications ultérieures. Pour régler ce problème, utilisez la commande `SLAVE STOP` , vérifiez la variable de statut `Slave_open_temp_tables` pour vérifier si elle vaut bien 0, puis exécutez `mysqladmin shutdown` . Si le nombre n'est pas 0, redémarrez l'esclave avec la commande `SLAVE START` et voyez si vous avez plus de chance la prochaine fois. Il y aura une solution plus élégante, mais il doit attendre la version 4.0. Dans les anciennes versions, les tables temporaires n'étaient pas répliquée : nous vous suggérons de mettre à jour votre version ou d'exécuter la commande `SET SQL_LOG_BIN=0` sur vos clients avant toute requête qui utiliserait les tables temporaires.
- MySQL ne supporte qu'un seul maître, et plusieurs esclaves. En version 4.x, nous allons ajouter un algorithme de sélection pour que le maître soit modifié si le maître initial flanche. Nous allons aussi introduire une notion d'agent dans les processus, pour aider la répartition de charge en répartissant les requêtes sur les esclaves.
- Depuis la version 3.23.26, il est possible de connecter les serveurs MySQL en chaîne bouclée (chaque serveur est le maître du précédent et l'esclave du suivant, en boucle), avec l'activation de l'option `log-slave-updates` . Notez que de nombreuses requêtes ne vont pas fonctionner dans ce type de configuration à moins que votre code client ne soit écrit avec beaucoup de soin, pour qu'il se charge des problèmes qui pourraient arriver dans différentes séquences de modifications sur différents serveurs.Cela signifie que vous pouvez réaliser

une configuration comme ceci :

```
A -> B -> C -> A
```

Cette configuration va fonctionner si vous ne faites que des modifications non exclusives sur les tables. En d'entre terme, si vous insérez des données dans les serveurs A et C, vous ne devez pas insérer une valeur dans le serveur A qui pourrait être en conflit avec une ligne insérée dans le serveur C. Vous ne devez pas non plus modifier de ligne sur deux serveurs en même temps, pour éviter les conflits de valeurs.Notez que le format de log a été modifié en version 3.23.26 de façon à ce que les esclaves des versions antérieure à la 3.23.26 ne soient pas capable de les lire.

- Si la requête sur l'esclave génère une erreur, le thread esclave s'arrête, et un message sera ajouté dans le fichier d'erreurs `.err` . Vous devriez vous connecter pour corriger manuellement les données de l'esclave, puis relancer l'esclave avec la commande `SLAVE START` (disponible depuis la version 3.23.16. En version 3.23.15, vous devrez redémarrer le serveur.
- Si la connexion au maître est perdue, l'esclave tente de se reconnecter immédiatement, et en cas d'échec, il va retenter toutes les `master-connect-retry` (par défaut, 60) secondes. A cause de cela, il est sage d'éteindre le serveur maître et de le redémarrer régulièrement. L'esclave sera capable de gérer les problèmes réseau.
- Eteindre l'esclave proprement est sûr, car il garde la trace du point où il en est rendu. Les extinctions sauvages vont produire des problèmes, surtout si le cache disque n'a pas été écrit sur le disque avant que le système ne s'arrête. Votre niveau de tolérance aux pannes sera grandement amélioré si vous avez de bons onduleurs.
- Si le maître écoute sur un port non−standard, vous devez aussi le spécifier dans le paramètre `master-port` du fichier `my.cnf` .
- En version 3.23.15, toutes les tables et bases seront répliquées. Depuis la version 3.23.16, vous pouvez limiter la réplication à certaines bases, grâce à l'option `replicate-do-db` du fichier `my.cnf` ou simplement d'exclure des bases avec `replicate-ignore-db` . Notez que jusqu'à la version 3.23.23, il y avait un bogue qui ne traitait correctement les commandes `LOAD DATA INFILE` si vous les utilisiez dans une base qui étaient exclue de la réplication.
- Depuis la version 3.23.16, `SET SQL_LOG_BIN = 0` va désactiver l'enregistrement des données de réplication (binaire) sur le maître, et `SET SQL_LOG_BIN = 1` la réactivera. Vous aurez besoin des droits de `SUPER` (en MySQL 4.0.2 et plus récent), ou `PROCESS` (dans les anciens droits MySQL) pour faire cela.
- Depuis la version 3.23.19, vous pouvez nettoyer les restes de la réplication lorsqu'un problème est survenu avec les commandes `FLUSH MASTER` et `FLUSH SLAVE` . En version 3.23.26, nous avons renommé cette commande en `RESET MASTER` et `RESET SLAVE` respectivement, pour clarifier leur action. Les anciennes variantes de `FLUSH` fonctionnent encore, pour la compatibilité.
- Depuis la version 3.23.23, vous pouvez modifier les maîtres et ajuster la position de log avec la commande `CHANGE MASTER TO` .
- Depuis la version 3.23.23, vous pouvez dire au maître que les modifications de certaines bases ne doivent pas être enregistrées dans le fichier de log binaire, avec l'option `binlog-ignore-db` .
- Depuis la version 3.23.26, vous pouvez utiliser l'option `replicate-rewrite-db` pour dire à l'esclave d'appliquer les modifications d'une base dans une autre.
- Depuis la version 3.23.28, vous pouvez utiliser la commande `PURGE MASTER LOGS TO 'log-name'` pour vous débarasser des anciens logs sur l'esclave. Cela va supprimer les vieux logs, mais pas le log appelé `'log-name'` .
- Etant donné la nature non transactionnelle des tables MySQL, il est possible qui va ne faire qu'une partie de la modification, et retourner une erreur. Cela peut arriver, par exemple, dans une insertion multiple dont une des lignes viole une contrainte d'unicité, ou si un très long `UPDATE` est interrompu au milieu du stock de ligne. Si cela arrive sur le maître, l'esclave va s'arrêter et attendre que l'administrateur décide quoi faire, à moins que l'erreur soit légitime, et que la requête arrive à la même conclusion. Si le code d'erreur n'est pas désirable, certaines erreurs (voire même toutes), peuvent être masquées avec l'option `slave-skip-errors` , depuis la version 3.23.47.

- Alors que des tables particulières peuvent être exclues de la réplication avec les options `replicate-do-table` et `replicate-ignore-table` ou `replicate-wild-do-table` et `replicate-wild-ignore-table` , il y a actuellement des problèmes de conception, qui provoque des résultats inattendus dans certains cas très rares. Le protocole de réplication n'informe pas explicitement l'esclave quelles tables vont être modifiées par la requête : l'esclave doit donc analyser lui−même la requête pour le savoir. Pour éviter une analyse répétitive des requêtes qui seront finalement exécutées, l'exclusion de table est actuellement implémentées en envoyant la requête à l'analyseur MySQL, qui va court circuiter la requête, et qui indiquera la réussite si une table utilisée doit être ignorée. En plus de plusieurs sources de ralentissement, cette approche va fatalement engendrer des bogues, et il en existe deux depuis la version 3.23.49 : car l'analyseur ouvre automatiquement la table lors de l'analyse de la requête, la table qui doit être ignorée doit donc exister sur l'esclave. L'autre bogue est que si le thread esclave modifie partiellement la table à ignorer, le thread ne va pas réaliser que la table aurait du être ignorée, et va suspendre la réplication. Alors que le premier bogue est conceptuellement simple à corriger, nous n'avons pas encore trouvé de moyen pour corriger le second sans intervenir massivement dans le code, et donc, en compromettre la stabilité. Il existe un palliatif pour les deux, dans les cas rares ou cela arrive à votre application : utilisez `slave-skip-errors` .

## 4.10.5 Options de réplication dans le fichier my.cnf

Si vos utilisez la réplication, nous vous recommandons d'utiliser MySQL version 3.23.30 ou plus récent. Les vieilles versions fonctionneront, mais elles présentent des bogues et manquent les nouvelles fonctionnalités. Certaines des options qui seront présentées ici ne seront pas forcément disponibles dans votre version de MySQL, si vous n'utilisez pas la version la plus récente. Pour toutes les options spécifiques à la version 4.0, il y a une note qui l'indique. Sinon, si vous découvrez une option qui vous interesse mais qui n'est pas disponible dans votre version 3.23, et que vous en avez vraiment besoin, mettez à jour votre installation.

Notez bien que la version 4.0 est toujours en phase alpha, et que certaines fonctionnalités ne fonctionneront peut être pas aussi bien que prévu. Si vous voulez vraiment essayer de nouvelles fonctionnalités de la version 4.0, nous vous recommandons de le faire de façon à ce que cela ne perturbe pas vos applications critiques.

Sur le maître et l'esclave, vous devez utiliser l'option `server-id` . Elle doit représenter un identifiant unique de réplication. Vous devriez choisir une valeur unique entre 1 et 2^32−1 pour chaque maître et esclave. Example: `server-id=3`

La table suivante décrit les options que vous pouvez utiliser avec `MASTER` :

| Option | Description |
|---|---|
| log−bin=filename | Ecrit le fichier de log binaire à la position indiquée. Notez que vis vous donner un paramètre avec une extension (par exemple, log−bin=/mysql/logs/replication.log ), les versions jusqu'en 3.23.24 ne fonctionneront pas correctement avec la réplication, si vous utilisez la commande FLUSH LOGS . Le problème a été corrigé avec la version 3.23.25. Si vous utilisez ce type de nom de fichier de log, FLUSH LOGS sera ignoré sur les logs binaires. Pour nettoyer le log, exécutez la commande FLUSH MASTER , et n'oubliez pas la commande FLUSH SLAVE sur chacun des esclaves. En version 3.23.26 et plus récent, vous devriez utiliser RESET MASTER et RESET SLAVE . |
| log−bin−index=filename | Comme l'utilisateur peut exécuter une commande FLUSH LOGS , nous devons savoir quel fichier de log est actuellement actif, lequels ont été déjà utilisés, et dans quel ordre. Cette information est stockée dans le fichier d'index de log. Par défaut, ce fichier s'appelle `hostname`.index . Nous n'avez pas à le modifier.Example: log−bin−index=db.index |
| sql−bin−update−same | Si cette option est activée, en donnant une valeur à SQL_LOG_BIN vous donnerez automatiquement la même valeur à SQL_LOG_UPDATE et vice−versa. |
| binlog−do−db=database_name | Indique au maître qu'il doit mettre les modifications dans le fichier de log binaire, si la base courante est la base database_name . Toutes les autres bases sont ignorées. Notez que vous si vous utilisez cette option, vous devez vous assurer que vous ne faites des modifications que dans la base courante.<br><br>Example: `binlog−do−db=sales` |

| | |
|---|---|
| binlog-ignore-db=database_name | Indique au maître que les modifications dont la base courante est database_name ne doivent pas être stockées dans le log binaire. Notez que vous si vous utilisez cette option, vous devez vous assurer que vous ne faites des modifications que dans la base courante.Example: binlog-ignore-db=accounting |

La table suivante décrit les options que vous pouvez utiliser avec SLAVE :

| Option | Description |
|---|---|
| master-host=host | Le nom d'hôte du maître, ou l'adresse IP de réplication. Si cette valeur n'est pas spécifiée, l'esclave ne démarrera pas. Notez que la configuration de master-host sera ignorée si il existe un fichier master.info valide. Il est probable qu'un nom mieux choisi pour cette option aurait été bootstrap-master-host , mais il est trop tard aujourd'hui.<br><br>Example: master-host=db-master.mycompany.com |
| master-user=username | Le nom d'utilisateur que l'esclave va utiliser pour s'identifier lors de la connexion au maître. Cet utilisateur doit avoir les droits de FILE . S'utilisateur maître n'est pas configuré, l'utilisateur test est utilisé. La valeur dans master.info aura priorité, si elle peut être lue.<br><br>Example: master-user=scott |
| master-password=password | Le mot de passe que l'esclave va utiliser automatiquement avec le maître. Si il n'est pas configuré, le mot de passe vide sera utilisé. La valeur dans master.info aura priorité, si elle peut être lue.<br><br>Example: master-password=tiger |
| master-port=portnumber | Le port sur lequel le maître écoute. Si cette valeur n'est pas configurée, la valeur compilée de MYSQL_PORT est utilisée. Si vous n'avez pas joué avec les options du script configure , ce devrait être le port 3306. La valeur dans master.info aura priorité, si elle peut être lue.<br><br>Example: master-port=3306 |
| master-connect-retry=seconds | Le nombre de secondes que le thread esclave va attendre avant de tenter à nouveau de se connecter au maître, dans le cas où le maître n'est plus joignable. Par défaut, c'est 60.<br><br>Example: master-connect-retry=60 |
| master-ssl | Disponible après la version 4.0.0. Active le mode SSL pour la réplication. Soyez prévenu que c'est une fonctionnalité très récente.<br><br>Example: master-ssl |
| master-ssl-key | Disponible après la version 4.0.0. Le nom du fichier de clé maître SSL. Uniquement valable si l'option master-ssl est active.<br><br>Example: master-ssl-key=SSL/master-key.pem |
| master-ssl-cert | Disponible après la version 4.0.0. Le nom du fichier de certificat SSL. Uniquement valable si l'option master-ssl est active.Example: master-ssl-key=SSL/master-cert.pem |
| master-info-file=filename | La position du fichier qui enregistre la dernière opération faite avec le maître, durant la réplication. Le fichier par défaut est master.info dans le dossier de données. Vous n'avez pas besoin de modifier cela.Example: master-info-file=master.info |
| report-host | Disponible après la version 4.0.0. Le nom d'hôte ou l'adresse IP de l'esclave, qui doit être enregistrée lors de l'identification. Cette valeur apparaîtra dans le résultat de la commande SHOW SLAVE HOSTS . Laissez la vide si vous ne souhaitez pas que l'esclave s'enregistre lui-même. Notez qu'il n'est pas suffisant pour le maître de simplement lire l'IP de l'esclave dans la socket de connexion. A cause du NAT et d'autres problèmes de routage, cette IP peut ne pas êtra valable pour la connexion du maître sur l'esclave.Example: report-host=slave1.mycompany.com |
| report-port | Disponible après la version 4.0.0. Port de connexion à l'esclave indiqué au maître durant l'enregistrement. Utilisez cette option uniquement si l'esclave doit écouter sur un port non standard, ou si vous avez un tunnel particulier pour le maître. En cas de doute, laissez le vide. |
| replicate-do-table=db_name.table_name | Restreint la réplication à certaines tables. Pour spécifier plus d'une table, vous pouvez utiliser cette directive plusieurs fois, une pour chaque table. Cette option va fonctionner pour les modifications inter-bases, contrairement à replicate-do-db .<br><br>Example: replicate-do-table=some_db.some_table |
| replicate-ignore-table=db_name.table_name | Indique à l'esclave d'ignorer les commandes qui modifie les tables spécifiées (même si d'autres tables de la même commande sont aussi modifiées). Pour spécifier plus d'une table, vous pouvez utiliser cette directive plusieurs fois, une pour chaque table. Cette option va fonctionner pour les modifications inter-bases, contrairement à replicate-ignore-db .<br><br>Example: replicate-ignore-table=db_name.some_table |
| replicate-wild-do-table=db_name.table_name | Restreint la réplication à certaines tables qui vérifient le masque d'expression régulière indiqué. Pour spécifier plus d'une table, vous pouvez utiliser cette directive plusieurs fois, une pour chaque table. Cette option va fonctionner pour les modifications inter-bases. |

| | |
|---|---|
| | Example: `replicate-wild-do-table=foo%.bar%` va répliquer uniquement les modifications qui utilisent une table dont le nom de base commence par `foo` , et le nom de table commence par `bar` . |
| `replicate-wild-ignore-table=db_name.table_name` | Interdit la réplication à certaines tables qui vérifient le masque d'expression régulière indiqué. Pour spécifier plus d'une table, vous pouvez utiliser cette directive plusieurs fois, une pour chaque table. Cette option va fonctionner pour les modifications inter-bases.<br><br>Example: `replicate-wild-ignore-table=foo%.bar%` va ignorer la réplication des modifications qui utilisent une table dont le nom de base commence par `foo` , et le nom de table commence par `bar` . |
| `replicate-ignore-db=database_name` | Restreint la réplication aux bases de données indiquées. Pour spécifier plus d'une base, vous pouvez utiliser cette directive plusieurs fois, une pour chaque base. Vous ne devez pas utiliser cette option si vous faites des modifications inter-bases et que vous ne voulez pas que ces modifications soient prises en compte.<br><br>La principale raison de ce comportement est qu'il est difficile d'utiliser uniquemnt la ligne de commande pour savoir si une requête doit être répliquée ou pas. Par exemple, si vous utilisez une commande d'effacement multi-table en MySQL 4.x, qui utilise plusieurs bases. Il est aussi très facile de vérifier la base courante, car cela ne s'applique qu'une fois au moment de la connexion, ou d'un changement de base.<br><br>Si vous voulez que des modifications inter-bases fonctionnent, assurez vous que vous avez la version 3.23.28 ou plus récente, et utilisez `replicate-wild-ignore-table=db_name.%` .<br><br>Example: `replicate-ignore-db=some_db` |
| `replicate-do-db=database_name` | Restreint la réplication aux commandes dont le nom de base courante est `database_name` . Pour spécifier plus d'une base, vous pouvez utiliser cette directive plusieurs fois, une pour chaque base. Notez que cette option ne va pas autoriser la réplication de requêtes inter-bases telles que `UPDATE some_db.some_table SET foo='bar'` lorsque vous avez sélectionné une autre base (ou qu'aucune base n'est sélectionnée). Si vous avez besoin de faire des modifications inter-bases, assurez vous que vous avez la version 3.23.28 ou plus récente, et utilisez `replicate-wild-do-table=db_name.%` .<br><br>Example: `replicate-do-db=some_db` |
| `log-slave-updates` | Indique à l'esclave d'enregistrer les modifications du thread esclave dans le log binaire. Par défaut, cette option est inactive (off). Vous aurez besoin de cette fonctionnalité si vous installez les esclaves en cascade. |
| `replicate-rewrite-db=from_name->to_name` | Reporte les modifications qui ont lieu dans une base, dans une autre.Example: `replicate-rewrite-db=master_db_name->slave_db_name` |
| `slave-skip-errors= [err_code1,err_code2,... \| all]` | Disponible depuis la version 3.23.47 et plus récent. Indique au thread esclave de continuer la réplication si une requête génère une erreur issue de la liste fournie. Normalement, la réplication va s'arrêter dès qu'une erreur survient, pour que l'utilisateur puisse corriger les incohérence manuellement. N'utilisez pas cette option si nous ne comprenez pas complètement les raisons qui génèrent ces erreurs. Si il y n'a pas de bogues dans vos configurations de réplication, et pas de bogues dans MySQL, vous ne devriez jamais interrompre la réplication. Utiliser cette option à tort et à travers va désynchroniser les esclaves et vous n'aurez aucune idée des problèmes qui sont survenus.Pour les codes d'erreurs, vous devez utiliser les numéros d'erreurs qui sont fournis dans le log d'erreur de l'esclave, et dans le résultat de la commande `SHOW SLAVE STATUS` . La liste complète des message d'erreurs sont disponibles dans le fichier source `Docs/mysqld_error.txt` .<br><br>Vous pouvez (mais ne devriez pas) aussi utiliser la valeur hautement dangeureuse de `all` (tous), pour ignorer toutes les erreurs de réplication, et continuer comme si de rien n'était. Inutile de vous dire que si vous l'utilisez, nous ne pouvons rien garantir quand à l'intégrité de vos données. Ne venez pas vous plaindre que les données de vos esclaves sont très différentes des données du maître : vous aurez été prévenus.<br><br>Exemple :<br><br>`slave-skip-errors=1062,1053` or `slave-skip-errors=all` |
| `skip-slave-start` | Indique au serveur esclave de ne pas démarrer l'esclave au démarrage. L'utilisateur le démarrera manuellement plus tard, avec `SLAVE START` . |
| `slave_compressed_protocol=#` | Si cette option vaut 1, alors le protocole compressé est utilisé entre le maître et l'esclave, si les deux le supporte. |

| | |
|---|---|
| slave_net_timeout=# | Le nombre de secondes d'attente de données depuis le maître, avant d'abandonner la lecture. |

## 4.10.6 Commandes SQL liées à la réplication

La réplication peut être contrôlée depuis l'interface SQL. Voici un résumé des commandes disponibles :

| Commande | Description |
|---|---|
| SLAVE START | Démarre le thread esclave. Depuis MySQL 4.0.2, vous pouvez ajouter les options IO_THREAD ou SQL_THREAD à cette commande, pour démarrer le thread E/S ou le thread SQL. Le thread E/S lit les requêtes depuis le maître, et les stocke dans le log de relais. Le thread SQL lit le log de relais, et exécute les requêtes. (Esclave) |
| SLAVE STOP | Stoppe le thread esclave. Comme SLAVE START , cette commande peut être utilisée avec les options IO_THREAD et SQL_THREAD . (Esclave) |
| SET SQL_LOG_BIN=0 | Désactive le log binaire, si l'utilisateur a les droits de SUPER . Ignoré sinon. (Maître) |
| SET SQL_LOG_BIN=1 | Réactive le log binaire, si l'utilisateur a les droits de SUPER . Ignoré sinon. (Maître) |
| SET GLOBAL SQL_SLAVE_SKIP_COUNTER=n | Ignore les n prochains événements du maître. Uniquement valide lorsque le thread esclave ne fonctionne pas, sinon une erreur est affichée. Très pratique pour corriger des petits problèmes de réplications. |
| RESET MASTER | Efface tous les logs binaires dans le fichier d'index, et vide le fichier de log binaire. Dans les versions antérieures à la 3.23.26, utilisez FLUSH MASTER . (Maître) |
| RESET SLAVE | Efface la position de réplication de l'esclave dans les logs du maître. Dans les versions antérieures à la 3.23.26, cette commande s'appelait FLUSH SLAVE . (Esclave) |
| LOAD TABLE tblname FROM MASTER | Télécharge une copie des tables depuis le maître vers l'esclave. Implémentée principalement pour le débogage de LOAD DATA FROM MASTER , mais certains utilisateurs ``gourmets'' pourront la trouver pratique pour d'autres utilisations. Ne l'utilisez pas si vous vous considérez comme un utiliser ``non-hacker''. (Esclave) |
| LOAD DATA FROM MASTER | Disponible depuis 4.0.0. Prend une copie des données du maître et les transfère vers l'esclave. Met à jour les valeurs de MASTER_LOG_FILE et MASTER_LOG_POS de manière à ce que l'esclave démarre la réplication depuis la position correcte. Cette commande respecte les règles d'exclusion des tables et des bases spécifiées dans les options replicate-* . Actuellement fonctionne uniquement avec les tables MyISAM et pose un verrou global en lecture sur toutes les tables, pour faire la sauvegarde. Dans le futur, il est prévu de rendre cette commande compatible avec les tables InnoDB et de supprimer le besoin du verrou global pour faire une sauvegarde non bloquante. |
| CHANGE MASTER TO master_def_list | Change les paramètres du maître avec les valeurs spécifiées dans l'argument master_def_list et redémarre le thread esclave. master_def_list est une liste séparée par des virgules, avec les variables suivantes : MASTER_HOST , MASTER_USER , MASTER_PASSWORD , MASTER_PORT , MASTER_CONNECT_RETRY , MASTER_LOG_FILE et MASTER_LOG_POS . Par exemple :<br>`CHANGE MASTER TO`<br>`MASTER_HOST='master2.mycompany.com',`<br>`MASTER_USER='replication',`<br>`MASTER_PASSWORD='bigs3cret',`<br>`MASTER_PORT=3306,`<br>`MASTER_LOG_FILE='master2-bin.001',`<br>`MASTER_LOG_POS=4;`<br>Vous n'avez pas besoin de spécifier les valeurs que vous ne souhaitez pas changer. Les valeurs omises conserveront leur valeur actuelle, même si vous modifiez l'hôte ou le port. Dans ce cas, l'esclave supposer que puisque vous vous connectez à un nouvel hôte ou un nouveau port, le maître est différent. Par conséquent, il les anciennes valeurs de log et position ne sont plus applicables, et il va automatiquement leur donner la valeur de la chaîne vide, et de 0, respectivement, qui sont les valeurs de démarrage. Notez qui si vous redémarrez l'esclave il va se souvenir de son ancien maître. Si vous ne le souhaitez pas, il suffit d'effacer le fichier master.info avant de le redémarrer, et l'esclave va lire la valeur depuis le fichier my.cnf ou la ligne de commande.Cette commande est pratique pour configurer un esclave lorsque vous avez une sauvegarde du maître, et que vous avez sauvé la valeur du log et de son offset chez le maître. Vous pouvez alors exécuter la commande CHANGE MASTER TO MASTER_LOG_FILE='log_name_on_master', MASTER_LOG_POS=log_offset_on_master sur l'esclave après avoir restauré la sauvegarde. (Esclave) |
| SHOW MASTER STATUS | Fournit les informations de statut sur le point de contrôle du fichier de log binaire du maître. (Maître) |
| SHOW SLAVE HOSTS | Disponible depuis la version 4.0.0. Cette commande liste les esclaves actuellement enregistré auprès du maître. (Maître) |
| SHOW SLAVE STATUS | Fournit les informations essentielles sur le statut de l'esclave. (Esclave) |
| SHOW MASTER LOGS | Uniquement disponible depuis la version 3.23.28. Liste les logs binaires sur le maître. Vous devriez utiliser cette commande avant d'utiliser la commande PURGE MASTER LOGS TO pour savoir lesquels vous voulez effacer. (Maître) |
| SHOW BINLOG EVENTS [ IN 'logname' ] [ FROM pos ] [ LIMIT [offset,] rows ] | Affiche les événements dans le log binaire. Initialement utilisé pour le test et le débogage dans le log binaire, mais il peut aussi être utilisé par les clients habituels qui, pour une raison ou une autre, ont besoin de lire le contenu du log binaire. (Maître) |
| SHOW NEW MASTER FOR SLAVE WITH MASTER_LOG_FILE='logfile' AND MASTER_LOG_POS=pos AND MASTER_LOG_SEQ=log_seq AND MASTER_SERVER_ID=server_id | Cette commande est utilisée lorsque l'esclave d'un maître probablement mort ou indisponible doit être redirigé vers un autre maître qui fait la réplication du même maître. La commande va retourner les nouvelles coordonnées de réplication (le nom du point de contrôle dans le fichier de log binaire et son offset). Le résultat de la commande peut être utilisé dans un appel à la commande CHANGE MASTER TO . Les utilisateurs normaux ne doivent jamais exécuter cette commande. Elle est réservée à l'utilisation interne par le code de réplication sans erreur. Nous risquons de changer cette syntaxe, si vous trouvons un autre moyen plus intuitif de faire cette opération. |
| PURGE MASTER LOGS TO 'logname' | Disponible depuis la version 3.23.28. Efface tous les logs de réplication qui sont listés dans le fichier d'index de réplication, comme étant des logs déjà utilisés. Le log qui est spécifié devient alors log courant. Exemple :<br>`PURGE MASTER LOGS TO 'mysql-bin.010'`<br>Cette commande ne fait rien et échoue avec une erreur si vous avez un esclave actif qui est actuellement en train de lire un des fichiers de logs que vous essayez d'effacer. Cependant, si vous avez un esclave qui dort, et que vous effacez un des logs qu'il voudra lire plus tard, l'esclave ne pourra pas effectuer la réplication lorsque son temps arrivera. La commande est sûre lorsque d'autres esclaves effectue la réplication : vous n'aurez pas à les arrêter. |

| | |
|---|---|
| | Vous devez d'abord vérifier tous les esclaves, avec la commande `SHOW SLAVE STATUS` pour voir quels logs ils utilisent, pour faire une liste des logs sur le maître avec la commande `SHOW MASTER LOGS` , trouver le plus vieux log de tous (si tous les esclaves sont mis à jours, cela doit être le dernier de la liste), effectuer une sauvegarde optionnelle de tous les logs, puis supprimer les logs du serveur. |

## 4.10.7 FAQ de la réplication

**Q** : Comment puis−je configurer un esclave si le maître fonctionne déjà, et que je ne veux pas le stopper?

**R** : Il y a plusieurs solutions. Si vous avez effectué une sauvegarde du maître à un moment et enregistré le nom et l'offset du binlog (issu du résultat de la commande `SHOW MASTER STATUS` ) correspondant à la sauvegarde, faites ceci :

- Assurez vous qu'un identifiant unique est assigné à l'esclave.
- Exécutez la commande `CHANGE MASTER TO MASTER_HOST='master−host−name', MASTER_USER='master−user−name', MASTER_PASSWORD='master−pass', MASTER_LOG_FILE='recorded−log−name', MASTER_LOG_POS=recorded_log_pos`
- Exécutez la commande `SLAVE START`

Si vous n'avez pas de copie de sauvegarde, voici un moyen rapide d'en faire une :

- `FLUSH TABLES WITH READ LOCK`
- `gtar zcf /tmp/backup.tar.gz /var/lib/mysql` ( ou toute variation de cette commande)
- `SHOW MASTER STATUS` – assurez vous de noter le résultat. Vous en aurez besoin ultérieurement.
- `UNLOCK TABLES`

Après cela, suivez les instructions comme pour le cas où vous avez déjà votre sauvegarde, et que vous avez enregistré le nom et l'offset du point de contrôle du log binaire. Tant que les logs binaires du serveur sont toujours là, vous allez pouvoir rattrapper tout ce qui se fait sur le serveur principal. Vous pourriez même attendre plusieurs jours ou mois avant de mettre en place votre esclave. En théorie, le temps d'attente peut être infini. En pratique, les limitations sont l'espace disque du maître, et le temps que cela prendra à l'esclave pour rattrapper le temps.En version 4.0.0 et plus récentes, vous pouvez aussi utiliser `LOAD DATA FROM MASTER` . C'est une commande pratique pour faire une copie de la base, l'envoyer à l'esclave, et ajutser le point de contrôle du log binaire, tout en une seule commande. Dans le future, `LOAD DATA FROM MASTER` sera la méthode recommandée pour configurer un esclave. Soyez prévenus, que le verrou de lecture posé par la commande sur le serveur peut rester en place un très long moment, si vous utilisez cette commande : elle n'est pas encore implémentée de manière efficace. Si vous avez de grandes tables, préférez donc la méthode qui utilise la sauvegarde via l'utilitaire `tar` après avoir exécuté la commande `FLUSH TABLES WITH READ LOCK` .

**Q** : Est ce que l'esclave doit être connecté en permanance au serveur?

**R** : Non, il n'est pas obligé. Vous pouvez éteindre l'esclave et le laisser déconnecter plusieurs heures ou jours, puis le reconnecter pour le voir récupérer les modifications et rattrapper le temps. Puis, se déconnecter à nouveau. De cette façon, vous pouvez, par exemple, configurer un esclave via une connexion modem, qui n'utilise que de brève période de connexions. L'implication de cela est qu'il n'est jamais garantit que l'esclave soit synchronisé avec le maître, à moins que vous ne preniez des mesures pour cela. Dans le futur, nous allons avoir l'option de bloquer le maître jusqu'à ce que au moins un des esclaves soit synchronisé.

**Q** : Comment puis−je forcer le maître à bloquer les modifications jusqu'à ce que l'esclave ait tout rattrappé?

**R** : Exécutez les commandes suivantes :

- Master: `FLUSH TABLES WITH READ LOCK`
- Master: `SHOW MASTER STATUS` – notez le nom du point de contrôle et son offset.
- Esclave : `SELECT MASTER_POS_WAIT('recorded_log_name', recorded_log_offset)` Lorsque la sélection est terminée, l'esclave est synchronisé avec le maître.
- Maître : `UNLOCK TABLES` – Le maître reprend ses opréations.

**Q** : Pourquoi est ce que je vois parfois plusieurs threads `Binlog_Dump` sur le maître, une fois que j'ai redémarré un esclave? **R** : `Binlog_Dump` est un processus continu, qui est géré par le serveur de cette manière :

- Rattrappe les modifications.
- Une fois qu'il n'y a plus de modifications, va dans `pthread_cond_wait()` , pour que nous puissions être prévenus d'une modificaiton ou d'un arrêt.
- Au moment de l'alerte, vérifie la raison. Si nous ne sommes pas supposés nous arrêter, contine la boucle `Binlog_dump` .
- Si il y a une erreur fatale, par exemple la détection d'un client mort, termine la boucle.

Si le thread de l'esclave s'arrête sur l'esclave, le thread correspondant `Binlog_Dump` sur le maître ne va pas le remarquer jusqu'à la prochaine modification sur le serveur (ou mort de processus), qui est nécessaire pour l'alerter, via `pthread_cond_wait()` . Dans le même temps, l'esclave peut avoir ouvert une autre connexion, qui a généré un autre thread `Binlog_Dump` .Le problème ci−dessus ne doit plus être présent depuis a version 3.23.26 et plus récent. En version 3.23.26, nous avons ajouté un identifiant `server−id` pour chaque serveur de réplication et désormais, les vieux zombies sont tués sur le maître, lorsqu'une nouvelle connexion de réplication se connecte sur le maître avec le même identifiant d'esclave.

**Q** : Comment faire tourner les logs?

**R** : En version 3.23.28, vous devez utiliser la commande `PURGE MASTER LOGS TO` après avoir déterminé quels logs peuvent être effacés, et optionnellement, avoir fait la sauvegarde des premiers. Dans les versions plus anciennes, le processus était bien plus douloureux, et ne pouvait être réalisé sans arrêter tous les esclaves, pour pouvoir réutiliser les noms des points de contrôle. Vous devez stopper les threads esclaves, éditer le fichier d'index de log à la main, effacer tous les vieux logs, redémarrer le serveur, redémarrer les esclaves, puis effacer les fichiers de logs.

**Q** : Comment puis−je passer à une configuration de réplication à chaud?

**R** : Si vous faîtes une mise à jour depuis les versions antérieures à la 3.23.26, vous devez simplement verrouiller les tables maîtres, laisser les esclaves rattrapper le temps, puis utiliser la commande `FLUSH MASTER` sur le master, et `FLUSH SLAVE` sur l'esclave pour redémarrer les logs, et redémarrer la nouvelle version du maître et de l'esclave. Notez que l'esclave peut être inactif pour un certain temps : comme le maître note toutes les modifications, l'esclave sera capable de rattrapper, dès qu'il peut se reconnecter.

Depuis la version 3.23.26, nous avons verrouillé le protocole de réplication pour les modifications, ce qui fait que vous pouvez modifier le maître et les esclaves à la volée vers une nouvelle version de la série des 3.23 et vous pouvez même avoir différentes versions de MySQL comme esclave et comme maître, tant qu'ils sont plus récents que la 3.23.26.

**Q** : Quels sont vos conseils concernant la réplication bi−bidirectionnelle?

**R** : La réplication MySQL ne supporte aucun protocole de verrouillage entre le maître et l'esclave

pour garantir l'atomicité d'une modification entre les serveurs. En d'autres termes, il est possible pour un client A de faire une modification sur le serveur 1 et que dans le même temps, avant que cela ne se soit propagé au serveur 2, un client B se connecte au serveur 2, et fasse une modification sur le serveur 2 qui ne débouchera pas sur le même état que celui dans lequel le serveur 1 est. C'est ainsi qu'il ne faut pas lier de cette façon deux serveurs, à moins que les modifications ne puisse se faire dans n'importe quel ordre, ou que vous sachiez prendre en charge des modifications anarchiques.

Vous devez aussi réaliser que la réplication bi−directionnelle n'améliore pas beaucoup les performances, tout au moins au niveau des modifications. Les deux serveurs doivent faire la même quantité de modifications, ainsi qu'un serveur seul le ferait. La seule différence est qu'il va y avoir moins de verrous, car les modifications qui proviennent d'un autre serveur seront optimisé par l'esclave. Cet avantage peut aussi être annulé par les délais réseau.

**Q** : Comment puis−je utiliser la réplication pour améliorer les performances de mon système ?

**R** : Vous devez configurer un serveur en maître et y diriger toutes les écritures, puis configurer les autres en esclaves dans la limite de vos moyens, et y distribuer les lectures. Vous pouvez aussi démarrer les esclaves en mode `--skip-bdb` , `--low-priority-updates` et `--delay-key-write=ALL` pour accélérer les esclaves. Dans ce cas, l'esclave va utiliser les tables non transactionnelles `MyISAM` au lieu des tables `BDB` pour obtenir plus de vitesse.

**Q** : Que dois−je faire pour préparer mon code client à la réplication?

**R** : Si la partie de votre code qui réalise les accès aux bases de données a été proprement modularisée, la convertir en une configuration qui supporte la réplication ne sera pas un problème : modifiez simplement votre base pour qu'elle aille lire sur les esclaves et le maître, mais ne fasse que des modifications avec le maître. Si votre code n'a pas ce niveau d'abstraction, l'installation du système de réplication vous donnera alors la motivation ou la raison pour le faire. Vous devriez commencer par créer une couche d'abstraction ou un module avec les fonctions suivantes :

- `safe_writer_connect()`
- `safe_reader_connect()`
- `safe_reader_query()`
- `safe_writer_query()`

`safe_` signifie que la fonction devra prendre en charge toutes les conditions d'erreurs.Vous devriez alors convertir votre code client pour qu'il utilise cette librairie. Cela peut être un processus laborieux et déroutant, mais il va s'avérer payant dans le long terme. Toutes les applications qui suivent la technique ci−dessus pourront alors prendre avantage des solutions de réplication. Le code sera aussi bien plus facilement entretenu, et ajouter des options sera trivial. Vous devrez modifier une ou deux fonctions, comme par exemple pour enregistrer le temps de calcul de certaines requêtes, ou les requêtes qui vous retournent des erreurs. Si vous avez écrit beaucoup de code jusqu'ici, vous pourriez vouloir automatiser la conversion en utilisant l'utilitaire de Monty, `replace` , qui est distribué avec la distribution standard de MySQL, ou bien simplement en écrivant un script Perl. Avec un peu de chance, votre code suit des conventions connues. Si ce n'est pas le cas, alors vous serez peut être conduit à réécrire votre application de toutes manières, ou bien, à lui appliquer des méthodes à la main.

Notez que, bien sur, vous pouvez utiliser différents nom pour les fonctions. Ce qui est important, c'est que vous ayez des interfaces unifiées pour vous connecter en lecture ou en écriture, et pour exécuter des lectures et écritures.

**Q** : Quand et combien de réplications de MySQL permettront d'améliorer les performances de mon système?

**R** : La réplication MySQL est particulièrement avantageuse pour les systèmes qui gèrent des lectures fréquentes, et des écritures plus rares. En théorie, en utilisant uniquement un maître et beaucoup d'esclaves, vous pouvez augmenter les performances de votre système jusqu'à saturation de la bande passante ou du maître, pour les modifications.

Afin de déterminer le nombre d'esclaves que vous pouvez obtenir voir les performances de votre système s'améliorer, vous devez bien connaître les types de requêtes que vous utilisez, et empiriquement déterminer la relation entre le nombre de lectures et d'écritures (par secondes, ou maximum absolu), pour un maître et un esclave. L'exemple ci−dessous va vous montrer comment faire des calculs simples.

Imaginons que votre charge système soit constituée de 10% d'écriture et de 90% de lectures. Nous avons aussi déterminé que le maximum de lectures `max_reads` = 1200 − 2 * `max_writes` , ou, en d'autres mots, notre système peut voir des pics de 1200 lectures par secondes sans aucune écritures, notre temps d'écriture moyen est deux fois plus temps qu'une lecture, et la relation est linéaire. Supposons que notre maître et notre esclave sont de la même capacité, et que nous avons N esclaves et un maître. Nous avons alors pour chaque serveur (maître ou esclave) :

`lectures = 1200 − 2 * écriture` (issue des tests)

`lectures = 9* écriture / (N + 1)` (lectures réparties, mais toutes les écritures vont à tous les serveurs)

```
9*écriture/(N+1) + 2 * écriture = 1200
```

```
écriture = 1200/(2 + 9/(N+1)
```

Si N = 0, ce qui signifie que nous n'avons pas de réplication, notre système peut gérer 1200/11, environs 109 écritures par secondes, ce qui signifie (que nous aurons 9 fois plus de lectures que d'écritures, étant donné la nature de notre application).

Si N = 1, nous pouvons monter à 184 écriture par seconde.

Si N = 8, nous pouvons monter à 400 écriture par seconde.

Si N = 17, nous pouvons monter à 480 écriture par seconde.

Eventuellement, si N se rapproche de l'infini (et notre budget de l'infini négatif), nous pourrons nous rapprocher de 600 écritures par secondes, en améliorant le système 5,5 fois. Toutefois, avec 8 serveurs, nous avons pu améliorer le système de 4 fois.

Notez que nos calculs ont supposés une bande passante infinie, et que nous avons négligé des facteurs qui pourraient être significatifs pour notre système. Dans de nombreux cas, nous ne pourrions pas faire de calculs précis pour prédire l'état de notre système avec N esclaves de réplication. Toutefois, répondre aux questions ci−dessus vous permettra de décider si la réplication est une solution à votre problème ou pas.

- Quel est le ratio d'écriture/lecture de votre système?
- Quelle est la charge maximale d'un serveur en écriture, si vous pouvez limiter les lectures?
- Combien d'esclaves votre réseau peut supporter?

**Q** : Comment puis−je utiliser la réplication pour fournir un système à haute tolérance de panne?

**R** : Avec les fonctionnalités actuellement disponible, vous devez configurer un serveur et un esclave (ou plusieurs esclaves), et écrire un script qui va surveiller le maître pour voir si il fonctionne , et instruire votre applcation et les esclaves d'un changement de maître en cas d'échec. Voici des

suggestions :

- Utilisez la commande `CHANGE MASTER TO` pour changer un esclave en maître.
- Un bon moyen de garder votre application informé du maître courant est d'utiliser les DNS dynamiques, vous pouvez attribuer au maître. Avec `bind` , vous pouvez utiliser `nsupdate` pour modifier dynamiquement votre DNS.
- Vous devez faire fonctionner vos esclaves avec l'option `log-bin` et sans l'option `log-slave-updates` . De cette façon, l'esclave sera prêt à prendre le relais dès que vous lui enverrez la commande `STOP SLAVE` ; envoyez `RESET MASTER` et `CHANGE MASTER TO` aux autres esclaves. Cela va aussi vous aider à intercepter des modifiations contagieuses qui pourraient intervenir dûes à des mauvaise configurations d'esclaves (idéalement, vous allez donner des droits d'accès pour que personne ne puisse modifier l'esclave, hormis le thread esclave), combinées avec des bogues dans vos programmes clients (ils ne doivent jamais mettre à jour un esclave directement).

Nous travaillons actuellement à l'intégration automatique de l'élection d'un nouveau maître, mais jusqu'à ce que ce soit près, vous devez créer votre propre outil de surveillance.

## 4.10.8 Correction de problèmes courants

Si vous avez suivi les instructions, et que votre configuration de réplication ne fonctionne pas, commencez par supprimer les problèmes liés à l'utilisateur comme ceci :

- Est ce que le maître enregistre dans le log binaire ? Vérifiez avec la commande `SHOW MASTER STATUS` . Si il le fait, la variable `Position` doit être non nulle. Si ce n'est pas le cas, vérifiez que vous avez donné au serveur l'option `log-bin` et que vous lui avez donné un `server-id` .
- Est ce que l'esclave fonctionne? Vérifiez le avec `SHOW SLAVE STATUS` . La réponse se trouve dans la colonne `Slave_running` . Si ce n'est pas le cas, vérifiez les options de l'esclave, et vérifiez le fichier de log d'erreurs.
- Si l'esclave fonctionne, as-t-il établit une connexion avec le maître? Exécutez la commande `SHOW PROCESSLIST` , et recherchez un utilisateur avec la valeur `system user` dans la colonne `User` et `none` dans la colonne `Host` , et vérifiez la colonne `State` . Si elle indique `connecting to master` , vérifiez les droits de connexion pour l'utilisateur de réplication sur le serveur, ainsi que le nom de l'hôte, votre configuration DNS, le fonctionnement du maître, et si tout est OK, vérifiez le fichier de log d'erreurs.
- Si l'esclave fonctionnait, mais s'est arrêté, vérifiez le résultat de la commande SHOW SLAVE STATUS, et vérifiez le fichier de log d'erreurs. Il arrive que certaines requêtes réussissent sur le maître mais échouent sur l'esclave. Cela ne devrait pas arriver si vous avez pris la bonne sauvegarde du maître, et que vous n'avez jamais modifié les données sur le serveur esclave, autrement que par le truchement de l'esclave de réplication. Si c'est le cas, c'est un bogue, et vous devez le rapporter. Voyez plus loin pour savoir comment rapporter un bogue.
- Si une requête qui a réussit sur le maître, refuse de s'exécuter sur l'esclave, et qu'une synchronisation complète de la base ne semble pas possible, essayez ceci :
    - Commencez par voir si il n'y a pas de lignes dans le chemin. Essayez de comprendre comment a plus se trouver la, effacez la, et essayer de redémarrer l'esclave avec `SLAVE START`
    - Si la solution ci-dessus ne fonctionne pas ou ne s'applique pas, essayez de comprendre si c'est risqué de faire une correction à la main (au besoin) puis, ignorez la prochaine requête du maître.
    - Si vous avez décidé que vous pouvez vous passer de la prochaine requête, exécutez les commandes `SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1; SLAVE START;` pour ignorer toutes les requêtes qui n'utilisent pas `AUTO_INCREMENT` ou `LAST_INSERT_ID()` , ou `SET GLOBAL`

`SQL_SLAVE_SKIP_COUNTER=2; SLAVE START;` . La raison pour laquelle des requêtes qui utilisent `AUTO_INCREMENT` or `LAST_INSERT_ID()` sont différentes, est que elles représentent deux événements dans le log binaire du maître.

    - Si vous êtes sûr que l'esclave a été démarré en parfaite synchronisation avec le maître, et que personne n'a fait de modification dans les tables impliquées, en dehors de l'esclave, lisez les rapport de bogues, pour vous éviter d'essayer tous les trucs ci−dessus.
- Assurez vous que vous ne rencontrez pas un des vieux bogues que vous pourriez supprimer en changeant de versions.
- Si tout cela échoue, lisez les logs d'erreurs. Si ils sont gros, utilisez la commande `grep −i slave /path/to/your−log.err` sur l'esclave. Il n'y a pas de schéma de recherche particulier a appliquer au maître, car les seules erreurs qu'il stocke sont des erreurs générales : si il le peut, il envoie l'erreur à l'esclave.

Lorsque vous avez épuisé toutes les possibilités, et qu'il n'y a pas d'erreur utilisateur impliquée, et que la réplication ne fonctionne plus du tout ou qu'elle est très instable, il est temps de préparer un rapport de bogues. Essayer de passer un peu de temps sur ce rapport pour en faire un bon. Idéalement, vous souhaitons recevoir un cas de test au format présenté dans le dossier `mysql−test/t/rpl*` , du dossier source. Si vous soumettez un test comme cela, vous pouvez vous attendre à recevoir un patch dans un ou deux journées, même si la durée de réaction peut varier en fonction de nombreux facteurs. La deuxième meilleure option est simplement d'écrire un programme qui va facilement configurer les arguments de connexion de votre maître et esclave, et qui va illustrer le problème sur nos systèmes. Vous pouvez l'exécuter en C ou en Perl, suivant votre langage de programmation.

Si vous pouvez utiliser l'une des méthodes ci−dessus pour démontrer votre bogue, utilisez le script `mysqlbug` pour préparer le rapport de bogue, et envoyez le à bugs@lists.mysql.com . Si vous avez un problème fantôme (un bogue qui survient, mais que vous ne pouvez pas reproduire à coup sûr) :

- Vérifiez qu'il y n'y a pas d'autre utilisateur impliqué. Par exemple, si vous modifiez l'esclave en dehors du thread l'esclave, les données seront désynchronisées, et vous pourriez aboutir à des violations de contraintes, auquel cas, vous devez arrêter l'esclave et nettoyer manuellement les tables pour les synchroniser.
- Exécutez l'esclave avec les options `log−slave−updates` et `log−bin` −− cela va conserver un log de toutes les modifications dans le cadre de l'esclave.
- Sauvez toutes les preuves avant d'éteindre la réplication. Si nous n'avons que des informations schématiques, cela nous prendra beaucoup de temps pour étudier le problème. Les preuves que vous devriez obtenir sont :
    - Le fichier de log binaire du maître
    - Tous les fichiers de logs binaires sur l'esclave
    - Le résultat de la commande `SHOW MASTER STATUS` sur le maître, au moment de la découverte du problème.
    - Le résultat de la commande `SHOW SLAVE STATUS` sur le maître, au moment de la découverte du problème.
    - Les fichiers de log d'erreurs du maître et de l'esclave.
- Utilisez `mysqlbinlog` pour examiner les logs binaires. Cette instructions doit être pratique pour rechercher des requêtes problématiques :

```
mysqlbinlog −j pos_from_slave_status /path/to/log_from_slave_status | head
```

Une fois que vous avez rassemblé les preuves du problème fantôme, essayez au maximum de l'isoler dans un cas de test. Puis, rapporter le problème à bugs@lists.mysql.com avec autant d'information possibles.

# 5 Optimisation de MySQL

L'optimisation est une tâche complexe car elle nécessite une parfaite compréhension du système en entier. Alors qu'il serait possible de faire quelques optimisations localement avec une faible connaissance de votre système ou de votre application, plus vous voulez un système optimal, plus il est nécessaire de le connaître.

Ce chapitre va tenter d'expliquer et de donner des exemples de différentes manières d'optimiser MySQL. Souvenez−vous, malgré tout, qu'il existe toujours d'autres moyens (de plus en plus difficiles) de rendre le système plus véloce.

## 5.1 Vue d'ensemble de l'optimisation

Le plus important pour obtenir un système rapide est bien sûr le schéma utilisé. Il faut également savoir le genre de choses que fera le système et où seront les goulots d'étranglement.Les ralentissements les plus courants sont :

- Les recherches sur les disques. Il faut du temps pour que le disque trouve un morceau de donnée. Avec les disques modernes en 1999, le temps d'accès est généralement inférieur à 10ms, donc on peut en théorie faire près de 100 recherches par seconde. Le temps s'amliore un peu avec les nouveaux disques et il est difficile de l'optimiser pour une seule table. La méthode pour l'optimiser est de disséminer les données sur plus d'un disque.
- Les accès disques en lecture/écriture. Une fois le disque à la bonne position, il faut lire les données. Avec des disques modernes en 1999, un disque fournit environ 10 à 20Mo/s. Ceci est plus facile à optimiser que les recherches car on peut lire en parallèle sur plusieurs disques.
- Les cycles CPU. Quand les données sont dans la mémoire centrale (ou si elles y étaient déjà) il faut les traiter pour obtenir le résultat. Avoir des tables de petite taille par rapport à la mémoire est le facteur limitant le plus courant. Mais avec des petites tables, la célérité n'est souvent pas un souci.
- La bande passante de la mémoire. Quand le CPU a besoin de plus d'informations qu'il ne peut en stocker dans son cache, la bande passante de la mémoire centrale devient le goulot d'étranglement. C'est un étranglement peu courant pour la majorité des systèmes, mais il faut être averti.

### 5.1.1 Limitations et inconvénients des choix conceptuels de MySQL

Avec les tables de type MyISAM, MySQL utilise un verrouillage extrêmement rapide (plusieurs lectures / une seule écriture). Le plus gros problème avec ce type de table survient quand vous avez un mélange de flux de modifications et des sélections lentes sur la même table. Si c'est une problème sur plusieurs tables, vous pouvez utiliser un autre type de table pour celles ci. Types de tables MySQL .

MySQL peut utiliser à la fois des tables transactionnelles et des tables non−transactionnelle. Pour pouvoir travailler tranquillement avec des tables non−transactionnelles (qui n'ont pas la possibilité de revenir en arrière si quelque chose se passe mal) MySQL suit les règles suivantes:

- Toutes les colonnes ont une valeur par défaut.
- Si vous insérez une mauvaise valeur dans une colonne (par exemple `NULL` dans une colonne `NOT NULL` , ou encore une valeur numérique trop grande dans une colonne numérique), MySQL prendra en compte "la meilleure valeur possible" plutôt que de sortir une erreur. Pour les

valeurs numériques, il s'agit de 0, de la valeur la plus petite possible, ou de la valeur la plus grande possible. Pour les chaînes, il s'agit soit d'une chaîne vide, soit de la chaîne la plus longue que peut contenir la colonne.
- Toutes les expressions calculées retournent une valeur qui peut être utilisées à la place d'afficher un message d'erreur. Par exemple, 1/0 retourne `NULL`

La raison des règles précitées est que nous ne pouvons pas vérifier les conditions avant que la requête ne commence à être exécutée. Si nous rencontrons un problème après avoir mis à jour plusieurs lignes, nous ne pouvons plus revenir en arrière puisque la table ne le supporte pas. Nous ne pouvons pas nous arrêter car le travail de mise à jour ne serait fait qu'à moitié, ce qui est sans doute le pire qui puisse arriver. Dans ce cas, il vaut mieux faire du mieux qu'on peut et continuer comme si rien ne s'était passé.Ce qui précède signifie qu'il ne faut pas que le contrôle du contenu des champs soit fait au niveau de MySQL, mais au niveau de l'application.

## 5.1.2 Portabilité

Comme tous les serveurs SQL implémentent différemment le langage SQL, cela prend de solides connaissances pour écrire des applications SQL portables. Pour les insertions et sélections simples, c'est très simple, mais plus vos besoins se complexifient, plus c'est abscons. Si vous voulez une application qui fonctionne rapidement sur de nombreuses bases de données, c'est même encore plus difficile.

Pour rendre une application complexe portable, vous pouvez commencer par choisir une panoplie de serveurs SQL avec lesquels travailler.

Vous pouvez utiliser le programme/page web de MySQL appelé `crash−me` http://www.mysql.com/information/crash−me.php pour trouver les fonctions, types et limites que vous pouvez utiliser avec un panel de serveurs de bases de données. Les tests de `crash−me` ne vérifient pas tout, mais il est déjà très exhaustif avec plus de 450 points de tests.

Par exemple, vous ne devriez pas avoir de nom de colonne supérieur à 18 caractères, si vous voulez pouvoir utiliser Informix ou DB2.

Les programmes de tests `crash−me` et de performances de MySQL sont très indépendants du serveur. En regardant comment nous avons géré ces situations, vous pouvez comprendre comment rendre votre propre code indépendant du serveur. Les tests de performances sont situés dans le dossier `sql−bench` de la distribution source de MySQL. Ils sont écrits en Perl avec l'interface DBI, ce qui résout les problèmes de connexion.

Voyez http://www.mysql.com/information/benchmarks.html pour connaître les résultats de ces benchmarks.

Comme vous pouvez le voir avec ces résultats, toutes les bases de données ont leur point faible. En réalité, elles ont toutes une approche différente du même problème, et cela conduit à des comportements spécifiques.

Si vous avez besoin de l'indépendance au serveurs de bases de données, vous devez bien connaître les faiblesses de chaque serveur. MySQL est très rapide pour lire et modifier les données, mais peine lorsque les lectures et écritures sont lentes sur la même table. Oracle, d'un autre coté, a de gros problèmes lorsque vous essayez d'accéder aux données que vous avez modifié récemment (jusqu'à ce qu'elles soient écrites sur le disque). Les bases de données transactionnelles en général ne sont pas très douées pour générer des tables résumés à partir des tables de log, car dans ce cas, le verrouillage de ligne est inutile.

Pour rendre votre application **reellement** indépendante de la base de données, vous devez définir un classe très souple à travers laquelle vous allez vous interfacer pour manipuler vos données. Comme le langage C++ est disponible sur la plupart des systèmes, cela rend les classes C++ très pratiques pour cette tâche.

Si vous utilisez une fonctionnalité spécifique d'une base de données (comme la commande REPLACE de MySQL), il vous faut aussi coder la même commande pour les autres serveurs (qui sera alors plus lente). Avec MySQL, vous pouvez aussi utiliser la syntaxe /*! */ pour utiliser des mots clés spécifiques de MySQL dans une requête. Le code entre /* */ sera alors traité comme un commentaire et ignoré par la plupart des autres serveurs SQL.

Si les hautes performances sont plus importantes que l'exactitude, comme pour les applications web, il est possible de créer une couche application qui met en cache les résultats et vous donne de meilleures performances. En laissant les anciens résultats se périmer, vous pouvez garder un cache à jour. Cela vous donne une méthode pour gérer les grandes charges, durant lesquelles vous pouvez augmenter la taille du cache, et augmenter la durée de vie.

Dans ce cas, les informations de création de tables doivent contenir les informations de taille initiale du cache, et la fréquence de rafraîchissement des tables.

### 5.1.3 Pour quoi avons nous utilisé MySQL ?

Pendant le développement initial de MySQL, les fonctions de MySQL ont été créées pour convenir à un maximum de clients. Celles ci supporte des entrepôts de données pour deux des plus gros revendeurs suédois.

Nous recevons chaque semaine le résumé de toutes les transactions par carte de toutes les boutiques, et nous sommes chargés de fournir des informations utiles aux gérants des boutiques pour les aider à comprendre comment leurs propres campagnes publicitaires touchent leurs clients.

Les données sont assez énormes (près de 7 millions de résumés de transactions par mois), et nous avec les données de 4−10 ans que nous présentons aux utilisateurs. Nous avons chaque semaine des requêtes des clients qui veulent un accès 'instantané' aux nouveaux rapports sur ces données.

Nous avons réussi en stockant toutes les informations dans des tables de 'transactions' compressées. Nous avons une série de macros (scripts) qui génère des tables de résumés groupés par différents critères (groupe de produits, identifiant de client, boutique ...). ces rapports sont des pages web générées dynamiquement par un petit script Perl qui parcours une page web, exécute les requêtes SQL, et insère les résultats. Nous aurions bien utilisé PHP ou mod_perl à la place, mais ils n'étaient pas disponibles à cette époque.

Nous avons écrit un outil en C pour la représentation graphique des données qui génère des GIFs à partir du résultat de requêtes SQL (avec quelques traitements sur le résultat). Ceci est également effectué dynamiquement par le script Perl qui parcourt les fichiers HTML .

Pour la plupart des cas, un nouveau rapport peut simplement être fait en copiant un script existant, et en modifiant la requête SQL qu'il exécute. Dans certains cas, nous aurons besoin d'ajouter des champs à une table de résumé existante ou d'en générer une nouvelle, mais c'est tout de même toujours assez simple, car nous gardons toutes les tables de transactions sur disque. (Actuellement, nous avons au moins 50 Go de tables de transactions et 200 Go d'autres données sur les clients.)

Nous donnons également accès aux tables de résumés à nos clients directement avec ODBC, de

sorte que les utilisateurs avancés puissent traiter les données eux−mêmes .

Nous n'avons eu aucun problème à supporter tout cela avec une relativement modeste Sun Ultra SPARCStation (2x200 MHz). Nous avons récemment amélioré l'un de nos serveurs en un bi−CPU 400 MHz UltraSPARC, et nous projetons actuellement de supporter les transactions au niveau du produit, ce qui signifie un décuplement des données. Nous pensons pouvoir y arriver uniquement en ajoutant des disques supplémentaires à nos systèmes.

Nous expérimentons aussi Intel−Linux, pour pouvoir avoir plus de puissance CPU pour moins cher. Comme nous utilisons désormais le format binaire portable pour les bases de données (nouveauté de la version 3.23), nous utiliserons cela pour quelques parties de l'application.

Nous avons au départ le sentiment que Linux s'acquitera mieux des faibles et moyennes charges tandis que Solaris fonctionnera mieux sur les grosses charges à cause des I/O disques extrêmes, mais nous n'avons actuellement aucune conclusion à ce propos. Après quelques discussion avec un développeur du noyau Linux, un effet de bord de Linux pourrait tant de ressources aux travaux de traitement que les performances de l'interface interactive peut devenir vraiment lente. Cela fait apparaître la machine très lente et sans réponse lorsque de gros traitements sont en cours. Heureusement, cela sera mieux géré dans les futurs noyaux de Linux.

## 5.1.4 La suite de tests MySQL

Ceci devrait comprendre une description technique de la suite de tests de performances de MySQL (et `crash−me` ), mais cette description n'est pas encore écrite. Actuellement, vous pouvez vous faire une idée des tests en regardant le code et les résultats dans le répertoire `sql−bench` dans toutes les distributions de sources de MySQL.

Cette suite de test est censée permettre à utilisateur de comparer ce qu'une implémentation SQL donnée réussi bien ou mal.

Sachez que ces tests de performances lancent en un seul thread, donc il mesure le temps minimum pour chaque opération. Nous projetons pour le futur d'ajouter de nombreux tests multi−thread à cette suite de tests.

Par exemple, (tous ont été lancés sur une même machine NT 4.0)

| **Lecture de 2000000 lignes indexées** | **Secondes** | **Secondes** |
|---|---|---|
| mysql | 367 | 249 |
| mysql_odbc | 464 | |
| db2_odbc | 1206 | |
| informix_odbc | 121126 | |
| ms−sql_odbc | 1634 | |
| oracle_odbc | 20800 | |
| solid_odbc | 877 | |
| sybase_odbc | 17614 | |

| **Insertion de lignes (350768** | **Secondes** | **Secondes** |
|---|---|---|
| mysql | 381 | 206 |
| mysql_odbc | 619 | |
| db2_odbc | 3460 | |
| informix_odbc | 2692 | |
| ms−sql_odbc | 4012 | |
| oracle_odbc | 11291 | |
| solid_odbc | 1801 | |
| sybase_odbc | 4802 | |

Le test ci−dessus a été exécuté avec un index de cache de 8Mo.

Nous avons rassemblé d'autres résultats de tests à http://www.mysql.com/information/benchmarks.html .

Notez que Oracle n'est pas inclus dans ces tests car ils ont demandé à être retirés. Tous les tests d'Oracle doivent être faits par Oracle! Nous croyons qu cette politique va biaiser **fortement** les tests en faveur de Oracle, car les tests ci−dessus sont supposé montrer ce qu'une installation simple peut faire pour un client simple.

Pour exécuter la suite de tests, vous devez télécharger la distribution source de MySQL, installer les pilotes perl DBI et perl DBD pour les bases qui vous intéressent.

```
 cd sql−bench
perl run−all−tests −−server=#
```

où # est un des serveurs supportés. Vous pouvez obtenir une liste de toutes les options et des serveurs supportés avec l'option de ligne de commande `run−all−tests −−help` .

`crash−me` essaie de déterminer quelles fonctionnalités un serveur supporte, et quelles sont ses limitations. Par exemple, le test détermine :

- Les types de colonnes supportés
- Le nombre d'index supportés
- Les fonctions supportées
- La taille maximale d'une requête
- La taille maximale d'une colonne `VARCHAR`

Vous pouvez retrouver les résultats de `crash−me` sur de nombreuses bases de données à http://www.mysql.com/information/crash−me.php .

## 5.1.5 Utiliser vos propres tests de performance

Vous devriez vraiment penser à préparer des tests de performances pour votre application et base, afin d'identifier les opérations les plus lentes. En les corrigeant (ou en remplaçant ces opérations des 'modules simples') vous pouvez facilement identifier les autres opérations lentes (et ainsi de suite...). Même si la performance générale de votre application est suffisante, vous devriez prévoir où seront les prochains freins, et décider d'anticiper leur résolution, avant que vous n'ayez vraiment besoin de ces performances.

Pour avoir un exemple de programme de tests portables, voyez la suite de tests MySQL. La suite de tests comparatifs de MySQL . Vous pouvez prendre n'importe quel programme de cette suite, le modifier pour l'adapter à vos besoins, et essayer différentes solutions à votre problème : il suffit de tester et d'identifier la solution la plus rapide pour vous.

Il est très fréquent que des problèmes surviennent lorsque le système subit une forte charge. Nous avons de nombreux clients qui nous contactent lorsqu'ils ont mis leur système en production, et rencontré des problèmes de charge. Pour chacun d'entre eux, les problèmes étaient des problèmes simples de conceptions (les scans de tables ne sont **pas bons** sous forte charge) ou des problèmes liés au système d'exploitation ou les librairies. La plupart auraient été vraiment **plus simples** à tester si le système n'était pas déjà en production.

Pour éviter des problèmes comme ceux−là, vous devriez mettre quelques efforts dans les tests de votre application dans son ensemble, avant de la mettre dans les pires conditions. Vous pouvez utiliser le programme Super Smack pour cela, qui est disponible à http://www.mysql.com/Downloads/super−smack/super−smack−1.0.tar.gz . Comme son nom le suggère, il va mettre votre système à genoux si vous lui demandez, alors assurez vous de ne l'utiliser qu'avec votre système de développement.

# 5.2 Optimisation des SELECT s et autres requêtes

Premièrement, ce qui affecte toutes les requêtes : plus votre système de droits est compliqué, plus vous aurez des baisses de performances.

Si vous n'avez aucun `GRANT` effectué, MySQL optimisera les vérifications de droits. Donc, si vous avez un système volumineux, il serait bénéfique d'éviter les grants. Sinon les performances seront réduites.

Si votre problème est du à des fonctions natives de MySQL, vous pouvez toujours effectuer les tests suivants avec le client MySQL :

```
 mysql> SELECT BENCHMARK(1000000,1+1);
+------------------------+
| BENCHMARK(1000000,1+1) |
+------------------------+
|                      0 |
+------------------------+
1 row in set (0.32 sec)
```

Ce qui précède montre que MySQL peut exécuter 1 000 000 d'additions en 0.32 secondes sur un `PentiumII 400MHz` .

Toutes les fonctions MySQL sont sensé être optimisées, mais il peut y avoir quelques exceptions et la fonction `BENCHMARK(nombre_de_fois,expression)` est un très bon moyen de trouver ce qui cloche dans vos requêtes.

## 5.2.1 Syntaxe de EXPLAIN (Obtenir des informations sur les SELECT )

```
     EXPLAIN nom_de_table
ou  EXPLAIN SELECT select_options
```

`EXPLAIN nom_de_table` est un synonyme de `DESCRIBE nom_de_table` ou `SHOW COLUMNS FROM nom_de_table` .Lorsque vous ajoutez au début d'une commande `SELECT` le mot clé `EXPLAIN` , MySQL va expliquer le processus qu'il va suivre pour exécuter la commande `SELECT` , et donner des détails sur la façon avec laquelle il va joindre les tables, et dans quel ordre.

Avec l'aide de `EXPLAIN` , vous pouvez identifier les index à ajouter pour accélérer les commandes `SELECT` .

Vous devriez souvent utiliser la commande `ANALYZE TABLE` pour mettre à jour les statistiques de cardinalité de vos tables, qui affectent les choix de l'optimisateur. Syntaxe de `ANALYZE TABLE` .

Vous pouvez aussi voir si l'optimisateur fait les jointures dans un ordre vraiment optimal. Pour forcer l'optimisateur à utiliser un ordre spécifique de jointure dans une commande `SELECT` , ajoutez l'attribut `STRAIGHT_JOIN` à la clause.

Pour les jointures complexes, `EXPLAIN` retourne une ligne d'information pour chaque table utilisée dans la commande `SELECT` . Les tables sont listées dans l'ordre dans lequel elles seront lues. MySQL résout toutes les jointures avec une seule passe multi−jointure. Cela signifie que MySQL lit une ligne dans la première table, puis recherche les lignes qui correspondent dans la seconde, puis dans la troisième, etc. Lorsque toutes les tables ont été traitées, MySQL affiche les colonnes demandées, et il remonte dans les tables jusqu'à la dernière qui avait encore des lignes à traiter. La prochaine ligne est alors traitée de la même façon.

Avec MySQL version 4.1 l'affichage de `EXPLAIN` a été modifié pour mieux fonctionner avec les structures comme `UNION` , sous−requêtes, et tables dérivées. La plus importante évolution est l'addition de deux nouvelles colonnes : `id` et `select_type` .

Le résultat de la commande `EXPLAIN` est constitué des colonnes suivantes :

*id*
: identifiant de `SELECT` , le numéro séquentiel de cette commande `SELECT` dans la requête.

*select_type*
: Type de clause `SELECT` , qui peut être :

  *SIMPLE*
  : Simple `SELECT` (sans `UNION` s ou sous−requêtes).

  *PRIMARY*
  : `SELECT` extérieur.

  *UNION*
  : Second et autres `UNION SELECT` s.

  *DEPENDENT UNION*
  : Second et autres `UNION SELECTS` s, dépend de la commande extérieure.

  *SUBSELECT*
  : Premier `SELECT` de la sous−requête.

  *DEPENDENT SUBSELECT*
  : Premier `SELECT` , dépendant de la requête extérieure.

  *DERIVED*
  : Table dérivée `SELECT` .

*table*
: La table à laquelle la ligne fait référence.

*type*
: Le type de jointure. Les différents types de jointures sont les suivants, dans l'ordre du plus efficace au plus lent :

  *system*
  : La table a une seule ligne (c'est une table système). C'est un cas spécial du type de jointure `const` .

  *const*
  : La table a au plus une ligne correspondante, qui sera lue dès le début de la requête. Comme il n'y a qu'une seule ligne, les valeurs des colonnes de cette ligne peuvent être considérées comme des constantes pour le reste de l'optimisateur. Les tables `const` sont très rapides, car elles ne sont lues qu'une fois.

  *eq_ref*
  : Une ligne de cette table sera lue pour chaque combinaison de ligne des tables précédentes. C'est le meilleur type de jointure possible, à l'exception des précédents. Il est utilisé lorsque toutes les parties d'un index sont utilisées par la jointure, et que l'index est `UNIQUE` ou `PRIMARY KEY` .

  *ref*
  : Toutes les lignes avec des valeurs d'index correspondantes seront lues dans cette table, pour chaque combinaison des lignes précédentes. `ref` est utilisé si la jointure n'utilise que le préfixe de gauche de la clé, ou si la clé n'est pas `UNIQUE` ou `PRIMARY KEY` (en d'autres termes, si la jointure ne peut pas sélectionner qu'une seule ligne en fonction de la clé). Si la clé qui est utilisée n'identifie que quelques lignes à chaque fois, la jointure est bonne.

  *range*
  : Seule les lignes qui sont dans un intervalle donné seront lues, en utilisant l'index pour sélectionner les lignes. La colonne `key` indique

quel est l'index utilisé. `key_len` contient la taille de la partie de la clé qui est utilisée. La colonne `ref` contiendra la valeur `NULL` pour ce type.

*index*

C'est la même chose que `ALL` , sauf que seul l'arbre d'index sera lu et scanné. C'est généralement plus rapide que `ALL` , car le fichier d'index est généralement plus petit que le fichier de données.

*ALL*

Une analyse complète de la table sera faîte pour chaque combinaison de lignes issue des premières tables. Ce n'est pas bon si la première table n'est pas une jointure de type `const` et c'est **très** mauvais dans les autres cas. Normalement vous pouvez éviter ces situations de `ALL` en ajoutant des index basée sur des parties de colonnes.

*possible_keys*

La colonne `possible_keys` indique quels index MySQL va pouvoir utiliser pour trouver les lignes dans cette table. Notez que cette colonne est totalement dépendante de l'ordre des tables. Cela signifie que certaines clés de la colonne `possible_keys` pourraient ne pas être utilisées dans d'autres cas d'ordre de tables.

Si cette colonne est vide, il n'y a pas d'index pertinent. Dans ce cas, vous pourrez améliorer les performances en examinant votre clause `WHERE` pour voir si des colonnes sont susceptibles d'être indexée. Si c'est le cas, créez un index ad hoc, et examinez le résultat avec la commande `EXPLAIN` . Syntaxe de `ALTER TABLE` .

Pour connaître tous les index d'une table, utilisez le code `SHOW INDEX FROM nom_de_table` .

*key*

La colonne `key` indique l'index que MySQL va décider d'utiliser. Si la clé vaut `NULL` , aucun index n'a été choisi. Pour forcer MySQL à utiliser un index listé dans la colonne `possible_keys` , utilisez `USE KEY/IGNORE KEY` dans votre requête. Syntaxe des `SELECT` .De plus, exécuter `myisamchk --analyze` ( Syntaxe d'invocation de `myisamchk` ) ou `ANALYZE TABLE` ( Syntaxe de `ANALYZE TABLE` ) sur la table va aider l'optimiseur à choisir les index.

*key_len*

La colonne `key_len` indique la taille de la clé que MySQL a décidé d'utiliser. La taille est `NULL` si la colonne `key` vaut `NULL` . Notez que cela vous indique combien de partie d'une clé multiple MySQL va réellement utiliser.

*ref*

La colonne `ref` indique quelle colonne ou quelles constantes sont utilisées avec la clé `key` , pour sélectionner les lignes de la table.

*rows*

La colonne `rows` indique le nombre de ligne que MySQL estime devoir examiner pour exécuter la requête.

*Extra*

Cette colonne contient des informations additionnelle sur comment MySQL va résoudre la requête. Voici une explication des différentes chaînes que vous pourriez trouver dans cette colonne :

*Distinct*

MySQL ne va pas continuer à chercher d'autres lignes que la ligne courante, après en avoir trouvé une.

*Not exists*

MySQL a été capable d'appliquer une optimisation de type `LEFT JOIN` sur la requête, et ne va pas examiner d'autres lignes de cette table pour la combinaison de lignes précédentes, une fois qu'il a trouvé une ligne qui satisfait le critère de `LEFT JOIN` .

Voici un exemple de cela :

```
SELECT * FROM t1 LEFT JOIN t2 ON t1.id=t2.id WHERE t2.id IS NULL;
```

Supposons que `t2.id` est défini comme `NOT NULL` . Dans ce cas, MySQL va scanner `t1` et rechercher des lignes dans `t2` via `t1.id` . Si MySQL trouve une ligne dans `t2` , il sait que `t2.id` ne peut pas être `NULL` , et il ne va pas scanner le reste des lignes de `t2` qui ont le même `id` . En d'autres termes, pour chaque ligne de `t1` , MySQL n'a besoin que de faire une recherche dans `t2` , indépendamment du nombre de lignes qui sont trouvées dans `t2` .

`range checked for each record (index map: #)`

MySQL n'a pas trouvé d'index satisfaisant à utiliser. Il va, à la place, pour chaque combinaison de lignes des tables précédentes, faire une vérification de quel index utiliser (si il en existe), et utiliser cet index pour continuer la recherche. Ce n'est pas très rapide, mais c'est plus rapide que de faire une recherche sans aucun index.

*Using filesort*

MySQL va avoir besoin d'un autre passage pour lire les lignes dans l'ordre. Le tri est fait en passant en revue toutes les lignes, suivant le `type de jointure` est stocker la clé de tri et le pointeur de la ligne pour chaque ligne qui satisfont la clause `WHERE` . Alors, les clés sont triées. Finalement, les lignes sont triées dans l'ordre.

*Using index*

Les informations de la colonne sont lues de la table, en utilisant uniquement les informations contenues dans l'index, sans avoir à faire d'autres lectures. Cela peut arriver lorsque toutes les colonnes utilisées dans une table font partie de l'index.

*Using temporary*

Pour résoudre la requête, MySQL va avoir besoin de créer une table temporaire pour contenir le résultat. C'est typiquement ce qui arrive si vous utilisez une clause `ORDER BY` sur une colonne différente de celles qui font partie de `GROUP BY` .

*Using where*

Une clause `WHERE` sera utilisée pour restreindre les lignes qui seront trouvées dans la table suivante, ou envoyée au client. Si vous n'avez pas cette information, et que la table est de type `ALL` ou `index` , vous avez un problème dans votre requête (si vous ne vous attendiez pas à tester toutes les lignes de la table).

Si vous voulez rendre vos requêtes aussi rapide que possible, vous devriez examiner les lignes qui utilisent `Using filesort` et `Using temporary` .

Vous pouvez obtenir une bonne indication de la qualité de votre jointure en multipliant toutes les valeurs de la colonne `rows` dans la table de la commande `EXPLAIN` . Cela est une estimation du nombre de lignes que MySQL va examiner pour exécuter cette requête. C'est aussi ce nombre qui sera utilisé pour interrompre votre requête, grâce à la variable `max_join_size` . Choix des paramètres du serveur .L'exemple ci−dessous illustre comme une requête `JOIN` peut être optimisée avec les résultats de la commande `EXPLAIN` .

Supposons que vous avez la requête `SELECT` suivante, et que vous l'examinez avec `EXPLAIN` :

```
EXPLAIN SELECT tt.TicketNumber, tt.TimeIn,
           tt.ProjectReference, tt.EstimatedShipDate,
           tt.ActualShipDate, tt.ClientID,
           tt.ServiceCodes, tt.RepetitiveID,
           tt.CurrentProcess, tt.CurrentDPPerson,
           tt.RecordVolume, tt.DPPrinted, et.COUNTRY,
           et_1.COUNTRY, do.CUSTNAME
       FROM tt, et, et AS et_1, do
       WHERE tt.SubmitTime IS NULL
           AND tt.ActualPC = et.EMPLOYID
           AND tt.AssignedPC = et_1.EMPLOYID
```

```
          AND tt.ClientID = do.CUSTNMBR;
```

Pour cette exemple, nous supposons que :

- Les colonnes utilisées sont déclarées comme ceci :

| Table | Colonne | Type de colonne |
|---|---|---|
| tt | ActualPC | CHAR(10) |
| tt | AssignedPC | CHAR(10) |
| tt | ClientID | CHAR(10) |
| et | EMPLOYID | CHAR(15) |
| do | CUSTNMBR | CHAR(15) |

- Les tables ont les index suivants :

| Table | Index |
|---|---|
| tt | ActualPC |
| tt | AssignedPC |
| tt | ClientID |
| et | EMPLOYID (primary key) |
| do | CUSTNMBR (primary key) |

- Les valeurs de `tt.ActualPC` ne sont pas réparties également.

Initialement, avant toute optimisation, la commande `EXPLAIN` produit les informations suivantes :

```
 table type possible_keys                key  key_len ref  rows  Extra
et    ALL  PRIMARY                     NULL NULL    NULL 74
do    ALL  PRIMARY                     NULL NULL    NULL 2135
et_1  ALL  PRIMARY                     NULL NULL    NULL 74
tt    ALL  AssignedPC,ClientID,ActualPC NULL NULL    NULL 3872
      range checked for each record (key map: 35)
```

Comme le type `type` vaut `ALL` pour chaque table, le résultat indique que MySQL fait un scan complet de toutes les tables. Cela va prendre un très long temps de calcul, car le nombre de lignes à examiner de cette façon est le produit du nombre de lignes de toutes les tables : dans notre cas, cela vaut `74 * 2135 * 74 * 3872 = 45,268,558,720` lignes. Si les tables étaient plus grandes, cela serait encore pire.

Le premier problème que vous avons ici, est que MySQL ne peut pas (encore) utiliser d'index sur les colonnes, si elles sont déclarées différemment. Dans ce contexte, les colonnes `VARCHAR` et `CHAR` sont les mêmes, mais elles ont été déclarée avec des tailles différentes. Comme `tt.ActualPC` est déclarée comme `CHAR(10)` et que `et.EMPLOYID` est déclaré comme `CHAR(15)` , il y a un problème de taille.

Pour corriger cette disparité, utilisez la commande `ALTER TABLE` pour agrandir la colonne `ActualPC` de 10 caractères à 15 :

```
 mysql> ALTER TABLE tt MODIFY ActualPC VARCHAR(15);
```

Maintenant, `tt.ActualPC` et `et.EMPLOYID` sont tous les deux des colonnes de type `VARCHAR(15)` . Exécuter la commande `EXPLAIN` produit maintenant le résultat suivant :

```
 table type   possible_keys   key     key_len ref         rows    Extra
tt    ALL    AssignedPC,ClientID,ActualPC NULL NULL NULL 3872    Using where
do    ALL    PRIMARY         NULL    NULL    NULL        2135
      range checked for each record (key map: 1)
et_1  ALL    PRIMARY         NULL    NULL    NULL        74
      range checked for each record (key map: 1)
et    eq_ref PRIMARY         PRIMARY 15      tt.ActualPC 1
```

Ce n'est pas parfait, mais c'est bien mieux. Le produit de toutes les lignes a été divisé par 74). Cette version s'exécute en quelques secondes.

Une autre modification peut être faîte pour éliminer les problèmes de taille de colonne pour `tt.AssignedPC = et_1.EMPLOYID` et `tt.ClientID = do.CUSTNMBR` :

```
 mysql> ALTER TABLE tt MODIFY AssignedPC VARCHAR(15),
    ->                MODIFY ClientID   VARCHAR(15);
```

Maintenant, `EXPLAIN` produit le résultat suivant :

```
 table type   possible_keys   key      key_len ref           rows Extra
et    ALL    PRIMARY         NULL     NULL    NULL          74
tt    ref    AssignedPC,     ActualPC 15      et.EMPLOYID   52   where used
             ClientID,
             ActualPC
et_1  eq_ref PRIMARY         PRIMARY  15      tt.AssignedPC 1
do    eq_ref PRIMARY         PRIMARY  15      tt.ClientID   1
```

C'est presque aussi bon que cela pourrait l'être.Le problème final est que, par défaut, MySQL support que les valeurs de la colonne `tt.ActualPC` sont uniformément répartie, et que ce n'est pas le cas pour la table `tt` . Mais il est facile de le dire à MySQL :

```
 shell> myisamchk --analyze PATH_TO_MYSQL_DATABASE/tt
shell> mysqladmin refresh
```

Maintenant, la jointure est parfaite, et la commande `EXPLAIN` produit ce résultat :

```
 table type   possible_keys key     key_len ref           rows Extra
tt    ALL    AssignedPC    NULL    NULL    NULL          3872 where used
             ClientID,
             ActualPC
et    eq_ref PRIMARY       PRIMARY 15      tt.ActualPC   1
et_1  eq_ref PRIMARY       PRIMARY 15      tt.AssignedPC 1
do    eq_ref PRIMARY       PRIMARY 15      tt.ClientID   1
```

Notez que la colonne `rows` dans le résultat de `EXPLAIN` est une prédiction éclairée de l'optimisateur de jointure MySQL. Pour optimiser une requête, vous devriez vérifier si ces nombres sont proches de la réalité. Si ce n'est pas le cas, vous pourriez obtenir de meilleures performances avec l'attribut `STRAIGHT_JOIN` dans votre commande `SELECT` , et en choisissant vous même l'ordre de jointure des tables dans la clause `FROM` .

## 5.2.2 Mesurer les performances d'une requête

Dans la plupart des cas, vous pouvez mesurer la performance d'une requête en comptant le nombre d'accès disques. Pour les tables de petite taille, vous pouvez généralement obtenir une seule lecture (car l'index est probablement en cache). Pour les tables plus grandes, vous pouvez estimer que vous aurez besoin de (en utilisant les index B tree) : `log(row_count) / log(index_block_length / 3 * 2 / (index_length + data_pointer_length)) + 1` lectures pour trouver une ligne.

Pour MySQL, un bloc d'index vaut généralement 1024 octets, et le pointeur de données vaut 4 octets. Une table de 500,000 avec un index de taille 3 (entier moyen) vous donnera : `log(500,000)/log(1024/3*2/(3+4)) + 1` = 4 lectures.

Comme l'index ci−dessus vous serait de taille 500,000 * 7 * 3/2 = 5.2Mo, (en supposant que les index des tampons sont remplit aux 2/3, ce qui est typique), vous aurez probablement l'essentiel de l'index en mémoire, et vous n'aurez alors besoin que de 1 ou 2 lectures pour lire le reste des lignes.

Pour les écritures, toutefois, vous aurez besoin de 4 lectures (comme ci−dessus), pour trouver la place du nouvel index, et normalement, deux autres lectures pour modifier l'index et la ligne.

Notez que le raisonnement ci−dessus n'indique pas que votre application va dégénérer en fonction du log N ! Tant que toute est mis en cache par l'OS ou le serveur SQL, les performances ne vont se réduire que marginalement, même si la table grossit beaucoup. Une fois que les données seront trop grosses pour être en cache, votre application va ralentir car le serveur devra faire des lectures sur le disque (ce qui va accroître le log N). Pour éviter cela, augmentez le cache d'index au fur et à mesure que votre index grossit. Choix des paramètres du serveur .

## 5.2.3 Vitesse des requêtes SELECT

En général, lorsque vous voulez rendre un `SELECT ... WHERE` plus rapide, la première chose à faire est de voir si vous pouvez ajouter des index. Comment MySQL utilise les index . Toutes les références entre les tables doivent normalement être faites avec des index. Vous pouvez utiliser la commande `EXPLAIN` pour déterminer les index utilisés pour le `SELECT` . Syntaxe de `EXPLAIN` (Obtenir des informations

`SELECT` ) .

Quelques conseils généraux :

- Pour aider MySQL à mieux optimiser les requêtes, exécutez `myisamchk --analyze` sur une table après l'avoir remplie avec quelques données consistantes. Cela met à jour une valeur pour chaque partie de l'index qui indique le nombre moyen de lignes qui ont la même valeur. (Pour les index uniques, c'est toujours 1, bien sûr.) MySQL utilisera cela pour décider quel index choisir pour connecter deux tables avec une 'expression non−constante'. Vous pouvez vérifier le retour de l'exécution d' `analyze` en faisant `SHOW INDEX FROM nom_de_table` et examiner la colonne `Cardinality` .
- Pour trier un index et des données par rapport à un index, utilisez `myisamchk --sort−index --sort−records=1` (si vous voulez trier selon le premier index). Si vous avec un index unique à partir duquel vous voulez lire toutes les lignes en prenant comme ordre cet index, c'est un bon moyen de rendre les traitements plus rapides. Notez, toutefois, que ce tri n'est pas le plus optimal et prendra beaucoup de temps pour une grosse table !

## 5.2.4 Comment MySQL optimise les clauses WHERE

Les optimisation de la clause `WHERE` sont présentées avec la commande `SELECT` car elles sont généralemnt utilisées avec la commande `SELECT` , mais les mêmes optimisations peuvent s'appliquer aux clauses `WHERE` des commandes `DELETE` et `UPDATE` .

Notez aussi que cette section est incomplète. MySQL fait de très nombreuses optimisations, et nous n'avons pas eu le temps de toutes les documenter.

Certaines des optimisations effectuées par MYSQL sont présentées ici :

- Suppression des parenthèses inutiles :
```
    ((a AND b) AND c OR (((a AND b) AND (c AND d))))
-> (a AND b AND c) OR (a AND b AND c AND d)
```
- Remplacement des constantes :
```
    (a<b AND b=c) AND a=5
-> b>5 AND b=c AND a=5
```
- Suppression des conditions constantes (nécessaires pour le remplacement des constantes) :
```
    (B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)
-> B=5 OR B=6
```
- Les expressions constantes utilisées par les index sont évaluées une fois.
- `COUNT(*)` sur une table simple, sans clause `WHERE` est lu directement dans les informations de la table pour les tables `MyISAM` et `HEAP` . Cela peut aussi être fait avec les expressions `NOT NULL` lorsqu'elles sont utilisées sur une seule table. table.
- Détection précoce est expressions constantes invalides. MySQL détecte rapidement les commandes `SELECT` qui sont impossibles, et ne retourne aucune ligne.
- `HAVING` est combiné avec la clause `WHERE` si vous n'utilisez pas la clause `GROUP BY` ou les fonctions de groupe ( `COUNT()` , `MIN()` ...).
- Pour chaque sous−jointure, une clause `WHERE` simplifiée est construite pour accélérer l'évaluation de `WHERE` pour chaque sous−joinutre, et aussi essayer d'ignorer les lignes le plus tôt possible.
- Toutes les tables constantes sont lues en premier, avant toute autre table de la requête. Une table constante est une table :
    - Une table vide ou une table d'une ligne.
    - Une table qui est utilisée avec la clause `WHERE` sur un index de type `UNIQUE` , ou avec une clé primaire `PRIMARY KEY` , dont toutes les parties sont des expressions constantes, et les parties de l'index sont identifiées comme `NOT NULL` .

Toutes les tables suivantes sont considérées comme constantes :

```
 mysql> SELECT * FROM t WHERE primary_key=1;
mysql> SELECT * FROM t1,t2
    ->          WHERE t1.primary_key=1 AND t2.primary_key=t1.id;
```

- La meilleure combinaison de jointure est obtenue en testant toutes les possibilités. Si toutes les colonnes des clauses `ORDER BY` et `GROUP BY` proviennent de la même table, cette table sera utilisée de préférence comme première table dans la jointure.
- Si il ya une clause `ORDER BY` et une clause `GROUP BY` différente, ou si la clause `ORDER BY` ou `GROUP BY` contient des colonnes issues des tables autres que la première, une table temporaire est créée.
- Si vous utilisez `SQL_SMALL_RESULT` , MySQL va utiliser une table temporaire en mémoire.
- Chaque index de table est interrogé, et le meilleur index qui représente moins de 30% des lignes est utilisé. Si un tel index ne peut être identifié, un scan rapide de la table est fait.
- Dans certains cas, MySQL peut lire des lignes depuis l'index sans même consulter le fichie de données. Si toutes les colonnes de l'index sont des nombres, alors seul l'arbre d'index sera utilisé pour résoudre la requête.
- Avant chaque affichage de ligne, celles qui ne satisfont pas les critères de la clause `HAVING` sont ignorées.

Quelques exemples de requêtes très rapides :

```
 mysql> SELECT COUNT(*) FROM tbl_name;
mysql> SELECT MIN(key_part1),MAX(key_part1) FROM tbl_name;
mysql> SELECT MAX(key_part2) FROM tbl_name
    ->        WHERE key_part_1=constant;
mysql> SELECT ... FROM tbl_name
    ->        ORDER BY key_part1,key_part2,... LIMIT 10;
mysql> SELECT ... FROM tbl_name
    ->        ORDER BY key_part1 DESC,key_part2 DESC,... LIMIT 10;
```

Les requêtes suivantes ne sont résolues qu'avec l'arbre d'index (en supposant que les colonnes sont numériques) :

```
 mysql> SELECT key_part1,key_part2 FROM tbl_name WHERE key_part1=val;
mysql> SELECT COUNT(*) FROM tbl_name
    ->        WHERE key_part1=val1 AND key_part2=val2;
mysql> SELECT key_part2 FROM tbl_name GROUP BY key_part1;
```

Les requêtes suivantes utilisent l'indexation pour lire les lignes dans un ordre donnés, dans faire de tri supplémentaire :

```
 mysql> SELECT ... FROM tbl_name
    ->            ORDER BY key_part1,key_part2,... ;
mysql> SELECT ... FROM tbl_name
    ->            ORDER BY key_part1 DESC,key_part2 DESC,... ;
```

## 5.2.5 Comment MySQL optimise la clause DISTINCT

`DISTINCT` est converti en `GROUP BY` sur toutes les colonnes, `DISTINCT` combiné avec un `ORDER BY` aura dans la plupart des cas recours à une table temporaire.

Quand vous combinerez `LIMIT #` avec `DISTINCT` , MySQL stoppera dès qu'il trouvera `#` lignes uniques.

Si vous n'utilisez pas de colonnes de toutes les tables utilisées, MySQL arrête de scanner la table non−utilisée dès qu'il trouve la première correspondance.

```
 SELECT DISTINCT t1.a FROM t1,t2 where t1.a=t2.a;
```

Dans ce cas, en supposant que `t1` est utilisée avant `t2` (vérifiez avec `EXPLAIN` ), MySQL arrêtera de lire à partir de `t2` (pour cette ligne particulière de `t1` ) lorsque la permière ligne de `t2` est trouvée.

## 5.2.6 Comment MySQL optimise LEFT JOIN et RIGHT JOIN

`A LEFT JOIN B` est implémenté dans MySQL comme suit :

- La table `B` est censée être dépendante de la table `A` et de toutes les tables dont dépend `A` .
- La table `A` est censée être dépendante de toutes les tables (à part `B` ) qui sont utilisées dans la condition du `LEFT JOIN` .
- Toutes les conditions du `LEFT JOIN` sont transmises à la clause `WHERE` .
- Toutes les optimisations standards de jointures sont effectuées, à l'excepté qu'une table est toujours lue après celles dont elle dépend. S'il y a une dépendance circulaire, MySQL retournera une erreur.
- Toutes les optimisations standards de `WHERE` sont effectuées.
- S'il y a une ligne dans `A` qui répond à la clause `WHERE` , mais qu'il n'y avait aucune ligne dans `B` qui répondait à la condition du `LEFT JOIN` , alors une ligne supplémentaire de `B` est générée avec toutes les colonnes mises à `NULL` .
- Si vous utilisez `LEFT JOIN` pour trouver les enregistrements qui n'existent pas dans d'autres tables et que vous effectuez le test suivant : `nom_colonne IS NULL` dans la partie `WHERE` , où `nom_colonne` est une colonne qui est déclarée en tant que `NOT NULL` , alors MySQL arrêtera de chercher d'autres lignes (pour une combinaison de clefs particulière) après avoir trouvé une ligne qui répond à la condition du `LEFT JOIN` .

`RIGHT JOIN` est implémenté de manière analogue à `LEFT JOIN` .

L'ordre de lecture de tables forcé par `LEFT JOIN` et `STRAIGHT JOIN` aidera l'optimiseur de jointures (qui calcule l'ordre dans lequel les tables doivent être jointes) à faire son travail plus rapidement, puisqu'il y aura moins de permutations de tables à vérifier.

Notez que ce qui précède signifie que si vous faites une requête de la sorte :

```
SELECT * FROM a,b LEFT JOIN c ON (c.key=a.key) LEFT JOIN d (d.key=a.key)
        WHERE b.key=d.key
```

MySQL fera une recherche complète sur `b` puisque le `LEFT JOIN` forcera sa lecture avant celle de `d` . Un palliatif est de changer la requête en :

```
SELECT * FROM b,a LEFT JOIN c ON (c.key=a.key) LEFT JOIN d (d.key=a.key)
        WHERE b.key=d.key
```

## 5.2.7 Comment MySQL optimise les clauses ORDER BY

Dans certain cas, MySQL peut utiliser un index pour répondre à une requête `ORDER BY` ou `GROUP BY` sans faire aucun tri.

L'index peut être utilisé même si le `ORDER BY` ne correpond pas exactement à l'index, tant que toutes les parties inutilisée de l'index et les colonnes du `ORDER BY` sont constantes dans la clause `WHERE` . Les requêtes suivantes utilisent l'index pour répondre aux parties `ORDER BY` / `GROUP BY` :

```
 SELECT * FROM t1 ORDER BY partie_clef1,partie_clef2,...
SELECT * FROM t1 WHERE partie_clef1=constante ORDER BY partie_clef2
SELECT * FROM t1 WHERE partie_clef1=constante GROUP BY partie_clef2
SELECT * FROM t1 ORDER BY partie_clef1 DESC,partie_clef2 DESC
SELECT * FROM t1 WHERE partie_clef1=1 ORDER BY partie_clef1 DESC,partie_clef2 DESC
```

Quelques cas où MySQL ne peut **pas** utiliser les index pour répondre à `ORDER BY` : (Notez que MySQL utilisera quand même les indexes pour trouver les lignes qui correspondent à la clause `WHERE` ) :

- Vous effectuez un `ORDER BY` sur des clefs différentes : `SELECT * FROM t1 ORDER BY key1,key2`
- Vous effectuez un `ORDER BY` en utilisant des parties de clef non consécutives. `SELECT * FROM t1 WHERE key2=constante ORDER BY partie_clef2`
- Vous mélangez `ASC` et `DESC` . `SELECT * FROM t1 ORDER BY partie_clef1 DESC,partie_clef2 ASC`
- La clef utilisée pour extraire les résultats n'est pas la même que celle utilisée lors du groupement `ORDER BY` : `SELECT * FROM t1 WHERE key2=constant ORDER BY key1`
- Vous faites une jointure entre plusieurs tables et les colonnes sur lesquelles vous faites un `ORDER BY` ne font pas toutes parties de la première table non− `const` qui est utilisée pour récupérer les lignes (C'est la première table dans l'affichage d' `EXPLAIN` qui n'utilise pas une

méthode de récupération sur une ligne constante).
- Vous avez plusieurs expressions `ORDER BY` et `GROUP BY` .
- L'index de table utilisé est un type d'index qui n'enregistre pas les lignes dans l'ordre. (comme le type d'index `HASH` dans les tables `HEAP` ).
- Les colonnes index peuvent contenir des valeurs `NULL` et l'une d'elles utilise `ORDER BY ... DESC` . Cela vient du fait que en SQL, les valeurs `NULL` sont toujours triées avant les valeurs normales, que vous utilisiez `DESC` ou non.

Dans les cas où MySQL doit trier les résultats, il utilisera l'algorithme suivant :

- Read all rows according to key or by table scanning. Les lignes qui ne répondent pas à la clause `WHERE` sont évitées.
- Store the sort−key in a buffer (of size `sort_buffer` ).
- Lorsque le tampon est plein, exécuter un qsort dessus et enregistrer le résultat dans un fichier temporaire. Sauvegarder un pointeur sur le block enregistré. (Dans le cas où toutes les lignes rentre dans le tampon de tri, aucun fichier temporaire n'est crée)
- Répeter ce qui précède jusqu'à ce que toutes les lignes soient lues.
- Do a multi−merge of up to `MERGEBUFF` (7) regions to one block in another temporary file. Repeat until all blocks from the first file are in the second file.
- Répeter ce qui précède jusqu'à ce qu'il y ait moins de `MERGEBUFF2` (15) blocks à traiter.
- On the last multi−merge, only the pointer to the row (last part of the sort−key) is written to a result file.
- Now the code in `sql/records.cc` will be used to read through them in sorted order by using the row pointers in the result file. To optimise this, we read in a big block of row pointers, sort these and then we read the rows in the sorted order into a row buffer ( `record_rnd_buffer` ) .

Vous pouvez vérifier avec `EXPLAIN SELECT ... ORDER BY` si MySQL peut utiliser des index pour répondre à cette requête. Si vous obtenez un `Using filesort` dans la colonne `extra` , c'est que MySQL ne peut utiliser d'index pour résoudre cet `ORDER BY` . Syntaxe de `EXPLAIN` (Obtenir des informations `SELECT` ) .

Si vous voulez plus de rapidité avec les `ORDER BY` , vous devez d'abord voir si vous pouvez faire en sorte que MySQL utilises des index au lieu de passer par des phases de tri en plus. Si cela se révèle impossible, vous pouvez :

- Augmenter la taille de la variable `sort_buffer` .
- Augmenter la taille de la variable `record_rnd_buffer` .
- Changer `tmpdir` pour qu'il pointe vers un disque dédié avec beaucoup d'espaces libres.

## 5.2.8 Comment MySQL optimise la clause LIMIT

Dans certains cas, MySQL va gérer la requête différemment avec la clause `LIMIT #` , si la clause `HAVING` n'est pas utilisée :

- Si vous ne sélectionnez que quelques lignes avec `LIMIT` , MySQL va utiliser les index dans certains cas, où il aurait préféré utiliser un scan de table complet.
- Si vous utilisez `LIMIT #` avec la clause `ORDER BY` , MySQL va arrêter de trier dès qu'il a trouvé la première `#` au lieu de trier toute la table.
- Lorsque vous combinez `LIMIT #` avec `DISTINCT` , MySQL va s'arrêter dès qu'il a trouvé `#` lignes distinctes.
- Dans certains cas, la clause `GROUP BY` peut être appliquée en lisant les clés dans l'ordre (ou en faisant un tri sur la clé), puis en calculant un sommaire, jusqu'à ce que la clé soient modifiée. Dans ce cas, `LIMIT #` ne va pas appliquer les éléments non nécessaires de la clause `GROUP BY` s.
- Aussitôt que MySQL a envoyé les premières `#` lignes au client, il annule le reste de la

requête (si vous n'utilisez pas la fonction `SQL_CALC_FOUND_ROWS` ).
- `LIMIT 0` va toujours retourner rapidement un résultat vide. C'est pratique pour vérifier une requête et lire les types de colonnes du résultat, sans exécuter réellement la requête.
- Lorsque le serveur utilise des tables temporaire pour résoudre les requêtes, la clause `LIMIT #` est utilisée pour calculer l'espace nécessaire.

## 5.2.9 Vitesse des requêtes INSERT

Le temps d'insertion d'une ligne est constitué comme ceci :

- Connexion : (3)
- Envoi au serveur : (2)
- Analyse de la requête : (2)
- Insertion de la ligne : (1 x taille de la ligne)
- Insertion des index : (1 x nombre d'index)
- Fermeture : (1)

où les nombres représentent une partie proportionnelle du temps total. Le calcul ne prend pas en compte les coûts d'administration initiaux de l'ouverture des tables (qui est fait une fois pour chaque requête simultanée).La taille de la table ralentit les opérations d'insertion des index par un facteur de log N (B−trees).

Quelques méthodes pour accélérer les insertions :

- Si vous insérerz plusieurs lignes depuis le même client, en même temps, utilisez les valeurs multiples de la commande `INSERT` . C'est bien plus rapide (et parfois beaucoup plus rapide) que d'utiliser des commandes `INSERT` distinctes. Si vous ajoutez des données dans une table non vide, vous pouvez ajuster la variable `bulk_insert_buffer_size` pour l'accélérer encore plus. `SHOW VARIABLES` .
- Si vous insérez de nombreuses lignes depuis différents clients, vous pouvez accélérer les insertions en utilisant la commande `INSERT DELAYED` . Syntaxe des `INSERT` .
- Notez qu'avec les tables `MyISAM` , vous pouvez insérer des lignes en même temps que vous utilisez des commandes `SELECT` , du moment qu'il n'y a pas d'effacement de ligne dans la table.
- Lorsque vous chargez une table depuis un fichier texte, utilisez la commande `LOAD DATA INFILE` . Elle est généralement 20 fois plus rapide que l'équivalent en commandes `INSERT` . Syntaxe des `LOAD DATA INFILE` .
- Il est possible, avec un peu de travail supplémentaire, d'accélérer encore la vitesse des commandes `LOAD DATA INFILE` . Utilisez la procédure standard :
    - Créez optionnellement une table avec `CREATE TABLE` . Par exemple, en utilisant `mysql` ou Perl−DBI.
    - Exécutez une commande `FLUSH TABLES` ou la commande en ligne shell `mysqladmin flush−tables` .
    - Utilisez `myisamchk −−keys−used=0 −rq /path/to/db/tbl_name` . Cela va supprimer l'utilisation des index dans la table.
    - Insérez vos données dans la table, avec `LOAD DATA INFILE` . Les index ne seront pas modifiés, et donc, très rapides.
    - Si vous allez uniquement lire la table dans le futur, utilisez `myisampack` pour la réduire de taille. Caractéristiques des tables compressées .
    - Re−créez les index avec `myisamchk −r −q /path/to/db/tbl_name` . Cette commande va créer l'arbre d'index en mémoire, avant de l'écrire sur le disque, ce qui est bien plus rapide, car il n'y a que peu d'accès disques. L'arbre final sera aussi parfaitement équilibrés.
    - Exécutez une commande `FLUSH TABLES` ou utilisez la commande en ligne shell `mysqladmin flush−tables` .

  Notez que la ocmmande `LOAD DATA INFILE` fait aussi les optimisations ci−dessus, si vous faites les

insertions dans une table vide. La différence principale avec la procédure ci−dessus est que vous pouvez laisser `myisamchk` allouer plus de mémoire temporaire pour la création d'index, que vous ne pourriez le faire pour chaque recréation.Depuis MySQL 4.0 vous pouvez aussi utiliser `ALTER TABLE tbl_name DISABLE KEYS` au lieu de `myisamchk −−keys−used=0 −rq /path/to/db/tbl_name` et `ALTER TABLE tbl_name ENABLE KEYS` au lieu de `myisamchk −r −q /path/to/db/tbl_name` . De cette façon, vous pouvez aussi éviter l'étape `FLUSH TABLES` .

- Vous pouvez accélérer les insertions qui sont faites avec plusieurs requêtes en verrouillant vos tables :

```
 mysql> LOCK TABLES a WRITE;
mysql> INSERT INTO a VALUES (1,23),(2,34),(4,33);
mysql> INSERT INTO a VALUES (8,26),(6,29);
mysql> UNLOCK TABLES;
```

La principale différence de vitesse est que l'index de buffer est écrit sur le disque une fois, après toutes les insertions `INSERT` terminées. Normalement, il aurait du y avoir de nombreuses écritures, une pour chaque commande `INSERT` . Le verrouillage n'est pas nécessaire si vous pouvez insérer toutes les lignes d'une seule commande.

Pour les tables transactionnelles, vous devriez utiliser `BEGIN/COMMIT` au lieu de `LOCK TABLES` pour accélérer les opérations.

Le verrouillage va aussi réduire le nombre total de tests de connexions, mais le temps d'attente maximum de certains threads va augmenter (car il va y avoir la queue pour les verrous). Par exemple :

```
 thread 1 fait 1000 insertions
thread 2, 3, et 4 font 1 insertion
thread 5 fait 1000 insertions
```

Si vous ne voulez pas utiliser le verrouillage, les threads 2, 3 et 4 auront fini avant les 1 et 5. Si vous utilisez le verrouillage, 2, 3 et 4 me finiront probablement pas avant 1 ou 5, mais la durée globale de l'opération sera 40% plus courte.Comme les commandes `INSERT` , `UPDATE` et `DELETE` sont très rapides avec MySQL, vous obtiendrez de meilleures performances générales en ajoutant des verrous autour de toutes vos opérations de 5 insertions o modifications simultanées. Si vous faîtes de très nombreux insertions dans une ligne, vous pouvez utiliser `LOCK TABLES` suivi de `UNLOCK TABLES` une fois de temps en temps (par exemple, toutes les 1000) pour permettre aux autres threads d'accéder à la table. Cela vous donnera quand même une bonne accélération.Bien sur, `LOAD DATA INFILE` reste bien plus rapide pour charger les données.

Pour accélérer `LOAD DATA INFILE` et `INSERT` , agrandissez le buffer de clé. Choix des paramètres du serveur .

## 5.2.10 Vitesses des commandes UPDATE

Les requêtes de modification sont optimisées comme les requêtes de `SELECT` avec le coût supplémentaire de l'écriture. La vitesse d'écriture dépend de la taille des données qui sont modifiées, et du nombre d'index que cela va impacter. Les index ne sont pas modifiés tant que la ligne n'est pas écrite. Les index qui ne sont pas modifiés ne seront pas réécrit.

De plus, une autre méthode pour obtenir des accélérations avec les modifications est de retarder les modifications, et d'en faire plusieurs d'un coup. Faire plusieurs modifications d'un coup est bien plus rapide que d'en faire une à chaque fois.

Notez que, avec le format de ligne dynamique, la modification d'une ligne peut déboucher sur la fragmentation de la ligne. Si vous le faite souvent, il est très important d'appliquer `OPTIMIZE TABLE` sur ces tables, pour les optimiser. Syntaxe de `OPTIMIZE TABLE` .

## 5.2.11 Rapidité des requêtes DELETE

Si vous voulez effacer toutes les lignes d'une table, vous devez utiliser `TRUNCATE TABLE nom_de_table` . Syntaxe des `TRUNCATE` .

Le temps de suppression d'une ligne est exactement proportionnel au nombre d'index. Pour effacer les enregistrements plus rapidement, vous pouvez augmenter la taille du cache d'index. Choix des paramètres du serveur .

## 5.2.12 Autres conseils d'optimisation

Quelques conseils en vrac pour accélérer le serveur :

- Utilisez les connexions persistantes à la base, pour éviter les coûts récurents de connexion. Si vous ne pouvez pas utiliser de connexions persistantes, et que vous faites de nombreuses connexions à la base, essayez de modifier la valeur de la variable `thread_cache_size` . Choix des paramètres du serveur .
- Vérifiez toujours que vos requêtes utilisent vraiment les index que vous avez créé dans les tables. Avec MySQL, vous pouvez utiliser la commande `EXPLAIN` . Syntaxe de `EXPLAIN` (Obtenir des informations `SELECT` ) .
- Essayez d'éviter les requêtes `SELECT` complexes sur les tables `MyISAM` qui sont souvent modifiées. Cela évitera des problèmes de verrouillage.
- Les nouvelles tables `MyISAM` peuvent insérer des lignes sans en effacer d'autre, tout en lisant dans cette table. Si c'est important pour vous, vous pouvez considérer d'autres méthodes où vous n'avez pas à effacer de lignes, ou bien utilisez `OPTIMIZE TABLE` après avoir effacé beaucoup de lignes.
- Utilisez `ALTER TABLE ... ORDER BY expr1,expr2...` si vous lisez les colonnes dans l'ordre `expr1,expr2...` . Avec cette option, après de grosses modifications dans la table, vous pourriez obtenir de meilleures performances.
- Dans certains cas, cela vaut la peine d'ajouter une colonne qui est une combinaison ('hashed') des informations des autres colonnes. Si cette colonne est courte, et plutôt exemptes de doublons, elle peut se révéler plus rapide qu'un gros index sur plusieurs colonnes. Avec MySQL, il est très facile d'utiliser une telle colonne : `SELECT * FROM table_name WHERE hash=MD5(CONCAT(col1,col2)) AND col_1='constant' AND col_2='constant'`
- Pour les tables qui sont souvent modifiées, vous devriez essayer d'éviter les colonnes `VARCHAR` et `BLOB` . Vous obtiendrez des lignes à format dynamique si vous utilisez ne serait−ce qu'une seule colonne `VARCHAR` ou `BLOB` . Types de tables MySQL .
- Normalement, cela ne sert à rien de séparer une table en différentes tables plus petites, juste parce que vos lignes deviennent 'grosses'. Pour accéder à une ligne, le plus long est le temps d'accès au premier octets de la ligne. Après cela, les disques modernes vont lire très rapidement la ligne, et suffisament pour la plus par des applications. Le seul cas où cela peut être important est si vous êtes capables de dégager une table à format de ligne fixe (voir ci−dessus), ou si vous avez besoin de scanner régulièrement la table, mais que vous n'avez pas besoin de toutes les colonnes. Types de tables MySQL .
- Si vous avez besoin de calculer souvent des expressions en fonction des informations placées dans de nombreuses lignes (comme compter des lignes), il est probablement plus efficace d'introduire une nouvelle table qui va mettre à jour ce compteur en temps réel. Une modification du type type `UPDATE table set count=count+1 where index_column=constant` est très rapide!C'est très important lorsque vous utilisez les types de tables MyISAM et ISAM, qui ne dispose que d'un verrouillage de table (plusieurs lecteurs, un seul qui écrit). Cela va aussi améliorer les performances avec la plus par des bases, car le gestionnaire de verrouillage de ligne aura moins de tâches à faire.
- Si vous devez rassembler des statistiques issues de grosses tables de log, utiliser les tables

de sommaires plutôt que la table complète. Entretenir un sommaire est bien plus rapide que de regénérer des tables à partir des logs à chaque modification (suivant la criticité de vos informations), plutôt que de modifier l'application qui fonctionne.

- Si possible, essayez de marquer les rapports comme 'direct' ou 'statistique', où les données nécessaires pour les rapports statistiques ne sont générées qu'à partir de tables de sommaires, calculées depuis les données réelles.
- Utilisez les valeurs par défaut des colonnes. N'insérez des valeurs explicitement que lorsque la valeur diffère de la valeur par défaut. Cela réduit le temps d'analyse de MySQL, et améliore les insertions.
- Dans certains cas, il est pratique de compacter et stocker les données dans un blob. Dans ce cas, vous devez ajouter du code supplémentaire pour compacter et décompacter les données dans le blob, mais cela pourra vous faire économiser de nombreux accès. C'est pratique lorsque vous avez des données qui ne peuvent s'adapter facilement à une structure de base de données.
- Normalement, vous devriez essayer de garder vos données non redondantes (ce qui s'appelle la troisième forme normale dans les théories de bases de données), mais ne vous empêchez pas de duplique des données ou de créer des tables de sommaire, pour gagner de la vitesse.
- Les procédures stockées ou UDF (fonctions utilisateur) peuvent être une bonne façon de gagner en performance. Dans ce cas, vous devriez avoir une méthode pour appliquer les mêmes fonctions d'une autre manière, si votre base ne supporte les Les procédures stockées.
- Vous pouvez aussi gagner de la vitesse en utilisant des caches de requêtes dans vos applications, et en essayant de rassembler les nombreuses insertions ou modifications. Si votre base de données supporte le verrouillage de table (comme MySQL et Oracle), cela vous aidera à vous assurer que le cache d'index est vidé après chaque modifications.
- Utilisez `INSERT /*! DELAYED */` lorsque vous n'avez pas besoin d'être assuré que vos données sont écrites. Cela accélère les insertions, car de nombreuses lignes seront écrites en une seule fois.
- Utilisez `INSERT /*! LOW_PRIORITY */` lorsque vous voulez que vos sélections soient prioritaires.
- Utilisez `SELECT /*! HIGH_PRIORITY */` pour rendre les sélections prioritaires. C'est à dire, les sélections seront désormais faites même si un autre programme attend pour écrire.
- Utilise la commande `INSERT` multiple pour insérer plusieurs lignes en une seule commande SQL (plusieurs serveurs SQL le supporte).
- Utilisez `LOAD DATA INFILE` pour charger de grande quantité de données dans une table. C'est généralement plus rapide que des insertions, et sera même encore plus rapide une fois que `myisamchk` sera intégré dans `mysqld` .
- Utilisez les colonnes `AUTO_INCREMENT` pour avoir des valeurs uniques.
- Utilisez `OPTIMIZE TABLE` une fois de temps en temps, pour éviter la fragmentation lors de l'utilisation de tables avec un format de ligne dynamique. Syntaxe de `OPTIMIZE TABLE` .
- Utilisez la tables de type `HEAP` pour accélérer les traitements au maximum. Types de tables MySQL .
- Avec un serveur web normal, les images doivent être stockées dans des fichiers. C'est à dire, ne stockez qu'une référence au fichier d'image dans la base. La raison principale à cela est qu'un serveur web est bien meilleur pour mettre en cache des fichiers que le contenu d'une base de données. Il est donc plus rapide si vous utilisez des fichiers.
- Utilisez des tables en mémoire pour les données non critiques, qui ont besoin d'être lues souvent (comme des informations sur la dernière bannière affichée pour les utilisateurs sans cookies).
- Les colonnes contenant des informations identiques dans différentes tables doivent être déclarées identiquement lors de la création des tables, et porter des noms identiques. Avant la version 3.23, vous pouviez ralentir les jointures.Essayez de garder des noms simples (utilisez `nom` au lieu de `nom_du_client` dans la table de clients). Pour rendre vos noms de colonnes portables vers les autres serveurs SQL, vous devriez essayer de les garder plus petits que

18 caractères.

- Si vous avez vraiment besoin de très haute vitesse, vous devriez considérer les interfaces de bas niveau pour le stockage des données que les différents serveurs SQL supportent. Par exemple, en accédant directement aux tables MySQL `MyISAM` , vous pourriez obtenir un gain de vitesse de l'ordre de 2 à 5 fois, en comparaison avec l'interface SQL. Pour cela, les données doivent être sur le même serveur que l'application, et généralement, elles ne doivent être manipulées que par un seul programme à la fois (car le verrouillage externe de fichiers est très lent). Vous pouvez éliminer ces problèmes en créan des commandes `MyISAM` de bas niveau dans le serveur MySQL (cela peut se faire facilement pour améliorer les performances). Soyez très prudent dans la conception de votre interface, mais il est très facile de supporter ce type d'optimisation.
- Dans de nombreux cas, il est plus rapide d'accéder aux données depuis une base (en utilisant une connexion ouverte) que d'accéder à un fichier texte, car la base de données est plus compacte que le fichier texte (si vous utilisez des données numériques), et cela entraine moins d'accès disques. Vous allez aussi économiser du code, car vous n'aurez pas à analyser le fichier texte pour repérer les limites de lignes.
- Vous pouvez aussi utiliser la réplication pour accélérer le serveur. Réplication avec MySQL .
- Déclarer une table avec `DELAY_KEY_WRITE=1` va accélérer la mise à jour des index, car ils ne seront pas écrit sur le disque jusqu'à ce que le fichier de données soit refermé. L'inconvénient est que vous devez exécuter l'utilitaire `myisamchk` sur ces tables avant de lancer `mysqld` pour vous assurer que les index sont bien à jour, au cas où le processus aurait été interrompu avant d'enregistrer les données. Comme les informations d'index sont toujours regénérables, vous ne perdrez pas de données avec `DELAY_KEY_WRITE` .

# 5.3 Verrouillage de tables

## 5.3.1 Comment MySQL verrouille les tables

Vous pouvez trouver une discussion sur les différentes méthodes de verrouillage dans l'annexe. Méthodes de verrouillage .

Toutes les méthodes de verrouillage de MySQL sont exemptes de blocage, sauf pour les tables `InnoDB` et `BDB` . Ceci fonctionne en demandant tous les verrous d'un seul coup, au début de la requête, et en verrouillant les tables toujours dans le même ordre.

Les tables `InnoDB` obtiennent automatiquement leur verrou de ligne et les tables `BDB` leur verrou de page, durant le traitement de la requête SQL, et non pas au démarrage de la transaction.

La méthode de verrouillage des tables de MySQL en écriture ( `WRITE` ) fonctionne comme ceci :

- Si il n'y a pas de verrou sur la table, pose un verrou en écriture dessus.
- Sinon, soumet une requête de verrouillage dans la queue de verrous d'écriture.

La méthode de verrouillage des tables de MySQL en lecture ( `READ` ) fonctionne comme ceci :

- Si il n'y a pas de verrou sur la table, pose un verrou en écriture dessus.
- Sinon, soumet une requête de verrouillage dans la queue de verrou de lecture.

Lorsqu'un verrou est libéré, le verrou est donné aux threads de la queue de verrou en écriture, puis à ceux de la queue de verrou en lecture.Cela signifie que si vous avez de nombreuses modifications dans une table, la commande `SELECT` va attendre qu'il n'y ait plus d'écriture avant de lire.

Pour contourner ce problème dans les cas où vous voulez faire de nombreuses `INSERT` et `SELECT` sur la même table, vous pouvez insérer les lignes dans une table temporaire, et ne modifier la table réelle que de temps en temps, à partir de la table temporaire.

Ceci peut être fait comme ceci :

```
 mysql> LOCK TABLES real_table WRITE, insert_table WRITE;
mysql> INSERT INTO real_table SELECT * FROM insert_table;
mysql> TRUNCATE TABLE insert_table;
mysql> UNLOCK TABLES;
```

Vous pouvez aussi utiliser les options `LOW_PRIORITY` de `INSERT` , `UPDATE` ou `DELETE` ou `HIGH_PRIORITY` avec `SELECT` si vous voulez donner d'autres priorités aux actions. Vous pouvez aussi démarrer `mysqld` avec l'option `--low-priority-updates` pour obtenir ce type de comportement.

Utiliser l'option `SQL_BUFFER_RESULT` peut aussi réduire la durée des verrous de tables. Syntaxe des `SELECT` .

Vous pouvez aussi changer le code de verrouillage dans le fichier `mysys/thr_lock.c` , pour utiliser une queue simple. Dans ce cas, les verrous en écriture et en lecture auront la même priorité, ce qui peut aider certaines applications.

## 5.3.2 Problème de verrouillage de tables

Le système de verrou de MySQL est exempt de blocage.

MySQL utilise le verrouillage de table (au lieu du verrouillage de ligne ou de colonne) sur tous les types de tables, sauf `InnoDB` et `BDB` , pour obtenir un système de verrou à très haute vitesse. Pour les grandes tables, le verrouillage de table est bien plus rapide que le verrouillage de page, mais il y a aussi des inconvénients.

Pour les tables `InnoDB` et `BDB` , MySQL n'utilise le verrouillage de table que vous le demandez explicitement avec `LOCK TABLES` . Pour ces tables, nous vous recommandons de ne jamais utiliser la commande `LOCK TABLES` , car `InnoDB` utilise un verrouillage de ligne automatique, et `BDB` utilise un verrouillage de pages, pour assurer l'isolation des transactions.

Depuis MySQL version 3.23.7, vous pouvez insérer des lignes dans les tables `MyISAM` , en même temps que d'autres threads y lisent. Notez que, actuellement, cela ne fonctionne que si il n'y a pas de trous après les lignes effacées dans la table, au moment de l'insertion. Lorsque tous les trous ont été bouchés avec de nouvelles données, les insertions simultanées seront automatiquement réactivées.

Le verrouillage de table permet à de nombreux threads de lire dans la même table, mais si un thread désire écrire dans la table, il doit obtenir un verrou en écriture pour avoir un accès exclusif. Durant la modification, les autres threads qui voudront lire dans cette table, devront attendre.

Comme les modifications de tables sont considérées comme plus importantes que les lectures avec `SELECT` , toutes les commandes qui modifient la table ont priorités sur les lectures. Cela devrait vous assurer que les modifications ne sont pas retenues trop longtemps, à cause de nombreuses lectures sur une même table. Vous pouvez toutefois modifier cela avec l'option `LOW_PRIORITY` des commandes de modification, et l'option `HIGH_PRIORITY` de `SELECT` ).

Depuis MySQL version 3.23.7, vous pouvez utiliser la variable `max_write_lock_count` pour forcer MySQL à laisser temporairement la place à toutes les commandes `SELECT` , après un certain nombre de modifications dans la table.

Le verrouillage de table est une mauvaise technique dans les situations suivantes :

- Un client exécute une commande `SELECT` qui prend très longtemps.
- Un autre client exécute une commande `UPDATE` sur la table. Ce client va devoir attendre que la commande `SELECT` soit finie.
- Un autre client exécute une autre commande `SELECT` sur la même table. Comme `UPDATE` a la priorité sur `SELECT` , cette commande `SELECT` va attendre que `UPDATE` soit finit. Il va donc attendre que le premier `SELECT` soit fini.
- Un thread attend des erreurs comme `full disk` , auquel cas tous les threads qui veulent accéder à la table seront en attente de plus d'espace sur le disque.

Des solutions aux problèmes sont :

- Essayez d'accélérer au maximum les commandes `SELECT` . Vous pourriez passer par une table de sommaire pour cela.
- Démarrez `mysqld` avec l'option `--low-priority-updates` . Cela va donner aux commandes de modification une priorité plus faible que `SELECT` . Dans ce cas, c'est la commande `SELECT` du précédent scénario qui s'exécutera avant la commande `INSERT` .
- Vous pouvez donner à une commande spécifique `INSERT` , `UPDATE` ou `DELETE` , une priorité plus basse avec l'attribut `LOW_PRIORITY` .
- Démarrez `mysqld` avec une valeur faible pour `max_write_lock_count` afin de donner plus souvent la chance aux verrous `READ` la possibilité de lire des données, entre deux verrous `WRITE` .
- Vous pouvez spécifier que toutes les modifications d'un thread spécifique doivent être faites avec un priorité basse, en utilisant la commande SQL : `SET LOW_PRIORITY_UPDATES=1` . Syntaxe de `SET` .
- Vous pouvez spécifier qu'une requête particulière `SELECT` est très importante, en utilisant l'attribut `HIGH_PRIORITY` . Syntaxe des `SELECT` .
- Si vous avez des problèmes avec des `INSERT` combinés avec des `SELECT` , utilisez les tables `MyISAM` car elle supportent les commandes `SELECT` s et `INSERT` simultanées.
- Si vous voulez mélanger les commandes `INSERT` et `SELECT` , utilisez l'attribut `DELAYED` de la commande `INSERT` pour résoudre ce problème. Syntaxe des `INSERT` .
- Si vous avez des problèmes avec `SELECT` et `DELETE` , l'option `LIMIT` de `DELETE` peut aider. Syntaxe des `DELETE` .

# 5.4 Optimisation de la structure de la base de données

## 5.4.1 Conception

MySQL conserve les données et les index dans deux fichiers séparés. De nombreux (et en fait presque toutes) les autres bases mélangent les données et les index dans le même fichier. Nous pensons que le choix de MySQL est bien meilleur pour un grand nombre de systèmes modernes.

Une autre méthode de stockage des données est de conserver les informations de chaque colonne dans une zone séparée (par exemple SDBM et Focus). Cela va réduire les performances qui accèdent à plus d'une colonne. Comme cela dégénère vite lorsque plus d'une colonne est utilisée, nous pensons que ce modèle n'est pas bon pour une base de données généraliste.

Les cas les plus courants sont que les index et les données sont stockées ensemble (comme Oracle/Sybase et al). Dans ce cas, vous aurez aussi les informations de lignes dans la page finale de l'index. L'intérêt d'une telle organisation est que, dans de nombreuses situations, dépendamment du cache d'index, vous économisez des lectures disques. Les problèmes de cette organisation sont :

- Le scan des tables est bien plus lent, car vous devez lire les index pour obtenir les données.
- Vous ne pouvez pas utiliser uniquement l'index pour lire des données pour une requête.

- Vous utilisez beaucoup d'espace, et vous devez dupliquer des index de noeuds (car vous ne pouvez pas simplement stocker des lignes dans les noeuds).
- Les suppressions vont perturber la table (comme les index ne sont pas modifiés lors de l'effacement).
- Il est plus difficile de ne mettre en cache que les données.

## 5.4.2 Rendre vos tables aussi compactes que possible

Une des optimisations simple est de réduire au maximum la taille de vos données et de vos index sur le disque et en mémoire. Cela peut donner des accélérations impressionnantes, car les lectures sur le disque sont plus rapides, et moins de mémoire centrale sera utilisée. L'indexation de colonnes de petites taille prend aussi moins de ressources.

MySQL supporte un grand nombre de type de tables et de format de ligne. Choisir ces types peut vous conduire à des améliorations de performances. Types de tables MySQL .

Vous pouvez obtenir des gains de performances sur les tables et minimiser l'espace disque en utilisant les techniques ci−dessous :

- Utilisez les types les plus efficaces et les plus petits possibles. MySQL a différents types spécialisés qui épargnent de l'espace disque et de la mémoire.
- Utilisez les types d'entiers les plus petits possible pour réduire les tables. Par exemple, `MEDIUMINT` est souvent préférable à `INT` .
- Déclarez les colonnes pour qu'elle soient `NOT NULL` si possible. Cela accélère les traitements, et vous fait gagner un bit par colonne. Notez que si vous avez vraiment besoin d'une valeur `NULL` dans votre application, il est recommandé de l'utiliser. Evitez simplement de l'utiliser par défaut sur toutes les colonnes.
- Si vous n'avez pas de colonne de taille variable ( `VARCHAR` , `TEXT` ou `BLOB` ), un format de ligne à taille fixe est utilisé. C'est plus rapide, mais cela prend plus d'espace sur le disque. Format de table `MyISAM` .
- La clé primaire doit être aussi courte que possible. Cela rend l'identification des lignes plus efficace.
- Pour chaque table, vous devez décider quel méthode de stockage et d'indexation vous allez utiliser. Types de tables MySQL .
- Ne créez que des index dont vous avez besoin. Les index sont bons pour accélérer les lectures, mais sont plus lents lorsque vous écrivez des données. Si vous accéder essentiellement à votre table en lecture avec des combinaisons de colonnes, faites un index avec ces colonnes. Le premier index doit être la colonne la plus utilisée. Si vous utilisez **constamment** de nombreuses colonnes, vous devriez utiliser la colonne avec le plus de doublons en premier, pour obtenir une meilleure compression.
- Si il est probable qu'une colonne a un préfixe unique avec les premiers caractères, il est mieux de n'indexer que ce préfixe. MySQL supporte les index sur une partie de colonne. Les index les plus courts sont les plus efficaces car ils prennent moins d'espace disque, et aussi, car ils absorbent plus de requêtes grâce au cache en mémoire. Choix des paramètres du serveur .
- Dans certaines circonstances, il peut être intéressant de séparer en deux une table qui est scannée très souvent. C'est particulièrement vrai pour les formats de tables dynamiques, et si possible, utilisez un format de table statique pour les colonnes les plus pertinentes.

## 5.4.3 Comment MySQL utilise les index

Les index sont utilisés pour trouver des lignes de résultat avec une valeur spécifique, très rapidement. Sans index, MySQL doit lire successivement toutes les lignes, et à chaque fois, faire les comparaisons nécessaires pour extraire un résultat pertinent. Plus la table est grosse, plus c'est coûteux. Si la table dispose d'un index pour les colonnes utilisées, MySQL peut alors trouver rapidement les positions des lignes dans le fichier de données, sans avoir à fouiller toute la table. Si une table à 1000 lignes, l'opération sera alors 100 fois plus rapide qu'une lecture séquentielle. Notez que si vous devez lire la presque totalité des 1000 lignes, la lecture séquentielle se révélera alors plus rapide, malgré tout.

Tous les index de MySQL ( `PRIMARY` , `UNIQUE` et `INDEX` ) sont stockés sous la forme de B−trees. Les chaînes sont automatiquement préfixée et leurs espaces terminaux sont supprimés. Syntaxe de `CREATE INDEX` .Les index sont utilisés pour :

- Trouver rapidement des lignes qui satisfont une clause `WHERE` .
- Lire des lignes dans d'autres tables lors des jointures.
- Trouver les valeurs `MAX()` et `MIN()` pour une colonne indexée. C'est une opération qui est optimisée par le préprocesseur, qui vérifie si vous utilisez la constante `WHERE` key_part_# = sur toute les parties de clés inférieures à < N. Dans ce cas, MySQL va faire une simple recherche de clé et remplacer l'expression par une constante. Si toutes les expressions sont remplacées par des constantes, la requête va alors être rapidement calculée :
```
SELECT MIN(key_part2),MAX(key_part2) FROM table_name where key_part1=10
```
- Trier ou grouper des lignes dans une table, si le tri ou le regroupement est fait avec un préfixe à gauche utilisable (par exemple, `ORDER BY key_part_1,key_part_2` ). La clé est lue en ordre inverse, si toutes les parties de clés sont suivies du mot clé `DESC` . Comment MySQL optimise `ORDER BY` .
- Dans certains cas, la requête peut être optimisée pour lire des valeurs sans consulter le fichier de données. Si cette possibilité est utilisée avec des colonnes qui sont toutes numériques, et forme le préfixe de gauche d'une clé, les valeurs peuvent être lues depuis l'index, à grande vitesse :
```
SELECT key_part3 FROM table_name WHERE key_part1=1
```

Supposez que vous utilisiez la commande `SELECT` suivante :

```
mysql> SELECT * FROM tbl_name WHERE col1=val1 AND col2=val2;
```

Si un index multi−colonne existe sur les colonnes `col1` et `col2` , les lignes appropriées seront directement lues. Si des index séparés sur les colonnes `col1` et `col2` existent, l'optimiseur va essayer de trouver l'index le plus restrictif des deux, en décidant quel index débouche sur le moins de lignes possibles. Si une table a un index multi−colonne, tout préfixe d'index peut être utilisé par l'optimiseur pour trouver des lignes. Par exemple, si vous avez un index à trois colonnes `(col1,col2,col3)` , vous pouvez faire des recherches accélérées sur les combinaisons de colonnes `(col1)` , `(col1,col2)` et `(col1,col2,col3)` .

MySQL ne peut utiliser d'index partiel sir les colonnes ne forment pas un préfix d'index. Supposez que vous avez la commande `SELECT` suivante :

```
 mysql> SELECT * FROM tbl_name WHERE col1=val1;
mysql> SELECT * FROM tbl_name WHERE col2=val2;
mysql> SELECT * FROM tbl_name WHERE col2=val2 AND col3=val3;
```

Si un index existe sur les colonnes `(col1,col2,col3)` , seule la première requête pourra utiliser l'index ci−dessus. Les deux autres requêtes utilisent des colonnes indexées, mais les colonnes `(col2)` et `(col2,col3)` ne font pas partie du préfixe des colonnes `(col1,col2,col3)` .

MySQL utilise aussi les index lors des comparaisons avec l'opérateur `LIKE` si l'argument de `LIKE` est une chaîne constante qui ne commence pas par un caractère joker. Par exemple, les requêtes `SELECT` suivantes utilisent des index :

```
 mysql> SELECT * FROM tbl_name WHERE key_col LIKE "Patrick%";
mysql> SELECT * FROM tbl_name WHERE key_col LIKE "Pat%_ck%";
```

Dans le premier exemple, seules les lignes avec `"Patrick" <= key_col < "Patricl"` sont considérées. Dans le second exemple, `"Pat" <= key_col < "Pau"` sont considérées.Les commandes `SELECT` suivantes n'utilisent pas d'index :

```
 mysql> SELECT * FROM tbl_name WHERE key_col LIKE "%Patrick%";
mysql> SELECT * FROM tbl_name WHERE key_col LIKE other_col;
```

Dans la première requête, la valeur associée à `LIKE` commence avec un caractère joker. Dans le second exemple, la valeur associée à `LIKE` n'est pas une valeur constante.MySQL 4.0 fait une autre optimisation avec l'opérateur `LIKE` . Si vous utilisez `... LIKE "%string%"` et que `string` est plus grand que 3 caractères, MySQL va utiliser l'algorithme `Turbo Boyer-Moore` qui prend une valeur initiale pour résoudre le masque, et l'exploite pour accélérer la recherche.

Les recherches qui utilisent la fonction `column_name IS NULL` vont utiliser les index si `column_name` sont des index.

MySQL normalement utilise l'index qui génère le moins de lignes possible. Un index est utilisé avec les colonnes que vous spécifiez, et les opérateurs suivants : `=` , `>` , `>=` , `<` , `<=` , `BETWEEN` et l'opérateur `LIKE` sans préfixe joker, c'est à dire de la forme `'quelquechose%'` .

Un index qui ne s'applique pas à tous les niveaux de `AND` dans une requête `WHERE` , ne sera pas utilisé pour optimiser la requête. En d'autres termes, pour être capable d'utiliser un index pour optimiser une requête, un préfixe de l'index doit être utilisé dans toutes les parties de la formule logique contenant `AND` .

Les clauses `WHERE` suivantes utilisent des index :

```
 ... WHERE index_part1=1 AND index_part2=2 AND other_column=3
... WHERE index=1 OR A=10 AND index=2      /* index = 1 OR index = 2 */
... WHERE index_part1='hello' AND index_part_3=5
          /* optimisé par "index_part1='hello'" */
... WHERE index1=1 and index2=2 or index1=3 and index3=3;
          /* peut utiliser un index sur index1 mais pas sur index2 ou index 3 */
```

Ces clauses `WHERE` n'utilisent **PAS** d'index :

```
 ... WHERE index_part2=1 AND index_part3=2  /* index_part_1 n'est pas utilisé */
... WHERE index=1 OR A=10                 /* Index n'est pas utilisé sur les deux parties du AND */
... WHERE index_part1=1 OR index_part2=10  /* Aucun index ne s'applique à toutes les colonnes */
```

Notez que dans certains cas, MySQL ne va pas utiliser un index, même si il en reste de disponible. Voici certains de ces cas :

- Si l'utilisation de l'index requiert que MySQL accède à plus de 30% des lignes de la table (dans ce cas, un scan de table est probablement plus rapide, et demandera moins d'accès disques). Notez que si une telle requête utilise la clause `LIMIT` pour ne lire qu'une partie des lignes, MySQL utilisera tout de même l'index, car il va trouver plus rapidement les quelques lignes de résultat.

## 5.4.4 Index de colonnes

Tous les types de colonnes ed MySQL peuvent être indexés. L'utilisation des index sur les colonnes pertinentes est la meilleur façon d'améliorer les performances de opérations de `SELECT` .

Le nombre maximum de clefs et la longueur maximale des index sont définis pour chaque type de table. Types de tables MySQL . Vous pouvez avec tous les gestionnaires de tables avoir au moins 16 clefs et une taille totale d'index d'au moins 256 octets.

Pour les colonnes `CHAR` et `VARCHAR` , il est possible d'indexer un préfixe de la colonne. C'est plus rapide et plus économe en espace disque que l'indexation de la colonne entière. La syntaxe pour indexer le début d'une colonne au moment de la création de la table ressemble à cela:

```
KEY nom_de_l_index (nom_de_la_colonne(longueur))
```

L'exemple suivant crée un index sur les 10 premiers caractères de la colonne `nom` :

```
mysql> CREATE TABLE test (
   ->        nom CHAR(200) NOT NULL,
   ->        KEY index_du_nom (nom(10)));
```

Pour les colonnes `BLOB` et `TEXT` , vous devez indexer le début de la colonne. Vous ne pouvez pas indexer une colonne entière.Dans les versions 3.23.23 et supérieures de MySQL, vous pouvez aussi créer les index spéciaux `FULLTEXT` . Ils sont utilisé pour les recherches sur les textes en entier. Seules les tables de type `MyISAM` supportent les index `FULLTEXT` . Il peuvent être créés depuis des colonnes `CHAR` , `VARCHAR` et `TEXT` . L'indexation se fait toujours par rapport à la colonne en entier, et l'indexation partielle n'est pas supportée. Voir Recherche full−text avec MySQL pour plus d'informations.

## 5.4.5 Index sur plusieurs colonnes

MySQL peut créer des index sur plusieurs colonnes. Un index peut comprendre jusqu'à 15 colonnes. (sur les colonnes de type `CHAR` ou `VARCHAR` , vous pouvez utiliser uniquement le début de la colonne pour l'indexation.)

Un index sur plusieurs colonnes peut être compris comme un tableau trié contenant des valeurs créées par concaténation des valeurs des colonnes indexées.

MySQL utilise les index sur plusieurs colonnes de telle sorte que les requêtes sont accélérées quand on spécifie une quantité connue de la première colonne de l'index dans un clause `WHERE` , même si on ne spécifie pas la valeur des autres colonnes.

On suppose qu'une table est créée avec les paramètres suivant:

```
mysql> CREATE TABLE test (
   ->       id INT NOT NULL,
   ->       nom CHAR(30) NOT NULL,
   ->       prenom CHAR(30) NOT NULL,
   ->       PRIMARY KEY (id),
   ->       INDEX nom_index (nom,prenom));
```

Alors l'index `nom_index` est un index de `nom` et de `prenom` . Cela sera utile pour les requêtes qui spécifient des valeurs dans une gamme donnée de `nom` , ou pour à la fois `nom` et `prenom` . Ainsi l'index `nom_index` sera utilisé pour les requêtes suivantes:

```
mysql> SELECT * FROM test WHERE nom="Widenius";

mysql> SELECT * FROM test WHERE nom="Widenius"
   ->                    AND prenom="Michael";

mysql> SELECT * FROM test WHERE nom="Widenius"
   ->                    AND (prenom="Michael" OR prenom="Monty");

mysql> SELECT * FROM test WHERE nom="Widenius"
   ->                    AND prenom >="M" AND prenom < "N";
```

Cependant, l'index `nom_index` ne sera PAS utilisé pour les requêtes suivantes:

```
mysql> SELECT * FROM test WHERE prenom="Michael";

mysql> SELECT * FROM test WHERE nom="Widenius"
   ->                    OR prenom="Michael";
```

Pour plus d'informations sur la méthode de MySQL pour utiliser les index dans le but d'améliorer les performance des requêtes, regardez Comment MySQL utilise les index .

## 5.4.6 Pourquoi tant de tables ouvertes ?

Quand vous utiliserez la commande `mysqladmin status` , vous verrez quelque chose de ce genre :

```
Uptime: 426 Running threads: 1 Questions: 11082 Reloads: 1 Open tables: 12
```

Cela vous laissera perplexe si vous n'avez que 6 tables.MySQL est multi−threadé, il peut donc exécuter plusieurs requêtes sur la même table simultanément. Pour minimiser les interférences entre deux threads ayant différentes actions sur le même fichier, la table est ouverte indépendamment par chacun des threads. Cela nécessite un peu de mémoire, mais augmente les performances. Avec les tables au format `ISAM` et `MyISAM` , cela requière aussi un fichier additionnel de description du fichier des données. Avec ce type de tables, le fichier décrivant l'index est partagé entre tous les threads.

Vous pourrez lire plus sur le sujet à la section suivante : Comment MySQL ouvre et ferme les tables .

## 5.4.7 Quand MySQL ouvre et ferme les tables

`table_cache` , `max_connections` et `max_tmp_tables` affectent le nombre maximum de tables que le serveur garde ouvertes. Si vous augmentez l'une de ces valeurs, vous pourriez rencontrer une des limites de votre système d'exploitation. Cependant, vous pourrez augmenter ces limites sur de nombreux systèmes d'exploitation. Consultez votre documentation système pour voir comment faire cela, car la méthode pour modifier la limite est différente pour chaque système.

`table_cache` est lié au `max_connections` . Par exemple, pour 200 connexions simultanées, vous devriez avoir un cache de table d'environ `200 * n` , où `n` est le nombre maximum de table dans une jointure. Vous devez aussi réserver des pointeurs de fichiers supplémentaires pour les tables temporaires et les fichiers.

Assurez vous que votre système d'exploitation peut gérer le nombre de pointeurs de fichiers demandé par l'option `table_cache` . Si `table_cache` est trop grand, MySQL peut être à court de pointeurs, et refuser des connexions, échouer à l'exécution de requêtes, ou être très instable. Vous devez aussi prendre en compte que les tables `MyISAM` peuvent avoir besoin de deux pointeurs de fichiers pour chaque table différente. Vous pouvez augmenter le nombre de pointeurs de fichiers disponibles pour MySQL avec l'option de démarrage `−−open−files−limit=#` . Fichier non trouvé .

Le cache de tables ouvertes reste au niveau de `table_cache` entrées (par défaut, 64; cela peut être modifié avec l'option `−O table_cache=#` de `mysqld` ). Notez que MySQL peut ouvrir temporairement plus de tables, pour être capable d'exécuter des requêtes.

Une table qui n'est pas utilisée est refermée, et supprimée du cache de table, dans les circonstances suivantes :

- Lorsque le cache est plein, et qu'un thread essaie d'ouvrir une table qui n'est pas dans le cache.
- Lorsque le cache contient plus de `table_cache` lignes, et qu'aucun thread n'utilise cette table.
- Lorsque quelqu'un utilise la commande `mysqladmin refresh` ou `mysqladmin flush−tables` .
- Lorsque quelqu'un exécute la commande `FLUSH TABLES` .

Lorsque le cache de table se remplit, le serveur utilise la procédure suivante pour identifier une entrée du cache, pour la supprimer :

- Les tables qui n'est pas en cours d'utilisation est libérée, en utilisant la table qui a été accédé depuis plus longtemps en premier.
- Si le cache est plein, et qu'aucune table ne peut être libérée, mais qu'une nouvelle table doit être ouverte, le cache est temporairement étendu.
- Si le cache est dans un état d'extension, et qu'une table passe de l'état d'utilisation à non utilisation, la table est immédiatement fermée et libérée du cache.

Une table est ouverte pour chaque accès simultané. Cela signifie que si vous avez deux threads qui accèdent à la même table, ou accèdent à la même table deux fois dans la requête (avec `AS` ), la table devra être ouverte deux fois. La première ouverture d'une table prendre deux pointeurs de fichiers. Chaque utilisation supplémentaire de la même table ne prendra qu'un pointeur supplémentaire. Le pointeur de fichier supplémentaire de la première table est celui du fichier d'index. Ce pointeur est partagé entre les threads.Si vous ouvrez une table avec `HANDLER table_name OPEN` , un objet de table dédié sera alloué pour le thread. Cet objet de table n'est pas partagé avec les autres threads, et il ne sera pas fermé avant que le thread n'appelle `HANDLER table_name CLOSE` , ou que le thread ne meurt. Syntaxe des `HANDLER` . Lorsque cela arrive, la table est placée dans le cache de table (si il n'est pas plein).

Vous pouvez vérifier si votre cache de table n'est pas trop petit en vérifiant la variable de `mysqld` appelée `Opened_tables` . Si cette valeur est grande, même si vous n'avez pas trop abusé de la commande `FLUSH TABLES` , vous devrez augmenter la taille du cache. `SHOW STATUS` .

### 5.4.8 Inconvénients de la création d'un grand nombre de tables dans la même base de données

Si vous avez beaucoup de fichiers dans un dossier, les opérations d'ouverture, fermeture, et création seront ralenties. Si vous exécutez une requête `SELECT` sur plusieurs tables, il y aura une légère perte lorsque le cache de tables sera plein, car pour chaque table ouverte, une autre doit être fermée. Vous pouvez réduire cette table en augmentant la taille du cache de tables.

## 5.5 Optimisation du serveur MySQL

### 5.5.1 Réglage du système, au moment de la compilation, et paramètres du démarrage

Nous démarrons par le niveau du système, car certaines décisions à ce niveau doivent être prises très tôt. Dans d'autres cas, un regard rapide à cette partie doit suffire, car ce n'est pas tellement important pour les gros gains. Toute fois, il est toujours sympathique de sentir combien on peut gagner en changeant des choses à ce niveau.

Le choix du système d'exploitation est vraiment important! Pour utiliser au maximum les capacités de machines multi−processeurs, il vaut mieux choisir Solaris (car les threads marchent vraiment très bien) ou Linux (car le noyau 2.2 supporte très bien le SMP). Mais les plate−formes Linux 32 bits limitent par défaut la taille des fichiers à 2 Go. Heureusement, cela sera bientot réparé avec l'arrivée des nouveaux systèmes de fichier (XFS/Reiserfs). Si vous souhaitez désespéremment utiliser des fichiers de plus de 2 Go sur Linux−intel 32bits, vous devriez utiliser le patch de LFS pour le système de fichier ext2

Comme nous n'avons pas utilisé MySQL en production sur énormément de plate−formes, nous vous conseillons de tester votre plate−forme avant de la choisir définitivement.

Autres astuces:

- Si vous avez suffisamment de RAM, vous pouvez supprimer toutes les partitions d'échange (swap). Certains systèmes d'exploitation utilisent parfois la partition d'échange quand bien même il reste de la mémoire libre.
- L'utilisation de l'option `--skip-external-locking` de MySQL empêche les verrous externes. Cela n'influencera pas les fonctionnalités de MySQL tant que vous n'utilisez qu'un seul serveur. Il faut cependant penser à arreter le serveur (ou bien de verrouiller les parties pertinantes) avant d'utiliser `myisamchk` . Sur certains systèmes, cette option est inutile car les verrous externes ne fonctionnent pas du tout.

  L'option `--skip-external-locking` est activée par défaut quand on compile avec MIT−pthreads, car `flock()` n'est pas totalement supporté sur toutes les plate−formes par MIT−pthreads. Elle l'est également sur Linux, cat le verrouillage des fichiers de Linux n'est pas encore sûr.

  Les seuls cas où on ne peut pas utiliser `--skip-external-locking` sont si on utilise plusieurs **servers** (pas de clients) MySQL sur les mêmes données, ou si on lance `myisamchk` sur une table sans vider son tampon et sans la verrouiller au préalable.Il est toujours possible d'utiliser `LOCK TABLES` / `UNLOCK TABLES` même si vous utilisez `--skip-external-locking` .

## 5.5.2 Réglage des paramètres du serveur

Vous pouvez obtenir les tailles par défaut des tampons du serveur `mysqld` avec la commande:

```
shell> mysqld --help
```

Cette commande génère une liste de toutes les options de `mysqld` et des variables configurables. Cette sortie comprend les valeurs par défaut et ressemble à cela:

```
 Possible variables for option --set-variable (-O) are:
back_log                 current value: 5
bdb_cache_size           current value: 1048540
binlog_cache_size        current value: 32768
connect_timeout          current value: 5
delayed_insert_timeout   current value: 300
delayed_insert_limit     current value: 100
delayed_queue_size       current value: 1000
flush_time               current value: 0
interactive_timeout      current value: 28800
join_buffer_size         current value: 131072
key_buffer_size          current value: 1048540
lower_case_table_names   current value: 0
long_query_time          current value: 10
max_allowed_packet       current value: 1048576
max_binlog_cache_size    current value: 4294967295
max_connections          current value: 100
max_connect_errors       current value: 10
max_delayed_threads      current value: 20
max_heap_table_size      current value: 16777216
max_join_size            current value: 4294967295
max_sort_length          current value: 1024
max_tmp_tables           current value: 32
max_write_lock_count     current value: 4294967295
myisam_sort_buffer_size  current value: 8388608
net_buffer_length        current value: 16384
net_retry_count          current value: 10
net_read_timeout         current value: 30
net_write_timeout        current value: 60
read_buffer_size         current value: 131072
record_rnd_buffer_size   current value: 131072
slow_launch_time         current value: 2
sort_buffer              current value: 2097116
table_cache              current value: 64
thread_concurrency       current value: 10
tmp_table_size           current value: 1048576
thread_stack             current value: 131072
wait_timeout             current value: 28800
```

Il est à noter que `--set-variable` n'est plus utile depuis MySQL 4.0, et on peut utiliser `--var=option` directement.

Si un serveur `mysqld` est en cours d'exécution, vous pouvez voir les valeurs que les variables utilisent réellement en exécutant la commande :

```
shell> mysqladmin variables
```

Vous pouvez trouver une description complète de toutes les variables dans la section `SHOW VARIABLES` de ce manuel. `SHOW VARIABLES` .

Vous pouvez aussi voir des statistiques sur un serveur en cours d'exécution en utilisant la commande `SHOW STATUS` . `SHOW STATUS` .

MySQL utilise des algorithmes très extensibles, donc vous pouvez utiliser très peu de mémoire. Si malgré tout vous fournissez plus de mémoire à MySQL, vous obtiendrez également de meilleures performances.

Les deux variables les plus importantes au moment du réglage d'un serveur MySQL sont `key_buffer_size` et `table_cache` . Vous devriez vous assurer que celles sont sont bien paramétrées avant de modifier les autres variables.

Si vous avez beaucoup de mémoire (>=256Mo) et beaucoup de tables, et que vous désirez des performances maximales avec un faible de nombre de clients, vous devriez essayer quelque chose cela:

```
shell> safe_mysqld -O key_buffer=64M -O table_cache=256 \
          -O sort_buffer=4M -O read_buffer_size=1M &
```

Si vous n'avez que 128Mo et seulement quelques tables, mais que vous demandez beaucoup de classements, vous pouvez essayer cela :

```
shell> safe_mysqld -O key_buffer=16M -O sort_buffer=1M
```

Si vous avez peu de mémoire et beaucoup de connections, essayez cela:

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=100k \
          -O read_buffer_size=100k &
```

Ou encore:

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=16k \
          -O table_cache=32 -O read_buffer_size=8k -O net_buffer_length=1K &
```

Si vous utilisez `GROUP BY` ou `ORDER BY` sur des fichiers de taille supérieure à la mémoire disponible, vous devriez augmenter la valeur de `record_rnd_buffer` pour accélérer la lecture des lignes après que le classement ait été fait.

A l'installation de MySQL, un répertoire `support-files` est créé, et contient plusieurs exemples de fichiers `my.cnf` : `my-huge.cnf` , `my-large.cnf` , `my-medium.cnf` et `my-small.cnf` . Vous pouvez les utiliser comme base pour optimiser votre système.

Si vous avez vraiment beaucoup de connections, des problèmes peuvent apparaître avec le fichier d'échange si `mysqld` n'a pas été configuré pour utiliser peu de mémoire pour chaque connexion. `mysqld` fonctionne mieux si vous avec suffisamment de mémoire pour toutes les connections, bien sûr !

Notez que si vous changez une option de `mysqld` , elle ne prendra effet qu'au prochain démarrage du serveur.

Pour voir les effets d'un changement de paramètre, essayez quelque chose comme ca:

```
shell> mysqld -O key_buffer=32m --help
```

Assurez vous bien de la présence de l'option `--help` en fin de ligne; si ce n'est pas le cas, les options listées après dans la ligne de commande ne seront pas prises en compte à la sortie.

## 5.5.3 Influences de la compilation et des liaisons sur la vitesse de MySQL

La plupart des tests suivants ont été réalisés sous Linux avec les outils comparatifs de MySQL, mais ils peuvent donner quelques indications pour d'autres systèmes d'exploitation et sur une charge de travail différente.

Les exécutables les plus rapides sont obtenus en liant avec `-static` .

Sur Linux, le code le plus rapide sera obtenu en compilant avec `pgcc` et `-O3` . Pour compiler `sql_yacc.cc` avec ces options, il faut environ 200 Mo de mémoire car `gcc/pgcc` demande beaucoup de mémoire pour créer toutes les fonctions d'une traite. Il est aussi possible d'utiliser `CXX=gcc` à la configuration de MySQL pour éviter l'inclusion de la bibliothèque `libstdc++` (qui n'est pas nécessaire). Sachez que pour certaines versions de `pgcc` , le code résultant ne fonctionnera que sur de vrais processeurs Pentium, même si vous utilisez l'option du compilateur qui doit génerer du code fonctionnant sur tour les types de processeurs x586 (comme AMD).

L'utilisation du meilleur compilateur et/ou de la meilleur option de compilation permet de gagner 10 à 30% de vitesse dans vos applications. C'est très important quand vous compilez le serveur SQL vous−même !

Nous avons compilé avec les compilateurs de Cygnus CodeFusion et de Fujitsu, mais aucun des deux n'était suffisamment exempt d'erreurs pour permettre la compilation de MySQL avec l'optimisation.

A la compilation de MySQL, vous devriez uniquement utiliser le support des caractères que vous allez utiliser. (Option `--with-charset=xxx` .) Les distributions binaires standards de MySQL sont compilées avec le support de toutes les gammes de caractères.

Voici une liste des mesures que nous avons effectués:

- L'utilisation de `pgcc` et la compilation complète avec l'option `-O6` donne un serveur `mysqld` 1% plus rapide qu'avec `gcc` 2.95.2.
- Si vous utilisez la liaison dynamique (sans `-static` ), le résultat est 13% plus lent sur Linux. Sachez que vous pouvez néanmoins utiliser la liaison dynamique pour les bibliothèques de MySQL. Seul le serveur a des performances critiques.
- Si vous allégez votre binaire `mysqld` avec l'option `strip libexec/mysqld` , vous obtenez un binaire jusqu'à 4% plus rapide.
- Si vous utilisez TCP/IP plutôt que les "sockets" Unix, le résultat est 7.5% plus lent sur le même ordinateur. (Si vous vous connectez sur `localhost` , MySQL utilisera les sockets par défaut.)
- Si vous vous connectez en TCP/IP depuis un autre ordinateur avec un lien Ethernet 100M, le résultat sera 8 à 11% plus lent.
- L'utilisation de connections sécurisées ( toutes les données chiffrées par le support interne de SSL) pour nos tests comparatifs a provoqué une perte de vitesse de 55%.
- Si vous compilez avec `--with-debug=full` , vous perdrez 20% de performances sur la plupart des requêtes, mais la perte peut être plus importante sur certaines requêtes (La suite de tests de MySQL tourne 35% plus lentement). Si vous utilisez `--with-debug` , vous ne perdrez que 15%. En démarrant une version de `mysqld` , compilée avec `--with-debug=full` , avec with `--skip-safemalloc` , le résultat final devrait être proche d'une compilation avec `--with-debug` .
- Sur un Sun UltraSPARC−IIe, Forte 5.0 est 4% plus rapide que `gcc` 3.2.
- Sur un Sun UltraSPARC−IIe, Forte 5.0 est 4% plus rapide en mode 32 bit qu'en mode 64 bit.
- La compilation avec `gcc` 2.95.2 sur UltraSPARC avec l'option `-mcpu=v8 -Wa,-xarch=v8plusa` améliore les performances de 4%.

- Sur Solaris 2.5.1, MIT−pthreads est 8−12% plus lent que la gestion native des threads de Solaris sur mono−processeur. Avec plus de charge ou de CPU, la différente devrait être encore plus grande.
- Le fonctionnement avec `--log-bin` rend mysqld 1% plus lent.
- La compilation sur Linux−x86 avec gcc sans les pointeurs `-fomit-frame-pointer` ou `-fomit-frame-pointer -ffixed-ebp` rend `mysqld` 1 à 4% plus rapide.

Autrefois les distributions fournies par MySQL AB de MySQL−Linux étaient compilées avec `pgcc` , mais nous avons dû revenir au simple gcc à cause d'un bogue dans `pgcc` qui génerait du code qui ne fonctionnait pas sur AMD. Nous continuerons à utiliser gcc tant que ce bogue ne sera pas corrigé. Néanmoins, si vous avez une machine non−AMD, vous pouvez obtenir des binaires plus rapides en compilant avec `pgcc` . Le binaire standard de MySQL pour Linux est lié statiquement pour être plus rapide et plus portable.

## 5.5.4 Comment MySQL gère la mémoire

La liste suivante indique certaines techniques utilisées par le serveur `mysqld` pour gérer la mémoire. Lorsque c'est possible, la variable serveur liée à la mémoire est indiquée :

- Le buffer de clés (variable `key_buffer_size` ) est partagé par tous les threads. Les autres buffers sont sont alloués par le serveur suivant les besoins. Choix des paramètres du serveur .
- Chaque connexion utilise un espace spécifique au thread : une pile (par défaut, 64ko, variable `thread_stack` ), un buffer de connexion (variable `net_buffer_length` ) et un buffer de résultat (variable `net_buffer_length` ). Le buffer de connexion et celui de résultat sont dynamiquement élargit jusqu'à `max_allowed_packet` suivant les besoins. Lorsque la requête s'exécute, une copie de la chaîne de requête est aussi allouée.
- Tous les threads partagent la même mémoire de base.
- Seules les tables compressées `ISAM` / `MyISAM` sont copiées en mémoire. Ceci est dû au fait que pour un espace de 32 bits, il n'y a pas de place pour les grosses tables en mémoire. Lorsque les systèmes de 64 bits seront plus répandus, nous pourrions généraliser le support pour la copie en mémoire.
- Chaque requête qui effectue une analyse séquentielle d'une table, alloue un buffer de lecture (variable `record_buffer` ).
- Lors de la lecture de lignes en ordre 'aléatoire' (par exemple, après un tri), un buffer de lecture aléatoire est allouée pour éviter les accès disques (variable `record_rnd_buffer` ).
- Toutes les jointures sont faîtes en une seule passe, et la plupart des jointure sont faîtes sans utiliser de table temporaire. La plupart des table temporaires sont faîtes en mémoire (table `HEAP` ). Les tables temporaires avec beaucoup de données (calculées comme la somme des tailles de toutes les colonnes) ou qui contiennent des colonnes de type `BLOB` sont sauvées sur le disque.

  Un problème avec les versions de MySQL antérieures à la version 3.23.2 est que si une table `HEAP` dépassait la taille maximale de `tmp_table_size` , vous obteniez une erreur `The table tbl_name is full` . Dans les nouvelles versions, ce problème est géré en passant automatiquement la table `HEAP` en une table `MyISAM` sur le disque. Pour contourner ce problème, vous pouvez augmenter la taille maximale des tables en mémoire en modifiant l'option `tmp_table_size` de `mysqld` , ou en modifiant l'option SQL `BIG_TABLES` dans le programme client. Syntaxe de `SET` . En MySQL version 3.20, la taille maximale de la table temporaire est `record_buffer*16` , ce qui fait que si vous utilisez cette version, vous aurez à augmenter la valeur de `record_buffer` . Vous pouvez aussi démarrer `mysqld` avec l'option `--big-tables` pour toujours stocker les tables temporaires sur le disque. Cependant, cela va affecter la vitesse de votre serveur pour les requêtes complexes.
- La plupart des requêtes qui sont triées allouent un buffer de tri, et entre 0 et 2 fichiers

temporaires, suivant la taille du résultat. Où MySQL conserve les fichiers temporaires .

- Toute l'analyse et les calculs sont faits en mémoire locale. Aucune mémoire supplémentaire n'est nécessaire pour les petits calculs, et les allocations et libérations de mémoire sont évités. La mémoire n'est allouée que pour les chaînes très grandes (ceci se fait via `malloc()` et `free()` ).
- Chaque fichier d'index est ouvert une fois, et le fichier de données est ouvert pour chaque thread concurrent. Pour chaque thread concurrent, une structure de table, une structure de colonne pour chaque colonne et un buffer de taille `3 * n` est alloué (où `n` est la taille maximale de ligne, en dehors des colonnes de type `BLOB` ). Une colonne de type `BLOB` utilise 5 à 8 octets de plus que la taille des données du `BLOB` . Les gestionnaires de table `ISAM` / `MyISAM` utilisent un buffer d'une ligne de plus pour leur utilisation interne.
- Pour chaque table qui a une colonne `BLOB` , un buffer est dynamiquement agrandi pour lire les valeurs `BLOB` . Si vous analysez toute une table, un buffer aussi grand que la plus grande valeur de la colonne `BLOB` sera alloué.
- Les gestionnaires de tables pour les tables en cours d'utilisation sont sauvées dans un cache, et géré comme une pile FIFO. Normalement, ce cache contient 64 lignes. Si une table doit être utilisée par deux threads concurrents simultanément, le cache contiendra deux entrées pour la table. Comment MySQL ouvre et ferme les tables .
- La commande `mysqladmin flush-tables` ferme toute les tables qui ne sont pas utilisées, et marque toutes les tables en cours d'utilisation pour qu'elles soient fermées dès la fin du thread. Cela va libérer l'essentiel de la mémoire utilisée.

`ps` et d'autres commandes de statut système peuvent indiquer que `mysqld` utilise beaucoup de mémoire. Ceci est peut être dû a des erreurs de comptabilité. Par exemple, sous Solaris, `ps` compte la mémoire inutilisée entre les threads comme de la mémoire utilisée. Vous pouvez le vérifier en regardant l'état de la swap avec `swap -s` . Nous avons testé `mysqld` avec les détecteurs de fuite mémoire commerciaux, et il n'y a aucune fuite.

## 5.5.5 Comment MySQL utilise le DNS

Quand un nouveau thread se connecte à `mysqld` , `mysqld` crée nouveau thread pour traiter la requête. Ce thread contrôle d'abord si le nom de l'hôte est dans le cache des noms d'hôte. Si ce n'est pas le cas, le thread va appeler `gethostbyaddr_r()` et `gethostbyname_r()` pour résoudre le nom de l'hôte.

Si le système d'exploitation ne supporte pas les appels précédents, le thread va verrouiller un "mutex" et appeler `gethostbyaddr()` et `gethostbyname()` à la place. Sachez que dans ce cas, aucun autre thread ne peut résoudre de nom d'hôte qui n'est pas dans le cache tant que le premier thread n'a pas fini.

Il est possible de désactiver la recherche du nom par DNS en démarrant `mysqld` avec l'option `--skip-name-resolve` . Dans ce cas, il est toujours possible d'utiliser les adresses IP dans les tables de privilèges de MySQL.

Si votre service DNS est très lent et que vous avez beaucoup d'hôtes, vous pouvez améliorer les performances soit en désactivant le DNS avec `--skip-name-resolve` , soit en augmentant la taille de `HOST_CACHE_SIZE` (par défaut: 128) et en recompilant `mysqld` .

Il est possible de désactiver le cache de noms d'hôte avec `--skip-host-cache` . Il est possible de vider le cache des noms d'hôtes avec `FLUSH HOSTS` ou avec `mysqladmin flush-hosts` .

Si vous ne voulez pas autoriser les connections par `TCP/IP` , vous pouvez utiliser l'option `--skip-networking` au démarrage de `mysqld` .

## 5.5.6 Syntaxe de SET

```
SET [GLOBAL | SESSION] variable_sql=expression, [[GLOBAL | SESSION] variable_sql=expression...]
```

`SET` permet de configurer plusieurs options qui affectent le comportement de votre serveur ou de votre client.

Les exemples suivants montrent les différentes syntaxes qu'on peut utiliser pour configurer des variables :

Dans les anciennes versions de MySQL nous avons permis l'utilisation de la syntaxe `SET OPTION` , mais celle−ci est à présent désapprouvée.

Dans la version 4.0.3 de MySQL nous avons ajouté les options `GLOBAL` et `SESSION` et l'accès aux variables de configuration les plus importantes.

`LOCAL` peut être utilisé en tant que synonyme de `SESSION` .

Si vous configurez plusieurs variables sur une seule ligne de commande, le dernier mode `GLOBAL | SESSION` utilisé est pris en compte.

```
 SET sort_buffer_size=10000;
SET &#64;&#64;local.sort_buffer_size=10000;
SET GLOBAL sort_buffer_size=1000000, SESSION sort_buffer_size=1000000;
SET &#64;&#64;sort_buffer_size=1000000;
SET &#64;&#64;global.sort_buffer_size=1000000, &#64;&#64;local.sort_buffer_size=1000000;
```

La syntaxe `@@nom_variable` est supportée pour rendre MySQL compatible avec d'autres bases de données.

Le différentes variables système qu'on peut configurer sont décrites dans la section variables système de ce manuel. Variables système .

Si vous utilisez `SESSION` (par défaut) l'option que vous configurez garde son effet jusqu'à ce que la session courante se termine, ou que vous modifiez à nouveau cette option. Si vous utilisez `GLOBAL` , qui requière le privilège `SUPER` , l'option est gardée en mémoire et utilisée pour les nouvelles connexion jusqu'au redémarrage du serveur. Si vous voulez qu'un changement reste permanent, vous devez l'effectuer dans l'un des fichiers d'options de MySQL. Fichier d'options `my.cnf` .

Pour éviter un mauvais usage, MySQL donnera une erreur si vous utilisez `SET GLOBAL` avec une variable qui ne peut être inutilisée que par `SET SESSION` ou si vous n'utilisez pas `SET GLOBAL` avec une variable globale.

Si vous voulez configurer une variable `SESSION` à une valeur `GLOBAL` ou une valeur `GLOBAL` à la valeur par défaut de MySQL, vous pouvez la configurer à `DEFAULT` .

```
SET max_join_size=DEFAULT;
```

Ceci est identique à :

```
SET &#64;&#64;session.max_join_size=&#64;&#64;global.max_join_size;
```

Si vous voulez poser une limite sur la valeur maximale qu'une valeur de variable peut prendre avec la commande `SET` , vous pouvez le spécifier en utilisant l'option de ligne de commande `−−maximum−nom−variable` . Options en ligne de commande de `mysqld` .Vous pouvez obtenir une liste de la plupart des variables avec `SHOW VARIABLES` . `SHOW VARIABLES` . Vous pouvez obtenir la valeur d'une variable spécifique avec la syntaxe `@@[global.|local.]nom_variable` .

```
 SHOW VARIABLES like "max_join_size";
SHOW GLOBAL VARIABLES like "max_join_size";
SELECT &#64;&#64;max_join_size, &#64;&#64;global.max_join_size;
```

Vous trouverez ici une description des variables qui utilisent une syntaxe non−standard de `SET` . Les définitions des autres variables peuvent être trouvées dans la section des variables système, avec les options de démarrage ou dans la description de `SHOW VARIABLES` . Variables système . Options en ligne de commande de `mysqld` . `SHOW VARIABLES` .

*CHARACTER SET nom_jeu_de_caractères DEFAULT*

Cela change le jeu de caractère dans toutes les chaînes du et vers le client avec le jeu donné. Jusqu'à maintenant, la seule option pour `nom_jeu_de_caractères` est `cp1251_koi8` , mais vous pouvez facilement ajouter d'autres possibilités en éditant le fichier `sql/convert.cc` dans la distribution des sources MySQL. Le jeu de caractères par défaut peut être restauré en utilisant la valeur `DEFAULT` de `nom_jeu_de_caractères DEFAULT` .

Notez que la syntaxe pour configurer l'option `CHARACTER SET` diffère de la syntaxe pour configurer les autres options.

*PASSWORD = PASSWORD('un mot de passe')*

Change le mot de passe pour l'utilisateur courant. Chaque utilisateur non anonyme peut changer son propre mot de passe !

*PASSWORD FOR utilisateur = PASSWORD('un mot de passe')*

Configure un mot de passe pour un utilisateur spécifique sur le serveur courant. Seul un utilisateur avec l'accès à la base de données `mysql` peut faire cela. L'utilisateur doit être donné au format `utilisateur@hote` , où `utilsateur` et `hote` sont exactement comme ils sont listés dans les colonnes `User` et `Host` des entrées de la table `mysql.user` . Par exemple, si vous avez une entrée avec dans les champs `User` et `Host` les entrées `'bob'` et `'%.loc.gov'` , vous écrirez :

```
mysql> SET PASSWORD FOR bob&#64;"%.loc.gov" = PASSWORD("nouveaupass");
```

Ce qui est équivalent à :

```
mysql> UPDATE mysql.user SET password=PASSWORD("nouveaupass")
   ->                  WHERE user="bob' AND host="%.loc.gov";
```

*SQL_AUTO_IS_NULL = 0 | 1*

Si définie à `1` (par défaut) alors on peut trouver la dernière ligne insérée dans une table avec une colonne `AUTO_INCREMENT` avec la construction suivante : `WHERE auto_increment_column IS NULL` . Ceci est utilisé par des programmes ODBC tel que Access.

*AUTOCOMMIT= 0 | 1*

Si définie à `1` tous les changements dans une table se feront en une seule fois. Pour démarrer une transaction multi−commandes, vous devez utiliser la commande `BEGIN` . Syntaxe des `BEGIN/COMMIT/ROLLBACK` . Si définie à `0` vous devez utiliser `COMMIT` / `ROLLBACK` pour accepter/annuler cette transaction. Syntaxe des `BEGIN/COMMIT/ROLLBACK` . Notez que quand vous passez du mode non `AUTOCOMMIT` vers le mode `AUTOCOMMIT` , MySQL fera un `COMMIT` automatique sur toutes les transactions en cours.

*BIG_TABLES = 0 | 1*

Si définie à `1` , toutes les tables temporaires sont stockées sur le disque plutôt qu'en mémoire. Cela sera un peu plus lent, mais vous n'obtiendrez jamais l'erreur `The table nom_de_table is full` pour les grands `SELECT` qui requièrent une table temporaire. La valeur par défaut pour une nouvelle connexion est `0` (qui est d'utiliser la mémoire pour les tables temporaires). Cette option se nommait avant `SQL_BIG_TABLES` .

*SQL_BIG_SELECTS = 0 | 1*

Si configuré à `0` , MySQL interrompra les requêtes `SELECT` qui prendront probablement trop de temps. C'est utile lorsqu'une clause `WHERE` déconseillée a été utilisée. Une grosse requête est définie comme étant un `SELECT` qui devra probablement étudier plus de `max_join_size` lignes. La valeur par défaut d'une nouvelle connexion est `1` (qui permet toutes les requêtes `SELECT` ).

*SQL_BUFFER_RESULT = 0 | 1*

`SQL_BUFFER_RESULT` forcera les résultats des requêtes `SELECT` à être placés dans une table temporaire. Cela aidera MySQL à libérer les verrous sur table plus tôt et améliorera les cas où le jeu de résultats de la requête prend trop de temps à être envoyée au

client.

*LOW_PRIORITY_UPDATES = 0 | 1*

Si vous configurez à `1` , toutes les commandes `INSERT` , `UPDATE` , `DELETE` , et `LOCK TABLE WRITE` attendent qu'il n'y ai plus de `SELECT` ou `LOCK TABLE READ` sur la table affectée. Cette option s'appelait avant `SQL_LOW_PRIORITY_UPDATES` .

*MAX_JOIN_SIZE = valeur | DEFAULT*

Ne pas permettre les requêtes `SELECT` qui auront besoin d'examiner plus de `valeur` combinaisons de lignes. En configurant cette valeur, vous pouvez capturer les requêtes `SELECT` où les clefs sont mal utilisées et qui prendront probablement beaucoup de temps à s'exécuter. Prendre une autre valeur que `DEFAULT` mettre à zéro l'option `SQL_BIG_SELECTS` . Si vous reconfigurez l'option `SQL_BIG_SELECTS` , la variable `SQL_MAX_JOIN_SIZE` sera ignorée. Vous pouvez configurer une valeur par défaut pour cette variable en démarrant `mysqld` avec `-O max_join_size=#` . Cette option s'appelait avant `SQL_MAX_JOIN_SIZE` .

Notez que si le résultat de la requête est déjà dans le cache de requêtes, le test précédent ne sera pas effectué. A la place, MySQL renverra le résultat au client. Puisque le résultat de la requête est déjà traité et qu'il sera aisé au serveur d'envoyer la réponse au client.

*QUERY_CACHE_TYPE = OFF | ON | DEMAND*

*QUERY_CACHE_TYPE = 0 | 1 | 2*

Configure le cache de requêtes pour ce processus.

| Option | Description |
|---|---|
| 0 ou OFF | Ne pas cacher ou récupérer les résultats. |
| 1 ou ON | Met tous les résultats en cache à part les requêtes SELECT SQL_NO_CACHE ... . |
| 2 ou DEMAND | Ne met en cache que les requêtes SELECT SQL_CACHE ... . |

*SQL_SAFE_UPDATES = 0 | 1*

Si définit à `1` , MySQL annulera si un `UPDATE` ou un `DELETE` est exécuté alors qu'il n'utilise pas de clef ou de `LIMIT` dans la clause `WHERE` . Cela permet de bloquer les requêtes erronées crées à la main.

*SQL_SELECT_LIMIT = valeur | DEFAULT*

Le nombre maximal des enregistrements que doivent retourner les requêtes `SELECT` . Si un `SELECT` possède une clause `LIMIT` , celle-ci est utilisée. La valeur par défaut pour une nouvelle connexion est ``illimitée.'' Si vous avez changé la limite, la valeur par défaut peut être retrouvée en utilisant la valeur `DEFAULT` avec `SQL_SELECT_LIMIT` .

*SQL_LOG_OFF = 0 | 1*

Si définie à `1` , aucune entrée ne sera écrite dans le log standard pour ce client, si le client à le privilège `SUPER` . Cela n'affecte pas le log des mises à jours !

*SQL_LOG_UPDATE = 0 | 1*

Si définie à `0` , aucune trace des requêtes ne sera gardée dans le log des mises à jour pour le client, si le client à le privilège `SUPER` . Cela n'affecte pas le log standard !

*SQL_QUOTE_SHOW_CREATE = 0 | 1*

Si vous le configurez à `1` , `SHOW CREATE TABLE` protégera les noms de tables et de colonnes. Ceci est **activé** par défaut, pour que la réplication des tables avec des noms à risques fonctionne. `SHOW CREATE TABLE` .

*TIMESTAMP = valeur_timestamp | DEFAULT*

Configure le temps pour ce client. C'est utilisé pour obtenir le timestamp d'origine si vous utilisez le log de mises à jour pour restaurer des lignes. `valeur_timestamp` doit être un timestamp Unix, et non un timestamp MySQL.

*LAST_INSERT_ID = #*

Configure la valeur qui doit être retournée par `LAST_INSERT_ID()` . C'est enregistré dans le log de mises à jour quand vous utilisez `LAST_INSERT_ID()` dans une commande qui met à jour une table.

*INSERT_ID = #*

Configure la valeur à utiliser par l'appel suivant à la commande `INSERT` ou `ALTER TABLE` lors de l'insertion d'une valeur `AUTO_INCREMENT` . Cela est souvent utilisé par le log des mises à

jour.

## 5.6 Problèmes avec les disques

- Comme mentionné plus tôt, les accès disques représentent une limitation. Ce problème devient de plus en plus apparent, au fur et à mesure que les données sont de plus en plus nombreuses, et que les techniques de cache deviennent impossibles. Pour les grandes bases de données, lorsque vous accédez aux données plus ou moins aléatoirement, vous pouvez être sûr que vous aurez besoin d'un accès disque pour lire, et de plusieurs autres pour écrire. Pour minimiser le problème, utilisez des disques avec des temps d'accès très faibles.
- Augmentez le nombre de disque disponibles (et donc, réduisez le coût d' un accès), en plaçant des données sur d'autres fichiers via des liens symboliques.
- Pour plus de confiance, vous pouvez utiliser des disques en RAID 0+1 (parallélisme et réplication), mais dans ce cas, vous aurez besoin de 2*N disques pour contenir vos données sur N disques. C'est probablement l'option la plus sûre, si vous avez le budget pour cela. Vous risquez aussi d'avoir à investir dans un système de gestion de gros volume de données pour gérer cela efficacement.
- Une bonne option est de garder les données semi−importantes (qui peuvent être regénérées) sur un disque RAID 0 tandis que les données vraiment importantes (comme les informations d'hôtes et les log) sur un disque de type RAID 0+1 ou RAID N. RAID N peut être un problème si vous avez de nombreux accès en écrire, à cause du temps de modification des bits de parité.
- Vous pouvez aussi optimiser les paramètres du système de fichier que la base de données utilise. Une modification simple est de monter le système de fichier avec l'option noatime. Cela fait que le système ne fait plus la modification de la date de dernier accès à l'inode, ce qui évite des accès disques.
- Sous Linux, vous pouvez améliorer les performances (jusqu'à 100% en charge n'est pas difficile) en utilisant `hdparm` pour configurer votre interface disque. La commande suivante doit être une série de bonnes options de `hdparm` pour MySQL (et probablement d'autres applications) :

  ```
  hdparm −m 16 −d 1
  ```

  Notez que la performances et la robustesse des solutions ci−dessus dépendent de votre matériel, et nous vous conseillons vivement de tester votre système soigneusement après avoir utilisé `hdparm` ! Consultez le manuel de `hdparm` pour plus de détails. Si `hdparm` n'est pas utilisé correctement, le système de fichiers peut être corrompu. Sauvegardez tout avant d'expérimenter.
- Sur de nombreux systèmes d'exploitation, vous pouvez monter des disques avec l'option `−o async` pour que le système de fichiers soit modifié de manière asynchrone. Si votre serveur est raisonnablement stable, vous devriez obtenir de bonne performances sans sacrifier la stabilité (cette option est activée par défaut sur Linux).
- Si vous n'avez pas besoin de savoir quand un fichier a été accédé la dernière fois (ce qui n'est pas utile avec un serveur de base de données), vous pouvez monter votre système de fichier avec l'option `−o noatime` .

### 5.6.1 Utiliser des liens symboliques

Vous pouvez déplacer les dossiers de bases de données et les placer dans un autre endroit, puis remplacer les dossiers eux−mêmes par des liens symboliques vers ces autres endroits. Vous pourriez vouloir faire cela pour mettre la base de données sur un système de fichier plus rapide, ou pour gagner de l'espace disque sur le système central, ou encore répartir vos tables sur différents disques.

Le mieux, pour cela, est de faire des liens symboliques des bases vers les différents disques, et de ne faire des liens symboliques sur les tables qu'en dernier ressort.

#### 5.6.1.1 Utiliser les liens symboliques pour les bases

Pour créer des liens symboliques sur les bases de données, vous devez commencer par créer un dossier sur un disque de destination, puis faire un lien symbolique depuis le dossier de données vers votre dossier de destination.

```
shell> mkdir /dr1/databases/test
shell> ln -s /dr1/databases/test mysqld-datadir
```

MySQL n'accepte pas que vous fassiez le lien depuis plusieurs bases sur le même dossier. Remplacer une base par un lien symbolique sera correct tant que vous n'essayez pas de faire des liens symboliques dans la même base. Supposez que vous la base `db1` dans le dossier de données MySQL, puis que vous fassiez un lien symbolique `db2` qui pointe sur `db1` :

```
shell> cd /path/to/datadir
shell> ln -s db1 db2
```

Maintenant, pour toute table `tbl_a` de `db1` , il en apparaît aussi `tbl_a` dans `db2` . Si un thead modifie `db1.tbl_a` et un autre `db2.tbl_a` , il va y avoir un conflit.Si vous avez vraiment besoin de cette fonctionnalité, vous devez changer le code suivant dans le fichier C `mysys/mf_format.c` :

```
if (flag & 32 || (!lstat(to,&stat_buff) && S_ISLNK(stat_buff.st_mode)))
```

en

```
if (1)
```

Sous Windows, vous pouvez utiliser des liens internes symboliques pour relier des bases en compilant MySQL avec l'option `-DUSE_SYMDIR` . Cela vous permettra de placer vos bases de données sur différentes partitions. Partager les données entre plusieurs disques avec Win32 .

#### 5.6.1.2 Utiliser des liens symboliques avec les tables

Avant MySQL 4.0, vous ne devez pas utiliser les liens symboliques avec les tables, si vous n'êtes pas très prudents avec. Le problème est que si vous exécutez `ALTER TABLE` , `REPAIR TABLE` ou `OPTIMIZE TABLE` sur une table symbolique, le lien sera supprimé et remplacé par le fichier original. Cela arrive car les commandes ci-dessus fonctionnent en créant un fichier temporaire dans le dossier de base, et lorsque l'opération est faite, l'original est remplacé par la copie.

Vous ne devez pas utiliser des liens symboliques sur les tables, sur les systèmes qui ne supportent pas complètement la fonction `realpath()` . (Au moins Linux et Solaris supportent `realpath()` )

En MySQL 4.0, les liens symboliques sont complètement supportés par les tables `MyISAM` . Les autres types de tables vous donneront des résultats étranges lorsque vous les utilisez comme indiqué ci-dessus.

La gestion des liens symboliques de MySQL 4.0 fonctionne comme ceci (uniquement pour les tables `MyISAM` ) :

- Dans le dossier de données, vous allez toujours trouver le fichier de définition de table, le fichier de structure et le fichier d'index.
- Vous pouvez utiliser un lien symbolique avec le fichier d'index et celui de données, pour placer ces fichiers dans d'autres dossiers.
- Le lien symbolique peut être fait via le système d'exploitation (si `mysqld` ne fonctionne pas) ou avec la commande `INDEX/DATA DIRECTORY="path-to-dir"` dans `CREATE TABLE` . Syntaxe de `CREATE TABLE` .

- `myisamchk` ne va pas remplacer un lien symbolique avec les données ou le fichier d'index, mais il va travailler directement sur le fichier vers lequel le lien pointe. Tous les fichiers temporaires seront créé dans le même dossier que le dossier qui contient les données ou le fichier d'index.
- Lorsque vous détruisez une table qui utilise un lien symbolique, le fichier et le lien symbolique sont détruits. C'est une bonne raison pour **ne pas** exécuter `mysqld` en tant que `root` ou donner des droits d'écriture à d'autres personnes dans les dossiers de données de MySQL.
- Si vous renommez une table avec `ALTER TABLE RENAME` vous n'avez pas à déplacer la table dans une autre base, le lien symbolique du dossier de base sera renommé avec le nouveau nom.
- Si vous utilisez la commande `ALTER TABLE RENAME` pour déplacer la table dans une autre base, la table sera déplacée dans l'autre base, et l'ancien lien symbolique et le fichier vers lequel il pointait seront détruits (en d'autres termes, la nouvelle table ne sera pas un lien symbolique).
- Si vous n'utilisez pas de lien symbolique, vous devriez utiliser l'option `--skip-symlink` de `mysqld` pour vous assurer que personne n'efface ou ne renomme un fichier en dehors du dossier de données de MySQL.

Ce qui n'est pas encore supporté :

- `ALTER TABLE` ignore toutes les options `INDEX/DATA DIRECTORY="path"` .
- `CREATE TABLE` n'indique pas si la table a des liens symboliques.
- `mysqldump` n'inclut pas d'informations sur les informations de liens.
- `BACKUP TABLE` et `RESTORE TABLE` ne respectent pas les liens symboliques.

# 6 Référence du langage MySQL

MySQL possède une interface SQL assez complexe, mais facile à apprendre. Ca chapitre décrit les différentes commandes, types, et fonctions que vous aurez besoin de connaître pour utiliser MySQL de façon optimale. Ce chapitre sert aussi de référence pour toutes les fonctionnalités incluses au sein de MySQL. Afin d'utiliser ce chapitre efficacement, vous pourrez trouver utile de vous reporter aux différents index.

## 6.1 Structure du langage

### 6.1.1 Literals: Comment écrire les chaînes et les nombres

Cette section décrit les différents façons d'écrire les chaînes et les nombres en MySQL. Elle couvre aussi les différentes nuances et quiproquos que vous pouvez rencontrer lorsque vous manipulez ces types de données.

#### 6.1.1.1 Chaînes

Une chaîne est une séquence de caractères, entourée de guillemets simples ( `'` ) ou doubles ( `"` ) (simple seulement si vous êtes en mode ANSI). Exemples:

```
 'une chaîne'
"une autre chaîne"
```

A l'intérieur d'une chaîne, certains séquences de caractères ont une signification spéciale. Chacune d'elle commence par un anti−slash ( `\` ), connu comme le **caractère d'échappement** . `MySQL` reconnaît les séquences suivantes :

*\0*
: Un 0 ASCII ( `NUL` ).

*\'*
: Un guillemet simple ( `'` ).

*\"*
: Un guillemet double ( `"` ).

*\b*
: Un effacement.

*\n*
: Une nouvelle ligne.

*\r*
: Un retour chariot.

*\t*
: Une tabulation.

*\z*
: ASCII(26) (Contrôle−Z). Ce caractère peut être encodé pour vous éviter des problèmes avec Windows, vu qu'il équivaut à une fin de fichier sur cet OS. (ASCII(26) vous posera des problèmes si vous utilisez `mysql base < fichier` .)

**
: Un anti−slash ( `\` ).

*\%*
: Un `%` . Cette notation est utilisée pour trouver les occurrences de `%` au cas où `%`

peut aussi être interprété comme un caractère spécial. Fonction de comparaison de chaînes de caractères .

_

Un '_' . Cette notation est utilisée pour trouver les occurrences de '_' au cas où '_' peut aussi être interprété comme un caractère spécial. Fonction de comparaison de chaînes de caractères .

Notez que si vous utilisez '\%' ou '_' dans un contexte de chaînes de caractères, ces séquences retourneront '\%' et '_' et non '%' et '_' .

Il y a plusieurs façons d'intégrer un guillemet dans une chaîne :

- Un ''' à l'intérieur d'une chaîne entourée de ''' peut être noté '''' .
- Un '"' à l'intérieur d'une chaîne entourée de '"' peut être noté '"" .
- Vous pouvez faire précéder le guillemet par caractère d'échappement ( '\' ).
- Un guillemet simple ''' à l'intérieur d'une chaîne à guillemets doubles '"' n'a besoin d'aucun traitement spécial (ni doublage, ni échappement). De même, aucun traitement spécial n'est requis pour un guillemet double '"' à l'intérieur d'une chaîne à guillemets simples ''' .

Le SELECT montré ici explique comment les guillemets et les échappements fonctionnent :

```
 mysql> SELECT 'bonjour', '"bonjour"', '""bonjour""', 'bon''jour', '\'bonjour';
+---------+-----------+-------------+----------+----------+
| bonjour | "bonjour" | ""bonjour"" | bon'jour | 'bonjour |
+---------+-----------+-------------+----------+----------+

 mysql> SELECT "bonjour", "'bonjour'", "'''bonjour'''", "bon""jour", "\"bonjour";
+---------+-----------+-------------+----------+----------+
| bonjour | 'bonjour' | ''bonjour'' | bon"jour | "bonjour |
+---------+-----------+-------------+----------+----------+

 mysql> SELECT "Voilà\n3\nlignes";
+--------------------+
| Voilà
3
lignes |
+--------------------+
```

Si vous voulez insérer des données binaires dans un champ chaîne (comme un BLOB ), les caractères suivants doivent être échappés :

*NUL*

ASCII 0. Représentez le avec '\0' (un anti−slash suivi du caractère ASCII '0' ).

\

ASCII 92, anti−slash. A représenter avec '\\' .

'

ASCII 39, guillemet simple. A représenter avec '\'' .

"

ASCII 34, guillemet double. A représenter avec '\"' .

Si vous écrivez du code C, vous pouvez utiliser la fonction mysql_real_escape_string() , fournie par l'API, pour échapper les caractères pour les requêtes INSERT . C API Function Overview . Avec Perl, vous pouvez utiliser la méthode quote du package DBI pour convertir les caractères spéciaux en leurs équivalents échappés. L'interface DBI .

Vous devez utiliser une fonction d'échappement pour chaque chaîne susceptible de contenir l'un des caractères spéciaux listés plus haut.

Alternativement, beaucoup d'API MySQL , fournissent des fonctionnalités qui vous permettent d'insérer des marqueurs spéciaux dans une requête puis de les remplacer par des valeurs lors de l'exécution de celle−ci. Dans ce cas, l'API s'occupe d'échapper les caractères spéciaux pour vous.

#### 6.1.1.2 Nombres

Les entiers sont représentés comme une séquence de chiffres. Les décimaux utilisent `'.'` comme séparateur. Tous les types de nombres peuvent être précédés d'un `'-'` pour indiquer une valeur négative.

Exemples d'entiers valides :

```
 1221
0
-32
```

Exemples de nombres à virgule flottante :

```
 294.42
-32032.6809e+10
148.00
```

Un entier peut être utilisé dans un contexte décimal, il sera interprété comme le nombre décimal équivalent.

#### 6.1.1.3 Valeurs hexadécimales

MySQL supporte les valeurs hexadécimales. Dans un contexte numérique, elles agissent comme des entiers (précision 64−bit). Dans un contexte de chaîne, elles agissent comme une chaîne binaire où chaque paire de caractères hexadécimaux est convertie en caractère :

```
 mysql> SELECT x'4D7953514C';
         -> MySQL
mysql> SELECT 0xa+0;
         -> 10
mysql> SELECT 0x5061756c;
         -> Paul
```

La syntaxe `x'chaînehexa'` (nouvelle en 4.0) est basée sur ANSI SQL et la syntaxe `0x` est basée sur ODBC. Les chaînes hexadécimales sont souvent utilisée par ODBC pour fournir des valeurs aux colonnes `BLOB` . Vous pouvez convertir une chaîne ou un nombre en hexadécimal avec la fonction `HEX()` .

#### 6.1.1.4 Valeurs NULL

La valeur `NULL` signifie ``pas de données'' et est différente des valeurs comme `0` pour les nombres ou la chaîne vide pour les types chaîne. Problèmes avec les valeurs `NULL` .

`NULL` peut être représenté par `\N` lors de la récupération ou écriture avec des fichiers ( `LOAD DATA INFILE` , `SELECT ... INTO OUTFILE` ). Syntaxe des `LOAD DATA INFILE` .

### 6.1.2 Noms de bases, tables, index, colonnes et alias

Les noms des bases de données, tables, index, colonnes et alias suivent tous les mêmes règles en MySQL.

Notez que les règles ont changé à partir de la version 3.23.6 de MySQL lorsqu'à été introduite la protection des noms d'identifiant avec les guillemets obliques `'`` . `'""'` fonctionne aussi si vous fonctionnez en mode AINSI. Exécuter MySQL en mode ANSI .

| Identifiant | Longueur maximale | Caractères autorisés |
|---|---|---|
| Base de données | 64 | Tous les caractères autorisés dans un nom de dossier à part '/' , '\' et '.' . |
| Table | 64 | Tous les caractères autorisés dans le nom d'un fichier à part '/' et '.' . |

| | | |
|---|---|---|
| Colonne | 64 | Tous. |
| Alias | 255 | tous. |

Notez qu'en plus de ce qui précède, vous n'avez pas droit au caractères ASCII(0), ASCII(255) ou aux guillemets dans un identifiant.

Notez que si un identifiant est un mot réservé, ou contient des caractères spéciaux, vous devez absolument le protéger avec ` :

```
mysql> SELECT * FROM `select` WHERE `select`.id > 100;
```

Est−ce que MySQL est sensible aux mots réservés ? .

Dans les versions antérieures à la 3.23.6, les règles étaient les suivantes :

- Un nom est constitué d'une séquence de caractères alpha−numériques, tiré du jeu de caractères courant, et en y ajoutant le souligné '_' et le signe dollar '$' . Le jeu de caractères par défaut est ISO−8859−1 Latin1; Il peut être modifié avec l'option de démarrage --default-character-set de mysqld . Le jeu de caractères utilisé pour les données et le stockage .
- Un nom peut commencer par n'importe quel caractère permis dans un nom. Un nom peut donc commencer par un chiffre. (Ce qui n'est pas le cas dans d'autres systèmes de base de données). Par contre, un nom ne peut consister **uniquement** de chiffres.
- Vous ne pouvez pas utiliser de point ( '.' ) dans les noms car ce caractère est utilisé pour se référer au colonnes (voir ce qui suit).

Il est conseillé de ne pas utiliser de nom comme 1e , car une expression comme 1e + 1 est ambigue. Ce nom pourra être interprété comme l'expression 1e + 1 ou le nombre 1e+1 .

Avec MySQL vous pouvez indiquer une colonne avec l'une des syntaxes suivantes :

| Référence | Signification |
|---|---|
| nom_colonne | La colonne nom_colonne de chaque table utilisée dans la requête qui comporte une telle colonne. |
| nom_de_table.nom_colonne | La colonne nom_colonne de la table nom_de_table de la base de données courante. |
| nom_de_base.nom_de_table.nom_colonne | La colonne nom_colonne de la table nom_de_table de la base nom_de_base . Cette forme est disponible à partir de la version 3.22 de MySQL. |
| `nom_colonne` | Une colonne qui est un mot réservé ou qui contient un caractère spécial. |

Vous n'avez pas besoin de spécifier nom_de_table ou nom_de_base.nom_de_table avant une colonne sauf si la référence risque d'être ambigue. Par exemple, supposons que t1 et t2 contiennent toutes les deux une colonne nommée c , et que vous lisez c dans une requête SELECT qui utilise t1 et t2 à la fois. Dans ce cas, c est ambigu car il peut se référer t1 aussi bien qu'à t2 . Vous devez donc indiquer la table ciblée en écrivant t1.c ou t2.c . De même, si vous lisez à partir d'une table t située dans la base db1 et de la table t dans la base db2 , vous devez vous référer aux colonnes de ces tables en écrivant db1.t.nom_colonne et db2.t.nom_colonne .

La syntaxe .nom_de_table référence la table nom_de_table dans la base de données courante. Cette syntaxe est acceptée pour assurer la compatibilité avec ODBC, car certains pilotes ODBC préfixent les noms des tables avec des points ( '.' ).

## 6.1.3 Sensibilité à la casse pour les noms

En MySQL, les bases et les tables correspondent à des dossiers et des fichiers. Par conséquent, la sensibilité à la casse du système détérminera la sensisbilité à la casse des noms de bases de données et tables. Cela signifie que les noms sont insensibles à la casse sous Windows, et sensibles sous la plupart des variétés Unix (Mac OS X étant une exception). Extensions MySQL à l'ANSI SQL92 .

**Note** : Même si les noms ne sont pas sensibles à la casse sous Windows, vous ne devez pas vous référer à une entité en utilisant différentes casse dans la même requête. La requête suivante ne fonctionnera pas car elle se réfère à une table avec `ma_table` et `MA_TABLE` :

```
mysql> SELECT * FROM ma_table WHERE MA_TABLE.col=1;
```

Les noms de colonnes et d'alias sont insensibles à la casse dans tous les cas.Les alias sur tables sont sensibles à la casse. La requête suivante ne marchera pas car elle se réfère à `a` et `A` :

```
mysql> SELECT nom_de_colonne FROM nom_de_table AS a
   ->                WHERE a.nom_de_colonne = 1 OR A.nom_de_colonne = 2;
```

Si vous avez du mal à vous souvenir de la casse des noms de bases et de tables, adoptez une convention, comme toujours créer les bases et les tables en utilisant des minuscules.Un moyen d'éviter ce problème est de démarrer `mysqld` avec `-O lower_case_table_names=1` . Par défaut, cette option vaut 1 sur Windows et 0 sur Unix.

Si `lower_case_table_names` est à 1 MySQL convertira tout les noms de tables en minuscules lors des sauvegardes et des récupérations. (A partir de la version 4.0.2, cette opération s'applique aussi aux noms de bases de données.). Notez que si vous changez cette option, vous devrez repasser vos anciens noms de tables en minuscules avant de démarrer `mysqld` .

Si vous déplacez les fichiers `MyISAM` d'un Windows à une architecture *nix, vous aurez peut être besoin d'utiliser l'utilitaire `mysql_fix_extensions` pour corriger la casse des extensions des fichiers dans chaque répertoire spécifique à la base de données (minuscules `.frm` , majuscules `.MYI` et `.MYD` ). `mysql_fix_extensions` peut être trouvé dans le sous-répertoire `script` .

## 6.1.4 Variables utilisateur

MySQL supporte les variables utilisateur spécifiques à la connexion avec la syntaxe `@variablename` . Un nom de variable consiste de caractères alpha-numériques, basés sur le jeu de caractères courant, de `'-'` , `'$'` , et `'.'` . Le jeu de caractères par défaut est ISO-8859-1 Latin1. Cette valeur peut être changée en utilisant l'option `--default-character-set` de `mysqld` . Le jeu de caractères utilisé pour les données et le stockage .

Les variables n'ont pas besoin d'être initialisés. Elles sont à `NULL` par défaut et peuvent contenir un entier, un réel ou une chaîne. Toutes les variables d'un thread sont automatiquement libérées lorsque le thread se termine.

Vous pouvez déclarer une variable avec la syntaxe de `SET` :

```
 SET &#64;variable= &#123; expression entier | expression réel | expression chaîne &#125;
[,&#64;variable= ...].
```

Vous pouvez aussi assigner une valeur à une variable avec d'autres commande que `SET` . Par contre, dans ce cas là, l'opérateur d'assignation est `:=` au lieu de `=` , parce que `=` est réservé aux comparaisons dans les requêtes autres que `SET` :

```
 mysql> SELECT &#64;t1:=(&#64;t2:=1)+&#64;t3:=4,&#64;t1,&#64;t2,&#64;t3;
+----------------------+------+------+------+
| &#64;t1:=(&#64;t2:=1)+&#64;t3:=4 | &#64;t1  | &#64;t2  | &#64;t3  |
+----------------------+------+------+------+
|                    5 |    5 |    1 |    4 |
+----------------------+------+------+------+
```

Les variables utilisateur peuvent être utilisés là où les expressions sont allouées. Notez que cela n'inclut pas pour l'instant les contextes où un nombre est explicitement requis, comme ce qui est le cas avec la clause `LIMIT` dans une requête `SELECT` , ou la clause `IGNORE nombre LINES` dans une requête `LOAD DATA` . **Note** : dans une requête `SELECT` , chaque expression est n'évaluée que lors de l'envoi au client. Cela signifie que pour les clauses `HAVING` , `GROUP BY` , ou `ORDER BY` , vous ne pouvez vous référer à une expression qui implique des variables qui sont définies dans la partie `SELECT` . Par exemple, la requête

suivante ne produira pas le résultat escompté :

```
mysql> SELECT (&#64;aa:=id) AS a, (&#64;aa+3) AS b FROM nom_de_table HAVING b=5;
```

La raison est que `@aa` ne contiendra pas la valeur de la ligne courante, mais celle de `id` pour la dernière ligne acceptée.

## 6.1.5 Variables système

A partir de la version 4.0.3 de MySQL, nous fournissons un meilleur accès à beaucoup de variables système et variables de connexion. On peut changer la plupart d'entre elle sans avoir à stopper le serveur.

Il y a deux sortes de variables système : Les variables spécifiques aux threads (ou à la connexion) qui sont uniques à la connexion courante et les variables globales qui sont utilisées pour configurer les évènements globaux. Les variables globales sont aussi utilisées pour configurer les valeurs initales des variables spécifiques au threads pour les nouvelles connexions.

Lorsque `mysqld` démarre, toutes les variables globales sont initialisées à partir des arguements passés en ligne de commande et des fichiers de configuration. Vous pouvez changer ces valeurs avec la commande `SET GLOBAL` . Lorsqu'un nouveau thread est créé, les variables spécifiques aux threads sont initialisées à partir des variables globales et ne changeront pas même si vous utilisez la commande `SET GLOBAL` .

Pour définir la valeur d'une variable `GLOBAL` , vous devez utiliser l'une des syntaxes suivantes : (Ici nous utilisons la variable `sort_buffer_size` à titre d'exemple)

```
 SET GLOBAL sort_buffer_size=valeur;
SET &#64;&#64;global.sort_buffer_size=valeur;
```

Pour définir la valeur d'une variable `SESSION` , vous devez utiliser l'une des syntaxes suivantes :

```
 SET SESSION sort_buffer_size=valeur;
SET &#64;&#64;session.sort_buffer_size=valeur;
SET sort_buffer_size=valeur;
```

Si vous ne spécifiez pas `GLOBAL` ou `SESSION` alors `SESSION` est utilisé. Syntaxe de `SET` . `LOCAL` est un synonyme de `SESSION` .

Pour récupérer la valeur d'une variable de type `GLOBAL` vous pouvez utiliser l'une des commandes suivantes :

```
 SELECT &#64;&#64;global.sort_buffer_size;
SHOW GLOBAL VARIABLES like 'sort_buffer_size';
```

Pour récupérer la valeur d'une variable de type `SESSION` vous pouvez utiliser l'une des commandes suivantes :

```
 SELECT &#64;&#64;session.sort_buffer_size;
SHOW SESSION VARIABLES like 'sort_buffer_size';
```

Lorsque vous **récuperez** une valeur de variable avec la syntaxe `@@nom_variable` et que vous ne spécifiez pas `GLOBAL` ou `SESSION` , MySQL retournera la valeur spécifique au thread ( `SESSION` ) si elle existe. Sinon, MySQL retournera la valeur globale.La raison d'imposer la présence du mot `GLOBAL` pour configurer une variable de type `GLOBAL` mais non pour la lire est pour être sur que vous n'aurez pas de problèmes plus tard si vous voulez introduire ou effacer une variable spécifique au thread qui aurait le même nom. Dans ce cas, vous pourriez changer accidentellement l'état du serveur pour toutes les connexions (et non la votre uniquement).

Voilà la liste complète de toutes les variables que que vous pouvez récuperer ou modifier et une

indication quant à l'utilisation de GLOBAL ou SESSION avec elles.

| Nom de la variable | Type de valeur | Type |
|---|---|---|
| autocommit | bool | SESSION |
| big_tables | bool | SESSION |
| binlog_cache_size | num | GLOBAL |
| bulk_insert_buffer_size | num | GLOBAL \| SESSION |
| concurrent_insert | bool | GLOBAL |
| connect_timeout | num | GLOBAL |
| convert_character_set | string | SESSION |
| delay_key_write | OFF \| ON \| ALL | GLOBAL |
| delayed_insert_limit | num | GLOBAL |
| delayed_insert_timeout | num | GLOBAL |
| delayed_queue_size | num | GLOBAL |
| error_count | num | LOCAL |
| flush | bool | GLOBAL |
| flush_time | num | GLOBAL |
| foreign_key_checks | bool | SESSION |
| identity | num | SESSION |
| insert_id | bool | SESSION |
| interactive_timeout | num | GLOBAL \| SESSION |
| join_buffer_size | num | GLOBAL \| SESSION |
| key_buffer_size | num | GLOBAL |
| last_insert_id | bool | SESSION |
| local_infile | bool | GLOBAL |
| log_warnings | bool | GLOBAL |
| long_query_time | num | GLOBAL \| SESSION |
| low_priority_updates | bool | GLOBAL \| SESSION |
| max_allowed_packet | num | GLOBAL \| SESSION |
| max_binlog_cache_size | num | GLOBAL |
| max_binlog_size | num | GLOBAL |
| max_connect_errors | num | GLOBAL |
| max_connections | num | GLOBAL |
| max_error_count | num | GLOBAL \| SESSION |
| max_delayed_threads | num | GLOBAL |
| max_heap_table_size | num | GLOBAL \| SESSION |
| max_join_size | num | GLOBAL \| SESSION |
| max_sort_length | num | GLOBAL \| SESSION |
| max_tmp_tables | num | GLOBAL |
| max_user_connections | num | GLOBAL |
| max_write_lock_count | num | GLOBAL |
| myisam_max_extra_sort_file_size | num | GLOBAL \| SESSION |
| myisam_max_sort_file_size | num | GLOBAL \| SESSION |
| myisam_sort_buffer_size | num | GLOBAL \| SESSION |
| net_buffer_length | num | GLOBAL \| SESSION |
| net_read_timeout | num | GLOBAL \| SESSION |
| net_retry_count | num | GLOBAL \| SESSION |
| net_write_timeout | num | GLOBAL \| SESSION |
| query_cache_limit | num | GLOBAL |
| query_cache_size | num | GLOBAL |
| query_cache_type | enum | GLOBAL |
| read_buffer_size | num | GLOBAL \| SESSION |
| read_rnd_buffer_size | num | GLOBAL \| SESSION |
| rpl_recovery_rank | num | GLOBAL |
| safe_show_database | bool | GLOBAL |
| server_id | num | GLOBAL |
| slave_compressed_protocol | bool | GLOBAL |
| slave_net_timeout | num | GLOBAL |
| slow_launch_time | num | GLOBAL |
| sort_buffer_size | num | GLOBAL \| SESSION |
| sql_auto_is_null | bool | SESSION |
| sql_big_selects | bool | SESSION |
| sql_big_tables | bool | SESSION |
| sql_buffer_result | bool | SESSION |
| sql_log_binlog | bool | SESSION |
| sql_log_off | bool | SESSION |
| sql_log_update | bool | SESSION |
| sql_low_priority_updates | bool | GLOBAL \| SESSION |
| sql_max_join_size | num | GLOBAL \| SESSION |
| sql_quote_show_create | bool | SESSION |

| | | |
|---|---|---|
| sql_safe_updates | bool | SESSION |
| sql_select_limit | bool | SESSION |
| sql_slave_skip_counter | num | GLOBAL |
| sql_warnings | bool | SESSION |
| table_cache | num | GLOBAL |
| table_type | enum | GLOBAL \| SESSION |
| thread_cache_size | num | GLOBAL |
| timestamp | bool | SESSION |
| tmp_table_size | enum | GLOBAL \| SESSION |
| tx_isolation | enum | GLOBAL \| SESSION |
| version | string | GLOBAL |
| wait_timeout | num | GLOBAL \| SESSION |
| warning_count | num | LOCAL |
| unique_checks | bool | SESSION |

Les variables suivies d'un `num` peuvent se voir affecter une valeur numérique. Celles marquées avec `bool` peuvent prendre 0, 1, `ON` ou `OFF` . Les variables qui sont du type `enum` doivent normalement comporter au moins l'une des valeurs disponibles pour la variable, mais peuvent aussi contenir les index des membres de la collection. (Le premier index est 0).

Voilà une description de quelques variables :

| Variable | Description |
|---|---|
| identity | Alias pour last_insert_id (compatibilité Sybase) |
| sql_low_priority_updates | Alias pour low_priority_updates |
| sql_max_join_size | Alias pour max_join_size |
| delay_key_write_for_all_tables | Si cette variable et delay_key_write sont définies, alors toutes les nouvelles tables MyISAM qui sont ouvertes utiliseront les écritures de clefs reportées. |
| version | Alias for VERSION() (compatibilité Sybase (?)) |

Une déscription des autres variables peut être trouvée dans la section des options de démarrage, la description de `SHOW VARIABLES` et dans la section de `SET` . Options en ligne de commande de `mysqld` . `SHOW VARIABLES` . Syntaxe de `SET` .

## 6.1.6 Syntaxe des commentaires

Le serveur MySQL supporte les commentaires de la forme `# jusqu'à la fin de la ligne` , `-- jusqu'à la fin de la ligne` et `/* dans la ligne ou sur plusieurs lignes */` :

```
 mysql> SELECT 1+1;     # Ce commentaire se continue jusqu'à la fin de la ligne
mysql> SELECT 1+1;     -- Ce commentaire se continue jusqu'à la fin de la ligne
mysql> SELECT 1 /* Ceci est un commentaire dans la ligne */ + 1;
mysql> SELECT 1+
/*
Ceci est un commentaire
sur plusieurs lignes
*/
1;
```

Notez que le style de commentaire `--` requiert au moins un espace après le second tiret !Même si le serveur interprète correctement les syntaxes de commentaires décrites plus haut, il y a quelques limitations dans la façon où le client interprète les commentaires multi−lignes :

- Les guillemets (simples et doubles) sont considérés comme des indications de début de chaîne, même dans un commentaire. Si le guillemet n'est pas refermé (par un second guillemet), l'analyseur ne réalisera pas que le commentaire est fini. Si vous utilisez mysql interactivement, vous pouvez vous en apercevoir, car il va modifier l'invite de commande de `mysql>` en `'>` ou `">` .
- Un point−virgule sert à indiquer la fin de la commande SQL, et tout ce qui suit un point−virgule est considéré comme étant une nouvelle requête.

Ces limitations s'appliquent aussi bien à `mysql` en ligne de commande, que lorsque vous demandez à `mysql` de lire des commandes depuis un fichier ( `mysql < un−fichier` ).

MySQL ne supporte les commentaires du style '--' ANSI SQL que si le second tiret est suivi d'un espace. '--' comme début de commentaire .

### 6.1.7 Est−ce que MySQL est sensible aux mots réservés ?

Un problème récurrent provient de la tentative de création de tables avec des noms de colonnes qui sont des types de champs ou des fonctions natives de MySQL, comme TIMESTAMP ou GROUP . Il vous st permis de le faire (par exemple ABS est permis comme nom de colonne), mais les espaces ne sont pas permis entre le nom d'une fonction et la première '(' suivante lors de l'utilisation de fonctions qui sont aussi des noms de colonnes.

Les mots suivants sont explicitement réservés en MySQL. La plupart sont interdits par ANSI SQL92 en tant que nom de colonnes ou de tables (par exemple, GROUP ). Quelques uns sont réservés parce que MySQL en a besoin et utilise (actuellement) un analyseur yacc :

Les symboles suivants (issus de la table ci−dessus) sont interdits par ANSI SQL mais permis par MySQL en tant que noms de colonnes ou de tables. Cela est du au fait que ces noms sont très courants, et de nombreux programmeur les ont déjà utilisés.

- ACTION
- BIT
- DATE
- ENUM
- NO
- TEXT
- TIME
- TIMESTAMP

## 6.2 Types de colonnes

MySQL supporte un grand nombre de types de colonnes, qui peuvent être rassemblés en trois groupes : les nombres, les dates et les chaînes de caractères. Cette section présente les types disponibles et leurs tailles de stockage, puis présente en détail chaque type. L'introduction est volontairement brève. Une section plus précise est dédiée à chaque type, qui présente tous les formats valides.

Les types de colonnes de MySQL sont listés ci−dessous. Les codes suivants sont utilisés dans les descriptions :

*M*
: Indique la taille maximale d'affichage. Le maximum légal est 255.

*D*
: S'applique aux nombres à virgule flottante, et indique le nombre de décimales qui suivent la virgule. Le nombre maximum est de 30, mais ne doit pas être plus grand que M −2.

Les crochets ( '[' et ']' ) indiquent les spécifications optionnelles.

Notez que si vous spécifiez ZEROFILL pour une colonne, MySQL ajoutera automatiquement l'attribut UNSIGNED à la colonne.

**Attention** : vous devez garder à l'esprit que lors de la soustraction de deux entiers dont l'un est de type UNSIGNED , le résultat ne sera pas signé ! Fonctions de transtypage .

*TINYINT[(M)] [UNSIGNED] [ZEROFILL]*

Un très petit entier. S'il est signé, sa valeur varie entre `-128` et `127` , sinon elle varie de `0` à `255` .

*BIT*

*BOOL*

Ce sont des synonymes de `TINYINT(1)` .

*SMALLINT[(M)] [UNSIGNED] [ZEROFILL]*

Un petit entier. S'il est signé, sa valeur varie entre `-32768` et `32767` , sinon elle varie de `0` à `65535` .

*MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]*

Un entier de taille moyenne. S'il est signé, sa valeur varie entre `-8388608` et `8388607` , sinon elle varie entre `0` et `16777215` .

*INT[(M)] [UNSIGNED] [ZEROFILL]*

Un entier. S'il est signé, sa valeur varie entre `-2147483648` et `2147483647` , sinon elle varie entre `0` et `4294967295` .

*INTEGER[(M)] [UNSIGNED] [ZEROFILL]*

C'est un synonyme de `INT` .

*BIGINT[(M)] [UNSIGNED] [ZEROFILL]*

Un grand entier. S'il est signé, sa valeur varie entre `-9223372036854775808` et `9223372036854775807` , sinon elle varie entre `0` et `18446744073709551615` .Quelques choses à savoir pour bien utiliser les colonnes de type `BIGINT` :

Toute l'arithmétique est effectuée en utilisant des `BIGINT` signés ou des valeurs de type `DOUBLE` , vous ne devez donc pas utiliser de grands entiers non-signés plus grand que `9223372036854775807` (63 bits) excepté avec les fonctions sur les bits ! Si vous le faites, certains des derniers chiffres du résultat risquent d'être faux à cause des erreurs d'arrondissage lors de la conversion de `BIGINT` à `DOUBLE` .MySQL 4.0 peut gérer les `BIGINT` dans les cas suivants :

Utilisation d'entiers pour stocker une grande valeur non-signée dans une colonne `BIGINT` .

Avec `MIN(colonne_big_int)` and `MAX(colonne_big_int)` .

Lors de l'utilisation des opérateurs ( `+` , `-` , `*` , etc.) lorsque les deux opérandes sont des entiers.

Vous pouvez toujours stocker la valeur exacte d'un entier dans une colonne de type `BIGINT` en l'enregistrant en tant que chaîne. Dans ce cas, MySQL éffectuera une conversion chaîne à nombre qui ne fait entrer en jeu aucune représentation intermédiaire en réel.

`'-'` , `'+'` , et `'*'` utiliseront l'arithmétique `BIGINT` quand les deux arguments sont des valeurs entières ! Cela signifie que si vous multipliez deux grands entiers (ou des résultats de fonctions qui retournent des entiers) vous obtiendrez peut être des résultats inattendus quand le résultat est supérieur à `9223372036854775807` .

*FLOAT(précision) [UNSIGNED] [ZEROFILL]*

Un nombre à virgule flottante. `précision` <=24 pour un nombre à virgule flottante de précision simple, entre 25 et 53 pour une précision double. Ces types correspondent aux types `FLOAT` et `DOUBLE` décrits ci-dessus. FLOAT(X) a le même intervalle de validité que `FLOAT` et `DOUBLE` , mais la taille d'affichage et le nombre de décimale sont indéfinies. Dans la version 3.23 de MySQL, c'est un véritable nombre à virgule flottante. Dans les anciennes versions de MySQL, `FLOAT(précision)` avait toujours deux décimales.

Notez que l'utilisation du type `FLOAT` peut vous créér des problèmes inattendus car tous les calculs internes de MySQL sont fait en double précision. Résoudre les problèmes des lignes non retournées .

Cette syntaxe est fournie pour assurer la compatibilité avec ODBC.

*FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]*

Un petit nombre à virgule flottante (précision simple). L'intervalle de validité va de `-3.402823466E+38` à `-1.175494351E-38` , `0` et de `1.175494351E-38` à `3.402823466E+38` . Si `UNSIGNED` est spécifié, les valeurs négatives sont interdites. `M` représente la taille d'affichage et `D` est le nombre de décimales. `FLOAT` sans arguments, ou avec un argument <= 24 représente un nombre à virgule flottante de précision simple.

*DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]*

Un nombre à virgule flottante (précision double). L'intervalle de validité va de `-1.7976931348623157E+308` à `-2.2250738585072014E-308` , `0` et de `2.2250738585072014E-308` à `1.7976931348623157E+308` . Si `UNSIGNED` est spécifié, les valeurs négatives sont interdites. `M` représente la taille d'affichage et `D` est le nombre de décimales. `FLOAT` sans arguments, ou avec un argument compris entre 25 et 53 (inclus) représente un nombre à virgule flottante de précision double.

*DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL]*

*REAL[(M,D)] [UNSIGNED] [ZEROFILL]*

C'est un synonyme de `DOUBLE` .

*DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]*

Un nombre à virgule flottante. Il doit être signé. Ce type se comporte comme une colonne de type `CHAR` : la valeur est stockée comme une chaîne, chaque caractère représentant un chiffre de la valeur. La virgule et le signe `'-'` des nombres négatifs ne sont pas comptés dans l'option `M` (mais de l'espace de stockage est reservé pour eux). Si `D` vaut 0, les valeurs n'auront pas de valeur décimale. L'intervalle de validité des valeurs `DECIMAL` est le même que `DOUBLE` , mais il peut être limité par les valeurs choisies pour `M` et `D` . Si `UNSIGNED` est spécifié, les valeurs négatives sont interdites.

Si `D` est ommis, la valeur par défaut est 0. Si `M` est ommis, la valeur par défaut est 10.

Avant la version 3.23 de MySQL, l'argument `M` doit inclure l'espace requis pour le signe et le point des décimales.

*DEC[(M[,D])] [UNSIGNED] [ZEROFILL]*

*NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]*

Ce sont des synonymes de `DECIMAL` .

*DATE*

Une date. L'intervalle de validité va de `'1000-01-01'` à `'9999-12-31'` . MySQL affiche les valeurs `DATE` au format `'AAAA-MM-JJ'` (année-mois-jour), mais vous pouvez assigner des valeurs aux colonnes `DATE` en utilisant différents formats numériques ou de chaînes de caractères. The `DATETIME` , `DATE` and `TIMESTAMP` types .

*DATETIME*

Une combinaison de date et d'heure. L'intervalle de validité va de `'1000-01-01 00:00:00'` à `'9999-12-31 23:59:59'` . MySQL affiche les valeurs `DATETIME` au format `'AAAA-MM-JJ HH:MM:SS'` (année-mois-jour heure:minutes:secondes), mais vous pouvez assigner des valeurs aux colonnes `DATETIME` en utilisant différents formats numériques ou de chaînes de caractères. The `DATETIME` , `DATE` and `TIMESTAMP` types .

*TIMESTAMP[(M)]*

Un timestamp. L'intervalle de validité va de `'1970-01-01 00:00:00'` à quelque part durant l'année 2037. MySQL affiche les valeurs de type `TIMESTAMP` au format `AAAAMMJJHHMMSS` , `AAMMJJHHMMSS` , `AAAAMMJJ` , ou `AAMMJJ` , suivant que le paramètre `M` vaut `14` (ou omis), `12` , `8` , ou `6` , mais vous pouvez assigner des valeurs aux colonnes `TIMESTAMP` sous forme de nombre ou de chaînes. Une colonne de type `TIMESTAMP` est pratique pour enregistrer une date, lors d'une commande `INSERT` ou `UPDATE` , car elle est automatiquement mise à la date et l'heure du moment de la commande, si vous ne fournissez pas de valeur vous-même. Vous pouvez aussi lui donner la date et l'heure courante en lui assignant la valeur `NULL` . Types dates et heures .

Un `TIMESTAMP` est toujours stocké sur 4 octets. L'argument `M` n'affecte que le mode

d'affichage de ce type de colonne.

Notez bien que les colonnes de type `TIMESTAMP(X)` où `X` vaut `8` ou `14` sont considérées comme des nombres tandis que les autres colonnes `TIMESTAMP(X)` sont considérées comme des chaînes. Ceci est fait pour s'assurer que l'on peut exporter et importer les tables avec ces types. The `DATETIME` , `DATE` and `TIMESTAMP` types .

*TIME*

Une heure. L'intervalle de validité va de `'−838:59:59'` à `'838:59:59'` . MySQL affiche les colonnes de type `TIME` au format `'HH:MM:SS'` , mais vous pouvez assigner une valeur de type `TIME` en lui passant des chaînes ou des entiers. The `TIME` type .

*YEAR[(2|4)]*

Une année au format 2 ou 4 chiffres (par défaut, c'est 4 chiffres). L'intervalle de validité va de `1901` à `2155` , `0000` pour le format à 4 chiffres, et de `1970` à `2069` pour le format à deux chiffres. MySQL affiche les valeurs de type `YEAR` au format `AAAA` , mais vous pouvez leur assigner des chaînes ou des nombres. (le type `YEAR` est nouveau depuis la version 3.22 de MySQL) The `YEAR` type .

*[NATIONAL] CHAR(M) [BINARY]*

Une chaîne de caractères de taille fixe, qui est toujours complétée à droite, par des espaces, lors du stockage. Le paramètre `M` peut valoir de 0 à 255 caractères. (1 à 255 dans les versions antérieures à la 3.23) Les espaces terminaux sont supprimés lorsque la chaîne est lue dans la base. Les valeurs de type `CHAR` sont triées et comparées sans tenir compte de la casse et en utilisant le jeu de caractères par défaut. Toutefois, vous pouvez utiliser l'opérateur BINARY pour effectuer des recherches sensibles à la casse. `NATIONAL CHAR` (forme courte : `NCHAR` ) est la dénomination ANSI SQL pour les colonnes de type `CHAR` qui doivent utiliser le jeu de caractères par défaut. C'est la configuration par défaut de MySQL.

MySQL autorise la création de colonne de type `CHAR(0)` . Cela ne sert réellement que si vous devez être compatible avec une vieille application, dont le bon fonctionnement repose sur l'existence de la colonne, mais qui n'utilise pas vraiment ses valeurs. C'est aussi pratique lorsque vous devez stocker un booléen (ou une valeur à deux états) dans une colonne : une colonne de type `CHAR(0)` , qui n'a pas l'attribut NOT NULL, n'occupera qu'un seul octet, et peut prendre deux valeurs : `NULL` et `""` . Les types `CHAR` et `VARCHAR` .

`CHAR` est une forme courte de `CHARACTER` .

*CHAR*

C'est un synonyme de `CHAR(1)` .

*[NATIONAL] VARCHAR(M) [BINARY]*

Une chaîne de caractères de longueur variable. **NOTE :** les espaces terminaux sont supprimés lors du stockage des valeurs (ce qui diffère des spécifications ANSI SQL). L'intervalle de taille de M va de 1 à 255 caractères. Les valeurs de type `VARCHAR` sont triées et comparées sans tenir compte de la casse et en utilisant le jeu de caractères par défaut. Toutefois, vous pouvez utiliser l'opérateur `BINARY` pour effectuer des recherches sensibles à la casse. Silent column changes .

`VARCHAR` est un forme courte de `CHARACTER VARYING` . Les types `CHAR` et `VARCHAR` .

*TINYBLOB*
*TINYTEXT*

Une colonne `BLOB` ou `TEXT` avec une longueur maximale de 255 (2^8 − 1) caractères. Silent column changes . Les types `BLOB` et `TEXT` .

*BLOB*
*TEXT*

Une colonne `BLOB` ou `TEXT` avec une longueur maximale de 65535 (2^16 − 1) caractères.

Silent column changes . Les types `BLOB` et `TEXT` .

*MEDIUMBLOB*
*MEDIUMTEXT*

Une colonne `BLOB` ou `TEXT` avec une longueur maximale de 16777215 (2^24 − 1) caractères. Silent column changes . Les types `BLOB` et `TEXT` .

*LONGBLOB*
*LONGTEXT*

Une colonne `BLOB` ou `TEXT` avec une longueur maximale de 4294967295 (2^32 − 1) caractères. Silent column changes . Notez que puisque le protocole serveur/client et les tables de type MyISAM ont une limitation de 16M par paquet communiqué / ligne, vous ne pourrez utiliser la totalité de la longueur de ce type. Les types `BLOB` et `TEXT` .

*ENUM('valeur1','valeur2',...)*

Une énumération. Une chaîne de caractères qui ne peut prendre qu'une valeur, issue d'une liste de valeurs `'valeur1'` , `'valeur2'` , `...` , `NULL` ou la valeur spéciale d'erreur `""` . Un `ENUM` peut avoir un maximum de 65535 valeurs distinctes. Le type `ENUM` .

*SET('valeur1','valeur2',...)*

Un ensemble. Une chaîne de caractères qui a zéro ou plusieurs valeurs issues d'une liste : `'valeur1'` , `'valeur2'` , `...` Un `SET` peut avoir au maximum 60 éléments. Le type `SET` .

## 6.2.1 Types numériques

MySQL supporte tous les types numériques de la norme ANSI/ISO SQL92. Ceux ci représentent les types numériques exacts ( `NUMERIC` , `DECIMAL` , `INTEGER` , et `SMALLINT` ), ainsi que les types approchés ( `FLOAT` , `REAL` , et `DOUBLE PRECISION` ). Le mot clef `INT` est un synonyme de `INTEGER` , et le mot clef `DEC` est un synonyme de `DECIMAL` .

Les types `NUMERIC` et `DECIMAL` sont considérés comme identiques par MySQL, comme l'autorise le standard SQL92. Ils sont utilisées par des valeurs dont il est primordial de conserver la précision exacte, comme pour des données financières. Lorsque vous déclarez des colonnes avec l'un de ces types, vous pouvez indiquer la précision et l'échelle comme ceci :

```
salaire DECIMAL(5,2)
```

Dans cet exemple, `5` ( `précision` ) représente le nombre de décimales signifiantes qui seront stockées pour les valeurs, et `2` ( `échelle` ) représente le nombre de chiffres qui seront stockés après le point des décimales. Dans ce cas, toutefois, l'intervalle des valeurs pouvant être stockés dans la colonne `salaire` varie de `-99.99` à `99.99` . (MySQL peut actuellement stocker des nombres allant jusqu'à `999.99` dans cette colonne car il n'a pas besoin de l'espace qui sert à stocker le signe pour un nombre positif)

En ANSI/ISO SQL92, la syntaxe `DECIMAL(p)` est équivalente à `DECIMAL(p,0)` . De manière similaire, la syntaxe `DECIMAL` est équivalente à to `DECIMAL(p,0)` , où l'implémentation est autorisée à choisir la valeur de `p` . MySQL ne supporte pas actuellement ces formes variantes des types de données `DECIMAL` / `NUMERIC` . Cela ne pose pas généralement de grand problèmes, vu que le principal bénéfice de ces types est de pouvoir contrôler explicitement la précision et l'échelle.

Les valeurs de type `DECIMAL` et `NUMERIC` sont stockées sous forme de chaînes de caractères, plutôt que comme des nombres à virgule flottante, afin de préserver la précision décimale des valeurs. Un caractère est donc nécessaire pour chaque chiffre, plus la virgule (si `scale` > 0), et le signe moins `'-'` (pour les nombres négatifs). Si `scale` vaut 0, les valeurs de type `DECIMAL` et `NUMERIC` ne comporteront pas de valeur décimale, ni de virgule.

L'intervalle de validité maximale des valeurs de type `DECIMAL` et `NUMERIC` est le même que pour le type `DOUBLE` , mais l'intervalle réel peut être limité par le choix des paramètres `précision` et `scale` . Lorsqu'une

valeur ayant trop de décimales est affectée à une colonne, la valeur est arrondie à `scale` décimales. Lorsqu'une valeur est hors des limites de validité de la colonne `DECIMAL` ou `NUMERIC` , MySQL enregistre la plus grande valeur qu'il peut à la place.

En extension de la norme ANSI/ISO SQL92, MySQL supporte aussi les types entiers `TINYINT` , `MEDIUMINT` , et `BIGINT` , comme présenté ci−dessus. Un autre extension supportée par MySQL permet de spécifier optionnellement la taille d'affichage, sous la forme d'une valeur entière entre parenthèses, juste après le mot clé spécifiant le type (par exemple, INT(4)). Cette spécification de taille est utilisée pour remplir à gauche, avec le caractère de remplissage par défaut, les nombres dont la taille est inférieure à celle spécifiée mais uniquement à l'affichage : cela ne réduit pas l'intervalle de validité des valeurs qui peuvent être stockées dans la colonne.

Lorsqu'elle est utilisée avec l'attribut de colonne optionnel `ZEROFILL` , le caractère de remplissage par défaut est remplacé par le caractère zéro. Par exemple, pour une colonne dont le type est `INT(5) ZEROFILL` , la valeur `4` sera lue `00004` .

Notez que si vous stockez des nombres plus grands que la taille maximale d'affichage, vous pouvez rencontrer des problèmes lors de jointures de tables particulièrement compliquées, surtout si MySQL génére des tables temporaires : dans ce cas, MySQL pense que les données étaient limitées par l'affichage.

Tous les types entiers ont un attribut optionnel (non−standard) `UNSIGNED` (non−signé, en français). Les valeurs non−signées peuvent être utilisées pour n'autoriser que des valeurs positives dans une colonne, ou bien pour exploiter un intervalle de validité plus haut. Depuis la version 4.0.2 de MySQL, les nombres à virgule flottante peuvent aussi être `UNSIGNED` Comme avec les types entiers, cet attribut interdit les valeurs négatives dans la colonne, mais n'élève pas l'intervalle de validité.

Le type FLOAT est utilisé pour représenter des données numériques approchées. La norme ANSI/ISO SQL92 permet la spécification optionnelle de la précision (mais pas de l'intervalle de validité) en fournissant le nombre de décimales voulues après la spécification de type, et entre parenthèses. L'implémentation de MySQL supporte aussi le paramétrage de la précision. Si le mot clé `FLOAT` est utilisé pour une colonne sans précision supplémentaire, MySQL utilise quatre octets pour stocker les valeurs. Une syntaxe alternative existe aussi, elle utilise deux paramètre optionnel après le mot clé `FLOAT` . Avec cette option, le premier nombre représente toujours la taille de stockage nécessaire pour la valeur, et le second nombre représente le nombre de chiffres à stocker et afficher, après la virgule décimale (comme pour les types `DECIMAL` et `NUMERIC` ). Lorsque MySQL stocke un nombre pour une telle colonne, et que cette valeur a plus de décimale que requis, la valeur est arrondie pour éliminer les chiffres surnuméraires.

Les types `REAL` et `DOUBLE PRECISION` n'acceptent pas de paramétrage de la précision. En extension du standard ANSI/ISO SQL92, MySQL reconnaît `DOUBLE` comme un synonyme du type `DOUBLE PRECISION` . Contrairement à la norme qui requiert que `REAL` soit plus petit que `DOUBLE PRECISION` , MySQL implémente ces deux types comme des nombres à virgule flottante de 8 octets, en double précision (lorsque le mode ``ANSI'' n'est pas activé). Pour une portabilité maximale, les applications réclamant le stockage de nombres approché doivent utiliser les types `FLOAT` ou `DOUBLE PRECISION` sans spécification de précision ou de nombre de décimales.

Lorsque MySQL doit stocker une valeur qui est hors de l'intervalle de validité d'une colonne, il ramène la valeur à la plus proche possible, et stocke cette valeur. Par exemple, l'intervalle de validité d'une colonne d'entiers `INT` va de `−2147483648` à `2147483647` . Si vous essayez d'insérer `−9999999999` dans une colonne de ce type, la valeur sera ramenée à la plus proche possible, c'est à dire `−2147483648` . De même, si vous essayez d'insérer `9999999999` , `2147483647` sera stocké à la place.

Si la colonne `INT` possède l'attribut `UNSIGNED` , l'intervalle de validité est aussi large, mais les valeurs

extrêmes se décalent vers `0` et `4294967295` . Si vous essayez de stocker `-9999999999` et `9999999999` dans cette colonne, vous obtiendrez respectivement `0` et `4294967296` .

Les dépassements de capacité entraînant des tronquage sont affichés comme des alertes (``warnings'') lors de l'utilisation des commandes `ALTER TABLE` , `LOAD DATA INFILE` , `UPDATE` , et les insertions `INSERT` multiples.

| Type | Octets | De | A |
|---|---|---|---|
| TINYINT | 1 | -128 | 127 |
| SMALLINT | 2 | -32768 | 32767 |
| MEDIUMINT | 3 | -8388608 | 8388607 |
| INT | 4 | -2147483648 | 2147483647 |
| BIGINT | 8 | -9223372036854775808 | 9223372036854775807 |

## 6.2.2 Les types date et heure

Les types dates et heures sont `DATETIME` , `DATE` , `TIMESTAMP` , `TIME` , et `YEAR` . Chacun d'eux à une échelle de valeurs légales, de même que la valeur ``zéro'' quand vous spécifiez une valeur illégale. A noter que MySQL vous permet d'enregistrer certaines dates qui ne sont pas strictement légales, par exemple `1999-11-31` . La raison est que nous pensons que la vérification des dates est à faire niveau application. Pour accélérer les tests, MySQL vérifie juste que le mois est entre 0 et 12 et que le jour est entre 0 et 31. Les intervalles précédentes sont définies de cette façon car MySQL vous permet d'enregistrer dans une colonne `DATE` ou `DATETIME` , des dates où le jour de la semaine ou le jour du mois est zéro. C'est extrêmement utile pour les applications où vous avez besoin d'enregistrer une date d'anniversaire pour laquelle vous n'avez pas la date exacte. Dans ce cas, vous enregistrez simplement la date comme `1999-00-00` ou `1999-01-00` . (Vous ne devez pas vous attendre à obtenir de valeurs correctes de fonctions tel que `DATE_SUB()` ou `DATE_ADD` pour des dates comme cela.)

Voici quelques considérations à garder à l'esprit quand vous manipulerez ce type de champs :

- MySQL extrait les valeurs d'un champ date ou heure dans un format standard, mais essaye d'interpréter une grande variétés de format pour les valeurs que vous donnez (par exemple, quand vous essayez de comparer ou attribuer une valeur à un champ de type date ou heure). Néanmoins, seul les formats décrits dans les sections suivantes sont disponibles. Il est attendu que vous fournissiez des valeurs légales et des erreurs imprévues peuvent survenir si vous utilisez d'autre formats.
- Même si MySQL essaye d'interpreter les valeurs sous différents formats, il s'attend toujours à ce que l'année soit dans la partie gauche de la valeur. Les dates doivent êtres données sous la forme année-mois-jour (exemple : `98-09-04` ), au lieu de mois-jour-année ou jour-mois-année qui sont très utilisés ailleurs (comme `09-04-98` ou `'04-09-98'` ).
- MySQL convertit automatiquement une date ou heure en nombre si la valeur est utilisée dans un contexte numérique et vice versa.
- Lorsque MySQL rencontre une valeur hors d'intervalle pour un type date ou heure qui est donc illégale pour ce type (voir le début de cette section), il la convertit à la valeur ``zéro'' de ce type. (L'exception est que les valeurs hors intervalles pour les champs `TIME` sont coupées à la limite appropriée des valeurs de `TIME` .) Le tableau suivant présente le format de la valeur ``zéro'' de chaque type :

| Column type | valeur du ``zéro'' |
|---|---|
| DATETIME | '0000-00-00 00:00:00' |
| DATE | '0000-00-00' |
| TIMESTAMP | 00000000000000 (la longueur dépend de la taille de l'affichage) |
| TIME | '00:00:00' |
| YEAR | 0000 |

- La valeur ``zéro'' est spéciale, mais vous pouvez l'enregistrer ou vous y référer explicitement en utilisant les valeurs contenues dans le tableau ci dessus. Vous pouvez aussi le faire en utilisant la valeur `'0'` ou `0` qui est plus facile à manipuler.

- La date ou le temps ``Zéro'' utilisé avec MyODBC est automatiquement convertie en NULL à partir de la version 2.50.12 de MyODBC , car ODBC ne peut manipuler de telles valeurs.

#### 6.2.2.1 An 2000 et les types date

MySQL lui même est compatible an 2000. ( Compatibilité an 2000 ), mais les valeurs manipulées par MySQL peuvent ne pas l'être. N'importe quelle valeur n'ayant que deux chiffres pour représenter l'année est ambigu, car le siècle n'est pas précisé. Ces valeurs doivent être interprétées comme des valeurs à 4 chiffres, car MySQL stocke les années en interne en utilisant 4 chiffres.

Pour les types DATETIME , DATE , TIMESTAMP , et YEAR , MySQL interprète les dates ambigus en se basant sur les règles suivantes :

- Les valeurs d'années comprises dans l'intervalle 00-69 sont converties en 2000-2069 .
- Les valeurs d'années comprises dans l'intervalle 70-99 sont converties en 1970-1999 .

Gardez bien à l'esprit que ces règles ne sont que la meilleure approximation possible d'une valeur. Si l'heuristique proposée par MySQL ne fournit pas les valeurs attendues, vous devrez fournir une valeur sans ambiguïté. (à 4 chiffres) ORDER BY ordonnera correctement les types YEAR/DATE/DATETIME à deux chiffres.

Notez aussi que quelques fonctions comme MIN() et MAX() convertiront un TIMESTAMP/DATE en nombre. Cela signifie qu'un timestamp avec une année à deux chiffres ne donneront pas de résultats corrects avec ces fonctions. Une solution dans ce cas est de convertir le TIMESTAMP/DATE en une année à 4 chiffres ou d'utiliser quelque chose comme MIN(DATE_ADD(timestamp,INTERVAL 0 DAYS)) .

#### 6.2.2.2 Les types DATETIME , DATE , et TIMESTAMP

Les types DATETIME , DATE , et TIMESTAMP sont liés. Cette section décrit leurs caractéristiques, leur similarités et leurs différences.

Le type DATETIME est prévu lorsque vous souhaitez stocker une date et une heure. MySQL affiche les valeurs de type DATETIME au format ' AAAA-MM-JJ HH:MM:SS '. L'intervalle de validité va de ' 1000-01-01 00:00:00 ' à ' 9999-12-31 23:59:59 '. (``validité'' signifie que même si d'autres valeurs plus anciennes peuvent être manipulées, il n'est pas garantit qu'elles le seront).

Le type DATE est prévu lorsque vous souhaitez stocker une date. MySQL affiche les valeurs de type DATE au format ' AAAA-MM-JJ '. L'intervalle de validité va de '1000-01-01' à '9999-12-31' .

Le type TIMESTAMP est prévu pour stocker automatiquement l'heure courante lors d'une commande INSERT ou UPDATE . Si vous avez plusieurs colonnes de type TIMESTAMP , seule la première colonne sera mise à jour automatiquement.

La modification automatique de la première colonne de type TIMESTAMP survient si l'une des conditions suivantes est remplie :

- La colonne n'est pas spécifiée explicitement dans la commande INSERT ou LOAD DATA INFILE .
- La colonne n'est pas spécifiée explicitement dans la commande UPDATE et d'autres colonnes changent de valeurs (Notez qu'une commande UPDATE qui affecte une valeur qui est déjà celle de la colonne sera ignorée, et la colonne TIMESTAMP ne sera pas modifiée, car la ligne n'est pas à proprement parlée modifiée. MySQL ignore alors ces modifications pour des raisons d'efficacité).

- La colonne `TIMESTAMP` est spécifiée explicitement à `NULL` .

Les autres colonnes de type `TIMESTAMP` , hormis la première, peuvent aussi prendre la valeur courante. Affectez lui alors la valeur `NULL` ou la fonction `NOW()` .

Vous pouvez affecter à n'importe quelle colonne de type `TIMESTAMP` une valeur différente de l'heure et la date courant en fournissant une valeur explicite. Cela s'applique aussi à la première colonne de type `TIMESTAMP` . Par exemple, si vous voulez affecter la date de création d'une ligne à une colonne de type `TIMESTAMP` , mais ne plus y toucher ultérieurement :

- Laissez MySQL donner la valeur de la colonne lors de la création de la ligne. Cela va initialiser la colonne à la date et heure courante.
- Lorsque vous faites des modifications ultérieures, affectez explicitement à la colonne `TIMESTAMP` sa propre valeur.

D'un autre coté, vous pouvez aussi facilement initialiser la colonne `TIMESTAMP` avec `NOW()` lors de sa création, puis ne plus la modifier ultérieurement.

L'intervalle de validité des valeurs `TIMESTAMP` va du début de l'année 1970 jusque quelque part durant l'année 2037, avec une précision d'une seconde. Les valeurs sont affichés comme des nombres entiers.

Le format d'affichage des valeurs `TIMESTAMP` dépend de la taille d'affichage, comme illustré ci−dessous. Le format total `TIMESTAMP` a 14 chiffres, mais les colonnes `TIMESTAMP` peuvent être créées avec des formats plus courts :

| Type de colonne | Format d'affichage |
|---|---|
| TIMESTAMP(14) | YYYYMMDDHHMMSS |
| TIMESTAMP(12) | YYMMDDHHMMSS |
| TIMESTAMP(10) | YYMMDDHHMM |
| TIMESTAMP(8) | YYYYMMDD |
| TIMESTAMP(6) | YYMMDD |
| TIMESTAMP(4) | YYMM |
| TIMESTAMP(2) | YY |

Toutes les colonnes de type `TIMESTAMP` ont la même taille de stockage, indépendamment de la taille d'affichage. Les formats les plus courants sont 6, 8, 12, et 14. Vous pouvez spécifier une taille arbitraire lors de la création de la table, mais 0 et les valeurs supérieures à 14 sont ramenées à 14. Les valeurs impaires sont aussi ramenées au nombre pair supérieur.

Vous pouvez spécifier les valeurs des colonnes `DATETIME` , `DATE` et `TIMESTAMP` , avec les formats communs suivants :

- Une chaîne au format `'AAAA−MM−JJ HH:MM:SS'` ou `'AA−MM−JJ HH:MM:SS'` . Une syntaxe ``relaxée'' est premise : tout caractère de ponctuation peut être utilisé comme délimiteur entre les parties de temps ou heure. Par exemple, `'98−12−31 11:30:45'` , `'98.12.31 11+30+45'` , `'98/12/31 11*30*45'` , et `'98@12@31 11^30^45'` sont équivalents.
- Une chaîne au format `'AAAA−MM−JJ HH:MM:SS'` ou `'AA−MM−JJ HH:MM:SS'`. Une syntaxe plus sympas est aussi permise : vous pouvez utiliser n'importe quel caractère de ponctuation comme délimiteur de date et d'heure. Par exemple, `'98/12/31 11*30*45'`, et `'98@12@31 11^30^45'` sont équivalents.
- Une chaîne au format `'AAAA−MM−JJ'` ou `'AA−MM−JJ'`. Une syntaxe plus sympas est aussi acceptée ici. Par exemple, `'98−12−31'`, `'98.12.31'`, `'98/12/31'`, et `'98@12@31'` sont équivalent.
- Une chaîne sans aucun délimiteurs sous la forme `'AAAAMMJJHHMMSS'` ou `'AAMMJJHHMMSS'`, en supposant qu'une telle chaîne ait un sens en terme de date. Par exemple `'19970523091528'` et `'970523091528'` sont interprétés comme `'1997−05−23 09:15:28'`, mais `'971122129015'` est invalide (les minutes ne sont pas valides) et devient alors `'0000−00−00 00:00:00'`.

- Une chaîne sans aucun délimiteurs sous la forme ' AAAAMMJJ ' ou ' AAMMJJ ', en supposant qu'une telle chaîne ait un sens en terme de date. Par exemple, ' 19970523 ' et ' 970523 ' sont interprétés comme ' 1997-05-23 ', mais ' 971332 ' est invalide (les mois ne sont pas valides) et devient alors ' 0000-00-00 '.
- Un nombre au format AAAAMMJJHHMMSS ou AAMMJJHHMMSS , en supposant qu'un tel nombre ait un sens en terme de date. Par exemple, 19830905132800 et 830905132800 sont interprétés comme ' 1983-09-05 13:28:00 '.
- Un nombre au format AAAAMMJJ ou AAMMJJ en supposant qu'un tel nombre ait un sens en terme de date. Par exemple, 19830905 et 830905 sont interprétés comme ' 1983-09-05 '.
- Un résultat de fonction qui retourne une valeur acceptable dans une colonne de type DATETIME , DATE , ou TIMESTAMP , tels que NOW() ou CURRENT_DATE .

Les valeurs invalides DATETIME , DATE , ou TIMESTAMP sont remplacées par la date ``zéro'' du type approprié (respectivement ' 0000-00-00 00:00:00 ', ' 0000-00-00 ', ou 00000000000000 ).

Pour la valeurs spécifiées sous forme de chaînes avec des délimiteurs de date, il n'est pas nécessaire de spécifier les deux chiffres pour les mois ou les dates qui sont inférieurs à 10 . Par exemple, ' 1979-6-9 ' est valide et est équivalent à ' 1979-06-09 '. Similairement, pour les valeurs spécifiées sous forme de chaîne avec des délimiteurs d'heure, il n'est pas obligatoire de spécifier les deux chiffres des heures, minutes et secondes qui sont inférieures à 10 . ' 1979-10-30 1:2:3 ' est valide et est équivalent à ' 1979-10-30 01:02:03 '.

Les valeurs spécifiées sous forme de nombres doivent avoir 6, 8, 12, ou 14 chiffres de long. Si le nombre a 8 ou 14 chiffres, MySQL suppose que le format est AAAAMMJJ ou AAAAMMJJHHMMSS (respectivement) et que l'année est représentées par les 4 premiers chiffres. Si le nombre a 6 ou 12 chiffres, MySQL suppose que le format est AAMMJJ ou AAMMJJHHMMSS (respectivement) et format et que l'année est représentées par les 2 premiers chiffres. Les nombres qui ne sont pas d'une taille valide, sont complétés avec des 0 jusqu'à la taille lisible la plus proche. Les valeurs spécifiées sous forme de chaînes sans délimiteurs sont interprétés en fonction de leur taille. Si la chaîne à 8 ou 14 caractères de long, l'année est supposée avoir 4 chiffres. Sinon, l'année est supposée avoir 2 chiffres. La chaîne est interprétée de gauche à droite, en lisant successivement l'année, le mois, la date, l'heure, les minutes et les secondes, tant qu'il y a des valeurs dans la chaîne. Cela signifie que vous ne devez pas utiliser de chaînes qui ont moins de 6 caractères. Par exemple, si vous spécifiez ' 9903 ', en pensant qu'il représente Mars 1999, vous vous apercevrez que MySQL insère à la place la date ``zéro'' dans votre table. Cela est dû au fait que si l'année et le mois sont 99 et 03, la date est 0, ce qui en fait une date invalide, qui est rejetée par MySQL.

Les colonnes TIMESTAMP stocke des valeurs légales en utilisant la précision maximum fournie, indépendamment de la taille d'affichage. Cela a plusieurs implications :

- Spécifiez toujours l'année, le mois et le jour, même si le type de colonne est TIMESTAMP(4) ou TIMESTAMP(2) . Sinon, la valeur ne sera pas légale et 0 sera stockée.
- Si vous utilisez la commande ALTER TABLE pour réduire la largeur d'une colonne TIMESTAMP , les informations qui étaient affichées sont désormais ``cachées'', mais pas détruites.
- Similairement, réduire une colonne de type TIMESTAMP ne cause aucune perte d'information, en dehors du fait que ces informations ne sont plus affichées.
- Bien que les valeurs TIMESTAMP soient stockées avec une précision d'une seconde, la seule fonction qui travaille directement avec ces valeurs est la fonction UNIX_TIMESTAMP() . Les autres fonctions opèrent sur des valeurs lues et formatées. Cela signifie que vous ne pouvez pas utiliser de fonctions telles que HOUR() ou SECOND() a moins que le format d'affichage de la valeur TIMESTAMP ne présente cette valeur. Par exemple, les heures ne sont jamais affichées dans une colonne de type TIMESTAMP à moins que la taille d'affichage de la colonne ne soit d'au moins 10. L'utilisation de la fonction HOUR() sur une valeur ayant un format d'affichage plus court que 10 retournera un résultat inutilisable.

Dans une certaines mesure, vous pouvez assigner des valeurs d'une colonne à une autre colonne d'un autre type. Cependant, vous devez vous attendre à quelques altération ou pertes de valeurs durant la conversion :

- Si vous assignez une valeur `DATE` à une colonne de type `DATETIME` ou `TIMESTAMP` , la partie représentant les heures vaudra ' `00:00:00` ', car les colonnes de type `DATE` ne contiennent pas d'information d'heure.
- Si vous assignez une valeur `DATETIME` ou `TIMESTAMP` à une colonne de type `DATE` , la composante heure sera perdue, car les colonnes de type `DATE` ne contiennent pas d'information d'heure.
- N'oubliez pas que même si les valeurs `DATETIME` , `DATE` , et `TIMESTAMP` peuvent être spécifiée avec différents formats, ces types n'ont pas les mêmes intervalle de validité. Par exemple, les valeurs de type `TIMESTAMP` ne peuvent pas prendre de valeur antérieure à 1970 ou postérieure à 2037. Cela signifie qu'une date telle que ' `1968-01-01` ', est légale dans les colonnes de type `DATETIME` , mais n'est pas valide pour les `TIMESTAMP` , et sera convertie en date zéro ( `0` ) si elle est assignée à une telle colonne.

Attention à certains pièges concernant les spécifications de dates :

- La syntaxe à délimiteur libre peut être une source de problème. Par exemple, une valeur telle que ' `10:11:12` ' ressemble à une heure, à cause du délimiteur ` `':'` ', mais avec une colonne de date, elle sera interprétée comme la date ' `2010-11-12` '. La valeur ' `10:45:15` ' sera convertie en ' `0000-00-00` ' car ' `45` ' n'est pas un mois valide.
- Le serveur MySQL effectue seulement la vérification de base la validité d'une date : jours `00-31` , mois `00-12` , années `1000-9999` . N'importe quelle date qui n'est pas dans cette marge retournera `0000-00-00` . Veuillez noter que ceci vous permet toujours de stocker les dates inadmissibles telles que `2002-04-31` . Il permet à des applications web de stocker des données d'une forme sans vérifier plus loin. Pour s'assurer qu'une date est valide, vous devrez effectuer un test dans votre application.
- Les années spécifiée avec deux chiffres seulement sont ambigues, car il manque le siècle. MySQL interprète les années à deux chiffres suivant ces règles :
    - Les années de l'intervalle `00-69` sont converties en `2000-2069` .
    - Les années de l'intervalle `70-99` sont converties en `1970-1999` .

### 6.2.2.3 Le type TIME

MySQL lit et affiche les colonnes de type `TIME` au format `'HH:MM:SS'` (ou `'HHH:MM:SS'` pour les grandes quantités d'heures). Les valeurs de `TIME` vont de `'-838:59:59'` à `'838:59:59'` . La raison de cet intervalle de validité si large est que les colonnes de type `TIME` peuvent être utilisés pour représenter non seulement des heures du jour, mais aussi des durées entre deux événements (ce qui peut dépasser largement les 24 heures, ou même, être négatif).

Vous pouvez spécifier une valeur de type `TIME` avec différents formats :

- Une chaîne au format `'D HH:MM:SS.fraction'` . (Notez que MySQL ne stockera pas la fraction d'une valeur `TIME` .)

  Vous pouvez aussi utiliser l'une des syntaxes alternatives suivantes : `HH:MM:SS.fraction` , `HH:MM:SS` , `HH:MM` , `D HH:MM:SS` , `D HH:MM` , `D HH` ou `SS` . Ici, `D` peut prendre des valeurs entre 0 et 33.
- Une chaîne sans délimiteur au format `'HHMMSS'` , en supposant que cela puisse avoir un sens en terme de date. Par exemple, `'101112'` est interprété comme `'10:11:12'` , mais `'109712'` est invalide (le nombre de minutes n'a pas de sens), et devient la date zéro : `'00:00:00'` .
- Un nombre au format `HHMMSS` , en supposant que cela puisse avoir un sens en terme de date. Par exemple, `101112` est interprété comme `'10:11:12'` . Les formats alternatifs sont aussi compris

: `SS` , `MMSS` , `HHMMSS` et `HHMMSS.fraction` . Notez que MySQL ne stocke pas encore les fractions de secondes.
- Le résultat d'une fonction qui retourne une valeur acceptable dans un contexte de valeurs `TIME` , comme `CURRENT_TIME` .

Pour les valeurs `TIME` spécifiées avec des délimiteurs, il n'est pas nécessaire de préciser deux chiffres pour les valeurs inférieurs à `10` pour les heures, minutes et secondes. `'8:3:2'` est la même chose que `'08:03:02'` .

Soyez soigneux lors de l'utilisation de valeurs ``courtes'' à une colonne de type `TIME` . MySQL interprète les valeurs en supposant que les chiffres de droite représentent les secondes (MySQL interprète les valeurs `TIME` comme des durées et non comme des heures d'une journée). Par exemple, vous pouvez penser que les valeurs `'11:12'` , `'1112'` et `1112` représentent `'11:12:00'` (12 minutes après 11 heures), mais MySQL les interprétera comme `'00:11:12'` (11 minutes, 12 secondes). Similairement, `'12'` et `12` représentent `'00:00:12'` . Les valeurs de `TIME` déclarées avec des `:` , au contraire, sont toujours traités comme des heures de journée. `'11:12'` signifiera `'11:12:00'` et non pas `00:11:12`

Les valeurs hors de l'intervalle de validité de `TIME` mais qui sont valides sont ramenées à la valeur maximale stockable la plus proche. Par exemple, `'−850:00:00'` et `'850:00:00'` sont respectivement converties en `'−838:59:59'` et `'838:59:59'` .

Les valeurs `TIME` non valides sont transformées en date zéro `'00:00:00'` . Notez que comme `'00:00:00'` est elle−même une valeur `TIME` valide, vous n'aurez pas le moyen de faire la différence entre une valeur `'00:00:00'` stockée en connaissance de cause, et `'00:00:00'` stockée à cause d'une erreur.

### 6.2.2.4 Le type YEAR

Le type `YEAR` est un type d'1 byte utilisé pour représenter les années.

MySQL extrait et affiche la valeur de `YEAR` au format `YYYY` . L'échelle va de `1901` à `2155` .

Vous pouvez spécifier la valeur de `YEAR` en plusieurs formats :

- Une chaîne de quatre chiffres entre `'1901'` et `'2155'` .
- Un nombre à quatre chiffres entre `1901` et `2155` .
- Une chaîne de deux chiffres entre `'00'` et `'99'` . Les valeurs entre `'00'` et `'69'` et entre `'70'` et `'99'` sont respectivement converties en valeurs `YEAR` comprises entre `2000` et `2069` d'une part, et `1970` et `1999` de l'autre.
- Une nombre de deux chiffres entre `1` et `99` . Les valeurs entre `1` et `69` et entre `70` et `99` sont respectivement converties en valeurs `YEAR` comprises entre `2001` et `2069` d'une part, et `1970` et `1999` d'autre part. Notez que le rang de valeurs pour les nombres à deux chiffres est totalement différent du rang pour les chaînes à deux chiffres parce que vous ne pouvez pas spécifier deux zéro directement en tant que nombre et le faire interpréter en tant que `2000` . Vous **devez** le spécifier comme chaîne `'0'` ou `'00'` sinon il sera interprété comme `0000` .
- En tant que résultat d'une fonction retournant une valeur acceptable dans le contexte de `YEAR` , comme as `NOW()` .

Les valeurs illégales pour `YEAR` sont converties en `0000` .

## 6.2.3 Les types chaînes

Les types chaînes sont CHAR , VARCHAR , BLOB , TEXT , ENUM , et SET . Cette section décrit comment ces types fonctionnent, leur besoins en espace et comment les utiliser dans vos requêtes.

| Type | Taille maximale | Bytes |
|---|---|---|
| TINYTEXT ou TINYBLOB | 2^8-1 | 255 |
| TEXT ou BLOB | 2^16-1 (64K-1) | 65535 |
| MEDIUMTEXT ou MEDIUMBLOB | 2^24-1 (16M-1) | 16777215 |
| LONGBLOB | 2^32-1 (4G-1) | 4294967295 |

### 6.2.3.1 Les types CHAR et VARCHAR

Les types CHAR et VARCHAR sont similaires, mais diffèrent dans la manière dont ils sont stockés et récupérés.

La longueur d'une colonne CHAR est fixée à la longueur que vous avez défini lors de la création de la table. La longueur peut être n'importe quelle valeur entre 1 and 255. (Dans la version 3.23 de MySQL, la longueuer est comprise entre 0 et 255.) Quand une valeur CHAR est enregistrée, elle est complété à droite avec des espaces jusqu'à atteindre la valeur fixée. Quand une valeur de CHAR est lue, les espaces en trop sont retirés.

Les valeurs contenues dans les colonnes de type VARCHAR sont de tailles variables. Vous pouvez déclarer une colonne VARCHAR pour que sa taille soit comprise entre 1 et 255, exactement comme pour les colonnes CHAR . Par contre, contrairement à CHAR , les valeurs de VARCHAR sont stockées en utilisant autant de caractères que nécessaire, plus un octet pour mémoriser la longueur. Les valeurs ne sont pas complétées. Au contraire, les espaces finaux sont supprimés avant stockage (ce qui ne fait pas partie des spécifications ANSI SQL).

Si vous assignez une chaîne de caractères qui dépasse la capacité de la colonne CHAR ou VARCHAR , celle ci est tronquée jusqu'à la taille maximale du champ.

Le tableau suivant illustre les différences entre les deux types de colonnes en montrant les différences entre l'enregistrement dans une colonne CHAR(4) ou VARCHAR(4) :

| Valeur | CHAR(4) | Espace requis | VARCHAR(4) | Espace requis |
|---|---|---|---|---|
| '' | '  ' | 4 octets | '' | 1 octet |
| 'ab' | 'ab  ' | 4 octets | 'ab' | 3 octets |
| 'abcd' | 'abcd' | 4 octets | 'abcd' | 5 octets |
| 'abcdefgh' | 'abcd' | 4 octets | 'abcd' | 5 octets |

Les valeurs lues dans les colonnes de type CHAR(4) et VARCHAR(4) seront les mêmes dans tous les cas, car les espaces finaux sont retirés des valeurs issues de colonnes de type CHAR lors de la lecture.

Les valeurs dans les colonnes CHAR et VARCHAR sont classées et comparées sans tenir compte de la casse, à moins que l'attribut BINARY n'ai été spécifié lors de la création de la table. L'attribut BINARY signifie que les valeurs sont classées et triées en tenant compte de la casse, suivant l'ordre des caractères ASCII de la machine ou est installé le serveur MySQL. BINARY n'affecte pas les méthodes de lecture et de stockage des valeurs.

L'attribut BINARY se propage dans une expression : il suffit qu'une seule colonne, utilisée dans une expression, ait l'attribut BINARY pour que toute l'expression ne tienne plus compte de la casse.

MySQL peut changer automatiquement le type d'une colonne CHAR ou VARCHAR lors de la création de la table. Silent column changes .

### 6.2.3.2 Les types BLOB et TEXT

Une valeur de type BLOB est un objet binaire de grande taille, qui peut contenir une quantité variable de données. Les quatre types BLOB ( TINYBLOB , BLOB , MEDIUMBLOB , et LONGBLOB ) ne différent que par la taille maximale de données qu'ils peuvent stocker. Capacités des colonnes .

Les quatre types TEXT ( TINYTEXT , TEXT , MEDIUMTEXT , et LONGTEXT correspondent aux types BLOB équivalents, et ont les mêmes contraintes de stockage. Les seules différences entre les colonnes de type BLOB et celles de type TEXT se situent aux niveau des tris et comparaisons : Les tris, faits sur les BLOB , contrairement à ceux faits sur les TEXT , tiennent compte de la casse. En d'autres termes, une valeur TEXT est une valeur BLOB insensible à la casse.

Si vous assignez une valeur trop grande à une colonne de type BLOB ou TEXT , la valeur sera tronquée à la taille maximale possible.

Dans la majorité des cas, vous pouvez considérer une colonne de type TEXT comme une colonne de type VARCHAR , aussi grande que vous le souhaitez. De même, vous pouvez considérer une colonne de type BLOB comme une colonne de type VARCHAR BINARY . Les seules différences sont :

- Vous ne pouvez indexer les colonnes de type BLOB ou TEXT qu'à partir de la version 3.23.2 de MySQL.
- Il n'y a pas de suppression des espaces finaux lors du stockage de valeur dans des colonnes de type BLOB et TEXT , ce qui est le cas dans pour les colonnes de type VARCHAR .
- Les colonnes BLOB et TEXT ne peuvent avoir de valeur par défaut. ( DEFAULT )

MyODBC considère les valeurs BLOB comme des LONGVARBINARY et les valeurs TEXT comme des LONGVARCHAR .

Vous pouvez rencontrer les problèmes suivants, à cause de la grande taille des colonnes de type BLOB et TEXT , lors de leur utilisation :

- Si vous voulez utiliser les commandes GROUP BY ou ORDER BY sur une colonne de type BLOB ou TEXT , vous devez d'abord la convertir en un objet de taille fixe. Le meilleur moyen est d'utiliser la fonction SUBSTRING . Par exemple :
```
mysql> SELECT comment FROM nom_de_table,SUBSTRING(comment,20) AS substr
    ->                ORDER BY substr;
```
  Si vous le ne faites pas, seuls les max_sort_length premiers octets de la colonne seront utilisés pour le tri. La valeur par défaut de max_sort_length est 1024. Cette valeur peut être modifiée en utilisant l'option -O au démarrage du serveur mysqld . Vous pouvez utiliser la commande GROUP BY sur une colonne de type BLOB ou TEXT en spécifiant la position de la colonne, ou avec un alias :
```
 mysql> SELECT id,SUBSTRING(blob_col,1,100) FROM nom_de_table GROUP BY 2;
mysql> SELECT id,SUBSTRING(blob_col,1,100) AS b FROM nom_de_table GROUP BY b;
```
- La taille maximale d'un objet BLOB ou TEXT est déterminée par son type, mais la valeur la plus grande que vous pouvez transmettre au programme client est déterminée par la quantité de mémoire disponible sur le serveur et par les tailles des buffers de communication. Vous pouvez changer la taille des buffers de communication, mais vous devez le faire sur le serveur et le client en même temps. Choix des paramètres du serveur .

Notez que chaque valeur BLOB ou TEXT est représentée en interne par un objet alloué séparément, contrairement à tous les autres types de colonne, pour lesquels la place de stockage est allouée une fois pour chaque colonne, lorsque la table est ouverte.

### 6.2.3.3 Le type ENUM

Une énumération ENUM est une chaîne dont la valeur est choisie parmi une liste de valeurs autorisées lors de la création de la table.

Cette chaîne peut aussi être la chaîne vide ( `""` ) ou `NULL` dans certaines circonstances :

- Si vous insérez une valeur illégale dans une énumération `ENUM` (c'est à dire, une chaîne qui n'est pas dans la liste de valeurs autorisées), la chaîne vide est insérée pour représenter une erreur. Cette chaîne peut être distinguée d'une chaîne vide 'normale' par le fait que cette chaîne à la valeur numérique 0. Nous reviendrons sur ce point plus tard.
- Si une colonne d'énumération est déclarée `NULL` , `NULL` devient aussi une valeur autorisée, et la valeur par défaut est alors `NULL` . Si une colonne d'énumération est déclarée `NOT NULL` , la valeur par défaut est le premier élément de la liste des valeurs autorisées.

Chaque élément de l'énumération dispose d'un index :

- Les valeurs de la liste des valeurs autorisées sont indexées à partir de 1.
- L'index de la chaîne vide (cas d'erreur) est 0. Cela signifie que vous pouvez utiliser la sélection suivante pour repérer les valeurs d'énumération invalides :
```
mysql> SELECT * FROM nom_de_table WHERE enum_col=0;
```
- L'index de la valeur `NULL` est `NULL` .

Par exemple, une colonne créée comme `ENUM("un", "deux", "trois")` peut prendre n'importe quelle valeur ci−dessous. L'index de chaque valeur est aussi présenté :

| Valeur | Index |
|---|---|
| NULL | NULL |
| "" | 0 |
| "un" | 1 |
| "deux" | 2 |
| "trois" | 3 |

Une énumération peut avoir un maximum de 65535 éléments.

A partir de la version 3.23.51, les espaces en début et fin de chaîne sont automatiquement supprimés des éléments de l'énumération `ENUM` lorsque la table est créée.

La casse des lettres est sans importance lors de l'assignation de valeurs dans une énumération. Cependant, les valeurs lues dans la base auront la même casse que celle spécifiée lors de la création de la table.

Si vous lisez le contenu d'une énumération dans un contexte numérique, l'index de la valeur `ENUM` sera retournée. Par exemple, vous pouvez lire des valeurs numériques comme ceci :

```
mysql> SELECT enum_col+0 FROM nom_de_table;
```

Si vous stockez un nombre dans une colonne de type `ENUM` , le nombre sera traité comme un index, et la valeur stockée sera celle de l'élément ayant cet index (Attention, cela ne fonctionnera pas avec les commandes `LOAD DATA` , car cette dernière traite toutes les valeurs comme des chaînes). Il est déconseillé de stocker des valeurs numériques dans un `ENUM` car cela engendre des confusions.

Les valeurs de type `ENUM` sont triées en fonction de l'ordre des éléments, fixé à la création de la table (en d'autres termes, les valeurs `ENUM` sont stockées en fonction de leur index). Par exemple, `"a"` précède `"b"` dans l'énumération `ENUM("a", "b")` , mais `"b"` précède `"a"` dans l'énumération `ENUM("b", "a")` . La chaîne vide précède toujours les chaînes non vides, et `NULL` précède toutes les valeurs.

Si vous voulez connaître toutes les valeurs possibles d'une colonne de type `ENUM` , pensez à utiliser cette commande : `SHOW COLUMNS FROM nom_de_table LIKE enum_column_name` , puis analysez la définition de la colonne de type `ENUM` (deuxième colonne dans le résultat).

### 6.2.3.4 Le type SET

Un `SET` est une chaîne qui peut avoir zéro ou plusieurs valeurs, chacune doit être choisie dans une liste de valeurs définies lors de la création de la table. Les valeurs des colonnes `SET` composées de plusieurs membres sont définies en séparant celles−ci avec des virgules ( `','` ). Ce qui fait que la valeur d'un membre de `SET` ne peut contenir lui même de virgule.

Par exemple, une colonne définie en tant que `SET("un", "deux") NOT NULL` peut avoir l'une de ces valeurs :

```
 ""
"un"
"deux"
"un,deux"
```

Un `SET` peut avoir au plus 64 membres.

A partir de la version 3.23.51, les espaces en trop sont automatiquement effacés des membres de `SET` lorsque la table est créée.

MySQL enregistre les valeurs de `SET` numériquement. Le bit de poids faible de la valeur correspond alors au premier élément de la liste. Si vous utilisez une valeur `SET` dans un contexte numérique, les bits des éléments dans cet ensemble seront mis à un, et les autres à zéro. Par exemple, vous pouvez obtenir un entier à partir d'un ensemble comme ceci :

```
mysql> SELECT col_set+0 FROM nom_de_table;
```

Si un nombre est enregistré dans une colonne `SET` , les bits un à un de ce nombre représenteront les éléments placés dans cet ensemble. Supposons qu'une colonne est spécifiée en tant que `SET("a","b","c","d")` , les membres ont alors les valeurs suivantes :

| `SET` **membre** | **Valeur décimale** | **Valeur binaire** |
|---|---|---|
| a | 1 | 0001 |
| b | 2 | 0010 |
| c | 4 | 0100 |
| d | 8 | 1000 |

Si vous assignez `9` à cette colonne, cela donne `1001` en binaire, ce qui fait que les valeurs du premier et quatrième membres `"a"` et `"d"` sont sélectionnés et la valeur résultante est `"a,d"` .

Pour les valeurs se composant de plus d'un membre du `SET` , l'ordre des membres n'a pas d'importance lors des insertions. Le nombre d'occurence d'un élément n'importe pas non plus. Lorsque la valeur sera lue ultérieurement, chaque élément n'apparatra qu'une seule fois, et dans l'ordre donné à la déclaration de la colonne. Par exemple, si une colonne est spécifiée comme `SET("a","b","c","d")` , alors `"a,d"` , `"d,a"` , et `"d,a,a,d,d"` seront tous représentés par `"a,d"` .

Si vous spécifiez une valeur incorrecte dans une colonne `SET` , la valeur sera ignorée.

Les valeurs de `SET` sont triées numériquement. La valeur `NULL` précède toutes les autres.

Normalement, vous exécuterez un `SELECT` sur une colonne `SET` en utilisant l'opérateur `LIKE` ou la fonction `FIND_IN_SET()` :

```
 mysql> SELECT * FROM nom_de_table WHERE set_col LIKE '%value%';
mysql> SELECT * FROM nom_de_table WHERE FIND_IN_SET('value',set_col)>0;
```

Mais ce qui suit fonctionnera aussi :

```
 mysql> SELECT * FROM nom_de_table WHERE set_col = 'val1,val2';
mysql> SELECT * FROM nom_de_table WHERE set_col & 1;
```

La première requête cherche les lignes qui correspondent exactement. La seconde ne cherche que les lignes contenant le premier membre du set.

Si vous voulez connaître toutes les valeurs possible d'une colonne `SET` , vous devez utiliser : `SHOW COLUMNS FROM nom_de_table LIKE nom_colonne_set` et étudier la définition du `SET` dans la seconde colonne.

## 6.2.4 Choisir le bon type de colonne

Pour une utilisation optimale des capacitées de stockage, essayez d'utiliser le type le plus optimal dans chaque cas. Par exemple, si une colonnes du type entier sera utilisée pour des valeurs entre `1` et `99999` , le type `MEDIUMINT UNSIGNED` sera le plus approprié.

La représentation des valeurs monétaires est un problème commun. Avec MySQL, vous devrez utiliser le type `DECIMAL` . Il est sauvegardé en tant que chaîne, aucune perte de précision ne devrait avoir lieu. Si la précision n'est pas très importante, vous pouvez utiliser le type `DOUBLE` .

Pour une haute précision, vous pouvez toujours transcrire en nombre décimaux, et les enregistrer dans des `BIGINT` . Cela vous permettra d'effectuer tout vos calculs avec des entiers et de convertir à nouveau en nombre décimaux au besoin.

## 6.2.5 Utilisation des types de données issues d'autres SGBDR

Pou faciliter l'importation de code SQL issu d'autres systèmes de gestion de bases de données, MySQL convertit les types de colonnes comme le montre le tableau suivant. Cette conversion facilite l'import de structures de tables :

| Other vendor type | MySQL type |
|---|---|
| BINARY(NUM) | CHAR(NUM) BINARY |
| CHAR VARYING(NUM) | VARCHAR(NUM) |
| FLOAT4 | FLOAT |
| FLOAT8 | DOUBLE |
| INT1 | TINYINT |
| INT2 | SMALLINT |
| INT3 | MEDIUMINT |
| INT4 | INT |
| INT8 | BIGINT |
| LONG VARBINARY | MEDIUMBLOB |
| LONG VARCHAR | MEDIUMTEXT |
| MIDDLEINT | MEDIUMINT |
| VARBINARY(NUM) | VARCHAR(NUM) BINARY |

La conversion du type de colonnes s'effectue lors de la création. Si vous créez une table avec des types issus d'un autre SGBDR puis que vous exécutez la commande `DESCRIBE nom_de_table` , MySQL fournira la structure de la table en utilisant les types équivalents.

## 6.2.6 Capacités des colonnes

Les capacités de stockage de chaque type de colonnes de MySQL sont listés par catégories.

Capacités de stockage des colonnes numériques

| Type de colonne | Espace requis |
|---|---|
| TINYINT | 1 octet |
| SMALLINT | 2 octets |
| MEDIUMINT | 3 octets |
| INT | 4 octets |
| INTEGER | 4 octets |
| BIGINT | 8 octets |

| | |
|---|---|
| FLOAT(X) | 4 if X <= 24 or 8 if 25 <= X <= 53 |
| FLOAT | 4 octets |
| DOUBLE | 8 octets |
| DOUBLE PRECISION | 8 octets |
| REAL | 8 octets |
| DECIMAL(M,D) | M+2 octets si D > 0, M+1 octets si D = 0 ( D +2, si M < D ) |
| NUMERIC(M,D) | M+2 octets si D > 0, M+1 octets si D = 0 ( D +2, si M < D ) |

Capacités de stockage des colonnes de date et heure

| Type de colonne | Espace requis |
|---|---|
| DATE | 3 octets |
| DATETIME | 8 octets |
| TIMESTAMP | 4 octets |
| TIME | 3 octets |
| YEAR | 1 octet |

Capacités de stockage des colonnes de texte

| Type de colonne | Espace requis |
|---|---|
| CHAR(M) | M octets, 1 <= M <= 255 |
| VARCHAR(M) | L +1 octets, avec L <= M et 1 <= M <= 255 |
| TINYBLOB , TINYTEXT | L +1 octets, avec L < 2^8 |
| BLOB , TEXT | L +2 octets, avec L < 2^16 |
| MEDIUMBLOB , MEDIUMTEXT | L +3 octets, avec L < 2^24 |
| LONGBLOB , LONGTEXT | L +4 octets, avec L < 2^32 |
| ENUM('valeur1','valeur2',...) | 1 ou 2 octets, suivant le nombre d'éléments de l'énumération (65535 au maximum) |
| SET('valeur1','valeur2',...) | 1, 2, 3, 4 ou 8 octets, suivant le nombre de membres de l'ensemble (64 au maximum) |

Les types VARCHAR , BLOB et TEXT sont de longueur variable, et l'espace disque requis dépend de la taille réelle de la valeur présente dans la colonne, (taille représentée par L dans le tableau précédent) et non pas de la taille maximale de la colonne. Par exemple une colonne VARCHAR(10) peut contenir une chaîne de 10 caractères. L'espace requis est dans ce cas là la longueur de la chaîne ( L ), plus 1 octet pour enregistrer la longueur de celle ci. Pour la chaîne 'abcd' , L est égal à 4 et l'espace requis est de 5 octets.

Les types BLOB et TEXT requièrent 1, 2, 3, ou 4 octets pour mémoriser la taille de la valeur dans la colonne, suivant la longueur maximale du type. Les types BLOB et TEXT .

Si une table inclus au moins une colonne de taille variable, la ligne sera de taille variable. Notez que lorsqu'une table est créée, MySQL peut, dans certaines circonstances, changer automatiquement une colonne de taille variable en colonne à taille fixe (et vice−versa). Silent column changes .

La taille d'un ENUM est détérminée par le nombre d'éléments de l'énumération. Un octet est nécessaire pour les énumérations ayant jusqu'à 255 valeurs possibles. Deux octets sont nécessaires pour les énumerations ayant jusqu'à 65535 valeurs possibles. Le type ENUM .

La taille d'un SET est détéerminé par le nombre de ses membres. Si il y'en a N , l'objet occupe (N+7)/8 octets, arrondis à 1, 2, 3, 4, or 8 octets. Un SET peut avoir au plus 64 membres. Le type SET .

La taille maximale d'une ligne dans une table MyISAM est de 65534 octets. Les colonnes BLOB et TEXT accepentent jusqu'à 5−9 octets en dessous de cette taille.

## 6.3 Fonctions à utiliser dans les clauses SELECT et WHERE

Une sélection ou une clause WHERE dans une requête SQL peut se former d'expressions utilisant les fonctions décrites ci−dessous.

Une expression contenant `NULL` produira toujours la valeur `NULL` comme résultat. (Sauf contre−indication dans le manuel)

**Note** : Il ne doit pas y avoir d'espace entre le nom d'une fonction et la parenthèse ouvrante la suivant. Cela aide le parseur MySQL à distinguer les appels à ces fonction des références aux tables ou colonnes ayant le même nom qu'une fonction. Les espaces autour des arguments sont autorisés.

Vous pouvez forcer MySQL à accepter les espaces après les nom de fonctions grâce à l'option `--ansi` de `mysqld` , ou en utilisant l'option `CLIENT_IGNORE_SPACE` avec `mysql_connect` . Dans ce cas, toutes les fonctions définies deviendront des mots strictement réservés. Exécuter MySQL en mode ANSI .

Dans un soucis de simplicité, les affichages des résultats de `mysql` sont fournis sous forme abrégée. Par exemple :

```
 mysql> SELECT MOD(29,9);
1 rows in set (0.00 sec)

 +-----------+
| mod(29,9) |
+-----------+
|         2 |
+-----------+
```

est affiché comme ceci :

```
 mysql> SELECT MOD(29,9);
        -> 2
```

## 6.3.1 Opérateurs et fonctions tout−types

### 6.3.1.1 Parenthèses

```
 ( ... )
```

Utilisez les parenthèses pour forcer l'ordre des évaluations dans une expression. Par exemple :

```
 mysql> SELECT 1+2*3;
        -> 7
mysql> SELECT (1+2)*3;
        -> 9
```

### 6.3.1.2 Opérateurs de comparaison

Les opérations de comparaison donnent comme résultats `1` (TRUE), `0` (FALSE), ou `NULL` . Ces fonctions fonctionnent pour les nombres comme pour les chaînes. Les nombres sont automatiquement transformés en chaînes et les chaînes en nombres si besoin en est. (comme en Perl)

MySQL effectue les comparaisons suivant les règles suivantes :

- Si l'un ou les deux arguments sont `NULL` , le résultat de la comparaison est `NULL` , exception faite pour l'opérateur `<=>` .
- Si les deux arguments de la comparaison sont des chaînes, ils seront comparés en tant que chaînes.
- Si les deux arguments sont des entiers, ils sont comparés en tant qu'entiers.
- Les valeurs hexadécimales sont traitées en tant que chaînes binaires si elles ne sont pas comparées à un nombre.
- Si l'un des arguments est une colonne de type `TIMESTAMP` ou `DATETIME` et que l'autre est une constante, celle ci est convertie en timestamp avant que la comparaison ne s'opère. Cela

est fait pour être mieux compatible avec ODBC.
- Dans tous les autres cas, les arguments sont comparés en tant que nombres à décimale flottante. (réels)

Par défaut, la comparaison des chaînes s'effectue d'une façcon insensible à la casse en utilisant le jeu de charactères courant (ISO−8859−1 Latin1 par défaut, qui fonctionne aussi très bien pour l'anglais).Les exemples suivants, montrent la conversion des chaînes en nombres pour les opérations de comparaison :

```
 mysql> SELECT 1 > '6x';
         -> 0
mysql> SELECT 7 > '6x';
         -> 1
mysql> SELECT 0 > 'x6';
         -> 0
mysql> SELECT 0 = 'x6';
         -> 1
```

=

Egal :

```
 mysql> SELECT 1 = 0;
        -> 0
mysql> SELECT '0' = 0;
        -> 1
mysql> SELECT '0.0' = 0;
        -> 1
mysql> SELECT '0.01' = 0;
        -> 0
mysql> SELECT '.01' = 0.01;
        -> 1
```

<>

*!=*

Différent :

```
 mysql> SELECT '.01' <> '0.01';
        -> 1
mysql> SELECT .01 <> '0.01';
        -> 0
mysql> SELECT 'zapp' <> 'zappp';
        -> 1
```

<=

Inférieur ou égal :

```
 mysql> SELECT 0.1 <= 2;
        -> 1
```

<

Strictement inférieur :

```
 mysql> SELECT 2 < 2;
        -> 0
```

>=

Supérieur ou égal :

```
 mysql> SELECT 2 >= 2;
        -> 1
```

>

Strictement supérieur :

```
 mysql> SELECT 2 > 2;
        -> 0
```

<=>

Egalité sure avec NULL :

```
 mysql> SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;
        -> 1 1 0
```

*IS NULL*

*IS NOT NULL*

Tester si une valeur est ou n'est pas `NULL` :

```
 mysql> SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;
        -> 0 0 1
mysql> SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;
        -> 1 1 0
```

Pour être compatible avec les autres programmes, MySQL gère les appels qui utilisent `IS NULL` de la façon suivante :

Vous pouvez trouver le dernier enregistrement inséré en utilisant :

```
SELECT * FROM nom_de_table WHERE auto_col IS NULL
```

Cela peut être interdit en mettant `SQL_AUTO_IS_NULL=0` . Syntaxe de `SET` .
For `NOT NULL DATE` and `DATETIME` columns you can find the special date `0000-00-00` by using:

```
SELECT * FROM nom_de_table WHERE date_column IS NULL
```

This is needed to get some ODBC applications to work (as ODBC doesn't support a `0000-00-00` date)

*expression BETWEEN min AND max*

Si `expression` est supérieure ou égale à `min` et `expression` est inférieure ou égale à `max` , `BETWEEN` retourne `1` , sinon `0` . Ceci est équivalent à l'expression `(min <= expression AND expression <= max)` si tous les arguments sont du même type. Dans tous les autres cas, la conversion de type prends place, selon les règles suivantes, mais appliquée aux trois arguments. **Notez** que avant la 4.0.5, les arguments étaient convertis au type de `expr` .

```
mysql> SELECT 1 BETWEEN 2 AND 3;
       -> 0
mysql> SELECT 'b' BETWEEN 'a' AND 'c';
       -> 1
mysql> SELECT 2 BETWEEN 2 AND '3';
       -> 1
mysql> SELECT 2 BETWEEN 2 AND 'x-3';
       -> 0
```

*expr NOT BETWEEN min AND max*

Même chose que `NOT (expr BETWEEN min AND max)` .

*expr IN (valeur,...)*

Retourne `1` si `expr` est l'une des valeurs dans la liste `IN` , sinon retourne `0` . Si toutes les valeurs sont des constantes, toutes les valeurs sont évaluées avec le type de `expr` et triées. La recherche de l'élément est alors faite en utilisant la recherche binaire. Cela signifie que `IN` est très rapide si les valeurs contenues dans la liste `IN` sont toutes des constantes. Si `expr` est une chaîne sensible à la casse, la comparaison est faite dans un contexte sensible à la casse :

```
mysql> SELECT 2 IN (0,3,5,'wefwf');
       -> 0
mysql> SELECT 'wefwf' IN (0,3,5,'wefwf');
       -> 1
```

*expr NOT IN (value,...)*

Même chose que `NOT (expr IN (valeur,...))` .

*ISNULL(expr)*

Si `expr` est `NULL` , `ISNULL()` retourne `1` , sinon il retourne `0` :

```
mysql> SELECT ISNULL(1+1);
       -> 0
mysql> SELECT ISNULL(1/0);
       -> 1
```

Notez que la comparaison de deux valeurs `NULL` en utilisant `=` donnera toujours false !

*COALESCE(list)*

Retourne le premier élément non- `NULL` de la liste :

```
mysql> SELECT COALESCE(NULL,1);
       -> 1
mysql> SELECT COALESCE(NULL,NULL,NULL);
       -> NULL
```

*INTERVAL(N,N1,N2,N3,...)*

Retourne `0` si `N` < `N1` , `1` si `N` < `N2` etc... Tous les arguments sont traités en tant qu'entiers. Il est requis que `N1` < `N2` < `N3` < `...` < `Nn` pour que cette fonction fonctionne correctement. Cela est due à la recherche binaire utilisée (très rapide) :

```
mysql> SELECT INTERVAL(23, 1, 15, 17, 30, 44, 200);
       -> 3
mysql> SELECT INTERVAL(10, 1, 10, 100, 1000);
       -> 2
mysql> SELECT INTERVAL(22, 23, 30, 44, 200);
       -> 0
```

Si vous comparez des chaînes non sensibles à la casse avec l'un des opérateurs standards ( `=` , `<>` ..., mais pas `LIKE` ) les espaces en début et fin de chaîne (espaces, tabulations et nouvelles lignes) seront ignorées.

```
mysql> SELECT "a" ="A \n";
       -> 1
```

### 6.3.1.3 Opérateurs logiques

En SQL, tous les opérateurs logiques évaluent à `TRUE` , `FALSE` ou `NULL` (INCONNU). En MySQL, c'est implémenté en `1` (TRUE), `0` (FALSE), et `NULL` . La plupart de ce qui suit est est commun entre les différents bases de données SQL, pourtant, certains système pourraient retourner une valeur non nulle pour TRUE (pas obligatoirement 1).

*NOT*
*!*

NOT (NON) logique. Evalue à `1` si l'opérande est `0` , à `0` si l'opérande est non nulle, et `NOT NULL` retourne `NULL` .

```
 mysql> SELECT NOT 10;
        -> 0
mysql> SELECT NOT 0;
        -> 1
mysql> SELECT NOT NULL;
        -> NULL
mysql> SELECT ! (1+1);
        -> 0
mysql> SELECT ! 1+1;
        -> 1
```

Le dernier exemple donne `1` car l'expression est évaluée comme `(!1)+1` .

*AND*
*&*

`AND` (ET) logique. Evalue à `1` si toutes les opérandes sont différentes de zéro et de `NULL` , à `0` si l'une des opérandes est `0` , dans les autres cas, `NULL` est retourné.

```
 mysql> SELECT 1 && 1;
        -> 1
mysql> SELECT 1 && 0;
        -> 0
mysql> SELECT 1 && NULL;
        -> NULL
mysql> SELECT 0 && NULL;
        -> 0
mysql> SELECT NULL && 0;
        -> 0
```

Notez que pour les versions antérieures à la 4.0.5 l'évaluation est interrompue lorsque `NULL` est rencontré, au lieu de continuer à tester une éventuelle existence de `0` . Cela signifie que dans ces versions, `SELECT (NULL AND 0)` retourne `NULL` au lieu de `0` . En 4.0.5 le code a été revu pour que le résultat réponde toujours au normes ANSI tout en optimisant le plus possible.

*OR*
*||*

OR (OU inclusif) logique. Evalue à `1` si aucune opérande n'est nulle, à `NULL` si l'une des opérandes est `NULL` , sinon `0` est retourné.

```
 mysql> SELECT 1 || 1;
        -> 1
mysql> SELECT 1 || 0;
        -> 1
mysql> SELECT 0 || 0;
        -> 0
mysql> SELECT 0 || NULL;
        -> NULL
mysql> SELECT 1 || NULL;
        -> 1
```

*XOR*

XOR (OU exclusif) logique. Retourne `NULL` si l'une des opérandes est `NULL` . Pour les opérandes non− `NULL` , évalue à `1` si un nombre pair d'opérandes est non−nul, sinon `0` est retourné.

```
 mysql> SELECT 1 XOR 1;
        -> 0
mysql> SELECT 1 XOR 0;
        -> 1
mysql> SELECT 1 XOR NULL;
        -> NULL
mysql> SELECT 1 XOR 1 XOR 1;
        -> 1
```

`a XOR b` est mathématiquement égal à `(a AND (NOT b)) OR ((NOT a) and b)` .

### 6.3.1.4 Les fonctions de contrôle

*IFNULL(expr1,expr2)*

Si l'argument `expr1` n'est pas `NULL` , la fonction `IFNULL()` retournera l'argument `expr1` , sinon elle retournera l'argument `expr2` . La fonction `IFNULL()` retourne une valeur numérique ou une chaîne de caractères, suivant le contexte d'utilisation :

```
mysql> SELECT IFNULL(1,0);
        -> 1
mysql> SELECT IFNULL(NULL,10);
        -> 10
mysql> SELECT IFNULL(1/0,10);
        -> 10
mysql> SELECT IFNULL(1/0,'oui');
        -> 'oui'
```

En version 4.0.6 et plus récent, le résultat par défaut de `IFNULL(expr1,expr2)` est le plus 'général' des deux expressions, dans l'ordre de type `STRING` , `REAL` ou `INTEGER` . La différence avec les anciennes versions de MySQL ne seront notables que si vous créez une table basée sur des expressions, ou si MySQL stocke en interne des valeurs issues de `IFNULL()` dans une table temporaire.

```
CREATE TABLE foo SELECT IFNULL(1,"test") as test;
```

En MySQL 4.0.6, le type de la colonne 'test' est `CHAR(4)` tandis que dans les versions plus anciennes, vous auriez obtenu un `BIGINT` .

*NULLIF(expr1,expr2)*

Si l'expression `expr1 = expr2` est vrai, la fonction retourne `NULL` sinon elle retourne `expr1` . Cela revient à faire `CASE WHEN x = y THEN NULL ELSE x END` :

```
mysql> SELECT NULLIF(1,1);
        -> NULL
mysql> SELECT NULLIF(1,2);
        -> 1
```

Notez que l'argument `expr1` est évalué deux fois dans MySQL si les arguments sont égaux.

*IF(expr1,expr2,expr3)*

Si l'arguement `expr1` vaut TRUE ( `expr1 <> 0` et `expr1 <> NULL` ) alors la fonction `IF()` retourne l'argument `expr2` , sinon, elle retourne l'argument `expr3` . La fonction `IF()` retourne une valeur numérique ou une chaîne de caractères, suivant le contexte d'utilisation :

```
mysql> SELECT IF(1>2,2,3);
        -> 3
mysql> SELECT IF(1<2,'oui','non');
        -> 'oui'
mysql> SELECT IF(STRCMP('test','test1'),'non','oui');
        -> 'non'
```

Si l'argument `expr2` ou `expr3` est explicitement `NULL` alors the le type du résultat de la fonction `IF()` est le type de la colonne non `NULL` . (Ce comportement est nouveau dans MySQL 4.0.3).L'argument `expr1` est évalué comme un entier, cela signifie que si vous testez un nombre à virgule flottante ou une chaîne de caractères, vous devez utiliser une opération de comparaison :

```
mysql> SELECT IF(0.1,1,0);
        -> 0
mysql> SELECT IF(0.1<>0,1,0);
        -> 1
```

Dans le premier exemple ci−dessus, `IF(0.1)` retourne `0` parce que `0.1` est converti en une chaîne de caractères, ce qui revient à tester `IF(0)` . Ce n'est certainement pas ce que vous désireriez. Dans le second exemple, la comparaison teste si le nombre à virgule flottante est différent de zéro. Le résultat de cette comparaison sera un entier.Le type de la fonction `IF()` (ce qui peut être important s'il est stocké dans une table temporaire) est calculé, dans la Version 3.23 de MySQL, comme suit :

| Expression | Valeur retournée |
|---|---|
| expr2 ou expr3 retourne une chaîne | chaîne |
| expr2 ou expr3 retourne un nombre à virgule | nombre à virgule |
| expr2 ou expr3 retourne un entier | entier |

Si expr2 et expr3 sont des chaînes de caractères, alors le résultat est insensible à la casse si les deux chaînes de caractères sont insensibles à la casse. (A partir de la version 3.23.51 de MySQL)

*CASE valeur WHEN [compare−value] THEN résultat [WHEN [compare−value] THEN résultat ...] [ELSE résultat] END*

*CASE WHEN [condition] THEN résultat [WHEN [condition] THEN résultat ...] [ELSE résultat] END*

La première version retourne `résultat` si `valeur=compare−value` . La seconde version retourne le résultat de la première condition qui se réalise. Si aucune des conditions n'est réalisé, alors le résultat de la clasue `ELSE` est retourné. Si il n'y a pas de clause `ELSE` alors `NULL` est retourné :

```
mysql> SELECT CASE 1 WHEN 1 THEN "un"
           WHEN 2 THEN "deux" ELSE "plus" END;
       -> "un"
mysql> SELECT CASE WHEN 1>0 THEN "vrai" ELSE "faux" END;
       -> "vrai"
mysql> SELECT CASE BINARY "B" WHEN "a" THEN 1 WHEN "b" THEN 2 END;
       -> NULL
```

Le type de la valeur retournée ( `INTEGER` , `DOUBLE` ou `STRING` ) est de même type que la première valeur retournée (l'expression après le premier `THEN` ).

## 6.3.2 Fonctions de chaînes de caractères

Les fonctions qui traitent les chaînes de caractères retournent `NULL` si la longueur du résultat finit par dépasser la taille maximale du paramètre `max_allowed_packet` , défini dans la configuration du serveur. Choix des paramètres du serveur .

Pour les fonctions qui opèrent sur des positions à l'intérieur d'une chaîne, la position initiale est 0.

*ASCII(str)*

Retourne le code ASCII du premier caractère de la chaîne de caractères `str` . Retourne `0` si la chaîne de caractère `str` est vide. Retourne `NULL` si la chaîne de caractères `str` est `NULL` :

```
mysql> SELECT ASCII('2');
       -> 50
mysql> SELECT ASCII(2);
       -> 50
mysql> SELECT ASCII('dx');
       -> 100
```

Voir aussi la fonction `ORD()` .

*ORD(str)*

Si le premier caractère de la chaîne `str` est un caractère multi−octets, la fonction retourne le code de ce caractère, calculé à partir du code ASCII retourné par cette formule : `((first byte ASCII code)*256+(second byte ASCII code))[*256+third byte ASCII code...]` . Si le premier caractère n'est pas un caractère multi−octet, la fonction retournera la même valeur que la fonction `ASCII()` :

```
mysql> SELECT ORD('2');
       -> 50
```

*CONV(N,from_base,to_base)*

Convertit des nombres entre différentes bases. Retourne une chaîne de caractères représentant le nombre `N` , convertit de la base `from_base` vers la base `to_base` . La fonction retourne `NULL` si un des arguments est `NULL` . L'argument `N` est interprété comme un entier, mais peut être spécifié comme un entier ou une chaîne de caractères. Le minimum pour la base est `2` et son maximum est `36` . Si `to_base` est un nombre négatif, `N` sera considéré comme un nombre signé. Dans le cas contraire, `N` sera traité comme un nombre non−signé. La fonction `CONV` travaille avec une précision de 64 bits :

```
mysql> SELECT CONV("a",16,2);
       -> '1010'
mysql> SELECT CONV("6E",18,8);
```

```
        -> '172'
mysql> SELECT CONV(-17,10,-18);
        -> '-H'
mysql> SELECT CONV(10+"10"+'10'+0xa,10,10);
        -> '40'
```

*BIN(N)*

Retourne une chaîne de caractères représentant la valeur binaire de l'argument `N` , où l'argument `N` est un nombre de type `BIGINT` . Cette fonction est un équivalent de `CONV(N,10,2)` . Retourne `NULL` si l'argument `N` est `NULL` :

```
mysql> SELECT BIN(12);
        -> '1100'
```

*OCT(N)*

Retourne une chaîne de caractères représentant la valeur octal de l'argument `N` , où l'argument `N` est un nombre de type `BIGINT` . Cette fonction est un équivalent de `CONV(N,10,8)` . Retourne `NULL` si l'argument `N` est `NULL` :

```
mysql> SELECT OCT(12);
        -> '14'
```

*HEX(N_or_S)*

Si l'argument N_OR_S est un nombre, cette fonction retournera une chaîne de caractère représentant la valeur hexadécimale de l'argument `N` , où l'argument `N` est de type `BIGINT` . Cette fonction est un équivalent de `CONV(N,10,16)` .Si N_OR_S est une chaîne de caractères, cette fonction retournera une chaîne de caractères hexadécimale de N_OR_S où chaque caractère de N_OR_S est converti en 2 chiffres hexadécimaux. C'est l'inverse de la chaîne `0xff` .

```
mysql> SELECT HEX(255);
        -> 'FF'
mysql> SELECT HEX("abc");
        -> 616263
mysql> SELECT 0x616263;
        -> "abc"
```

*CHAR(N,...)*

La fonction `CHAR()` interprète les arguments comme des entiers et retourne une chaîne de caractères, constituée des caractères, identifiés par leur code ASCII. Les valeurs `NULL` sont ignorées :

```
mysql> SELECT CHAR(77,121,83,81,'76');
        -> 'MySQL'
mysql> SELECT CHAR(77,77.3,'77.3');
        -> 'MMM'
```

*CONCAT(str1,str2,...)*

Retourne une chaîne représentant la concaténation des arguments. Retourne `NULL` si un des arguments est `NULL` . Cette fonction peut prendre plus de 2 arguments. Si un argument est un nombre, il sera converti en son équivalent sous forme de chaîne de caractères :

```
mysql> SELECT CONCAT('My', 'S', 'QL');
        -> 'MySQL'
mysql> SELECT CONCAT('My', NULL, 'QL');
        -> NULL
mysql> SELECT CONCAT(14.3);
        -> '14.3'
```

*CONCAT_WS(separator, str1, str2,...)*

La fonction `CONCAT_WS()` signifie `CONCAT With Separator` , c'est-à-dire "concaténation avec séparateur. Le premier argument est le séparateur utilisé pour le reste des arguments. Le séparateur peut être une chaîne de caractères, tout comme le reste des arguments. Si le séparateur est `NULL` , le résultat sera `NULL` . Cette fonction ignorera tous les arguments de valeur `NULL` et vides, hormis le séparateur. Le séparateur sera ajouté entre tous les arguments à concaténer :

```
mysql> SELECT CONCAT_WS(",","Premier nom","Deuxième nom","Dernier nom");
       -> 'Premier nom,Deuxième nom,Dernier nom'
mysql> SELECT CONCAT_WS(",","Premier nom",NULL,"Dernier nom");
       -> 'Premier nom,Dernier nom'
```

*LENGTH(str)*
*OCTET_LENGTH(str)*
*CHAR_LENGTH(str)*
*CHARACTER_LENGTH(str)*

Retourne le nombre de caractères de la chaîne `str` :

```
mysql> SELECT LENGTH('text');
       -> 4
mysql> SELECT OCTET_LENGTH('text');
       -> 4
```

Notez que pour la fonction `CHAR_LENGTH()` et la fonction `CHARACTER_LENGTH()` , les caractères multi−octets sont comptés une fois seulement.

*BIT_LENGTH(str)*

Retourne le nombre de bits de la chaîne de caractères `str` :

```
mysql> SELECT BIT_LENGTH('text');
       -> 32
```

*LOCATE(substr,str)*

*POSITION(substr IN str)*

Retourne la position de la première occurrence de la chaîne `substr` dans la chaîne de caractères `str` . Retourne `0` si `substr` ne se trouve pas dans la chaîne de caractères `str` :

```
mysql> SELECT LOCATE('bar', 'foobarbar');
       -> 4
mysql> SELECT LOCATE('xbar', 'foobar');
       -> 0
```

Cette fonction gère les caractères multi−octets. Dans la version 3.23 de MySQL, cette fonction est sensible à la casse, alors que dans la version 4.0 de MySQL, cette fonction sera sensible à la casse si l'argument est une chaîne de caractères binaire.

*LOCATE(substr,str,pos)*

Retourne la position de la première occurrence de la chaîne `substr` dans la chaîne de caractères `str` , à partir de la position `pos` . Retourne `0` si `substr` ne se trouve pas dans la chaîne de caractères `str` :

```
mysql> SELECT LOCATE('bar', 'foobarbar',5);
       -> 7
```

Cette fonction gère les caractères multi−octets. Dans la version 3.23 de MySQL, cette fonction est sensible à la casse, alors que dans la version 4.0 de MySQL, cette fonction sera sensible à la casse si l'argument est une chaîne de caractères binaire.

*INSTR(str,substr)*

Retourne la position de la première occurence de la chaîne `substr` dans la chaîne de caractères `str` . Cette fonction est exactement la même que la fonction `LOCATE()` , à la différence que ces arguments sont inversés :

```
mysql> SELECT INSTR('foobarbar', 'bar');
       -> 4
mysql> SELECT INSTR('xbar', 'foobar');
       -> 0
```

Cette fonction gère les caractères multi−octets. Dans la version 3.23 de MySQL, cette fonction est sensible à la casse, alors que dans la version 4.0 de MySQL, cette fonction sera sensible à la casse si l'argument est une chaîne de caractères binaire.

*LPAD(str,len,padstr)*

Retourne la chaîne de caractères `str` , complétée à gauche par la chaîne de caractères `padstr` jusqu'à ce que la chaîne de caractères `str` atteigne `len` caractères de long. Si la chaîne de caractères `str` est plus longue que `len'` caractères, elle sera raccourcie de `len` caractères.

```
mysql> SELECT LPAD('hi',4,'??');
       -> '??hi'
```

*RPAD(str,len,padstr)*

Retourne la chaîne de caractères `str` , complétée à droite par la chaîne de caractères `padstr` jusqu'à ce que la chaîne de caractères `str` atteigne `len` caractères de long. Si la chaîne de caractères `str` est plus longue que `len'` caractères, elle sera raccourcie de `len` caractères.

```
mysql> SELECT RPAD('hi',5,'?');
       -> 'hi???'
```

*LEFT(str,len)*

Retourne les `len` caractères les plus à gauche de la chaîne de caractères `str` :

```
mysql> SELECT LEFT('foobarbar', 5);
       -> 'fooba'
```

Cette fonction gère les caractères multi−octets.

*RIGHT(str,len)*

Retourne les `len` caractères les plus à droite de la chaîne de caractères `str` :

```
mysql> SELECT RIGHT('foobarbar', 4);
       -> 'rbar'
```

Cette fonction gère les caractères multi−octets.

*SUBSTRING(str,pos,len)*
*SUBSTRING(str FROM pos FOR len)*
*MID(str,pos,len)*

Retourne une chaîne de `len` caractères de long dela chaîne `str` , à partir de la position `pos` . La syntaxe ANSI SQL92 utilise une variante de la fonction `FROM` :

```
mysql> SELECT SUBSTRING('Quadratically',5,6);
       -> 'ratica'
```

Cette fonction gère les caractères multi−octets.

*SUBSTRING(str,pos)*
*SUBSTRING(str FROM pos)*

Retourne une portion de la chaîne de caractères `str` à partir de la position `pos` :

```
mysql> SELECT SUBSTRING('Quadratically',5);
       -> 'ratically'
mysql> SELECT SUBSTRING('foobarbar' FROM 4);
       -> 'barbar'
```

Cette fonction gère les caractères multi−octets.

*SUBSTRING_INDEX(str,delim,count)*

Retourne une portion de la chaîne de caractères `str` , située avant `count` occurences du délimiteur `delim` . Si l'argument `count` est positif, tout ce qui précède le délimiteur final sera retourné. Si l'argument `count` est négatif, tout ce qui suit le délimiteur final sera retourné :

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);
       -> 'www.mysql'
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', −2);
       -> 'mysql.com'
```

Cette fonction gère les caractères multi−octets.

*LTRIM(str)*

Retourne la chaîne de caractères `str` sans les espaces initiaux :

```
mysql> SELECT LTRIM('  barbar');
       -> 'barbar'
```

*RTRIM(str)*

Retourne la chaîne de caractères `str` sans les espaces finaux :

```
mysql> SELECT RTRIM('barbar   ');
       -> 'barbar'
```

Cette fonction gère les caractères multi−octets.

*TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)*

Retourne la chaîne de caractères `str` dont tous les préfixes et/ou suffixes `remstr` ont été supprimés. Si aucun des spécificateurs `BOTH` , `LEADING` ou `TRAILING` sont fournis, `BOTH` est utilisé comme valeur par défaut. Si `remstr` n'est pas spécifié, les espaces sont supprimés :

```
mysql> SELECT TRIM('  bar   ');
       -> 'bar'
mysql> SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');
       -> 'barxxx'
mysql> SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');
       -> 'bar'
mysql> SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');
       -> 'barx'
```

Cette fonction gère les caractères multi−octets.

*SOUNDEX(str)*

Retourne la valeur Soundex de la chaîne de caractères `str` . Deux chaînes qui ont des sonorités proches auront des valeurs soundex proches. Une chaîne Soundex standart possède 4 caractères, mais la fonction `SOUNDEX()` retourne une chaîne de longueur arbitraire. Vous pouvez utiliser la fonction `SUBSTRING()` sur ce résultat pour obtenir une chaine Soundex standart. Tout caractère non alpha−numérique sera ignoré. Tous les caractères internationaux qui ne font pas partie de l'alphabet de base (A−Z) seront considérés comme des voyelles :

```
mysql> SELECT SOUNDEX('Hello');
       -> 'H400'
mysql> SELECT SOUNDEX('Quadratically');
       -> 'Q36324'
```

*SPACE(N)*

Retourne une chaîne constituée de N espaces :

```
mysql> SELECT SPACE(6);
       -> '      '
```

*REPLACE(str,from_str,to_str)*

Retourne une chaîne de caractères str dont toutes les occurences de la chaîne from_str sont remplacées par la chaîne to_str :

```
mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');
       -> 'WwWwWw.mysql.com'
```

Cette fonction gère les caractères multi−octets.

*REPEAT(str,count)*

Retourne une chaîne de caractères constituée de la répétition de count fois la chaîne str . Si count <= 0 , retourne une chaîne vide. Retourne NULL si str ou count sont NULL :

```
mysql> SELECT REPEAT('MySQL', 3);
       -> 'MySQLMySQLMySQL'
```

*REVERSE(str)*

Retourne une chaîne dont l'ordre des caractères est l'inverse de la chaîne str :

```
mysql> SELECT REVERSE('abc');
       -> 'cba'
```

Cette fonction gère les caractères multi−octets.

*INSERT(str,pos,len,newstr)*

Retourne une chaîne de caractères str , après avoir remplacé la portion de chaîne commençant à la position pos et de longueur len caractères, par la chaîne newstr :

```
mysql> SELECT INSERT('Quadratic', 3, 4, 'What');
       -> 'QuWhattic'
```

Cette fonction gère les caractères multi−octets.

*ELT(N,str1,str2,str3,...)*

Retourne str1 si N = 1 , str2 si N = 2 , et ainsi de suite. Retourne NULL si N est plus petit que 1 ou plus grand que le nombre d'arguments. La fonction ELT() est un complément de la fonction FIELD() :

```
mysql> SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');
       -> 'ej'
mysql> SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');
       -> 'foo'
```

*FIELD(str,str1,str2,str3,...)*

Retourne l'index de la chaîne str dans la liste str1 , str2 , str3 , ... . Retourne 0 si str n'est pas trouvé. La fonction FIELD() est un complément de la fonction ELT() :

```
mysql> SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');
       -> 2
mysql> SELECT FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');
       -> 0
```

*FIND_IN_SET(str,strlist)*

Retourne une valeur de 1 à N si la chaîne str se trouve dans la liste strlist constituée de N chaînes. Une liste de chaîne est une chaîne composée de sous−chaînes séparées par une virgule ',' . Si le premier argument est une chaîne constante et le second, une colonne de type SET , la fonction FIND_IN_SET() est optimisée pour utiliser une recherche binaire très rapide. Retourne 0 si str n'est pas trouvé dans la liste strlist ou si la liste strlist est une chaîne vide. Retourne NULL si l'un des arguments est NULL . Cette fonction ne fonctionne pas correctement si le premier argument contient une virgule ',' :

```
mysql> SELECT FIND_IN_SET('b','a,b,c,d');
       -> 2
```

*MAKE_SET(bits,str1,str2,...)*

Retourne une liste (une chaîne contenant des sous−chaînes séparées par une virgule ',' ) constituée de chaînes qui ont le bit correspondant dans la liste bits . str1 correspond au bit 0, str2 au bit 1, etc... Les chaines NULL dans les listes str1 , str2 , ... sont ignorées :

```
mysql> SELECT MAKE_SET(1,'a','b','c');
       -> 'a'
mysql> SELECT MAKE_SET(1 | 4,'hello','nice','world');
       -> 'hello,world'
mysql> SELECT MAKE_SET(0,'a','b','c');
       -> ''
```

*EXPORT_SET(bits,on,off,[séparateur,[nombre_de_bits]])*

Retourne une chaîne dont tous les bits à 1 dans 'bit' sont représentés par la chaîne 'on', et dont tous les bits à 0 sont représentés par la chaîne 'off'. Chaque chaîne est séparée par 'séparateur' (par défaut, une virgule ',') et seul 'nombre_de_bits' (par défaut, 64) 'bits' est utilisé :

```
mysql> SELECT EXPORT_SET(5,'Y','N',',',4)
       -> Y,N,Y,N
```

*LCASE(str)*
*LOWER(str)*

Retourne une chaîne `str` dont tous les caractères ont été mis en minuscule, en accord avec le charset courant (le charset par défaut est ISO−8859−1 Latin1) :

```
    mysql> SELECT LCASE('QUADRATICALLY');
       -> 'quadratically'
```

Cette fonction gère les caractères multi−octets.

*UCASE(str)*
*UPPER(str)*

Retourne une chaîne `str` dont tous les caractères ont été mis en majuscule, en accord avec le charset courant (le charset par défaut est ISO−8859−1 Latin1) :

```
mysql> SELECT UCASE('Hej');
       -> 'HEJ'
```

Cette fonction gère les caractères multi−octets.

*LOAD_FILE(file_name)*

Lit le fichier `file_name` et retourne son contenu sous la forme d'une chaîne de caractères. Le fichier doit se trouver sur le serveur qui exécute MySQL, vous devez spécifier le chemin absolu du fichier et vous devezavoir les droits en lecture sur celui−ci. Le fichier doit pouvoir être lisible par tous et doit être plus petit que `max_allowed_packet` .

Si ce fichier n'existe pas ou ne peut pas être lu pour différentes raisons, la fonction retourne `NULL` :

```
mysql> UPDATE tbl_name
          SET blob_column=LOAD_FILE("/tmp/picture")
          WHERE id=1;
```

Si vous n'utilisez pas la version 3.23 de MySQL, vous devez lire le fichier depuis votre application et créer ainsi votre requête `INSERT` vous−même, pour mettre à jour la base de données avec le contenu de ce fichier. Une des possibilités pour réaliser ceci, si vous utilisez la librairie MySQL++, peut être trouvée à http://www.mysql.com/documentation/mysql++/mysql++−examples.html .

*QUOTE(str)*

Echappe les caractères d'une chaîne pour produire un résultat qui sera exploitable dans une requête SQL. Les caractères suivants seront précédés d'un anti−slash dans la chaîne retournée : le guillement simple ( `'` ), l'anti−slash ( `'\'` ), ASCII NUL, et le Control−Z. Si l'argument vaut `NULL` , la valeur retournée sera le mot ``NULL'' sans les guillements simples. La fonction `QUOTE` a été ajoutée en MySQL version 4.0.3.

```
 mysql> SELECT QUOTE("Don't");
        -> 'Don\'t!'
mysql> SELECT QUOTE(NULL);
        -> NULL
```

MySQL convertit automatiquement les nombres en une chaînes de caractères, si nécessaire et vice−versa :

```
 mysql> SELECT 1+"1";
        -> 2
mysql> SELECT CONCAT(2,' test');
        -> '2 test'
```

Si vous voulez convertir explicitement un nombre en une chaîne de caractères, passez−le en argument à la fonction `CONCAT()` .Si une fonction de chaînes de caractères prend en argument une

chaîne binaire, le résultat sera également une chaîne binaire. Un nombre converti en une chaîne sera traité comme étant une chaîne binaire. Cela affecte uniquement les comparaisons.

#### 6.3.2.1 Opérateurs de comparaison pour les chaînes de caractères

Normalement, si l'une des expressions dans une comparaison de chaîne est sensible à la casse, la comparaison est exécutée en tenant compte de la casse.

*expr LIKE pat [ESCAPE 'escape−char']*

La réalisation d'expression utilisant les expressions régulières simples de comparaison de SQL. Retourne `1` (TRUE) ou `0` (FALSE). Avec `LIKE` , vous pouvez utiliser les deux jokers suivants :

| Char | Description |
|---|---|
| % | Remplace n'importe quel nombre de caractères, y compris aucun |
| _ | Remplace exactement un caractère |

```
 mysql> SELECT 'David!' LIKE 'David_';
        -> 1
mysql> SELECT 'David!' LIKE '%D%v%';
        -> 1
```

Pour tester la présence littérale d'un joker, précédez−le d'un caractère d'échappement. Si vous ne spécifiez pas le caractère d'échappement `ESCAPE` , le caractère `'\'` sera utilisé :

| String | Description |
|---|---|
| \% | Remplace le caractère littéral % |
| _ | Remplace le caractère littéral _ |

```
 mysql> SELECT 'David!' LIKE 'David\_';
        -> 0
mysql> SELECT 'David_' LIKE 'David\_';
        -> 1
```

Pour spécifier un caractère d'échappement différent, utilisez la clause `ESCAPE` :

```
 mysql> SELECT 'David_' LIKE 'David|_' ESCAPE '|';
        -> 1
```

Les deux exemples suivants illustrent le fait que les comparaisons de chaînes de caractères ne sont pas sensibles à la casse à moins qu'une des opérandes soit une chaîne binaire.

```
 mysql> SELECT 'abc' LIKE 'ABC';
        -> 1
mysql> SELECT 'abc' LIKE BINARY 'ABC';
        -> 0
```

`LIKE` est également autorisé pour les expressions numériques. (C'est une extension MySQL à la norme ANSI SQL `LIKE` .)

```
 mysql> SELECT 10 LIKE '1%';
        -> 1
```

Note : Comme MySQL utilise la syntaxe d'échappement de caractères du langage C dans les chaînes (par exemple, `'\n'` ), vous devez doubler tous les slashs `'\'` que vous utilisez dans les expressions `LIKE` . Par exemple, pour rechercher les nouvelles lignes ( `'\n'` ), vous devez le spécifier comme cela : `'\\n'` . Pour rechercher un anti−slash ( `'\'` ), vous devez le spécifier comme cela : `'\\\\'` (les anti−slashs sont supprimés une première fois pas l'analyseur syntaxique, puis une deuxième fois par le moteur d'expression régulières, ce qui ne laisse qu'un seul anti−slash à la fin).

*expr NOT LIKE pat [ESCAPE 'escape−char']*

Equivalent à `NOT (expr LIKE pat [ESCAPE 'escape−char'])` .

*expr REGEXP pat*

*expr RLIKE pat*

Effectue une recherche de chaîne avec l'expression régulière `pat` . Le masque peut être une expression régulière étendue. Voir la section Expressions régulières de MySQL . Retourne `1` si `expr` correspond au masque `pat` , sinon, retourne `0` . `RLIKE` est un synonyme de `REGEXP` , fourni pour assurer la compatiblité avec `mSQL` . Note : Comme MySQL utilise la syntaxe d'échappement de caractères du langage C dans les chaînes (par exemple, `'\n'` ), vous devez doubler tous les anti−slashs `'\'` que vous

utilisez dans les expressions `REGEXP` . A partir de la version 3.23.4 de MySQL, `REGEXP` est insensible à la casse pour les comparaisons de chaînes normales (non binaires) :

```
mysql> SELECT 'Monty!' REGEXP 'm%y%%';
       -> 0
mysql> SELECT 'Monty!' REGEXP '.*';
       -> 1
mysql> SELECT 'new*\n*line' REGEXP 'new\\*.\\*line';
       -> 1
mysql> SELECT "a" REGEXP "A", "a" REGEXP BINARY "A";
       -> 1  0
mysql> SELECT "a" REGEXP "^[a-d]";
       -> 1
```

*`REGEXP` et `RLIKE` utilise le jeu de caractères courant (ISO−8859−1*
Latin1 par défaut) lorsqu'il faut décider du type d'un caractère.

*expr NOT REGEXP pat*
*expr NOT RLIKE pat*
Identique à `NOT (expr REGEXP pat)` .

*STRCMP(expr1,expr2)*
`STRCMP()` retourne `0` si les chaînes sont identiques, `-1` si la première chaîne est plus petite que la seconde et `1` dans les autres cas :

```
mysql> SELECT STRCMP('text', 'text2');
       -> -1
mysql> SELECT STRCMP('text2', 'text');
       -> 1
mysql> SELECT STRCMP('text', 'text');
       -> 0
```

*MATCH (col1,col2,...) AGAINST (expr)*
*MATCH (col1,col2,...) AGAINST (expr IN BOOLEAN MODE)*
`MATCH ... AGAINST()` est utilisé pour les recherches en texte plein et retourne une information de pertinence − pertinence mesuré entre le texte des colonnes `(col1,col2,...)` et la valeur `expr` . La pertinence est un nombre à virgule flottante positif. La pertinence zéro signifie qu'il n'y a aucune similitude. `MATCH ... AGAINST()` est utilisable dans les versions 3.23.23 ou suivantes de MySQL. L'extension `IN BOOLEAN MODE` a été ajoutée dans la version 4.0.1. Pour plus des détails ainsi que des exemples, voir Recherche full−text avec MySQL .

### 6.3.2.2 Sensibilité à la casse

`BINARY`
L'opérateur `BINARY` modifie la chaîne qui le suit en une chaîne binaire. C'est une solution simple pour forcer la comparaison de colonnes à être sensible à la casse même si la colonne n'est pas définie comme étant de type `BINARY` ou `BLOB` :

```
mysql> SELECT "a" = "A";
       -> 1
mysql> SELECT BINARY "a" = "A";
       -> 0
```

`BINARY string` est un raccourci pour `CAST(string AS BINARY)` . Fonctions de transtypage . `BINARY` a été introduit dans MySQL à partir de la version 3.23.0.

Notez que dans quelques cas, MySQL n'est pas capable d'utiliser l'index efficacement lorsque vous modifiez une colonne indexée en `BINARY` .

Si vous voulez comparer un champ de type `BLOB` d'une manière insensible à la casse, vous pouvez toujours le convertir en majuscules avant d'effectuer la comparaison :

```
SELECT 'A' LIKE UPPER(blob_col) FROM nom_de_table;
```

Nous avons planifié d'introduire bientôt de possibles modifications entre les différents jeux de caractères pour rendre les comparaisons de chaînes encore plus flexibles.

## 6.3.3 Fonctions numériques

### 6.3.3.1 Opérations arithmétiques

Les opérateurs arithmétiques usuels sont disponibles. Notez que dans le cas de `'-'` , `'+'` , and `'*'` , le résultat est calculé avec en `BIGINT` avec une précision de 64 bits si les deux arguments sont des entiers ! Si l'un des arguments est un entier non signé, et que l'autre argument est aussi un entier, le résultat sera un entier non signé. Fonctions de transtypage .

+

Addition :

```
mysql> SELECT 3+5;
       -> 8
```

–

Soustraction :

```
mysql> SELECT 3-5;
       -> -2
```

*

Multiplication :

```
mysql> SELECT 3*5;
       -> 15
mysql> SELECT 18014398509481984*18014398509481984.0;
       -> 324518553658426726783156020576256.0
mysql> SELECT 18014398509481984*18014398509481984;
       -> 0
```

Le résultat du dernier calcul est incorrect car le résultat de la multiplication des deux entiers à dépassé la capacité de calcul de `BIGINT` (64 bits).

/

Division :

```
mysql> SELECT 3/5;
       -> 0.60
```

La division par zéro produit un résultat `NULL` :

```
mysql> SELECT 102/(1-1);
       -> NULL
```

Une division sera calculée en `BIGINT` seulement si elle est effectuée dans un contexte où le résultat est transformé en entier.

### 6.3.3.2 Fonctions mathématiques

Toutes les fonctions mathématiques retournent `NULL` en cas d'erreur.

–

Le signe moins. Change le signe de l'argument :

```
mysql> SELECT - 2;
       -> -2
```

Notez que si cet opérateur est utilisé avec une valeur de type `BIGINT` , la valeur retournée sera de type `BIGINT` ! Cela signifie que vous devez éviter d'utiliser l'opérateur `-` sur les valeurs de type entier inférieures à `-2^63` !

*ABS(X)*

Retourne la valeur absolue de `X` :

```
mysql> SELECT ABS(2);
       -> 2
mysql> SELECT ABS(-32);
       -> 32
```

Cette fonction est utilisable avec les valeurs issues des champs `BIGINT` .

*SIGN(X)*

Retourne le signe de l'argument sous la forme `-1` , `0` , ou `1` , selon que `X` est négatif, zéro, ou positif :

```
mysql> SELECT SIGN(-32);
       -> -1
```

```
mysql> SELECT SIGN(0);
        -> 0
mysql> SELECT SIGN(234);
        -> 1
```

*MOD(N,M)*
*%*

Modulo (équivalent de l'opérateur `%` dans le langage C). Retourne le reste de la division de `N` par `M` :

```
 mysql> SELECT MOD(234, 10);
        -> 4
mysql> SELECT 253 % 7;
        -> 1
mysql> SELECT MOD(29,9);
        -> 2
```

Cette fonction ne pose pas de problèmes avec les `BIGINT` .

*FLOOR(X)*

Retourne la valeur entière inférieure de `X` :

```
 mysql> SELECT FLOOR(1.23);
        -> 1
mysql> SELECT FLOOR(-1.23);
        -> -2
```

Notez que la valeur retournée sera de type `BIGINT` !

*CEILING(X)*

Retourne la valeur entière supérieure de `X` :

```
 mysql> SELECT CEILING(1.23);
        -> 2
mysql> SELECT CEILING(-1.23);
        -> -1
```

Notez que la valeur retournée sera de type `BIGINT` !

*ROUND(X)*

Retourne l'entier le plus proche de `X` :

```
 mysql> SELECT ROUND(-1.23);
        -> -1
mysql> SELECT ROUND(-1.58);
        -> -2
mysql> SELECT ROUND(1.58);
        -> 2
```

Notez que le comportement de l'opérateur `ROUND()` , lorsque l'argument est exactement entre deux entiers, dépend de la librairie C active. Certaines arrondissent toujours à l'entier pair le plus proche, toujours vers le haut, toujours vers le bas, ou toujours vers zéro. Si vous avez besoin d'un certain type d'arrondissement, vous devez utiliser une fonction bien définie comme `TRUNCATE()` ou `FLOOR()` .

*ROUND(X,D)*

Retourne l'argument `X` , arrondi à un nombre à `D` décimales. Si `D` vaut `0` , le résultat n'aura ni de partie décimale, ni de séparateur de décimal :

```
 mysql> SELECT ROUND(1.298, 1);
        -> 1.3
mysql> SELECT ROUND(1.298, 0);
        -> 1
```

*EXP(X)*

Retourne la valeur de `e` (la base des logarithmes naturels) élevé à la puissance `X` :

```
 mysql> SELECT EXP(2);
        -> 7.389056
mysql> SELECT EXP(-2);
        -> 0.135335
```

*LN(X)*

Retourne le logarithme naturel de `X` (népérien) :

```
 mysql> SELECT LN(2);
        -> 0.693147
mysql> SELECT LN(-2);
        -> NULL
```

Cette fonction a été ajoutée à MySQL à partir de la version 4.0.3. C'est un synonyme de la fonction `LOG(X)` .

*LOG(X)*
*LOG(B,X)*

Appelée avec un seul paramètre, cette fonction retourne le logarithme naturel (népérien) de `X` :

```
mysql> SELECT LOG(2);
       -> 0.693147
mysql> SELECT LOG(-2);
       -> NULL
```

Appelée avec deux paramètres, cette fonction retourne le logarithme naturel de X pour une base B arbitraire :

```
mysql> SELECT LOG(2,65536);
       -> 16.000000
mysql> SELECT LOG(1,100);
       -> NULL
```

Cette base arbitraire a été ajoutée à MySQL à partir de la version 4.0.3. LOG(B,X) est l'équivalent de LOG(X)/LOG(B) .

*LOG2(X)*

Retourne le logarithme en base 2 de X :

```
mysql> SELECT LOG2(65536);
       -> 16.000000
mysql> SELECT LOG2(-100);
       -> NULL
```

LOG2() est utile pour trouver combien de bits sont nécessaires pour stocker un nombre. Cette fonction a été ajoutée à MySQL à partir de la version 4.0.3. Dans les versions antérieures, vous pouvez utiliser LOG(X)/LOG(2) en remplacement.

*LOG10(X)*

Retourne le logarithme en base 10 de X :

```
mysql> SELECT LOG10(2);
       -> 0.301030
mysql> SELECT LOG10(100);
       -> 2.000000
mysql> SELECT LOG10(-100);
       -> NULL
```

*POW(X,Y)*

*POWER(X,Y)*

Retourne la valeur de X élevée à la puissance Y :

```
mysql> SELECT POW(2,2);
       -> 4.000000
mysql> SELECT POW(2,-2);
       -> 0.250000
```

*SQRT(X)*

Retourne la racine carrée de X :

```
mysql> SELECT SQRT(4);
       -> 2.000000
mysql> SELECT SQRT(20);
       -> 4.472136
```

*PI()*

Retourne la valeur de PI. Par défaut, 5 décimales sont retournées, mais MySQL utilise la double précision pour PI.

```
mysql> SELECT PI();
       -> 3.141593
mysql> SELECT PI()+0.000000000000000000;
       -> 3.141592653589793116
```

*COS(X)*

Retourne le cosinus de X , où X est donné en radians :

```
mysql> SELECT COS(PI());
       -> -1.000000
```

*SIN(X)*

Retourne le sinus de X , où X est donné en radians :

```
mysql> SELECT SIN(PI());
       -> 0.000000
```

*TAN(X)*

Retourne la tangente de X , où X est donné en radians :

```
mysql> SELECT TAN(PI()+1);
       -> 1.557408
```

*ACOS(X)*

Retourne l'arccosinus de X , c'est à dire, la valeur de l'angle dont X est la cosinus. Retourne NULL si X n'est pas dans l'intervalle -1 – 1 :

```
mysql> SELECT ACOS(1);
       -> 0.000000
mysql> SELECT ACOS(1.0001);
       -> NULL
mysql> SELECT ACOS(0);
```

```
        -> 1.570796
```

*ASIN(X)*

Retourne l'arcsinus de X , c'est à dire, la valeur de l'angle dont le sinus est X . Retourne NULL si X n'est pas dans l'intervalle −1 – 1 :

```
 mysql> SELECT ASIN(0.2);
        -> 0.201358
mysql> SELECT ASIN('foo');
        -> 0.000000
```

*ATAN(X)*

Retourne l'arctangente de X , c'est à dire, la valeur de l'angle dont la tangente est X :

```
 mysql> SELECT ATAN(2);
        -> 1.107149
mysql> SELECT ATAN(-2);
        -> -1.107149
```

*ATAN(Y,X)*
*ATAN2(Y,X)*

Retourne l'arctangente des variables X et Y . Cela revient à calculer l'arctangennte de Y / X , excepté que les signes des deux arguments servent à déterminer le quadrant du résultat :

```
 mysql> SELECT ATAN(-2,2);
        -> -0.785398
mysql> SELECT ATAN2(PI(),0);
        -> 1.570796
```

*COT(X)*

Retourne la cotangente de X :

```
 mysql> SELECT COT(12);
        -> -1.57267341
mysql> SELECT COT(0);
        -> NULL
```

*RAND()*
*RAND(N)*

Retourne un nombre aléatoire à virgule flottante comprit dans l'intervalle 0 – 1.0 . Si l'argument entier N est spécifié, il est utilisé comme initialisation du générateur de nombres aléatoires :

```
 mysql> SELECT RAND();
        -> 0.9233482386203
mysql> SELECT RAND(20);
        -> 0.15888261251047
mysql> SELECT RAND(20);
        -> 0.15888261251047
mysql> SELECT RAND();
        -> 0.63553050033332
mysql> SELECT RAND();
        -> 0.70100469486881
```

Vous ne pouvez pas utiliser une colonne de valeur RAND() dans une clause ORDER BY , parce que ORDER BY va évaluer la colonne plusieurs fois. Dans la version 3.23 de MySQL, vous pouvez, tout de même, faire ceci : `SELECT * FROM nom_de_table ORDER BY RAND()`

Cette syntaxe est très pratique pour faire une sélection aléatoire de lignes : `SELECT * FROM table1,table2 WHERE a=b AND c<d ORDER BY RAND() LIMIT 1000` .

Notez que la fonction RAND() dans une clause WHERE sera réévaluée à chaque fois que WHERE sera exécuté.

RAND() n'est pas un générateur de nombres aléatoires parfait, mais reste une manière rapide de produire des nombres aléatoires ad hoc portables selon les différentes plate−formes pour une même version de MySQL.

*LEAST(X,Y,...)*

Avec deux arguments ou plus, cette fonction retourne la plus petite des valeurs des différents arguments. Les arguments sont comparés en respectant les règles suivantes :

Si la valeur retournée est dans un contexte INTEGER , ou bien que tous les arguments sont des entiers, ils sont comparés en tant qu'entiers.

Si la valeur retournée est dans un contexte REAL , ou bien que tous les

arguments sont des réels, ils sont comparés en tant que réels.
Si l'un des arguments est une chaîne sensible à la casse, les arguments sont comparés comme des chaînes sensibles à la casse.
Dans tous les autres cas, les arguments sont comparés comme des chaînes, insensibles à la casse.

```
mysql> SELECT LEAST(2,0);
       -> 0
mysql> SELECT LEAST(34.0,3.0,5.0,767.0);
       -> 3.0
mysql> SELECT LEAST("B","A","C");
       -> "A"
```

Dans les versions de MySQL 3.22.5 et antérieures, vous pouvez utiliser `MIN()` à la place de `LEAST` .

*GREATEST(X,Y,...)*

Retourne le plus grand des arguments. Les arguments sont comparés en suivant les mêmes règles que la fonction `LEAST` :

```
mysql> SELECT GREATEST(2,0);
       -> 2
mysql> SELECT GREATEST(34.0,3.0,5.0,767.0);
       -> 767.0
mysql> SELECT GREATEST("B","A","C");
       -> "C"
```

Dans les versions de MySQL 3.22.5 et antérieures, vous pouvez utiliser `MAX()` à la place de `GREATEST` .

*DEGREES(X)*

Retourne l'argument `X` , convertit de radians en degrés :

```
mysql> SELECT DEGREES(PI());
       -> 180.000000
```

*RADIANS(X)*

Retourne l'argument `X` , convertit de degrés en radians :

```
mysql> SELECT RADIANS(90);
       -> 1.570796
```

*TRUNCATE(X,D)*

Retourne l'argument `X` , tronqué à `D` décimales. Si `D` vaut `0` , le résultat n'aura ni séparateur décimal, ni partie décimale :

```
mysql> SELECT TRUNCATE(1.223,1);
       -> 1.2
mysql> SELECT TRUNCATE(1.999,1);
       -> 1.9
mysql> SELECT TRUNCATE(1.999,0);
       -> 1
mysql> SELECT TRUNCATE(-1.999,1);
       -> -1.9
```

A partir de MySQL 3.23.51 tous les nombres sont arrondis vers zéro.Si `D` est négatif, la partie entière du nombre est mise à zéro :

```
mysql> SELECT TRUNCATE(122,-2);
      -> 100
```

Notez que les nombres décimaux ne sont pas stockés exactement comme les nombres entiers , mais comme des valeurs doubles. Vous pouvez être dupés par le résultat suivant :

```
mysql> SELECT TRUNCATE(10.28*100,0);
      -> 1027
```

Ce résultat est normal car 10.28 est actuellement stocké comme cela 10.2799999999999999.

## 6.3.4 Fonctions de dates et d'heures

Voir Types dates et heures pour une description détaillée des intervalles de validité de chaque type, ainsi que les formats valides de spécifications des dates et heures.

Voici un exemple d'utilisation des fonctions de date. La requête suivante sélectionne toutes les lignes dont la colonne `date_col` représente une date de moins de 30 jours :

```
mysql> SELECT quelquechose FROM nom_de_table
         WHERE TO_DAYS(NOW()) - TO_DAYS(date_col) <= 30;
```

*DAYOFWEEK(date)*

Retourne l'index du jour de la semaine : pour `date` ( `1` = Dimanche, `2` = Lundi, ... `7` = Samedi). Ces index correspondent au standart ODBC :

```
mysql> SELECT DAYOFWEEK('1998-02-03');
       -> 3
```

*WEEKDAY(date)*

Retourne l'index du jour de la semaine, avec la convertion suivante : `date` ( `0` = Lundi, `1` = Mardi, ... `6` = Dimanche) :

```
mysql> SELECT WEEKDAY('1997-10-04 22:23:00');
       -> 5
mysql> SELECT WEEKDAY('1997-11-05');
       -> 2
```

*DAYOFMONTH(date)*

Retourne le jour de la date `date` , dans un intervalle de `1` à `31` :

```
mysql> SELECT DAYOFMONTH('1998-02-03');
       -> 3
```

*DAYOFYEAR(date)*

Retourne le jour de la date `date` , dans un intervalle de `1` à `366` :

```
mysql> SELECT DAYOFYEAR('1998-02-03');
       -> 34
```

*MONTH(date)*

Retourne le numéro du mois de la date `date` , dans un intervalle de `1` à `12` :

```
mysql> SELECT MONTH('1998-02-03');
       -> 2
```

*DAYNAME(date)*

Retourne le nom du jour de la semaine, en anglais, de la date `date` :

```
mysql> SELECT DAYNAME("1998-02-05");
       -> 'Thursday'
```

*MONTHNAME(date)*

Retourne le nom du mois de la date `date` :

```
mysql> SELECT MONTHNAME("1998-02-05");
       -> 'February'
```

*QUARTER(date)*

Retourne le numéro du trimestre de la date `date` , dans un intervalle de `1` à `4` :

```
mysql> SELECT QUARTER('98-04-01');
       -> 2
```

*WEEK(date)*
*WEEK(date,first)*

Avec un seul argument, retourne le numéro de la semaine dans l'année de la date `date` , dans un intervalle de `0` à `53` (oui, il peut y avoir un début de semaine numéro 53), en considérant que Dimanche est le premier jour de la semaine. Avec deux arguments, la fonction `WEEK()` vous permet de spécifier si les semaines commencent le Dimanche ou le Lundi et la valeur retournée sera dans l'intervalle `0-53` ou bien `1-52` .Voici un tableau explicatif sur le fonctionnement du second argument :

| **Value** | Meaning |
|---|---|
| 0 | La semaine comme le Dimanche et retourne une valeur dans l'intervalle 0-53 |
| 1 | La semaine comme le Lundi et retourne une valeur dans l'intervalle 0-53 |
| 2 | La semaine comme le Dimanche et retourne une valeur dans l'intervalle 1-53 |
| 3 | La semaine comme le Lundi et retourne une valeur dans l'intervalle 1-53 (ISO 8601) |

```
mysql> SELECT WEEK('1998-02-20');
       -> 7
mysql> SELECT WEEK('1998-02-20',0);
       -> 7
mysql> SELECT WEEK('1998-02-20',1);
       -> 8
mysql> SELECT WEEK('1998-12-31',1);
       -> 53
```

Note : Dans la version 4.0 de MySQL, `WEEK(#,0)` a été modifié pour être utilisé avec le calendrier USA.

Notez que si une semaine est la dernière semaine de l'année précédente, MySQL retournera 0 si vous n'utilisez pas 2 ou 3 comme argument optionnel :

```
mysql> SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0);
       -> 2000, 0
mysql> SELECT WEEK('2000-01-01',2);
       -> 52
```

On peut prétendre que la fonction `WEEK()` de MySQL retourne `52` , vu que la date donnée est en fait, la 52 ème semaine de l'année 1999.

Nous avons décidé de retourner 0 au lieu de vouloir que la fonction retourne 'le numéro de la semaine dans une année donnée'. Ceci rend l'utilisation de la fonction `WEEK()` fiable une fois combinée avec d'autres fonctions qui extraient une partie de date à partir d'une date :

```
mysql> SELECT YEARWEEK('2000-01-01');
       -> 199952
mysql> SELECT MID(YEARWEEK('2000-01-01'),5,2);
       -> 52
```

*YEAR(date)*

Retourne l'année de la date `date` , dans un intervalle de `1000` à `9999` :

```
mysql> SELECT YEAR('98-02-03');
       -> 1998
```

*YEARWEEK(date)*

*YEARWEEK(date,first)*

Retourne l'année et la semaine pour une date. Le second argument fonctionne exactement comme le second argument de la fonction `WEEK()` . Notez que l'année peut être différente de l'année de la date pouur la première et la dernière semaine de l'année :

```
mysql> SELECT YEARWEEK('1987-01-01');
       -> 198653
```

Notez que le numéro de la semaine est différent de la valeur retournée par la fonction `WEEK()` ( `0` ) pour un argument optionnel 0 ou 1, alors que la fonction `WEEK()` retourne une semaine dans une année donnée.

*HOUR(time)*

Retourne le nombre d'heures pour l'heure `time` , dans un intervalle de `0` à `23` :

```
mysql> SELECT HOUR('10:05:03');
       -> 10
```

*MINUTE(time)*

Retourne le nombre de minutes pour l'heure for `time` , dans un intervalle de `0` à `59` :

```
mysql> SELECT MINUTE('98-02-03 10:05:03');
       -> 5
```

*SECOND(time)*

Retourne le nombre de secondes pour l'heure `time` , dans un intervalle de `0` à `59` :

```
mysql> SELECT SECOND('10:05:03');
       -> 3
```

*PERIOD_ADD(P,N)*

Ajoute `N` mois à la période `P` (au format `YYMM` ou `YYYYMM` ). Retourne une valeur dans le format `YYYYMM` .

Notez que l'argument `P` **n'est pas** de type date :

```
mysql> SELECT PERIOD_ADD(9801,2);
       -> 199803
```

*PERIOD_DIFF(P1,P2)*

Retourne le nombre de mois entre les périodes `P1` et `P2` . `P1` et `P2` doivent être au format `YYMM` ou `YYYYMM` .Notez que les arguments `P1` et `P2` **ne sont pas** de type date :

```
mysql> SELECT PERIOD_DIFF(9802,199703);
       -> 11
```

*DATE_ADD(date,INTERVAL expr type)*

*DATE_SUB(date,INTERVAL expr type)*

*ADDDATE(date,INTERVAL expr type)*

*SUBDATE(date,INTERVAL expr type)*

Ces fonctions effectuent des calculs sur les dates. Elles sont nouvelles depuis la version 3.22 de MySQL. `ADDDATE()` et `SUBDATE()` sont synonymes de `DATE_ADD()` and `DATE_SUB()` . Dans la version 3.23 de MySQL, vous pouvez utiliser les opérateurs `+` et `-` en lieu et place des fonctions `DATE_ADD()` et `DATE_SUB()` si l'expression de droite est une colonne de type date ou datetime. (Voir l'exemple suivant) `date` est une valeur de type `DATETIME` ou `DATE` , indiquant la date du début. `expr` est une expression spécifiant la valeur de l'intervalle à ajouter ou à soustraire depuis le début de la date. `expr` est une chaîne ; elle peut commencer par le signe `'-'` pour les intervalles négatifs. `type` est un mot clé indiquant comment l'expression doit être interprétée.

La fonction `EXTRACT(type FROM date)` retourne le 'type' d'intervalle pour la date.

La table suivante illustre la relation entre les arguments `type` et `expr` :

| type **value** | **Expected** expr **format** |
|---|---|
| SECOND | SECONDES |
| MINUTE | MINUTES |
| HOUR | HEURES |
| DAY | JOURS |
| MONTH | MOIS |
| YEAR | ANNES |
| MINUTE_SECOND | "MINUTES:SECONDES" |
| HOUR_MINUTE | "HEURES:MINUTES" |
| DAY_HOUR | "JOURS HEURES" |
| YEAR_MONTH | "ANNEES-MOIS" |
| HOUR_SECOND | "HEURES:MINUTES:SECONDES" |
| DAY_MINUTE | "JOURS HEURES:MINUTES" |
| DAY_SECOND | "JOURS HEURES:MINUTES:SECONDES" |

MySQL autorise n'importe quel signe de ponctuation comme délimiteur dans `expr` . Ceux qui sont présentés dans la table ci-dessus ne sont que des suggestions. Si l'argument `date` est une `DATE` et que votre calcul implique seulement les années ( `YEAR` ), les mois ( `MONTH` ), et les jours ( `DAY` ) (c'est-à-dire, pas d'heures), le résultat est une `DATE` . Sinon, le résultat est une valeur de type `DATETIME` :

```
 mysql> SELECT "1997-12-31 23:59:59" + INTERVAL 1 SECOND;
        -> 1998-01-01 00:00:00
mysql> SELECT INTERVAL 1 DAY + "1997-12-31";
        -> 1998-01-01
mysql> SELECT "1998-01-01" - INTERVAL 1 SECOND;
       -> 1997-12-31 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
    ->                 INTERVAL 1 SECOND);
        -> 1998-01-01 00:00:00
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
    ->                 INTERVAL 1 DAY);
        -> 1998-01-01 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
    ->                 INTERVAL "1:1" MINUTE_SECOND);
        -> 1998-01-01 00:01:00
mysql> SELECT DATE_SUB("1998-01-01 00:00:00",
    ->                 INTERVAL "1 1:1:1" DAY_SECOND);
        -> 1997-12-30 22:58:59
mysql> SELECT DATE_ADD("1998-01-01 00:00:00",
    ->                 INTERVAL "-1 10" DAY_HOUR);
        -> 1997-12-30 14:00:00
mysql> SELECT DATE_SUB("1998-01-02", INTERVAL 31 DAY);
        -> 1997-12-02
```

Si vous spécifiez un intervalle de valeur qui est trop court (c'est-à-dire qu'il n'inclut pas toutes les valeurs auxquelles ont peu s'attendre pour le type `type` ), MySQL suppose que vous avez ignoré ces valeurs. Par exemple, si vous spécifiez un `type` de `DAY_SECOND` , la valeur `expr` doit avoir des jours, heures, minutes, secondes. Si vous spécifiez une valeur telle que `"1:10"` , MySQL supposera que les jours et les heures sont ignorés, et que la valeur fournit représente uniquement les minutes et les secondes. En d'autres termes, `"1:10" DAY_SECOND` s'interprète comme `"1:10" MINUTE_SECOND` . C'est comparable à la manière dont MySQL interprète les valeurs `TIME` qui représente une durée plutôt qu'une heure du jour.Notez que si vous ajoutez ou soustrayez à une date une valeur qui contient des heures, le résultat final sera automatiquement converti en DATETIME :

```
mysql> SELECT DATE_ADD("1999-01-01", INTERVAL 1 DAY);
       -> 1999-01-02
mysql> SELECT DATE_ADD("1999-01-01", INTERVAL 1 HOUR);
       -> 1999-01-01 01:00:00
```

Si vous utilisez des dates incorrectes, le résultat sera `NULL` . Si vous ajoutez des `MONTH` , `YEAR_MONTH` , ou `YEAR` , et que le résultat dépasse le nombre de jour dansle mois, ce nombre de jour sera ramené au maximun acceptable :

```
mysql> SELECT DATE_ADD('1998-01-30', INTERVAL 1 MONTH);
        -> 1998-02-28
```

Notez que dans l'exemple précédent, le mot `INTERVAL` et le type `type` ne sont pas sensible à la casse.

*EXTRACT(type FROM date)*

La fonction `EXTRACT()` utilise les mêmes types d'intervalles que la fonction `DATE_ADD()` ou la fonction `DATE_SUB()` , mais extrait des parties de date plutôt que des opérations de date.

```
mysql> SELECT EXTRACT(YEAR FROM "1999-07-02");
       -> 1999
mysql> SELECT EXTRACT(YEAR_MONTH FROM "1999-07-02 01:02:03");
       -> 199907
mysql> SELECT EXTRACT(DAY_MINUTE FROM "1999-07-02 01:02:03");
       -> 20102
```

*TO_DAYS(date)*

Retourne le nombre de jours depuis la date 0 jusqu'à la date `date` :

```
mysql> SELECT TO_DAYS(950501);
        -> 728779
mysql> SELECT TO_DAYS('1997-10-07');
        -> 729669
```

`TO_DAYS()` n'est pas fait pour travailler avec des dates qui précèdent l'avènement du calendrier Grégorien (1582), car elle ne prend pas en compte les jours perdus lors du changement de calendrier.

*FROM_DAYS(N)*

Retourne la date correspondant au nombre de jours ( `N` ) depuis la date 0 :

```
mysql> SELECT FROM_DAYS(729669);
        -> '1997-10-07'
```

`FROM_DAYS()` n'est pas fait pour travailler avec des dates qui précèdent l'avènement du calendrier Grégorien (1582), car elle ne prend pas en compte les jours perdus lors du changement de calendrier.

*DATE_FORMAT(date,format)*

Formate la date `date` avec le format `format` . Les spécificateurs suivants peuvent être utilisé dans la chaîne `format` :

| Specifier | Description |
|---|---|
| %M | Nom du mois ( January .. December ) |
| %W | Nom du jour de la semaine ( Sunday .. Saturday ) |
| %D | Jour du mois, avec un suffix anglais ( 1st , 2nd , 3rd , etc.) |
| %Y | Année, au format numérique, sur 4 chiffres |
| %y | Année, au format numérique, sur 2 chiffres |
| %X | Année, pour les semaines qui commencent le Dimanche, au format numérique, sur 4 chiffres, utilisé avec '%V' |
| %x | Année, pour les semaines qui commencent le Lundi, au format numérique, sur 4 chiffres, utilisé avec '%v' |
| %a | Nom du jour de la semaine, en abrégé et en anglais ( Sun .. Sat ) |
| %d | Jour du mois, au format numerique ( 00 .. 31 ) |
| %e | Jour du mois, au format numerique ( 0 .. 31 ) |
| %m | Mois, au format numerique ( 01 .. 12 ) |
| %c | Mois, au format numérique ( 1 .. 12 ) |
| %b | Nom du mois, en abrégé et en anglais ( Jan .. Dec ) |
| %j | Jour de l'année ( 001 .. 366 ) |
| %H | Heure ( 00 .. 23 ) |
| %k | Heure ( 0 .. 23 ) |
| %h | Heure ( 01 .. 12 ) |
| %I | Heure ( 01 .. 12 ) |
| %l | Heure ( 1 .. 12 ) |
| %i | Minutes, au format numerique ( 00 .. 59 ) |
| %r | Heures, au format 12-heures ( hh:mm:ss [AP]M ) |
| %T | Heures, au format 24-heures ( hh:mm:ss ) |
| %S | Secondes ( 00 .. 59 ) |
| %s | Secondes ( 00 .. 59 ) |
| %p | AM ou PM |
| %w | Numéro du jour de la semaine ( 0 =Sunday.. 6 =Saturday) |

| | |
|---|---|
| %U | Numéro de la semaine ( 00 .. 53 ), où Dimanche est le premier jour de la semaine |
| %u | Numéro de la semaine ( 00 .. 53 ), où Lundi est le premier jour de la semaine |
| %V | Numéro de la semaine ( 01 .. 53 ), où Dimanche est le premier jour de la semaine, utilisé avec '%X' |
| %v | Numéro de la semaine ( 01 .. 53 ), où Lundi est le premier jour de la semaine, utilisé avec '%x' |
| %% | Un signe pourcentage littéral '%' . |

Tous les autres caractères sont simplement copiés dans le résultat sans interprétation:

```
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%W %M %Y');
        -> 'Saturday October 1997'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');
        -> '22:23:00'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%D %y %a %d %m %b %j');
        -> '4th 97 Sat 04 10 Oct 277'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%H %k %I %r %T %S %w');
        -> '22 22 10 10:23:00 PM 22:23:00 00 6'
mysql> SELECT DATE_FORMAT('1999-01-01', '%X %V');
        -> '1998 52'
```

A partir de MySQL version 3.23, le caractère `'%'` est nécessaire avant tous les caractères de spécification de format. Dans les anciennes versions, il était optionnel.

*TIME_FORMAT(time,format)*

Cette fonction est utilisée exactement comme la fonction `DATE_FORMAT()` ci-dessus, mais la chaîne `format` ne doit utiliser que des spécificateurs d'heures, qui gèrent les heures, minutes et secondes. Les autres spécificateurs génèreront la valeur `NULL` ou `0` .

*CURDATE()*

*CURRENT_DATE*

Retourne la date courante au format `'YYYY-MM-DD'` ou `YYYYMMDD` , suivant le contexte numérique ou chaîne :

```
mysql> SELECT CURDATE();
        -> '1997-12-15'
mysql> SELECT CURDATE() + 0;
        -> 19971215
```

*CURTIME()*

*CURRENT_TIME*

Retourne l'heure courante au format `'HH:MM:SS'` ou `HHMMSS` , suivant le contexte numérique ou chaîne :

```
mysql> SELECT CURTIME();
        -> '23:50:26'
mysql> SELECT CURTIME() + 0;
        -> 235026
```

*NOW()*

*SYSDATE()*

*CURRENT_TIMESTAMP*

Retourne la date courante au format `'YYYY-MM-DD HH:MM:SS'` ou `YYYYMMDDHHMMSS` , suivant le contexte numérique ou chaîne :

```
mysql> SELECT NOW();
        -> '1997-12-15 23:50:26'
mysql> SELECT NOW() + 0;
        -> 19971215235026
```

Notez que `NOW()` est évalué une seule fois par requête, au début de l'exécution de celle-ci. Cela signifie que si il y a de multiples références à `NOW()` dans une requête, la même date sera utilisé.

*UNIX_TIMESTAMP()*

*UNIX_TIMESTAMP(date)*

Lorsqu'elle est appelé sans argument, cette fonction retourne un timestamp Unix (nombre de secondes depuis `'1970-01-01 00:00:00'` GMT). Si `UNIX_TIMESTAMP()` est appelé avec un argument `date` , elle retourne le timestamp correspondant à cette date. `date` peut être une chaîne de type `DATE` , `DATETIME` , `TIMESTAMP` , ou un nombre au format `YYMMDD` ou `YYYYMMDD` , en horaire local :

```
mysql> SELECT UNIX_TIMESTAMP();
        -> 882226357
mysql> SELECT UNIX_TIMESTAMP('1997-10-04 22:23:00');
        -> 875996580
```

Lorsque `UNIX_TIMESTAMP` est utilisé sur une colonne de type `TIMESTAMP` , la fonction reçoit directement la valeur, sans conversion explicite. Si vous donnez à `UNIX_TIMESTAMP()` une

date hors de son intervalle de validité, elle retourne 0.Si vous voulez soustraire une colonne de type `UNIX_TIMESTAMP()` , vous devez sûrement vouloir un résultat de type entier signé. Fonctions de transtypage .

*FROM_UNIXTIME(unix_timestamp)*

Retourne une représentation de l'argument `unix_timestamp` sous la forme `'YYYY-MM-DD HH:MM:SS'` ou `YYYYMMDDHHMMSS` , suivant si la fonction est utilisé dans un contexte numérique ou de chaîne.

```
 mysql> SELECT FROM_UNIXTIME(875996580);
        -> '1997-10-04 22:23:00'
mysql> SELECT FROM_UNIXTIME(875996580) + 0;
        -> 19971004222300
```

*SEC_TO_TIME(seconds)*

Retourne l'argument `seconds` , convertit en heures, minutes et secondes au format `'HH:MM:SS'` ou `HHMMSS` , suivant le contexte numérique ou chaîne :

```
 mysql> SELECT SEC_TO_TIME(2378);
        -> '00:39:38'
mysql> SELECT SEC_TO_TIME(2378) + 0;
        -> 3938
```

*TIME_TO_SEC(time)*

Retourne l'argument `time` , convertit en secondes :

```
 mysql> SELECT TIME_TO_SEC('22:23:00');
        -> 80580
mysql> SELECT TIME_TO_SEC('00:39:38');
        -> 2378
```

## 6.3.5 Fonctions de transtypage

La syntaxe de la fonction `CAST` est :

```
CAST(expression AS type)

ou

CONVERT(expression,type)
```

Où type est l'un des suivants :

- `BINARY`
- `CHAR` (Nouveau en 4.0.6)
- `DATE`
- `DATETIME`
- `SIGNED {INTEGER}`
- `TIME`
- `UNSIGNED {INTEGER}`

`CAST()` est la syntaxe ANSI SQL99 et `CONVERT()` est la syntaxe ODBC.

La fonction de transtypage est très pratique lorsque vous voulez créer une colonne avec un type spécifique dans une requête `CREATE ... SELECT` :

```
CREATE TABLE nouvelle_table SELECT CAST('2000-01-01' AS DATE);
```

`CAST(chaîne AS BINARY` est l'équivalent de `BINARY chaîne` .

Pour transformer une chaîne de caractères en une valeur numérique, vous ne devez rien faire de particulier ; juste utiliser la valeur de la chaîne en lieu et place de la valeur numérique :

```
mysql> SELECT 1+'1';
      -> 2
```

Si vous utilisez un nombre dans un contexte de chaîne, le nombre sera automatiquement converti en une chaîne binaire.

mysql> SELECT concat("salut toi ",2); −> "salut toi 2"

Si vous utilisez un nombre dans un contexte de chaîne, le nombre sera automatiquement converti en chaîne binaire ( `BINARY` ).

```
mysql> SELECT CONCAT("Salut vous ",2);
      ->  "Salut vous 2"
```

MySQL supporte l'arithmétique avec les valeurs 64 bits signées et non signées. Si vous utilisez une opération numérique (comme le signe `+` ) et qu'un des opérandes est de type `unsigned integer` , alors, le résultat sera une valeur non signé. Vous pouvez corriger cela en utilisant les opérateurs de transtypages `SIGNED` et `UNSIGNED` , qui transformeront l'opération respectivement en un entier signé sur 64 bits et un entier non signé sur 64 bits.

```
mysql> SELECT CAST(1−2 AS UNSIGNED)
       −> 18446744073709551615
mysql> SELECT CAST(CAST(1−2 AS UNSIGNED) AS SIGNED);
       −> −1
```

Notez que si l'une ou l'autre opération est une valeur à virgule flottante (Dans ce contexte, `DECIMAL()` est considéré comme une valeur à virgule flottante) le résultat devrait être une valeur à virgule flottante et ne sera pas affecté par la règle ci−dessus.

```
mysql> SELECT CAST(1 AS UNSIGNED) −2.0
       −> −1.0
```

Si vous utilisez une chaîne dans une opération arithmétique, elle sera converti en un nombre à virgule flottante.Les fonctions `CAST()` et `CONVERT()` ont été ajoutées dans la version 4.0.2 de MySQL.

Le rendu des valeurs non signées a été modifié dans la version 4.0 de MySQL pour pouvoir supporter correctement les valeurs de type `BIGINT` . Si vous voulez utiliser du code fonctionnant dans la version 4.0 et la version 3.23 de MySQL (dans ce cas, vous ne pouvez probablement pas utiliser les fonctions de transtypage), vous pouvez utiliser l'astuce suivante pour avoir un résultat signé lorsque vous soustrayez deux colonnes d'entier non signé :

```
SELECT (unsigned_column_1+0.0)−(unsigned_column_2+0.0);
```

L'idée est que les colonnes sont convertis en un point mobile avant de faire la soustraction.

Si vous rencontrez un problème avec les colonnes `UNSIGNED` dans vos anciennes applications MySQL lorsque vous effectuez le port sous la version 4.0 de MySQL , vous pouvez utiliser l'option `−−sql−mode=NO_UNSIGNED_SUBTRACTION` lorsque vous lancez `mysqld` . Notez cependant qu'aussi longtemps que vous employez ceci, vous ne serez pas capable d'utiliser efficacement les colonnes de type `UNSIGNED BIGINT` .

## 6.3.6 Autres fonctions

### 6.3.6.1 Fonctions sur les bits

MySQL utilise l'arithmétique des BIGINT (64−bits) pour les opérations sur les bits. Ces opérateurs travaillent donc sur 64 bits.

`|`

OU bit−à−bit (OR)

```
mysql> SELECT 29 | 15;
       −> 31
```

Le résultat est un entier de 64 bits non signé.

`&`

ET bit−à−bit (AND)

```
mysql> SELECT 29 & 15;
       −> 13
```

Le résultat est un entier de 64 bits non signé.

^

XOR bit−à−bit

```
mysql> SELECT 1 ^ 1;
       -> 0
mysql> SELECT 1 ^ 0;
       -> 1
mysql> SELECT 11 ^ 3;
       -> 8
```

Le résultat est un entier de 64 bits non signé.

<<

Décale les bits de l'entier ( `BIGINT` ) sur la gauche :

```
mysql> SELECT 1 << 2;
       -> 4
```

Le résultat est un entier de 64 bits non signé.

>>

Décale les bits de l'entier ( `BIGINT` ) sur la droite :

```
mysql> SELECT 4 >> 2;
       -> 1
```

Le résultat est un entier de 64 bits non signé.

~

Inverse tous les bits :

```
mysql> SELECT 5 & ~1;
       -> 4
```

Le résultat est un entier de 64 bits non signé.

*BIT_COUNT(N)*

Retourne le nombre de bits non nuls de l'argument `N` :

```
mysql> SELECT BIT_COUNT(29);
       -> 4
```

### 6.3.6.2 Fonctions diverses

*DATABASE()*

Retourne le nom de la base de données courante :

```
mysql> SELECT DATABASE();
       -> 'test'
```

Si aucune base de données n'a été sélectionnée, `DATABASE()` retourne une chaîne vide.

*USER()*
*SYSTEM_USER()*
*SESSION_USER()*

Retourne l'utilisateur MySQL courant :

```
mysql> SELECT USER();
       -> 'davida&#64;localhost'
```

A partir de la version 3.22.11 de MySQL, cette fonction inclut le nom de l'hôte du client tout comme le nom de l'utilisateur. Vous pouvez extraire le nom de l'utilisateur comme ceci (fonctionne également si le nom de l'hôte est présent) :

```
mysql> SELECT SUBSTRING_INDEX(USER(),"&#64;",1);
       -> 'davida'
```

*PASSWORD(str)*

Calcule un mot de passe chiffré à partir de la chaîne `str` . C'est cette fonction qui est utilisé pour crypter les mots de passes MySQL pour être stockés dans une colonne de type `Password` de la table `user` :

```
mysql> SELECT PASSWORD('badpwd');
       -> '7f84554057dd964b'
```

Le chiffrage par `PASSWORD()` n'est pas réversible.

`PASSWORD()` n'est pas un chiffrage comparable à la fonction Unix de chiffrage. Si votre mot de passe Unix et votre mot de passe MySQL sont identiques, il n'est ABSOLUMENT pas certain que Unix et MySQL le chiffreront de la même façon. Voir `ENCRYPT()` .

*ENCRYPT(str[,salt])*

Chiffre la chaîne `str` en utilisant la fonction `crypt()` . L'argument `salt` doit être une chaîne de deux caractères. (A partir de la version 3.22.16, l'argument `salt` peut être plus long que deux caractères.) :

```
mysql> SELECT ENCRYPT("hello");
       -> 'VxuFAJXVARROc'
```

Si la fonction `crypt()` n'est pas disponible sur votre système, la fonction `ENCRYPT()` retournera toujours `NULL` .

La fonction `ENCRYPT()` conserve uniquement les 8 premiers caractères de la chaîne `str` , au moins, sur certains système. Le comportement exact est directement déterminé par la fonction système `crypt()` sous−jacente.

*ENCODE(str,pass_str)*

Chiffre la chaîne `str` en utilisant la clé `pass_str` . Pour déchiffrer le résultat, utilisez la fonction `DECODE()` .Le résultat est une chaîne binaire de la même longueur que `string` . Si vous voulez sauvegarder le résultat dans une colonne, utilisez une colonne de type `BLOB` .

*DECODE(crypt_str,pass_str)*

Déchiffre la chaîne chiffrée `crypt_str` en utilisant la clé `pass_str` . `crypt_str` doit être une chaîne qui a été renvoyée par la fonction `ENCODE()` .

*MD5(string)*

Calcul la somme de vérification MD5 de la chaîne `string` . La valeur retournée est un entier hexadécimal de 32 caractères qui peut être utilisé, par exemple, comme clé de hachage :

```
mysql> SELECT MD5("testing");
       -> 'ae2b1fca515949e5d54fb22b8ed95575'
```

C'est l'algorithme RSA ("RSA Data Security, Inc. MD5 Message−Digest Algorithm").

*SHA1(string)*

*SHA(string)*

Calcul la somme de vérification SHA1 160 bit de la chaîne `string` , comme décrit dans la RFC 3174 (Secure Hash Algorithm). La valeur retournée est un entier hexadécimal de 40 caractères, ou bien `NULL` dans le cas où l'argument vaut `NULL` . Une des possibilités d'utilisation de cette fonction est le hachage de clé. Vous pouvez aussi l'utilisé comme fonction de cryptographie sûre pour stocker les mots de passe.

```
mysql> SELECT SHA1("abc");
       -> 'a9993e364706816aba3e25717850c26c9cd0d89d'
```

La fonction `SHA1()` a été ajoutée dans la version 4.0.2 de MySQL et peut être considérée comme une méthode de cryptographie plus sûre que la fonction `MD5()` . La fonction `SHA()` est un alias de la fonction `SHA1()` .

*AES_ENCRYPT(string,key_string)*

*AES_DECRYPT(string,key_string)*

Ces fonctions permettent le chiffrement/déchiffrement de données utilisant l'algorithme AES (Advanced Encryption Standard), anciennement connu sous le nom de Rijndael. Une clé de 128 bits est utilisé pour le chiffrement, mais vous pouvez l'étendre à 256 bits en patchant la source. Nous avons choisi 128 bits parce que c'est plus rapide et assez sécurisé.

Les arguments peuvent être de n'importe quelle taille. Si l'un des arguments est `NULL` , le résultat de cette fonction sera `NULL` .

Vu que AES est un algorithme de niveau bloc, le capitonnage est utilisé pour chiffrer des chaînes de longueur inégales et donc, la longueur de la chaîne résultante peut être calculée comme ceci : 16*(trunc(string_length/16)+1).

Si la fonction `AES_DECRYPT()` détecte des données invalides ou un capitonnage incorrect, elle retournera `NULL` . Il est également possible que la fonction `AES_DECRYPT()` retourne une valeur différente de `NULL` (valeur incohérente) si l'entrée de données ou la clé est

invalide.

Vous pouvez utiliser les fonctions AES pour stocker des données sous une forme chiffrée en modifiant vos requêtes:

```
INSERT INTO t VALUES (1,AES_ENCRYPT("text","password"));
```

Vous pouvez obtenir encore plus de sécurité en évitant de transférer la clé pour chaque requête, en la stockant dans une variable sur le serveur au moment de la connexion :

```
SELECT &#64;password:="my password";
INSERT INTO t VALUES (1,AES_ENCRYPT("text",&#64;password));
```

Les fonctions `AES_ENCRYPT()` et `AES_DECRYPT()` ont été ajoutées dans la version 4.0.2 de MySQL et peuvent être considérées comme étant les fonctions de cryptographie les plus sûres disponibles actuellement dans MySQL.

*DES_ENCRYPT(string_to_encrypt [, (key_number | key_string) ] )*

Chiffre la chaîne avec la clé donnée en utilisant l'algorithme DES.

Notez que cette fonction fonctionne uniquement si vous avez configuré MySQL avec le support SSL. Utilisation des connexions sécurisées .

La clé de hachage utilisée est choisie en suivant les recommandations suivantes :

| Argument | Description |
|---|---|
| Un seul argument | La première clé de des-key-file est utilisée. |
| Un numéro de clé | Le numéro de la clé donnée (0-9) de des-key-file est utilisée. |
| Une chaîne | La chaîne donnée key_string doit être utilisé pour chiffrer string_to_encrypt . |

La chaîne retournée doit être une chaîne binaire où le premier caractère doit être `CHAR(128 | key_number)` .

Le nombre 128 a été ajouté pour reconnaître facilement une clé de hachage. Si vous utilisez une chaîne comme clé, `key_number` doit être 127.

Si une erreur survient, la fonction retournera `NULL` .

La longueur de la chaîne de résultat doit être : `new_length= org_length + (8-(org_length % 8))+1` .

`des-key-file` a le format suivant :

```
key_number des_key_string
key_number des_key_string
```

Chaque `key_number` doit être un nombre dans l'intervalle 0 à 9. Les lignes dans le fichier peuvent être dans n'importe quel ordre. `des_key_string` est la chaîne qui permettera le chiffrage du message. Entre le nombre et la clé, il doit y avoir au moins un espace. La première clé est la clé par défaut qui sera utilisé si vous ne spécifiez pas d'autres clés en arguments de la fonction `DES_ENCRYPT()` .Vous pouvez demander à MySQL de lire de nouvelles valeurs de clé dans le fichier de clés avec la commande `FLUSH DES_KEY_FILE` . Cela requière le privilège `Reload_priv` .

Un des bénéfices d'avoir une liste de clés par défaut est que cela donne aux applications la possibilité de regarder l'existence de la valeur chiffrée de la colonne, sans pour autant donner la possibilité à l'utilisateur final de déchiffrer ces valeurs.

```
mysql> SELECT customer_address FROM customer_table WHERE
      crypted_credit_card = DES_ENCRYPT("credit_card_number");
```

*DES_DECRYPT(string_to_decrypt [, key_string])*

Déchiffre une chaîne chiffrée à l'aide de la fonction `DES_ENCRYPT()` .Notez que cette fonction fonctionne uniquement si vous avez configuré MySQL avec le support SSL.

Utilisation des connexions sécurisées .

Si l'argument `key_string` n'est pas donné, la fonction `DES_DECRYPT()` examine le premier bit de la chaîne chiffrée pour déterminer le numéro de clé DES utilisé pour chiffrer la chaîne originale, alors la clé est lu dans le fichier `des-key-file` pour déchiffrer le message. Pour pouvoir utiliser cela, l'utilisateur doit avoir le privilège `SUPER` .

Si vous passé l'argument `key_string` à cette fonction, cette chaîne est utilisée comme clé pour déchiffrer le message.

Si la chaîne `string_to_decrypt` ne semble pas être une chaîne chiffrée, MySQL retournera la chaîne `string_to_decrypt` .

Si une erreur survient, cette fonction retourne `NULL` .

*LAST_INSERT_ID([expr])*

Retourne la dernière valeur générée automatiquement qui a été insérée dans une colonne de type `AUTO_INCREMENT` . `mysql_insert_id()` .

```
mysql> SELECT LAST_INSERT_ID();
       -> 195
```

Le dernier ID généré est conservé par le serveur pour chaque connexion. Un autre client ne le modifiera donc pas. Il ne sera pas modifié non plus si vous modifiez directement la valeur d'une colonne `AUTO_INCREMENT` avec une valeur simple (c'est à dire, une valeur qui n'est ni NULL, ni 0). Si vous insérez plusieurs lignes au même moment avec une requête insert, `LAST_INSERT_ID()` retourne la valeur de la première ligne insérée. La raison de cela est de rendre possible de reproduire facilement la même requête `INSERT` sur d'autres serveurs.

Si `expr` est donnée en argument à la fonction `LAST_INSERT_ID()` , alors la valeur de l'argument sera retourné par la fonction et sera enregistré comme étant la prochaine valeur retournée par `LAST_INSERT_ID()` . Cela peut être utilisé pour simuler des séquences :

Commencez par créer la table suivante :

```
mysql> CREATE TABLE sequence (id INT NOT NULL);
mysql> INSERT INTO sequence VALUES (0);
```

Alors cette table peut être utilisé pour générer des séquences de nombres, comme ceci :

```
mysql> UPDATE sequence SET id=LAST_INSERT_ID(id+1);
```

Vous pouvez générer des séquences sans appeler la fonction `LAST_INSERT_ID()` , mais l'utilité d'utiliser cette fonction cette fois si est que la valeur ID est gérée par le serveur comme étant la dernière valeur générée automatiquement. (sécurité multi-utilisateur). Vous pouvez retrouver la nouvelle ID tout comme vous pouvez lire n'importe quelle valeur `AUTO_INCREMENT` dans MySQL. Par exemple, la fonction `LAST_INSERT_ID()` (sans argument) devrait retourner la nouvelle ID. La fonction C de l'API `mysql_insert_id()` peut être également utilisée pour trouver cette valeur.

Notez que la fonction `mysql_insert_id()` est incrémentée uniquement après des requêtes `INSERT` et `UPDATE` , donc, vous ne pouvez pas utiliser la fonction C de l'API pour trouver la valeur de `LAST_INSERT_ID(expr)` après avoir exécuté d'autres types de requêtes, comme `SELECT` ou bien `SET` .

*FORMAT(X,D)*

Formate l'argument `X` en un format comme `'#,###,###.##'` , arrondi à `D` décimales. Si `D` vaut `0` , le résultat n'aura ni séparateur décimal, ni partie décimale :

```
mysql> SELECT FORMAT(12332.123456, 4);
       -> '12,332.1235'
mysql> SELECT FORMAT(12332.1,4);
```

```
        -> '12,332.1000'
mysql> SELECT FORMAT(12332.2,0);
        -> '12,332'
```

*VERSION()*

Retourne une chaîne indiquant la version courante du serveur MySQL :

```
mysql> SELECT VERSION();
        -> '3.23.13-log'
```

Notez que si votre version se termine par `-log` , cela signifie que le système d'historique est actif.

*CONNECTION_ID()*

Retourne l'identifiant de connexion courant ( `thread_id` ). Chaque connexion a son propre identifiant unique :

```
mysql> SELECT CONNECTION_ID();
        -> 1
```

*GET_LOCK(str,timeout)*

Tente de poser un verrou nommé `str` , avec un délai d'expiration ( `timeout` ) exprimé en seconde. Retourne `1` si le verrou a été posé avec succès, `0` si il n'a pas pu être posé avant l'expiration du délai et `NULL` si une erreur est survenu (comme par exemple un manque de mémoire, ou la mort du thread lui−même, par `mysqladmin kill` ). Un verrou sera levé lorsque vous exécuterez la commande `RELEASE_LOCK()` , `GET_LOCK()` ou si le thread se termine. Cette fonction peut être utilisée pour implémenter des verrous applicatifs ou pour simuler des verrous de lignes. Les requêtes concurrentes des autres clients de même nom seront bloquées ; les clients qui s'entendent sur un nom de verrou peuvent les utiliser pour effectuer des verrouillages coopératifs :

```
mysql> SELECT GET_LOCK("lock1",10);
        -> 1
mysql> SELECT IS_FREE_LOCK("lock2");
        -> 1
mysql> SELECT GET_LOCK("lock2",10);
        -> 1
mysql> SELECT RELEASE_LOCK("lock2");
        -> 1
mysql> SELECT RELEASE_LOCK("lock1");
        -> NULL
```

Notez que le deuxième appel à `RELEASE_LOCK()` retourne `NULL` car le verrou `"lock1"` a été automatiquement libéré par le deuxième appel à `GET_LOCK()` .

*RELEASE_LOCK(str)*

Libère le verrou nommé `str` , obtenu par la fonction `GET_LOCK()` . Retourne `1` si le verrou a bien été libéré, `0` si le verrou n'a pas été libéré par le thread (dans ce cas, le verrou reste posé) et `NULL` si le nom du verrou n'existe pas. Le verrou n'existe pas si il n'a pas été obtenu par la fonction `GET_LOCK()` ou si il a déjà été libéré.La commande `DO` est utilisable avec `RELEASE_LOCK()` . Syntaxe des `DO` .

*IS_FREE_LOCK(str)*

Regarde si le verrou nommé `str` peut être librement utilisé (i.e., non verrouillé). Retourne `1` si le verrou est libre (personne ne l'utilise), `0` si le verrou est actuellement utilisé et `NULL` si une erreur survient (comme un argument incorrect).

*BENCHMARK(count,expr)*

La fonction `BENCHMARK()` exécute l'expression `expr` de manière répétée `count` fois. Elle permet de tester la vélocité de MySQL lors du traitement d'une requête. Le résultat est toujours `0` . L'objectif de cette fonction ne se voit que du côté client, qui permet à ce dernier d'afficher la durée d'exécution de la requête :

```
mysql> SELECT BENCHMARK(1000000,ENCODE("bonjour","au revoir"));
+-------------------------------------------------+
| BENCHMARK(1000000,ENCODE("bonjour","au revoir")) |
+-------------------------------------------------+
|                                               0 |
+-------------------------------------------------+
1 row in set (4.74 sec)
```

Le temps affiché est le temps côté client, et non pas les ressources processeurs consommées. il est conseillé d'utiliser `BENCHMARK()` plusieurs fois de suite pour interpréter un résultat, en dehors de charges ponctuelles sur le serveur.

*INET_NTOA(expr)*

Retourne l'adresse réseau (4 ou 8 octets), de l'expression numérique `exp` :

```
mysql> SELECT INET_NTOA(3520061480);
    ->  "209.207.224.40"
```

*INET_ATON(expr)*

Retourne un entier qui représente l'expression numérique de l'adresse réseau. Les adresses peuvent être des entiers de 4 ou 8 octets.

```
mysql> SELECT INET_ATON("209.207.224.40");
    ->  3520061480
```

Le nombre généré est toujours dans l'ordre des octets réseau ; par exemple, le nombre précédent est calculé comme ceci : `209*256^3 + 207*256^2 + 224*256 +40` .

*MASTER_POS_WAIT(log_name, log_pos)*

Bloque le maître jusqu'à ce que l'esclave atteigne une position donnée dans le fichier d'historique principal, durant une replication. Si l'historique principal n'est pas initialisé, retourne `NULL` . Si l'esclave n'est pas démarré, le maître restera bloqué jusqu'à ce que l'esclave soit démarré et ai atteint la position demandée. Si l'esclave a déjà dépassé cette position, la fonction se termine immédiatement. La valeur retournée est le nombre d'évènements qui a du être traité pour atteindre la position demandée, ou `NULL` en cas d'erreur. Cette fonction est très utile pour contrôler la synchronisation maître−esclave, mais elle a été initialement écrite pour faciliter les tests de replications.

*FOUND_ROWS()*

Retourne le nombre de lignes que la dernière commande `SELECT SQL_CALC_FOUND_ROWS ...` a retourné, si aucune limite n'a été défini par `LIMIT` .

```
mysql> SELECT SQL_CALC_FOUND_ROWS * FROM nom_de_table
    WHERE id > 100 LIMIT 10;
mysql> SELECT FOUND_ROWS();
```

Le second `SELECT` retourne un nombre indiquant combien de lignes le premier `SELECT` aurait retourné si il n'avait pas été écrit avec une clause `LIMIT` .Notez que si vous utilisez `SELECT SQL_CALC_FOUND_ROWS ...` , MySQL calcule toutes les lignes dans la liste des résultats. Ainsi, c'est plus rapide si vous n'utilisez pas de clause `LIMIT` et que la liste des résultats n'a pas besoin d'être envoyée au client.

`SQL_CALC_FOUND_ROWS` est utilisable à partir de la version 4.0.0 de MySQL.

## 6.3.7 Fonctions avec la clause GROUP BY

Si vous utilisez une fonction de groupement avec une commande qui n'utilise pas la clause `GROUP BY` , cela revient à regrouper toutes les lignes ensembles.

*COUNT(expr)*

Retourne le nombre de valeurs non−nulles ( `NULL` ) dans les lignes lues par la commande `SELECT` :

```
mysql> SELECT etudiant.nom_etudiant,COUNT(*)
   ->       FROM etudiant,course
   ->       WHERE etudiant.id_etudiant=course.id_etudiant
   ->       GROUP BY nom_etudiant;
```

`COUNT(*)` est légèrement différent car il retourne le nombre de lignes lues, qu'elles contiennent ou pas la valeur `NULL` . `COUNT(*)` est optimisé pour retourner très rapidement le résultat si la commande `SELECT` n'opère que sur une table, qu'aucune autre valeur n'est calculée en même temps, et qu'il n'y a pas de clause `WHERE` . Par exemple :

```
mysql> SELECT COUNT(*) FROM etudiant;
```

*COUNT(DISTINCT expr,[expr...])*

Retourne le nombre de lignes ayant des valeurs distinctes non−nulles ( `NULL` ) :

```
mysql> SELECT COUNT(DISTINCT resultat) FROM etudiant;
```

Avec MySQL, vos pouvez connaître le nombre de combinaison d'expressions distinctes qui ne contiennent pas la valeur `NULL` en donnant une liste d'expressions. En ANSI SQL, vous devriez faire la concaténation de toutes les expressions de clause `COUNT(DISTINCT ...)` .

*AVG(expr)*

Retourne la valeur moyenne de la colonne `expr` :

```
mysql> SELECT nom_etudiant, AVG(test_score)
   ->        FROM etudiant
   ->        GROUP BY nom_etudiant;
```

*MIN(expr)*
*MAX(expr)*

Retourne le minimum ou le maximum de l'expression `expr` . `MIN()` et `MAX()` acceptent aussi les chaînes comme argument ; dans ce cas, elles retournent le minimum ou le maximum de la valeur de la chaîne. Comment MySQL utilise les index .

```
mysql> SELECT nom_etudiant, MIN(test_score), MAX(test_score)
   ->        FROM etudiant
   ->        GROUP BY nom_etudiant;
```

Dans `MIN()` , `MAX()` et les autres fonctions d'agrégat, MySQL compare les colonnes de type `ENUM` et `SET` par la valeur de leur chaîne au lieu de la position relative de la chaîne dans la liste. Cela sera rectifié.

*SUM(expr)*

Retourne la somme de l'expression `expr` . Notez que si aucune ligne n'est sélectionnée, la fonction retournera NULL !

*VARIANCE(expr)*

Retourne la variance standard de `expr` . C'est une extension à la norme ANSI SQL. (disponible à partir de la version 4.1).

*STD(expr)*
*STDDEV(expr)*

Retourne la déviation standard de l'expression `expr` . C'est une extension à la norme ANSI SQL. La fonction `STDDEV()` est fournie pour assurer la compatiblité avec Oracle.

*BIT_OR(expr)*

Retourne le `OR` bit−à−bit de l'expression `expr` . Les calculs sont effectués avec une précision de 64 bits ( `BIGINT` ).

*BIT_AND(expr)*

Retourne le `AND` bit−à−bit de l'expression `expr` . Les calculs sont effectués avec une précisioon de 64 bits ( `BIGINT` ).

MySQL a étendu l'utilisation de la clause `GROUP BY` . Vous pouvez utiliser des noms de colonnes ou des expressions qui n'apparaissent pas dans la clause `GROUP BY` . Elles prennent alors **n'importe quelle valeur possible dans ce groupe** . Vous pouvez les utiliser pour améliorer les performances, en évitant de trier et regrouper des éléments non critiques. Par exemple, vous n'avez pas besoin de faire de groupement par `client.nom` dans la requête suivante :

```
mysql> SELECT order.custid,client.nom,MAX(payments)
   ->        FROM order,client
   ->        WHERE order.custid = client.custid
   ->        GROUP BY order.custid;
```

En ANSI SQL, vous devez ajouter `client.nom` à la clause `GROUP BY` . En MySQL, le nom est redondant si vous n'utilisez par le mode ANSI. **N'utilisez surtout pas cette caractéristique** si les colonnes que vous omettez n'ont pas de valeur unique au sein du groupe! Vous obtiendriez des résultas incohérents. Dans certains cas, vous pouvez utiliser `MIN()` et `MAX()` pour obtenir une valeur d'une colonne spécifique, même si elle n'est pas unique. La fonction suivante calcule la valeur de `column` dans la ligne contenant la plus petite valeur de la colonne `sort` :

```
SUBSTR(MIN(CONCAT(RPAD(sort,6,' '),column)),7)
```

La ligne contenant la plus grande valeur d'un certain champ par rapport à un groupe . Notez que si vous utilisez la version 3.22 de MySQL (ou plus récent), ou si vous essayez d'utiliser le mode ANSI SQL, vous ne pouvez pas utiliser d'expressions dans les clauses `GROUP BY` et `ORDER BY` . Vous pouvez contourner ces limitations en utilisant un alias d'expression :

```
mysql> SELECT id,FLOOR(value/100) AS val FROM nom_de_table
   ->        GROUP BY id,val ORDER BY val;
```

En MySQL version 3.23, vous pouvez faire :

```
mysql> SELECT id,FLOOR(value/100) FROM nom_de_table ORDER BY RAND();
```

# 6.4 Manipulation de données : SELECT , INSERT , UPDATE , DELETE

## 6.4.1 Syntaxe de SELECT

```
SELECT [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
      [DISTINCT | DISTINCTROW | ALL]
   select_expression,...
   [INTO &#123;OUTFILE | DUMPFILE&#125; 'nom_fichier' export_options]
   [FROM table_references
     [WHERE where_definition]
     [GROUP BY &#123;unsigned_integer | nom_de_colonne | formula&#125; [ASC | DESC], ...
     [HAVING where_definition]
     [ORDER BY &#123;unsigned_integer | nom_de_colonne | formula&#125; [ASC | DESC] ,...]
     [LIMIT [offset,] lignes]
     [PROCEDURE procedure_name(argument_list)]
     [FOR UPDATE | LOCK IN SHARE MODE]]
```

`SELECT` est utilisé pour obtenir des enregistrements venant d'une ou plusieurs tables. `select_expression` indique les champs que vous voulez obtenir. `SELECT` peut aussi être utilisé pour obtenir des enregistrements calculés sans aucune référence à une table. Par exemple :

```
mysql> SELECT 1 + 1;
        -> 2
```

Tous les mots−clés utilisés doivent être donnés exactement dans le même ordre que ci−dessus. Par exemple, une clause `HAVING` doit être placée après toute clause `GROUP BY` et avant toute clause `ORDER BY` .

- Une expression `SELECT` peut être aliasée en utilisant `AS` . L'alias est utilisé de la même façon que le nom du champ et peut être employé avec des clauses `ORDER BY` ou `HAVING` . Par exemple :
  ```
  mysql> SELECT CONCAT(last_name,', ',first_name) AS full_name
     FROM mytable ORDER BY full_name;
  ```
- Il n'est pas possible d'utiliser un alias de champ dans une clause `WHERE` , car la valeur du champ peut ne pas être définie lorsque la clause `WHERE` est exécutée. Problèmes avec les `alias` .
- La clause `FROM table_references` indique les tables à partir desquelles nous allons obtenir les enregistrements. Si vous indiquez le nom de plusieurs tables, vous faites une jointure. Pour davantage d'informations sur la syntaxe des jointures, consultez Syntaxe de `JOIN` . Pour chaque table spécifiée, vous pouvez éventuellement indiquer un alias.
  ```
  table_name [[AS] alias] [USE INDEX (key_list)] [IGNORE INDEX (key_list)]
  ```
  Depuis la version 3.23.12 de MySQL, vous pouvez donner des indications sur les index que MySQL doit utiliser lors de l'obtention des enregistrements venant d'une table. C'est pratique lorsque `EXPLAIN` montre que MySQL utilise le mauvais index parmi la liste des index possibles. En indiquant `USE INDEX (key_list)` , vous pouvez dire à MySQL d'utiliser un seul des index possibles pour trouver des enregistrements dans la table. L'autre syntaxe `IGNORE INDEX (key_list)` peut être utilisée pour dire à MySQL de ne pas utiliser un index en particulier. `USE/IGNORE KEY` sont des synonymes de `USE/IGNORE INDEX` .
- Vous pouvez vous référer à une table avec `nom_de_table` (au sein de la base de données courante), ou avec `dbname.nom_de_table` pour expliciter le nom de la base de données. Vous pouvez vous référer à un champ avec `nom_de_colonne` , `nom_de_table.nom_de_colonne` , ou `db_name.nom_de_table.nom_de_colonne` . Vous n'êtes pas obligés d'indiquer de préfixe `nom_de_table` ou `db_name.nom_de_table` pour une référence à un champ dans un `SELECT` , à moins que la référence ne soit ambigüe. Consultez Noms de base de données, tables, index, colonnes et alias , pour des exemples d'ambigüités qui nécessitent des formes plus explicites de référence à des champs.
- Une référence à une table peut être aliasée en utilisant `nom_de_table [AS] alias_name` :
  ```
  mysql> SELECT t1.name, t2.salary FROM employee AS t1, info AS t2
     ->        WHERE t1.name = t2.name;
  ```

```
mysql> SELECT t1.name, t2.salary FROM employee t1, info t2
    ->        WHERE t1.name = t2.name;
```

- Vous pouvez faire référence aux champs sélectionnés en sortie dans des clauses `ORDER BY` et `GROUP BY` en utilisant les noms des champs, les alias des champs ou bien les positions des champs. Les positions des champs commencent à 1 :

```
mysql> SELECT college, region, seed FROM tournament
    ->        ORDER BY region, seed;
mysql> SELECT college, region AS r, seed AS s FROM tournament
    ->        ORDER BY r, s;
mysql> SELECT college, region, seed FROM tournament
    ->        ORDER BY 2, 3;
```

  Pour trier dans l'ordre inverse, ajoutez le mot−clé `DESC` (descendant) au nom du champ dans la clause `ORDER BY` qui vous permet de trier. Par défaut, l'ordre ascendant est utilisé; ceci peut être indiqué de façon explicite en utilisant le mot−clé `ASC` .
- Vous pouvez, dans une clause `WHERE` , utiliser n'importe laquelle des fonctions que MySQL supporte. Fonctions .
- La clause `HAVING` peut faire référence à n'importe quel champs ou alias défini dans `select_expression` . C'est évalué en dernier lieu, juste avant que les éléments ne soient envoyés au client, sans aucune optimisation. N'utilisez pas `HAVING` pour des éléments qui devraient être dans la clause `WHERE` . Par exemple, n'écrivez pas ceci :

```
mysql> SELECT nom_de_colonne FROM nom_de_table HAVING nom_de_colonne > 0;
```

  Ecrivez plutôt cela :

```
mysql> SELECT nom_de_colonne FROM nom_de_table WHERE nom_de_colonne > 0;
```

  Dans les versions 3.22.5 et supérieures de MySQL, vous pouvez aussi écrire des requêtes ainsi :

```
mysql> SELECT user,MAX(salary) FROM users
    ->        GROUP BY user HAVING MAX(salary)>10;
```

  Dans des versions plus anciennes de MySQL, vous pouvez écrire à la place :

```
mysql> SELECT user,MAX(salary) AS sum FROM users
    ->        group by user HAVING sum>10;
```

- Les options `DISTINCT` , `DISTINCTROW` et `ALL` indiquent quels enregistrements avec doublons doivent être retournés. Par défaut, c'est ( `ALL` ), retournant ainsi tous les enregistrements. `DISTINCT` et `DISTINCTROW` sont synonymes et indique que les doublons doivent être éliminés du résultat.
- Toutes les options commençant par `SQL_` , `STRAIGHT_JOIN` , et `HIGH_PRIORITY` sont des extensions MySQL de ANSI SQL.
- `HIGH_PRIORITY` donne à une commande `SELECT` une plus grande priorité qu'une commande qui modifie une table. Vous devez l'utiliser seulement pour les requêtes qui sont très rapides et qui doivent être effectuées en premier lieu. Une requête `SELECT HIGH_PRIORITY` s'exécutera sur une table verrouillée en lecture même si une commande de mise à jour attend que la table soit libérée.
- `SQL_BIG_RESULT` peut être utilisé avec `GROUP BY` ou `DISTINCT` pour indiquer à l'optimiseur que le résultat comportera beaucoup d'enregistrements. Dans ce cas, MySQL utilisera si besoin directement les bases temporaires stockées sur le disque. MySQL préférera, dans ce cas, trier que d'obtenir une table temporaire avec une clé sur les éléments du `GROUP BY` .
- `SQL_BUFFER_RESULT` forcera le résultat à être stocké dans une table temporaire. Ceci va aider MySQL à libérer plus tôt les verrous des tables et aidera aussi dans les cas ou l'envoi du résultat au client prend un temps assez conséquent.
- `SQL_SMALL_RESULT` , une option spécifique à MySQL, peut être utilisée avec `GROUP BY` ou `DISTINCT` pour indiquer à l'optimiseur que le résultat sera petit. Dans ce cas, MySQL utilise des tables temporaires rapides pour stocker la table résultante plutôt que d'utiliser le tri. Dans MySQL 3.23, ceci n'est normalement pas nécessaire.
- `SQL_CALC_FOUND_ROWS` (version 4.0.0 et supérieure) indique à MySQL de calculer combien d'enregistrements seront dans le jeu de résultats, indépendamment de n'importe quelle clause `LIMIT` . Le nombre d'enregistrements peut alors être obtenu avec `SELECT FOUND_ROWS()` . Fonctions diverses .Veuillez noter que dans les versions antérieures à 4.1.0, ceci ne fonctionne pas avec `LIMIT 0` , qui est optimisé pour retourner le résultat instantanément (retournant bien entendu un nombre d'enregistrements égal à 0). Comment MySQL optimise `LIMIT` .

- `SQL_CACHE` indique à MySQL de stocker le résultat de la requête dans le cache de requêtes si vous utilisez `QUERY_CACHE_TYPE=2` ( `DEMAND` ). Cache de requêtes MySQL .
- `SQL_NO_CACHE` indique à MySQL de ne pas permettre au résultat de la requête d'être stocké dans le cache de requêtes. Cache de requêtes MySQL .
- Si vous utilisez `GROUP BY` , les enregistrements en sortie seront triés en accord avec le `GROUP BY` comme si vous aviez eu un `ORDER BY` sur tous les champs du `GROUP BY` . MySQL a étendu `GROUP BY` de façon à ce que vous puissiez spécifier `ASC` et `DESC` lors du `GROUP BY` :
```
SELECT a,COUNT(b) FROM test_table GROUP BY a DESC
```
- MySQL a étendu l'utilisation de `GROUP BY` pour permettre de sélectionner des champs qui ne sont pas mentionnés dans la clause `GROUP BY` . Si vous n'obtenez pas les résultats que vous attendez de votre requête, veuillez consulter la description de `GROUP BY` . Fonctions à utiliser dans les définitions `GROUP BY` .
- `STRAIGHT_JOIN` force l'optimiseur à joindre les tables dans l'ordre où elles sont listées dans la clause `FROM` . Vous pouvez utiliser cela pour accélérer une requête si l'optimiseur joint les tables dans un ordre qui n'est pas optimal. Syntaxe de `EXPLAIN` (Obtenir des informations `SELECT` ) .
- La clause `LIMIT` peut être utilisée pour limiter le nombre d'enregistrements retournés par la commande `SELECT` . `LIMIT` accepte un ou deux arguments numériques. Ces arguments doivent être des entiers constants.

  Si deux arguments sont donnés, le premier indique le décalage du premier enregistrement à retourner, le second donne le nombre maximum d'enregistrement à retourner. Le décallage du premier enregistrement est 0 (pas 1):

```
mysql> SELECT * FROM table LIMIT 5,10;  # Retourne les enregistrements 6 à 15
```

  Pour obtenir tous les enregistrement d'un certain décallage jusqu'à la fin du résultat, vous pouvez utiliser −1 en tant que second paramètre:

```
mysql> SELECT * FROM table LIMIT 95,-1; # Retourne les enregistrements de 96 jusqu'au dernier.
```

  Si un seul argument est donné, il indique le nombre maximum d'enregistrements à retourner :
```
mysql> SELECT * FROM table LIMIT 5;     # Retourne les 5 premiers enregistrements
```
  Autrement dit, `LIMIT n` est équivalent à `LIMIT 0,n` .
- La forme `SELECT ... INTO OUTFILE 'nom_fichier'` de `SELECT` écrit les lignes sélectionnées dans un fichier. Le fichier est crée sur le serveur et ne peut y être déjà présent (cela permet entre autre d'éviter la destruction des tables et de fichiers tel que `/etc/passwd` ). Vous devez avoir le droit `FILE` sur le serveur pour utiliser cette forme de `SELECT` . `SELECT ... INTO OUTFILE` à pour but principal de vous permettre de réaliser des dumps rapides des tables sur la machine serveur. Si vous voulez créer le fichier sur une autre machine, vous ne pouvez utiliser `SELECT ... INTO OUTFILE` . Dans ce cas là, vous pouvez utiliser à la place un programme client comme `mysqldump --tab` ou `mysql -e "SELECT ..." > fichier` pour générer le fichier.

  `SELECT ... INTO OUTFILE` est le complément de `LOAD DATA INFILE` ; La syntaxe pour la partie `export_options` de la requête se compose des mêmes clauses `FIELDS` et `LINES` que celles utilisées avec la commande `LOAD DATA INFILE` . Syntaxe des `LOAD DATA INFILE` .

  Dans le fichier résultant, seul les caractères suivants sont protégés par le caractère `ESCAPED BY` :

  - ♦ Le caractère `ESCAPED BY`
  - ♦ Les premier caractère de `FIELDS TERMINATED BY`
  - ♦ Les premier caractère de `LINES TERMINATED BY`

  De plus, `ASCII 0` est convertit en `ESCAPED BY` suivi de 0 ( `ASCII 48` ).La raison de ce qui précède est que vous devez **impérativement** protéger chaque caractère `FIELDS TERMINATED BY` , `ESCAPED BY` , ou `LINES TERMINATED BY` pour assurer une relecture fiable du fichier. Le caractère `ASCII 0` est échappé

pour assurer la lisibilité sur certains clients. Comme le fichier résultant ne se doit pas d'être syntaxiquement conforme à SQL, vous n'avez besoin d'échapper rien d'autre.

Voila un exemple de relecture de fichier au format utilisé par plusieurs anciens programmes.

```
 SELECT a,b,a+b INTO OUTFILE "/tmp/result.text"
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY "\n"
FROM test_table;
```

- Si vous utilisez `INTO DUMPFILE` au lieu de `INTO OUTFILE` , MySQL n'écrira qu'une seule ligne dans le fichier, sans aucun caractère de fin de ligne ou de colonne, ni d'échappement. Cela est utile lorsque vous voulez enregistrer un blob dans un fichier.
- Notez que chaque fichier crée par `INTO OUTFILE` et `INTO DUMPFILE` sera lisible par tout les utilisateurs ! La raison est que le serveur MySQL ne peut créer de fichier appartenant à autre que l'utilisateur qui l'a mit en route. (vous devez éviter d'exécuter `mysqld` en tant que root), le fichier doit se composer de mot lisible pour que les données puissent être récupérées.
- Si vous utilisez la clause `FOR UPDATE` avec un gestionnaire de tables qui gère les verrous de lignes ou de pages, les lignes seront verrouillées.

### 6.4.1.1 Syntaxe de JOIN

MySQL supporte les syntaxes suivantes de `JOIN` pour une utilisation dans les `SELECT` :

```
 reference_table, reference_table
reference_table [CROSS] JOIN reference_table
reference_table INNER JOIN reference_table condition_jointure
reference_table STRAIGHT_JOIN reference_table
reference_table LEFT [OUTER] JOIN reference_table condition_jointure
reference_table LEFT [OUTER] JOIN reference_table
reference_table NATURAL [LEFT [OUTER]] JOIN reference_table
&#123; OJ reference_table LEFT OUTER JOIN reference_table ON expr_conditionnelle &#125;
reference_table RIGHT [OUTER] JOIN reference_table condition_jointure
reference_table RIGHT [OUTER] JOIN reference_table
reference_table NATURAL [RIGHT [OUTER]] JOIN reference_table
```

où `reference_table` est définie de la manière suivante :

```
 nom_de_table [[AS] alias] [USE INDEX (liste_de_clefs)] [IGNORE INDEX (liste_de_clefs)]
```

et `condition_jointure` est définie comme suit :

```
 ON expr_conditionnelle |
USING (column_list)
```

Généralement, vous ne devez avoir aucune condition, dans la partie `ON` , qui soit utilisée pour spécifier les lignes que vous voulez obtenir en résultat. (il y a des exceptions à cette règle). Si vous voulez restreindre les lignes résultantes, vous devez le faire dans la clause `WHERE` .Notez que dans les versions antérieures à la 3.23.17, `INNER JOIN` ne prenait pas en compte `condition_jointure` !

La dernière syntaxe de `LEFT OUTER JOIN` vue plus haut, n'existe que pour assurer la compatibilité avec ODBC :

- On peut créer un alias sur une référence de table en utilisant `nom_de_table AS alias_name` ou `nom_de_table alias_name` .

  ```
  mysql> SELECT t1.name, t2.salary FROM employee AS t1, info AS t2
     ->        WHERE t1.name = t2.name;
  ```

- La condition `ON` est de la même forme qu'une condition pouvant être utilisée dans la clause `WHERE` .
- Si aucune ligne ne correspond dans la table de droite dans la partie `ON` ou `USING` du `LEFT JOIN` , une ligne avec toutes les colonnes mises à `NULL` est utilisé en remplacement. Vous pouvez utiliser ce fait pour trouver les enregistrements dans une table qui n'ont pas de correspondances dans une autre :

  ```
  mysql> SELECT table1.* FROM table1
     ->        LEFT JOIN table2 ON table1.id=table2.id
     ->        WHERE table2.id IS NULL;
  ```

Cet exemple retourne toutes les lignes trouvées dans `table1` avec une valeur de `id` qui n'est pas présente dans `table2` (autrement dit, toutes les lignes de `table1` sans correspondances dans la table `table2` ). Cela demande que `table2.id` soit déclaré `NOT NULL` , bien sur. Comment MySQL optimise `LEFT JOIN` .

- La clause `USING (column_list)` recense la liste des colonnes qui doivent exister dans les deux tables. Une clause `USING` comme :

```
A LEFT JOIN B USING (C1,C2,C3,...)
```

  est définie pour être sémantiquement identique à une expression `ON` comme :

```
A.C1=B.C1 AND A.C2=B.C2 AND A.C3=B.C3,...
```

- La jointure de deux tables avec `NATURAL [LEFT] JOIN` est défini pour être sémantiquement équivalent à un `INNER JOIN` ou un `LEFT JOIN` avec une clause `USING` qui nomme toutes les colonnes qui existent dans les deux tables.
- `INNER JOIN` et `,` (virgule) sont sémentiquement équivalents. Les deux opèrent une jointure totale sur les tables utilisées. Normalement, vous spécifiez les conditions de jointure dans la clause WHERE.
- `RIGHT JOIN` fonctionne de façon analogue à `LEFT JOIN` . Pour garder un code facilement portable, il est recommandé d'utiliser les `LEFT JOIN` à la place des `RIGHT JOIN` .
- `STRAIGHT_JOIN` est identique à `JOIN` , sauf que la table de gauche est toujours lues avant celle de droite. Cela peut être utilisé dans les cas (rares) où l'optimiseur des jointures place les tables dans le mauvais ordre.
- A partir de la version 3.23.12 de MySQL, vous pouvez donner des indications à propos de l'index à utiliser lors de la lecture d'informations d'une table. C'est utile si `EXPLAIN` montre que MySQL utilise un mauvais index de la liste de ceux disponibles. En spécifiant `USE INDEX (liste_de_clefs)` , vous pouvez forcer MySQL à utiliser un index spécifique pour trouver les enregistrements dans la table. Une alternative réside dans l'utilisation de `IGNORE INDEX (liste_de_clefs)` pour dire à MySQL de ne pas utiliser certains index. `USE/IGNORE KEY` sont des synonymes de `USE/IGNORE INDEX` .

Quelques exemples :

```
 mysql> SELECT * FROM table1,table2 WHERE table1.id=table2.id;
mysql> SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id;
mysql> SELECT * FROM table1 LEFT JOIN table2 USING (id);
mysql> SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id
    ->          LEFT JOIN table3 ON table2.id=table3.id;
mysql> SELECT * FROM table1 USE INDEX (key1,key2)
    ->          WHERE key1=1 AND key2=2 AND key3=3;
mysql> SELECT * FROM table1 IGNORE INDEX (key3)
    ->          WHERE key1=1 AND key2=2 AND key3=3;
```

Comment MySQL optimise `LEFT JOIN` .

### 6.4.1.2 Syntaxe de UNION

```
 SELECT ...
UNION [ALL]
SELECT ...
  [UNION
   SELECT ...]
```

`UNION` est implémentée en MySQL 4.0.0. `UNION` est utilisé pour combiner le résultat de plusieurs requêtes `SELECT` en un seul résultat.

Les colonnes listées dans la partie select_expression du `SELECT` doivent être du même type. Les noms de colonnes utilisés dans le premier `SELECT` seront utilisé comme nom de champs pour les résultats retournés.

Les commandes `SELECT` sont des select normaux, mais avec les restrictions suivantes :

- Seule la dernière commande `SELECT` peut avoir une clause `INTO OUTFILE` .

Si vous n'utilisez pas le mot clef `ALL` pour l' `UNION` , toutes les lignes retournées seront uniques, comme si vous aviez fait un `DISTINCT` pour l'ensemble du résultat. Si vous spécifiez `ALL` , vous aurez alors tout les résultats retournés par toutes les commandes `SELECT` .Si vous voulez utiliser un `ORDER BY` pour le résultat final de `UNION` , vous devez utiliser des parenthèses :

```
 (SELECT a FROM nom_de_table WHERE a=10 AND B=1 ORDER BY a LIMIT 10)
UNION
(SELECT a FROM nom_de_table WHERE a=11 AND B=2 ORDER BY a LIMIT 10)
ORDER BY a;
```

## 6.4.2 Syntaxe de HANDLER

```
 HANDLER nom_de_table OPEN [ AS alias ]
HANDLER nom_de_table READ nom_index &#123; = | >= | <= | < &#125; (value1,value2,...)
    [ WHERE ... ] [LIMIT ... ]
HANDLER nom_de_table READ nom_index &#123; FIRST | NEXT | PREV | LAST &#125;
    [ WHERE ... ] [LIMIT ... ]
HANDLER nom_de_table READ &#123; FIRST | NEXT &#125;
    [ WHERE ... ] [LIMIT ... ]
HANDLER nom_de_table CLOSE
```

La commande `HANDLER` fournit un accès direct à l'interface de gestion de la table `MyISAM` .

La première forme de `HANDLER` ouvre la table, la rendant accessible via la requête `HANDLER ... READ` qui la suit. Cette objet table n'est pas partagé par les autres threads et ne sera refermé que si le thread appelle `HANDLER nom_de_table CLOSE` ou que celui ci se termine.

La seconde forme récupère une ligne (ou plus, à spécifier dans la clause `LIMIT` ) où l'index spécifié remplit les conditions et où la clause `WHERE` est répondue. Si l'index se compose de plusieurs parties, (s'étend sur plusieurs colonnes) les valeurs sont spécifiées dans une liste séparée par des virgules, fournir des valeurs pour quelques premières colonnes est possible.

La troisième forme récupère une ligne (ou plus, à spécifier dans la clause `LIMIT` ) de la table dans l'ordre de l'index, qui répond à la clause `WHERE` .

La quatrième forme (sans spécifications relatives à l'index) récupère une ligne (ou plus, à spécifier dans la clause `LIMIT` ) de la table dans un ordre naturel des lignes (comme stocké dans le fichier de données) qui correspond à la condition `WHERE` . C'est plus rapide que `HANDLER nom_de_table READ nom_index` quand une lecture entière de la table est requise.

`HANDLER ... CLOSE` ferme une table qui a été ouverte avec `HANDLER ... OPEN` .

`HANDLER` est en quelque sorte une commande bas−niveau. Par exemple, elle ne propose pas de consistance. En clair, `HANDLER ... OPEN` ne se base **PAS** sur une image de la table, et ne verrouille **PAS** la table. Cela signifie qu'après l'exécution d'une requête `HANDLER ... OPEN` , les données de la table peuvent être modifiées (par ce ou un autre thread) et ces modifications peuvent apparaître partiellement dans les lectures de `HANDLER ... NEXT` ou `HANDLER ... PREV` .

Les raisons d'utiliser cette interface plutôt que les commandes MySQL usuelles sont :

- Plus rapide qu'un `SELECT` car :
  - Un pointeur sur table dédié est alloué au thread dans `HANDLER open` .
  - Il y a moins de traitements.
  - Pas de pertes de temps en optimisation ou vérifications de requêtes.
  - La table utilisée n'a pas besoin d'être verrouillée entre deux requêtes de gestion.
  - L'interface de gestion n'a pas à fournir une vue consistante des données (par exemple, les lectures corrompues sont autorisées), ce qui permet au gestionnaire d'effectuer des optimisations que SQL ne permet pas.
- Cela facilite le port des applications qui utilisent l'interface ISAM pour MySQL.

- Cela permet de traverse plus facilement la base de données qu'avec SQL (dans certains cas, cette opération est impossible avec SQL). L'interface de gestion amène une façon plus naturelle de manipuler les données lorsque vous travaillez avec des applications qui proposent une interface interactive entre l'utilisateur et la base de données.

## 6.4.3 Syntaxe de INSERT

```
     INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] nom_de_table [(nom_colonne,...)]
    VALUES ((expression | DEFAULT),...),(...),...
    ou  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] nom_de_table [(nom_colonne,...)]
    SELECT ...
    ou  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] nom_de_table
:    SET nom_colonne=(expression | DEFAULT), ...
```

`INSERT` insère une nouvelle ligne dans une table existante. La syntaxe `INSERT ... VALUES` insère une ligne à partir de valeurs explicitement fournies. La syntaxe `INSERT ... SELECT` insère des valeurs à partir d'une autre table. La syntaxe `INSERT ... VALUES` avec plusieurs valeurs est supportées à partir de MySQL Version 3.22.5 ou supérieure. la syntaxe `nom_colonne=expression` est supportée à partir de la version 3.22.10 de MySQL.

`nom_de_table` est le nom de la table dans laquelle les valeurs seront insérées. La liste de noms de colonne ou la clause `SET` indiquent les colonnes qui seront affectées:

- Si vous ne spécifiez pas de liste de colonnes avec `INSERT ... VALUES` ou `INSERT ... SELECT` , les valeurs pour toutes les colonnes doivent être fournies dans la clause `VALUES()` ou par la commande `SELECT` . Si vous ne connaissez pas l'ordre des colonnes, utilisez la commande `DESCRIBE nom_de_table` pour le connaître.
- A chaque fois qu'on ne donne pas explicitement une valeur pour une colonne, celle prend la valeur par défaut. Par exemple, si on définit une liste de colonnes qui ne compte pas toutes les colonnes de la tables, toutes les colonnes qui ne sont pas nommées prendront leur valeur par défaut. La définition de la valeur par défaut se fait avec Syntaxe de `CREATE TABLE` .Il est également possible d'utiliser le mot `DEFAULT` pour qu'une colonne prenne sa valeur par défaut. (Nouveauté de MySQL 4.0.3) Il est ainsi plus simple d'écrire des requêtes `INSERT` qui donnent des valeurs à toutes les colonnes sauf quelques unes, car cela permet de ne pas avoir à écrire la liste incomplète de `VALUES()` (une liste qui ne comprend pas une valeur pour chaque colonne de la table). Sans cela, il faudrait écrire la liste des noms de colonnes correspondant a chaque valeur de la liste `VALUES()` .

  MySQL a toujours une valeur par défaut pour chaque champs. C'est obligatoire pour MySQL pour pouvoir fonctionner aussi bien avec des tables supportant les transactions qu'avec des tables ne les supportant pas.

  Nous pensons que le contrôle du contenu des champs devrait être fait pas l'application et non par le serveur de base de données.
- Une `expression` peut faire référence à n'importe quelle colonne qui a été définie précédemment dans une liste de valeurs. Par exemple, on peut dire ceci:

  ```
  mysql> INSERT INTO nom_de_table (col1,col2) VALUES(15,col1*2);
  ```

  Mais pas cela:

  ```
  mysql> INSERT INTO nom_de_table (col1,col2) VALUES(col2*2,15);
  ```
- Si on spécifie le mot `LOW_PRIORITY` , l'exécution de `INSERT` sera retardé jusqu'à ce qu'il n'y ait plus de clients qui lisent la table. Dans ce cas le client doit attendre jusqu'à la fin de l'opération d'insertion, ce qui peut prendre beaucoup de temps si la table est fréquemment accédée. C'est la grande différence avec `INSERT DELAYED` , qui laisse le client continuer tout de suite. Syntaxe des `INSERT DELAYED` . On peut remarquer que `LOW_PRIORITY` ne devrait en principe pas être

utiliser avec des tables de type `MyISAM` , étant donné que celle si n'autorisent pas les inserts concurrents. Tables `MyISAM` .
- Si on spécifie le mot `IGNORE` dans un `INSERT` avec les valeurs de plusieurs lignes, chaque ligne qui qui ferait doublon avec une clé `PRIMARY` ou `UNIQUE` existante dans la table sera ignoré et ne sera pas insérée. Si on ne spécifie pas `IGNORE` , l'insert est abandonné si quelque ligne que ce soit fait doublon avec une clé existante. La fonction `mysql_info()` de l'API C permet de savoir combien de lignes ont été insérées dans la table.
- Si MySQL a été configuré avec l'option `DONT_USE_DEFAULT_FIELDS` , les opérations `INSERT` génère une erreur tant que les valeurs de toutes les colonnes qui attendent une valeur autre que `NULL` ne sont pas spécifiées explicitement. Les options habituelles de `configure` .
- Il est possible de récupérer la valeur utilisée pour une colonne `AUTO_INCREMENT` avec la fonction `mysql_insert_id` . `mysql_insert_id()` .

Si on fait une opération `INSERT ... SELECT` ou `INSERT ... VALUES` avec plusieurs listes de valeurs, la fonction `mysql_info()` de l'API C permet d'obtenir des informations relatives à la requête. Le format de la chaîne d'information est la suivante:

```
Records: 100 Duplicates: 0 Warnings: 0
```

`Duplicates` indique le nombre de lignes qui n'ont pas peut être insérées pour cause de conflit avec une clé unique existante. `Warnings` indique le nombre de tentatives d'inserts de valeurs dans une colonne qui ont généré des problèmes. Les `Warnings` peuvent apparaître dans les conditions suivantes:

- Insertion de `NULL` dans une colonne déclarée `NOT NULL` . Cette colonne a été enregistrée avec sa valeur par défaut.
- Enregistrement dans une colonne numérique d'une valeur qui déborde de la taille de la colonnes. Cette valeur a été tronquée à l'extrémité la plus adaptée de la colonne.
- Attribution à une colonne numérique d'une valeur telle que `'10.34 a'` . Celle valeur refusée est séparée, et la partie numérique résultante est insérée. Si cette valeur n'a pas une valeur numérique sensée, la valeur `0` est insérée.
- L'insertion d'une chaîne dans une colonne `CHAR` , `VARCHAR` , `TEXT` , ou `BLOB` qui dépasse la taille maximale de la colonne. La valeur est tronquée à la taille maximale de la colonne.
- L'insertion d'une valeur illégale pour une colonne de type date ou time. La colonne est alors enregistrée avec la valeur de zero appropriée pour le type.

### 6.4.3.1 Syntaxe de INSERT ... SELECT

```
INSERT [LOW_PRIORITY] [IGNORE] [INTO] nom_de_la_table [(liste des colonnes)] SELECT ...
```

La requête `INSERT ... SELECT` permet de rapidement insérer dans une table un grand nombre de lignes d'une ou plusieurs autres tables.

```
INSERT INTO tblTemp2 (fldID) SELECT tblTemp1.fldOrder_ID FROM tblTemp1 WHERE
tblTemp1.fldOrder_ID > 100;
```

Les conditions suivantes s'appliquent à la requête `INSERT ... SELECT` :

- La table de destination de la requête `INSERT` ne peut apparaître dans la clause `FROM` de la partie `SELECT` de la requête car il est interdit par le ANSI SQL de lire la table dans laquelle on est en train de faire un insert. (le problème est que le `SELECT` pourrait trouver des enregistrements qui aurait été insérés auparavant dans la même exécution. L'utilisation de "subselect" peut rendre la situation confuse !)
- Les colonnes `AUTO_INCREMENT` fonctionnent comme d'habitude.
- Il est possible d'utiliser la fonction `mysql_info()` de l'API C pour obtenir des informations sur la requête. Syntaxe des `INSERT` .
- Pour s'assurer que les journaux des modifications ou les journaux binaires puissent être

utilisés pour re−créer les tables originales, MySQL n'autorise pas les inserts concurrents pendant `INSERT ... SELECT` .

Il est bien sûr possible d'utiliser `REPLACE` à la place de `INSERT` pour remplacer les anciennes lignes.

## 6.4.4 Syntaxe de INSERT DELAYED

```
INSERT DELAYED ...
```

L'option `DELAYED` de la commande `INSERT` est une option spécifique à MySQL très utile si vos clients ne peuvent pas attendre que `INSERT` se termine. C'est un problème fréquent quand on utilise MySQL pour des logs, mais aussi quand on utilise souvent des commandes `SELECT` ou `UPDATE` qui prennent beaucoup de temps. `DELAYED` a été ajouté à MySQL dans la version 3.22.15. C'est une extension de MySQL au ANSI SQL 92.

`INSERT DELAYED` fonctionne avec les tables `ISAM` et `MyISAM` . On peut noter que, les tables `MyISAM` supportant les `SELECT` et les `INSERT` concurrents, s'il n'y a pas de blocs libres au milieu du fichier de données il est vraiment conseillé d'utiliser `INSERT DELAYED` avec les tables `MyISAM` . Tables `MyISAM` .

En utilisant `INSERT DELAYED` , le client reçoit immédiatement un aquitement, et la ligne sera insérée quand la table ne sera plus utilisée par un autre thread.

Un autre avantage de `INSERT DELAYED` est que les insertions des clients sont regroupés, et écrits d'un seul bloc. C'est beaucoup plus rapide que de faire des insertions séparés.

Actuellement, les lignes en attente sont uniquement stockées en mémoire tant qu'elle ne sont pas insérées dans la table. Cela signifie que si on tue `mysqld` violemment, ( `kill −9` ) ou si `mysqld` meurt accidentellement, toutes les lignes en attente qui n'auront pas été écrites sur le disque seront perdues !

Les paragraphes suivants décrivent en détail ce qu'il se passe quand on utilise l'option `DELAYED` dans une requête `INSERT` ou `REPLACE` . Dans cette description, ``thread'' est un thread qui reçoit une commande `INSERT DELAYED` ans ``handler'' est un thread qui gère toutes les opérations de `INSERT DELAYED` pour une table donnée.

- Quand un thread exécute une opération `DELAYED` sur une table, un thread de gestion est créé pour exécuter toutes les opérations `DELAYED` pour cette table − si ce thread de gestion n'existe pas.
- Le thread vérifie que a déjà reçu un verrou `DELAYED` ; sinon, il dit au thread de gestion de le faire. le verrou `DELAYED` peut être obtenu même si d'autres threads ont des verrous `READ` ou `WRITE` sur la table. Cependant le gestionnaire attendra que tous les verrous `ALTER TABLE` ou `FLUSH TABLES` soient finis pour s'assurer que la structure de la table est à jour.
- Le thread exécute une opération `INSERT` , mais plutôt que d'écrire la ligne dans la table, il va placer une copie de la ligne finale dans une file d'attente gérée par le thread de gestion. Le programme client est avertit de toutes les erreurs de syntaxe.
- Le client ne peut pas faire de rapport sur le nombre de duplicata ou sur la valeur de `AUTO_INCREMENT` de la ligne enregistrée; il ne peut pas les obtenir du serveur, car le `INSERT` est validé avant que l'opération d'insert n'ait été effectuée. Si vous utilisez l' API C, la fonction `mysql_info()` ne retourne pas de valeur intéressante, pour la même raison.
- Le journal de modification est mis à jour par le thread de gestion au moment où la ligne est insérée dans la table. Si plusieurs lignes sont insérées en même temps, le journal des modifications est mis à jour quand la première ligne est insérée.
- Une fois que toutes les lignes `delayed_insert_limit` sont écrites, le gestionnaire vérifie si des

requêtes SELECT sont en attente, et si c'est le cas, il leur permet de s'exécuter avant de continuer.

- Quand le thread de gestion n'a plus de ligne dans sa file, la table est déverrouillée. Si aucun INSERT DELAYED n'est reçu avant delayed_insert_timeout secondes, le gestionnaire s'arrête.
- Si plus de delayed_queue_size lignes sont déjà en attente d'un gestionnaire de file donné, le thread qui demande le INSERT DELAYED doit attendre qu'il y ait une place dans la file. Cela permet d'être sûr que mysqld n'utilisera pas toute la mémoire pour la mémoire des files d'attente d'insertions retardés.
- Le thread de gestion apparaîtra dans la liste des processus de MySQL avec delayed_insert dans la colonne Command . Il sera tué si on exécute une commande FLUSH TABLES ou si on le tue avec KILL thread_id . Cependant, il commencera par stocker toutes les lignes en attente dans la table avant de sortir. Pendant ce temps, il n'acceptera aucune commande INSERT d'aucun autre thread. Si on exécute une commande INSERT DELAYED après cela, un nouveau thread de gestion sera créé.

  Il faut noter que les commandes INSERT DELAYED ont une plus grande priorité que les commandes INSERT normales si un gestionnaire de INSERT DELAYED existe déjà! les autres commandes de modification devront attendre que la file d'attente de INSERT DELAYED soit vide, que quelqu'un tue le gestionnaire (avec KILL thread_id ), ou que quelqu'un exécute FLUSH TABLES ..
- Les variables suivantes fournissent des informations relatives à la commande INSERT DELAYED :

| Variable | Signification |
|---|---|
| Delayed_insert_threads | Nombre de threads de gestion |
| Delayed_writes | Nombre de lignes écrites avec INSERT DELAYED |
| Not_flushed_delayed_rows | Nombre de lignes en attente d'être écrites. |

  On peut voir ces variables avec la commande SHOW STATUS ou en exécutant la commande mysqladmin extended-status .

Il faut noter que INSERT DELAYED est plus lent qu'un INSERT normal si la table n'est pas utilisée. L'utilisation d'un thread de gestion séparé pour chaque table sur lesquelles on utilise INSERT DELAYED rajoute également une surcharge au serveur. Ce qui signifie qu'il vaut mieux utiliser INSERT DELAYED uniquement quand c'est vraiment nécessaire!

## 6.4.5 Syntaxe de UPDATE

```
UPDATE [LOW_PRIORITY] [IGNORE] nom_de_table
   SET nom_colonne1=expr1 [, nom_colonne2=expr2, ...]
   [WHERE where_definition]
   [ORDER BY ...]
   [LIMIT #]
```

UPDATE met à jour des enregistrements dans une tables avec de nouvelles valeurs. La clause SET indique les colonnes à modifier et les valeurs à leur donner. La clause WHERE , si fournie, spécifie les enregistrements à mettre à jour. Sinon, tous les enregistrements sont mis à jour. Si la clause ORDER BY est fournie, les enregistrements seront mis à jour dans l'ordre spécifié.

Si vous spécifiez le mot clef LOW_PRIORITY , l'exécution de l' UPDATE sera repoussé jusqu'à ce que aucun client ne lise plus de la table.

Si vous spécifiez le mot clef IGNORE , la mise à jour ne s'interrompra pas même si on rencontre des problèmes d'unicité de clefs durant l'opération. Les enregistrements posant problèmes ne seront pas mis à jour.

Si vous accédez à une colonne d'une table dans une expression, UPDATE utilisera la valeur courante de la colonne. Par exemple, la requête suivante ajoute une année à l'âge actuel de tout le monde :

```
mysql> UPDATE persondata SET age=age+1;
```

Les requêtes UPDATE sont évaluées de gauche à droite. Par exemple, la requête suivante double la valeur de la colonnes âge, puis l'incrémente :

```
mysql> UPDATE persondata SET age=age*2, age=age+1;
```

Si vous changez la valeur d'une colonne en lui spécifiant sa valeur actuelle, MySQL s'en aperçoit et ne fait pas la mise à jour. UPDATE retourne le nombre d'enregistrements ayant changé. Depuis la version 3.22 de MySQL, la fonction mysql_info() de l'API C retourne le nombre de colonnes qui correspondaient, le nombre de colonnes mises à jour et le nombre d'erreurs générées pendant l' UPDATE .

Dans la version 3.23 de MySQL, vous pouvez utilisez le code LIMIT # pour vous assurer que seul un nombre d'enregistrements bien précis est changé.

A partir de la version 4.0.4 de MySQL, vous pouvez aussi effectuer un UPDATE qui se base sur plusieurs tables :

```
 UPDATE items,month SET items.price=month.price
WHERE items.id=month.id;
```

## 6.4.6 Syntaxe de DELETE

```
DELETE [LOW_PRIORITY] [QUICK] FROM nom_de_table
      [WHERE clause_where]
      [ORDER BY ...]
      [LIMIT lignes]

ou

DELETE [LOW_PRIORITY] [QUICK] nom_de_table[.*] [,nom_de_table[.*] ...]
      FROM table-references
      [WHERE clause_where]

ou

DELETE [LOW_PRIORITY] [QUICK]
      FROM nom_de_table[.*], [nom_de_table[.*] ...]
      USING table-references
      [WHERE clause_where]
```

DELETE efface les enregistrements de nom_de_table qui satisfont la condition donnée par clause_where , et retourne le nombre d'enregistrements effacés.Si vous exécutez un DELETE sans clause WHERE , tous les enregistrements sont effacés. Si vous le faites en mode AUTOCOMMIT cela aura le même effet qu'un TRUNCATE . Syntaxe des TRUNCATE . Avec MySQL 3.23, DELETE sans clause WHERE retournera zéro comme nombre d'enregistrements affectés.

Si vous voulez vraiment savoir combien d'enregistrements ont été effacés quand vous videz une table, et que vous êtes prêts à souffrir d'un léger ralentissement, vous pouvez utiliser une requête DELETE de ce genre :

```
mysql> DELETE FROM nom_de_table WHERE 1>0;
```

Notez que c'est plus lent que DELETE FROM nom_de_table sans clause WHERE , parce que cela efface un enregistrement à la fois.

Si vous spécifiez le mot clef LOW_PRIORITY , l'exécution du DELETE sera repoussée jusqu'à ce qu'aucun client ne lise plus de la table.

Si vous spécifiez le mot QUICK , le gestionnaire de la table ne fusionnera pas les blancs dans les index durant la suppréssion, ce qui peut accélérer certain types de suppressions.

Dans les tables de type MyISAM , les enregistrements effacés sont maintenus dans une liste liée et les requêtes INSERT suivantes réutilisent les vieux emplacements. Pour recouvrir l'espace inutilisé ou

réduire la taille des fichiers, utilisez la commande `OPTIMIZE TABLE` ou l'utilitaire `myisamchk` pour réorganiser les tables. `OPTIMIZE TABLE` est plus simple, mais `myisamchk` est plus rapide. Voyez Syntaxe de `OPTIMIZE TABLE` et Optimisation de table .

Le premier format de suppression multi−tables est supporté à partir de MySQL 4.0.0. Le second format de suppression multi−tables est supporté à partir de MySQL 4.0.2.

L'idée est que seul les lignes concordante dans les tables énumérées **avant** le `FROM` ou avant la clause `USING` sont effacés. Le but est de pouvoir effacer des lignes de plusieurs tables en même temps tout en ayant d'autres tables pour les recherches.

Le code `.*` après les noms de tables n'est présent que pour assurer la compatibilité avec `Access` :

```
DELETE t1,t2 FROM t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id

ou

DELETE FROM t1,t2 USING t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id
```

Dans les cas précédents, nous n'avons supprimé les lignes correspondantes que dans les tables `t1` et `t2` . `ORDER BY` et l'utilisation de plusieurs tables dans une requête `DELETE` est supporté en MySQL 4.0.

Si une clause `ORDER BY` est utilisée, les enregistrements seront effacés dans cet ordre. Ceci n'est vraiment intéressant qu'en conjonction avec `LIMIT` . Par exemple:

```
DELETE FROM unlog
WHERE user = 'jcole'
ORDER BY timestamp
LIMIT 1
```

Cela effacera la plus vielle entrée (en se basant sur `timestamp` ) où les enregistrements satisfont la clause `WHERE` .

L'option `LIMIT lignes` spécifique à MySQL pour `DELETE` donne au serveur le nombre maximal de lignes à être effacé avant que le contrôle ne revienne au client. Cela peut être utilisé pour s'assurer qu'une commande `DELETE` spécifique ne prenne pas trop de temps. Vous pouvez répéter la commande jusqu'à ce que le nombre de lignes affectées soit inférieur à la valeur passée à l'option `LIMIT` .

## 6.4.7 Syntaxe de TRUNCATE

```
TRUNCATE TABLE nom_de_table
```

Dans la version 3.23, `TRUNCATE TABLE` est équivalent à `COMMIT ; DELETE FROM nom_de_table` . Syntaxe des `DELETE` .

`TRUNCATE TABLE` diffère de `DELETE FROM ...` des façons suivantes :

- Implémentée comme une destruction/création de table, ce qui accélère la suppression des enregistrements.
- Ne respecte pas les transactions. Vous aurez des erreurs si vous avez une transaction active ou une table protégée en écriture.
- Ne retourne pas le nombre de lignes effacées.
- Tant que le fichier de définition `nom_de_table.frm` est valide, la table peut être recréée, me si les données ou un index a été corrompu.

`TRUNCATE` est une extension Oracle SQL.

## 6.4.8 Syntaxe de REPLACE

```
     REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] nom_de_table [(nom_de_colonne,...)]
        VALUES (expression,...),(...),...
ou  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] nom_de_table [(nom_de_colonne,...)]
        SELECT ...
ou  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] nom_de_table
        SET col_name=expression, nom_de_colonne=expression,...
```

`REPLACE` fonctionne exactement comme `INSERT` , sauf que si une vieille ligne dans la table à la même valeur qu'une nouvelle pour un index `UNIQUE` ou une `PRIMARY KEY` , la vielle ligne sera effacée avant que la nouvelle ne soit insérée. Syntaxe des `INSERT` .En d'autres termes, vous ne pouvez pas accéder aux valeurs de l'ancienne ligne à partir d'une requête `REPLACE` . Dans quelques vieilles versions de MySQL, il apparaît que c'était possible, mais c'etait un dysfonctionnement qui a été corrigé depuis.

Pour utiliser `REPLACE` vous devez avoir les privilèges `INSERT` et `DELETE` sur la table.

Quand vous utilisez une commande `REPLACE` , `mysql_affected_rows()` retournera 2 si une nouvelle ligne en remplace une existante, et cela parce qu'il y aura eu une insertion puis une suppression.

Cela aide à savoir si `REPLACE` a ajouté ou a remplacé une ligne : Testez si le nombre de lignes affectées est égal à 1 (ajout) ou s'il est égal à 2 (remplacement).

Notez que si vous n'utilisez pas un index `UNIQUE` ou une `PRIMARY KEY` , utiliser un `REPLACE` n'a pas de sens vu que cela revient à utiliser un `INSERT` .

## 6.4.9 Syntaxe de LOAD DATA INFILE

```
 LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'nom_de_fichier.txt'
    [REPLACE | IGNORE]
    INTO TABLE nom_de_table
    [FIELDS
        [TERMINATED BY '\t']
        [[OPTIONALLY] ENCLOSED BY '']
        [ESCAPED BY '\\' ]
    ]
    [LINES TERMINATED BY '\n']
    [IGNORE nombre LINES]
    [(nom_de_colonne,...)]
```

La commande `LOAD DATA INFILE` lit les lignes dans un fichier texte et les insère à très grande vitesse. Si le mot clef `LOCAL` est spécifié, le fichier sera lu sur la machine du client. Si le mot clef `LOCAL` n'est pas spécifié, le fichier doit se trouver sur le serveur ( `LOCAL` est disponible à partir de la version 3.22.6 de MySQL).

Pour des raisons de sécurité, lorsque les fichiers sont lus sur le serveur, ils doivent se trouver dans le répertoire de la base de données courante, ou bien être lisible par tous. Pour utiliser la commande `LOAD DATA INFILE` sur des fichiers du serveur, vous devez avoir le droit de `FILE` sur le serveur. Droits fournis par MySQL .

Dans les versions 3.23.49 et 4.0.2 de MySQL, `LOCAL` ne fonctionnera que si vous n'avez pas démarré `mysqld` avec l'option `--local-infile=0` ou si vous n'avez pas activé le support de `LOCAL` pour votre client. LOAD DATA LOCAL .

Si vous spécifiez le mot clef `LOW_PRIORITY` , l'exécution de la commande `LOAD DATA` est ajournée jusqu'à ce qu'aucun client ne lise plus de la table.

Si vous spécifiez le mot clef `CONCURRENT` avec un table au format `MyISAM` , les autres threads pourront accéder à la table durant l'exécution de la commande `LOAD DATA` . L'utilisation de cette option ralentira un peu les performances de `LOAD DATA` même si aucun thread n'utilise la table en même si aucun autre thread n'accède à la table en même temps.

L'utilisation de `LOCAL` sera un peu plus lente que laisser le serveur accéder directement au fichiers, car le contenu du fichier devra transiter du client jusqu'au serveur. D'un autre côté, vous n'aurez pas besoin du droit `FILE` pour charger les fichiers locaux.

Si vous utilisez une version de MySQL antérieure à la 3.23.24 vous ne pouvez lire à partir d'un FIFO avec `LOAD DATA INFILE` . Si vous avez besoin de lire à partir d'un FIFO (par exemple la sortie de gunzip), utilisez `LOAD DATA LOCAL INFILE` .

Vous pouvez aussi charger des fichiers de données en utilisant l'utilitaire `mysqlimport` ; Il opère en envoyant la commande `LOAD DATA INFILE` au serveur. L'option `--local` fais que `mysqlimport` lit les fichiers de données chez le client. Vous pouvez spécifier l'option `--compress` pour avoir de meilleurs performances avec les connexions lentes si le client et le serveur supportent le protocole compressé.

Lorsque les fichiers de données sont sur le serveur, celui çi utilise les règles suivantes :

- Si un chemin absolu est fourni, le serveur utilise le chemin tel quel.
- Si un chemin relatif est fourni, avec un ou plusieurs éléments de dossiers, le serveur recherche le fichier relativement à son dossier de données.
- Si le fichier n'a pas d'éléments de dossier, le serveur recherche les données dans le dossier de base de données courante.

Notez que ces règles font qu'un fichier tel que `./myfile.txt` est lu dans le dossier de données du serveur, alors que s'il est nommé `myfile.txt` , il sera lu dans le dossier de base de données courante. Par exemple, la commande `LOAD DATA` suivante lit le fichier `donnees.txt` dans le dossier de la base `db1` car `db1` est la base de données courante, même si la commande charge explicitement le fichier dans la base de données `db2` :

```
 mysql> USE db1;
mysql> LOAD DATA INFILE "donnees.txt" INTO TABLE db2.ma_table;
```

Les mots réservés `REPLACE` et `IGNORE` contrôlent la méthode d'insertion de lignes lorsque des doublons apparaissent pour les clés uniques. Si vous spécifiez `REPLACE` , les nouvelles lignes remplaceront les anciennes. Si vous spécifiez `IGNORE` , les nouvelles lignes seront ignorées. Si vous ne spécifiez pas cette option, une erreur sera générée à chaque doublon, et le reste du fichier sera ignoré.

Si vous chargez un fichier sur votre machine client avec l'option `LOCAL` , le serveur ne peut pas interrompre la transmission du fichier au milieu de l'opération : par défaut, il utilisera l'option `IGNORE` .

Si vous utilisez `LOAD DATA INFILE` sur une table vide de type `MyISAM` , tous les index non−uniques seront créés dans un processus séparé (tout comme `REPAIR` ). Cela rend `LOAD DATA INFILE` beaucoup plus rapide si vous avez plusieurs index.

`LOAD DATA INFILE` est le complémentaire de `SELECT ... INTO OUTFILE` . Syntaxe des `SELECT` . Pour écrire des données depuis une table dans un fichier, utilisez `SELECT ... INTO OUTFILE` . Pour lire les données dans la table, utilisez `LOAD DATA INFILE` . La syntaxe des clauses `FIELDS` et `LINES` est la même pour les deux commandes. Ces deux clauses sont optionnelles, mais `FIELDS` doit précéder `LINES` , si les deux sont specifiées.

Si vous spécifiez la clause `FIELDS` , les sous−clauses `TERMINATED BY` , `[OPTIONALLY] ENCLOSED BY` , et `ESCAPED BY` sont aussi optionnelles, mais vous devez en spécifier au moins une.

Si vous ne spécifiez par de clause `FIELDS` , les valeurs par défaut sont :

```
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
```

Si vous ne spécifiez par de clause `LINES` , les valeurs par défaut sont :

```
LINES TERMINATED BY '\n'
```

En d'autres termes, les valeurs par défaut font que `LOAD DATA INFILE` lit les données comme suit :

- Recherche des limites de lignes parmis les nouvelles lignes.
- Scinde les lignes en champs avec les tabulations.
- Ne suppose pas que les champs sont entourés de guillemets.
- Interprète les occurrences de tabulation, nouvelle ligne, `'\'` précédées par `'\'` comme des caractères littéraux qui font partie de la valeur d'un champs.

A l'inverse, les valeurs par défaut font que `SELECT ... INTO OUTFILE` écrit les données comme ceci :

- Ecrit des tabulations entre les champs.
- N'entoure pas les champs de guillemets.
- Utilise `'\'` pour echapper les occurences de tabulation, nouvelle ligne, `'\'` trouvées dans les valeurs.
- Insère une nouvelle ligne entre les lignes.

Notez que pour utiliser `FIELDS ESCAPED BY '\\'` , vous devez spécifier deux antislashs pour que cette valeur soit interprétée comme un antislash simple.L'option `IGNORE nombre LINES` sert à ignorer une en−tête de fichier, telle que des noms de colonnes, qui débutent parfois un fichier à charger :

```
mysql> LOAD DATA INFILE "/tmp/nom_fichier" INTO TABLE test IGNORE 1 LINES;
```

Lorsque vous utilisez `SELECT ... INTO OUTFILE` conjointement avec `LOAD DATA INFILE` pour écrire des données dans un fichier et les relire dans une table, les options de `FIELDS` et `LINES` doivent être identiques. Sinon, `LOAD DATA INFILE` ne pourra pas interpréter le contenu du fichier correctement. Supposez que la commande `SELECT ... INTO OUTFILE` ait écrit un fichier délimité par des virgules :

```
mysql> SELECT * INTO OUTFILE 'donnees.txt'
   ->          FIELDS TERMINATED BY ','
   ->          FROM ...;
```

Pour lire ce fichier, la commande correcte serait :

```
mysql> LOAD DATA INFILE 'donnees.txt' INTO TABLE table2
   ->           FIELDS TERMINATED BY ',';
```

Si au contraire, vous essayez de lire le fichier avec la commande ci−dessous, cela ne fonctionnera pas, car la commande `LOAD DATA INFILE` essaie de lire des tabulations entre les champs :

```
mysql> LOAD DATA INFILE 'donnees.txt' INTO TABLE table2
   ->           FIELDS TERMINATED BY '\t';
```

Il est probable que chaque ligne d'entrée sera interprétée que comme un seul champ.

La commande `LOAD DATA INFILE` peut être utilisée pour lire des données issues d'autres sources. Par exemple, un fichier au format dBASE présente des champs séparés par des virgules, et entourés de guillemets doubles. Si les lignes sont terminées par de nouvelles lignes, la commande ci−dessous illustre la relecture d'un tel fichier avec MySQL :

```
mysql> LOAD DATA INFILE 'donnees.txt' INTO TABLE nom_de_table
   ->           FIELDS TERMINATED BY ',' ENCLOSED BY '"'
   ->           LINES TERMINATED BY '\n';
```

Les clauses `FIELDS` et `LINES` peuvent prendre des chaînes vides comme valeur. S'il la chaîne n'est pas vide, `FIELDS [OPTIONALLY] ENCLOSED BY` et `FIELDS ESCAPED BY` ne doivent avoir qu'un seul caractère. Les valeurs de `FIELDS TERMINATED BY` et `LINES TERMINATED BY` peuvent avoir plus d'un caractère. Par exemple, pour écrire des lignes terminées par le couple retour chariot−nouvelle ligne, ou pour lire un tel fichier, spécifiez la clause `LINES TERMINATED BY '\r\n'` .

Par exemple, pour charger un fichier de blagues, qui sont séparées par une ligne de `%%` , dans une table vous pouvez faire :

```
 CREATE TABLE blagues (
   a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    blague TEXT NOT NULL
   );
LOAD DATA INFILE "/tmp/blagues.txt" INTO TABLE blagues FIELDS TERMINATED BY ""
LINES TERMINATED BY "\n%%\n" (blague);
```

`FIELDS [OPTIONALLY] ENCLOSED BY` contrôle la mise entre guillemets des champs. Pour l'écriture de fichier ( `SELECT ... INTO OUTFILE` ), si vous omettez le mot `OPTIONALLY` , tous les champs seront entourés par le caractère spécifié dans la clause `ENCLOSED BY` . Par exemple, si la virgule est utilisée comme séparateur de champs :

```
 "1","une chaîne","100.20"
"2","une chaîne contenant une , virgule","102.20"
"3","une chaîne contenant un \" guillemet","102.20"
"4","une chaîne contenant un \", guillemet et une virgule","102.20"
```

Si vous spécifiez `OPTIONALLY` , le caractère `ENCLOSED BY` n'est utilisé que pour protéger les colonnes de types `CHAR` et `VARCHAR` :

```
 1,"une chaîne",100.20
2,"une chaîne contenant une , virgule",102.20
3,"une chaîne contenant un \" guillemet",102.20
4,"une chaîne contenant un \", guillemet et une virgule",102.20
```

Notez que les occurrences du caractère `ENCLOSED BY` dans un champs sont échappée en les préfixant avec le caractère `ESCAPED BY` . Notez aussi que si vous spécifiez un caractère d'échappement vide, il n'est pas possible de garantir que les champs seront correctement relus par `LOAD DATA INFILE` . Par exemple, l'exemple ci−dessus apparaîtra comme montré ci−dessous. Notez que le second champ de la quatrième ligne comporte une virgule suivant un guillemet qui semble (mais c'est faux) terminer la ligne :

```
 1,"une chaîne",100.20
2,"une chaîne contenant une , virgule",102.20
3,"une chaîne contenant un " guillemet",102.20
4,"une chaîne contenant un ", guillemet et une virgule",102.20
```

Lors des lectures, le caractère `ENCLOSED BY` , s'il est présent, est supprimé des extrémités de la valeur du champ. (ce qui est vrai, qu'il y ait l'option `OPTIONALLY` ou pas). Les occurences du caractère `ENCLOSED BY` , précédées par le caractère `ESCAPED BY` sont interprétées comme faisant partie de la valeur du champ. Les caractères `ENCLOSED BY` doublées, apparaissant dans la chaîne, sont interprétés comme le caractère `ENCLOSED BY` lui−même. Par exemple, si `ENCLOSED BY '"'` est spécifié, les guillemets sont gérés comme ceci :

```
 "Le ""GRAND"" chef"  -> Le "GRAND" chef
Le "GRAND" chef      -> Le "GRAND" chef
Le ""GRAND"" chef    -> Le ""GRAND"" chef
```

`FIELDS ESCAPED BY` contrôle les caractères spéciaux. Si le caractère `FIELDS ESCAPED BY` n'est pas vide, il est utilisé pour préfixer les caractères suivants en écriture :

- La caractère `FIELDS ESCAPED BY`
- Le caractère `FIELDS [OPTIONALLY] ENCLOSED BY`
- Le premier caractère des valeurs de `FIELDS TERMINATED BY` et `LINES TERMINATED BY`
- ASCII `0` (en fait, ce qui est écrit après le caractère d'échappement est le caractère ASCII `'0'` , et non pas le code ASCI de zéro)

Si le caractère `FIELDS ESCAPED BY` est vide, aucun caractère ne sera échappé. Ce n'est probablement pas une bonne idée de spécifier un caractère d'échappement vide, en particulier si les valeurs dans vos champs risquent d'utiliser l'un des caractères de la liste ci−dessus.En lecture, si le caractère `FIELDS ESCAPED BY` n'est pas vide, les occurrences de ce caractère sont supprimées, et le caractère suivant est lu littéralement. Les exceptions à cette règle sont `'0'` ou `'N'` (par exemple, `0` ou `\N` si le caractère d'échappement est `'\'` ). Ces séquences sont interprétées comme l'octet nul (ASCII `0` ) et la valeur `NULL` . Voyez plus bas pour la gestion des valeurs `NULL` .

Pour plus d'informations sur la syntaxe avec les caractères d'échappement `'\'` , consultez Comment écrire les chaînes et les nombres ? .

Dans certains cas, les options de `FIELDS` et `LINES` interfèrent entre elles :

- Si le caractère de `LINES TERMINATED BY` est une chaîne vide et que celui de `FIELDS TERMINATED BY` ne l'est pas, ce dernier sera celui utilisé pour `LINES TERMINATED BY` .
- Si les valeurs `FIELDS TERMINATED BY` et `FIELDS ENCLOSED BY` sont vides toutes les deux ( `''` ), un format à taille de champ fixe est utilisé. Avec ce format, aucun délimiteur n'est utilisé entre les champs. Au lieu de cela, les valeurs des colonnes sont écrites avec leur configuration d'affichage. Par exemple, si une colonne a été déclarée `INT(7)` , la valeur de cette colonne sera écrite avec 7 caractères. Lors de la relecture, la valeur de la colonne sera obtenue en lisant à nouveau 7 caractères. Ce format à taille fixe affecte la gestion de la valeur `NULL` ; voyez plus loin pour cela. Notez que ce format ne fonctionne pas avec les jeux de caractères multi−octets.

La gestion des valeurs `NULL` depend des options `FIELDS` et `LINES` que vous utilisez :

- Pour les valeurs par défaut de `FIELDS` et `LINES` , `NULL` est écrit `\N` et `\N` est lu `NULL` (en supposant que le caractère d'échappement est `'\'` ).
- Si `FIELDS ENCLOSED BY` n'est pas vide, un champ contenant le mot `NULL` comme valeur sera lu comme la valeur `NULL` (ce qui diffère du mot `NULL` , entouré du caractère `FIELDS ENCLOSED BY` , qui sera lu comme le mot `'NULL'` ).
- Si `FIELDS ESCAPED BY` est vide, `NULL` est écrit comme le mot `'NULL'` .
- Avec le format à taille fixe (ce qui arrive si `FIELDS TERMINATED BY` et `FIELDS ENCLOSED BY` sont tous les deux vides), les valeurs `NULL` sont écrites sous forme de chaîne vide. Notez que cela fait que `NULL` et les chaînes vides seront représentées par une valeur qui ne les distingue pas l'une de l'autre. Si vous avez besoin de différencier entre les deux, n'utilisez par ce format !

Certains cas ne sont pas supportés par `LOAD DATA INFILE` :

- Lignes à tailles fixes ( `FIELDS TERMINATED BY` et `FIELDS ENCLOSED BY` sont tous les deux vides) et les types de colonne `BLOB` ou `TEXT` .
- Si vous spécifiez un séparateur qui est le même qu'un autre préfixe, `LOAD DATA INFILE` ne sera pas capable de relire proprement le résultat. Par exemple, la clause `FIELDS` suivante posera sûrement des problèmes :
  ```
  FIELDS TERMINATED BY '"' ENCLOSED BY '"'
  ```
- Si `FIELDS ESCAPED BY` est vide, une valeur de colonne qui contient une occurence de `FIELDS ENCLOSED BY` ou de `LINES TERMINATED BY` suivi du caractère `FIELDS TERMINATED BY` interrompra la lecture de `LOAD DATA INFILE` trop tôt. Cela est dû au fait que `LOAD DATA INFILE` ne peut pas faire la différence entre la valeur dans le champ et la fin de la ligne.

L'exemple suivant charge toutes les colonnes de la table `persondata` :

```
mysql> LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata;
```

Aucun champ n'est spécifié, ce qui fait que `LOAD DATA INFILE` s'attend à ce que les lignes lues contiennent le bon nombre de champs. Les valeurs par défaut de `FIELDS` et `LINES` sont utilisées.

Si vous voulez charger uniquement quelques colonnes dans une table, spécifiez la liste des champs :

```
mysql> LOAD DATA INFILE 'persondata.txt'
   ->           INTO TABLE persondata (col1,col2,...);
```

Vous devez aussi spécifier les champs si l'ordre dans lequel ils seront lus diffère de l'ordre des

colonnes dans la table. Sinon, MySQL ne pourra pas savoir à quelle colonne correspond une valeur.

Si une ligne a trop peu de champs, les colonnes omises prendront leur valeur par défaut. Les affectations de valeurs par défaut sont décrites dans Syntaxe de `CREATE TABLE` .

Une valeur de champs vide et un champ manquant ne seront pas interprétés de la même façon :

- Pour les types chaîne, la colonne est remplie avec la chaîne vide.
- Pour les types numériques, la colonne est mise à `0` .
- Pour les types dates et heures, la colonne est mise au zéro approprié pour le type. Types dates et heures .

Notez que vous obtiendrez le même résultat en assignant à ces différents types de champs la chaîne vide dans une commande `INSERT` ou `UPDATE` .Les colonnes `TIMESTAMP` prendront la date et l'heure courante uniquement si on leur affecte la valeur NULL, ou (pour la première colonne `TIMESTAMP` seulement) si la colonne `TIMESTAMP` est ignorée de la liste des colonnes spécifiée.

Si une ligne d'entrée comporte trop de colonnes, les champs en trop sont ignorés, et le nombre d'alertes est incrémenté.

`LOAD DATA INFILE` considère toutes les valeurs lues comme des chaînes de caractères : vous ne pourrez donc pas utiliser la forme numérique des colonnes `ENUM` ou `SET` , comme d'habitude. Toutes les colonnes `ENUM` et `SET` doivent être spécifiée comme des chaînes ! Si vous utilisez l'API C, vous pouvez obtenir des informations à propos de la requête en utilisant la fonction `mysql_info()` quand `LOAD DATA INFILE` se termine. Le format de la chaîne d'informations est le suivant :

```
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

Les alertes sont générées dans les mêmes circonstances que pour la commande `INSERT` ( Syntaxe des `INSERT` ), excepté que `LOAD DATA INFILE` génère aussi des alertes s'il y a trop peu ou trop de champs dans une ligne. Les alertes ne sont pas stockées; le nombre d'alertes est la seule indication. Si vous recevez des alertes et vous voulez savoir exactement ce qui s'est passé, exécutez une commande `SELECT ... INTO OUTFILE` dans un autre fichier et comparez le avec le fichier original.Si vous voulez faire lire `LOAD DATA` à partir d'un pipe, vous pouvez utiliser l'astuce suivante :

```
 mkfifo /mysql/db/x/x
chmod 666 /mysql/db/x/x
cat < /dev/tcp/10.1.1.12/4711 > /nt/mysql/db/x/x
mysql −e "LOAD DATA INFILE 'x' INTO TABLE x" x
```

Si vous utilisez une version de MySQL supérieure à 3.23.25 vous ne pouvez que faire ce qui précède avec `LOAD DATA LOCAL INFILE` .Pour plus d'informations sur les performances de `INSERT` comparées à `LOAD DATA INFILE` et accélerer `LOAD DATA INFILE` : Vitesse des requêtes `INSERT` .

### 6.4.10 Syntaxe de DO

```
DO expression, [expression, ...]
```

Exécute l'expression mais ne retourne aucun résultat. C'est un alias de `SELECT expression, expression` , mais il a l'avantage d'être plus rapide quand on n'a pas besoin du résultat.Cela s'avère très utile avec les fonctions qui ont des effets secondaires, comme `RELEASE_LOCK` .

## 6.5 Définition de données : CREATE , DROP , ALTER

## 6.5.1 Syntaxe de CREATE DATABASE

```
CREATE DATABASE [IF NOT EXISTS] db_name
```

`CREATE DATABASE` crée une base de données avec le nom donné. La syntaxe des noms des bases de données est donnée ici : Noms de base de données, tables, index, colonnes et alias . Une erreur survient si la base de données existe déjà et que vous n'avez pas spécifié `IF NOT EXISTS` .Les bases de données MySQL sont implémentées comme des répertoires contenant des fichiers qui correspondent aux tables dans les bases de données. Puisqu'il n'y a pas de tables dans une base de données lors de sa création, la requête `CREATE DATABASE` créera seulement le dossier dans le répertoire de données de MySQL.

Vous pouvez aussi créer des bases de données avec `mysqladmin` . Scripts côté client et outils .

## 6.5.2 Syntaxe de DROP DATABASE

```
DROP DATABASE [IF EXISTS] db_name
```

`DROP DATABASE` détruit toutes les tables dans la base de données et l'efface elle même. Si vous utilisez la commande `DROP DATABASE` sur un lien symbolique pointant sur la base de données, le lien et la base seront effacés. **Soyez très prudents avec cette commande!**

`DROP DATABASE` retourne le nombre de fichiers qui ont été effacés du dossier de la base de données. Normalement, c'est égal à trois fois le nombre de tables, car chaque table, correspond normalement à un fichier `.MYD` , un fichier `.MYI` et un fichier `.frm` .

La commande `DROP DATABASE` efface tous les fichiers du dossier de la base de données ayant les extensions suivantes :

| Ext | Ext | Ext | Ext |
|---|---|---|---|
| .BAK | .DAT | .HSH | .ISD |
| .ISM | .ISM | .MRG | .MYD |
| .MYI | .db | .frm | |

Tous les sous−dossiers qui consistent de 2 chiffres (dossiers `RAID` ) sont aussi supprimés.

Depuis la version 3.22 de MySQL, vous pouvez utiliser le mot clef `IF EXISTS` pour éviter l'affichage d'erreurs si la base n'existe pas.

Vous pouvez aussi supprimer des bases de données avec `mysqladmin` . Scripts côté client et outils .

## 6.5.3 Syntaxe de CREATE TABLE

```
 CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nom_de_table [(definition_de_create,...)]
[options_de_table] [select_statement]

 definition_de_create:
  nom_de_colonne type [NOT NULL | NULL] [DEFAULT valeur_par_defaut] [AUTO_INCREMENT]
            [PRIMARY KEY] [definition_de_reference]
  ou    PRIMARY KEY (index_col_name,...)
  ou    KEY [nom_index] (index_col_name,...)
  ou    INDEX [nom_index] (index_col_name,...)
  ou    UNIQUE [INDEX] [nom_index] (index_col_name,...)
  ou    FULLTEXT [INDEX] [nom_index] (index_col_name,...)
  ou    [CONSTRAINT symbol] FOREIGN KEY [nom_index] (index_col_name,...)
            [reference_definition]
  ou    CHECK (expr)

 type:
        TINYINT[(longueur)] [UNSIGNED] [ZEROFILL]
  ou    SMALLINT[(longueur)] [UNSIGNED] [ZEROFILL]
  ou    MEDIUMINT[(longueur)] [UNSIGNED] [ZEROFILL]
```

```
 ou    INT[(longueur)] [UNSIGNED] [ZEROFILL]
 ou    INTEGER[(longueur)] [UNSIGNED] [ZEROFILL]
 ou    BIGINT[(longueur)] [UNSIGNED] [ZEROFILL]
 ou    REAL[(longueur,décimales)] [UNSIGNED] [ZEROFILL]
 ou    DOUBLE[(longueur,décimales)] [UNSIGNED] [ZEROFILL]
 ou    FLOAT[(longueur,décimales)] [UNSIGNED] [ZEROFILL]
 ou    DECIMAL(longueur,décimales) [UNSIGNED] [ZEROFILL]
 ou    NUMERIC(longueur,décimales) [UNSIGNED] [ZEROFILL]
 ou    CHAR(longueur) [BINARY]
 ou    VARCHAR(longueur) [BINARY]
 ou    DATE
 ou    TIME
 ou    TIMESTAMP
 ou    DATETIME
 ou    TINYBLOB
 ou    BLOB
 ou    MEDIUMBLOB
 ou    LONGBLOB
 ou    TINYTEXT
 ou    TEXT
 ou    MEDIUMTEXT
 ou    LONGTEXT
 ou    ENUM(valeur1,valeur2,valeur3,...)
 ou    SET(valeur1,valeur2,valeur3,...)

 index_nom_de_colonne:
        nom_de_colonne [(longueur)]

 definition_de_reference:
        REFERENCES nom_de_table [(index_nom_de_colonne,...)]
                   [MATCH FULL | MATCH PARTIAL]
                   [ON DELETE option_de_reference]
                   [ON UPDATE option_de_reference]

 option_de_reference:
        RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

 options_de_table:
    TYPE = &#123;BDB | HEAP | ISAM | InnoDB | MERGE | MRG_MYISAM | MYISAM &#125;
ou    AUTO_INCREMENT = #
ou    AVG_ROW_LENGTH = #
ou    CHECKSUM = &#123;0 | 1&#125;
ou    COMMENT = "phrase"
ou    MAX_ROWS = #
ou    MIN_ROWS = #
ou    PACK_KEYS = &#123;0 | 1 | DEFAULT&#125;
ou    PASSWORD = "mot"
ou    DELAY_KEY_WRITE = &#123;0 | 1&#125;
ou      ROW_FORMAT= &#123; default | dynamic | fixed | compressed &#125;
ou    RAID_TYPE= &#123;1 | STRIPED | RAID0 &#125; RAID_CHUNKS=#  RAID_CHUNKSIZE=#
ou    UNION = (nom_de_table,[nom_de_table...])
ou    INSERT_METHOD= &#123;NO | FIRST | LAST &#125;
ou      DATA DIRECTORY="chemin absolu vers dossier"
ou      INDEX DIRECTORY="chemin absolu vers dossier"

 select_statement:
    [IGNORE | REPLACE] SELECT ...  (une clause de séléction valide)
```

`CREATE TABLE` Crée une table avec le nom donné, dans la base de données courante. Les règles de nommage des tables sont disponibles dans Noms de base de données, tables, index, colonnes et alias . Une erreur est affichée s'il n'y a pas de base courante, ou si la table existe déjà.Depuis la version 3.22 de MySQL, vous pouvez utiliser le mot réservé `TEMPORARY` lorsque vous créez une table. Une table temporaire sera immédiatement effacée dès que la connexion se termine. Cela signifie que vous pouvez utiliser le même nom de table temporaire depuis deux connexions différentes sans risque de conflit entre les connexions. Vous pouvez aussi utiliser une table temporaire qui a le même nom qu'une table existante (la table existante est alors cachée tant que dure la table temporaire). En MySQL version 4.0.2 ou plus récent, vous avez juste à avoir le privilège `CREATE TEMPORARY TABLES` pour créer des tables temporaires.

Depuis la version 3.23 de MySQL, vous pouvez utiliser le mot réservé `IF NOT EXISTS` , de façon à ce qu'aucune erreur ne soit affiché si la table que vous essayez de créer existe déjà. Notez qu'il n'y a pas de comparaisons entre les structures de table lors du test d'existence.

En MySQL 4.1, vous pouvez utiliser `LIKE` pour créer une table basée sur la définition d'une autre table. En MySQL 4.1, vous pouvez aussi spécifier le nom de la colonne générée :

```
CREATE TABLE foo (a tinyint not null) SELECT b+1 AS 'a' FROM bar;
```

Chaque table nom_de_table est representée par des fichiers dans le dossier de la base de données. Dans le cas des tables de type MyISAM, ce sont les fichiers suivants :

| Fichier | Rôle |
|---|---|
| nom_de_table.frm | Fichier de définition de la table |
| nom_de_table.MYD | Fichier de données |
| nom_de_table.MYI | Fichier d'index |

Pour plus d'informations sur les propriétés des différentes colonnes et de leurs types, voyez Types de colonnes :

- Si ni `NULL` , ni `NOT NULL` n'est spécifié, une colonne utilisera par défaut l'attribut `NULL` (elle acceptera les valeurs `NULL` ).
- Une colonne de nombre entier peut se voir attribuer l'attribut `AUTO_INCREMENT` . Lorsque vous insérez la valeur `NULL` (recommandée) ou `0` dans une colonne `AUTO_INCREMENT` , la colonne prendra automatiquement la valeur de `value+1` , où value est la plus grande valeur positive courante dans cette colonne. La série des valeurs `AUTO_INCREMENT` commence à `1` . `mysql_insert_id()` .Si vous effacez la ligne contenant la valeur maximale dans la colonne `AUTO_INCREMENT` , cette valeur sera réutilisée dans les tables de type ISAM mais pas dans les tables de type MyISAM. Si vous effacez toutes les lignes dans la table avec la commande `DELETE FROM` nom_de_table (sans la clause `WHERE)` en mode `AUTOCOMMIT` , la série des valeurs `AUTO_INCREMENT` recommencera à zéro.

  **NOTE** : Il ne peut y avoir qu'une seule colonne de type `AUTO_INCREMENT` dans une table, et elle doit être indexée. MySQL version 3.23 ne fonctionnera correctement que si cette colonne n'accueille que des valeurs positives. Insérer un nombre négatif sera considéré comme inserer un nombre de très grande taille, mais positif. Ceci est fait pour éviter les problèmes de précision lorsque les nombres passe de positif à négatif lorsqu'ils atteignent leur valeur maximale positive. C'est aussi pour éviter qu'une colonne de type `AUTO_INCREMENT` ne contienne de valeur 0.

  En MyISAM et tables BDB, vous pouvez spécifier une colonne secondaire d' `AUTO_INCREMENT` dans une clef multi−colonne. Utiliser `AUTO_INCREMENT` .

  Pour rendre MySQL avec certaines applications ODBC, vous pouvez retrouver la valeur de la dernière valeur automatiquement générée avec la requête suivante :

  ```
  SELECT * FROM nom_de_table WHERE auto_col IS NULL
  ```

- `CREATE TABLE` effectue automatiquement la transaction courante d' `InnoDB` si le MySQL binlogging est employé.
- La valeur `NULL` est traitée différemment dans les colonnes de type `TIMESTAMP` . Vous ne pouvez pas stocker de valeur `NULL` littérale dans une colonne `TIMESTAMP` ; insérer une valeur `NULL` dans une telle colonne revient à insérer la date et l'heure courante. Car les colonnes `TIMESTAMP` ignorent les attributs NULL et `NOT NULL` .Cela facilite grandement l'utilisation des colonnes `TIMESTAMP` pour les clients MySQL : le serveur indique que ces colonnes peuvent se voir assigner une valeur `NULL` (ce qui est vrai), même si les colonnes `TIMESTAMP` ne contiendront jamais de valeur `NULL` . Vous pouvez le constater lorsque vous utiliser la commande `DESCRIBE` nom_de_table pôur avoir une description de votre table.

  Notez qu'affecter la valeur `0` à une colonne `TIMESTAMP` n'est pas la même chose que lui affecter la valeur `NULL` , car `0` est une valeur `TIMESTAMP` valide.
- Une valeur `DEFAULT` doit être une constante, ça ne peut être une fonction ou une expression.

  Si aucun valeur par défaut (attribut `DEFAULT` ) n'est spécifié, MySQL en assigne une automatiquement

  Si la colonne accepte les valeur `NULL` , la valeur par défaut sera la valeur `NULL` .

Si la colonne est déclarée comme NOT NULL (non−nulle), la valeur par défaut dépendra du type de colonne :

- Pour les types numériques sans l'attribut AUTO_INCREMENT , la valeur sera 0 . Pour une colonne AUTO_INCREMENT , la valeur par défaut sera la prochaine valeur de la série.
- Pour les types dates et heures autres que TIMESTAMP , la valeur par défaut est la date zéro appropriée. Pour les colonnes TIMESTAMP , la valeur par défaut est la date et l'heure courante. Types dates et heures .
- Pour les types de chaînes autres que ENUM , la valeur par défaut est la chaîne vide. Pour ENUM , la valeur par défaut est la première valeur de l'énumération.

Les valeurs par défaut doivent être des constantes. Cela signifie, par exemple, que vous ne pouvez pas donner de valeur par défaut en fonction de NOW() ou CURRENT_DATE .

- KEY est un synonyme de INDEX .
- Avec MySQL, une clé UNIQUE peut avoir uniquement avoir deux valeurs distinctes. Une erreur surviendra si vous essayez d'ajouter une ligne dont la clé corresond à une ligne existante.
- Une clé primaire ( PRIMARY KEY ) est un index UNIQUE avec la contrainte supplémentaire que les toutes les colonnes utilisées doit avoir l'attribut NOT NULL . En MySQL, cette clé est dite PRIMARY . Une table ne peut avoir qu'une seule clé primaire. Si vous n'avez pas de PRIMARY KEY et que des applications demandent la PRIMARY KEY dans vos tables, MySQL retournera la première clé UNIQUE , qui n'a aucune valeur NULL .
- Une PRIMARY KEY peut être multi−colonne. Cependant, vous ne pouvez pas créer d'index multi−colonne avec l'attribut PRIMARY KEY dans une spécification de colonne. En faisant cela, le seul résultat sera que cette seule colonne sera marquée comme clé primaire. Vous devez absolument utiliser la syntaxe PRIMARY KEY (index_nom_de_colonne, ...).
- Si une clé primaire ( PRIMARY ) ou unique ( UNIQUE ) est établit sur une seule colonne, et que cette colonne est de type entier, vous pouvez aussi faire référence à cette colonne sous le nom _rowid (nouveau en version 3.23.11).
- Si vous ne donnez pas de nom à un index, l'index prendra le nom de la première colonne qui le compose, avec éventuellement un suffixe (_2, _3, ...) pour le rendre unique. Vous pouvez voir les noms des index avec la commande SHOW INDEX FROM nom_de_table. Lire des informations sur les bases, colonnes, tables ou index .
- Seul, les formats de table MyISAM , InnoDB , et BDB supportent des index sur des colonnes qui peuvent contenir des valeurs NULL . Dans les autres situations, vous devez déclarer ces colonnes NOT NULL ou une erreur sera générée.
- Avec la syntaxe nom_de_colonne(longueur), vous pouvez spécifier un index qui n'utilise qu'une partie de la colonne CHAR ou VARCHAR . Cela peut réduire la taille des fichiers d'index. Index de colonne .
- Seul le format de table MyISAM supporte l'indexation des colonnes BLOB et TEXT . Colonnes TEXT . Lorsque vous ajoutez un index à une colonne BLOB ou TEXT , vous devez ABSOLUMENT spécifier une longueur d'index :
```
CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));
```
- Lorsque vous utilisez une clause ORDER BY ou GROUP BY sur une colonne de type TEXT ou BLOB , seuls, les max_sort_longueur premiers octets seront lus. Les types BLOB et TEXT .
- En MySQL version 3.23.23 ou plus récent, vous pouvez aussi créer des index spécial FULLTEXT . Ils sont utilisés pour faire des recherches en texte plein. Seul, le format de table MyISAM supporte les index FULLTEXT . Ils peuvent être créés uniquement pour les colonnes de type VARCHAR et TEXT . L'indexation est alors exécutée sur toute la colonne, et les indexations partielles ne sont pas supportées. supported. Voir Recherche full−text avec MySQL pour les détails.
- En MySQL version 3.23.44 et plus récent, les tables InnoDB supportent la vérification de clé étrangères. Tables InnoDB . Notez que la syntaxe des clés étrangères FOREIGN KEY de InnoDB est plus restrictive que la syntaxe présentée ci−dessus. InnoDB ne permet pas la spécification d'un index_name , et les colonnes de la table référencée doivent être explicitement nommées. Depuis la version 4.0.8, InnoDB supporte les clauses ON DELETE et ON UPDATE avec les clés étrangères. Voyez le manuel InnoDB pour la syntaxe précise. Tables InnoDB . Pour les autres types de tables, le

serveur MySQL n'analyse pas les clauses FOREIGN KEY , CHECK et REFERENCES dans les commandes CREATE TABLE , et aucune action n'est réalisée. Clés étrangères .

- Chaque colonne NULL requiert un bit supplémentaire, arrondi à l'octet supérieur le plus proche.
- La taille maximale d'enregistrement peut être calculée comme ceci :

```
row longueur = 1
            + (somme des longueurs de colonnes)
            + (nombre de colonnes NULL + 7)/8
            + (nombre de colonnes à taille variable)
```

- Les options options_de_table et SELECT ne sont implémentées que dans MySQL version 3.23 et plus récent.

Les différents types de tables sont :

| Table type | Description |
|---|---|
| BDB or BerkeleyDB | Tables avec transactions. Tables BDB ou BerkeleyDB . |
| HEAP | Les données de ces tables ne sont stockées qu'en mémoire. Tables HEAP . |
| ISAM | Le gestionnaire originel de tables. Tables ISAM . |
| InnoDB | Transaction−safe tables with row locking. Tables InnoDB . |
| MERGE | Un emsemble de tables MyISAM utilisées comme une seule et même table. Tables MERGE . |
| MRG_MyISAM | Un synonyme pour MERGE les tables. |
| MyISAM | Le nouveau gestionnaire de table binaire et portable. Tables MyISAM . |

Types de tables MySQL .

Si un type de table est demandé, mais que ce type particulier n'est pas disponible, MySQL va choisir le type de table le plus proche de celui qui est spécifié. Par exemple, si TYPE=BDB est spécifié, et que la distribution de MySQL ne supporte pas les tables BDB , la table qui sera créée sera du type MyISAM .

Les autres options de tables sont utilisées pour optimiser le comportement de la table. Dans la plupart des cas, vous n'avez pas à les spécifier. Les options fonctionnent pour tous les types de tables (sauf contre−indication) :

| Option | Description |
|---|---|
| AUTO_INCREMENT | La prochaine valeur auto_increment de votre table (MyISAM).. |
| AVG_ROW_LENGTH | La taille moyenne approchée des lignes de votre table. Vous ne devez fournir cette valeur que pour les tables à taille de ligne variable, de très grande taille. |
| CHECKSUM | Passez 1 si vous voulez que MySQL génère une somme de vérification (ce qui facilite la recherche des lignes corrompues, mais ralentit les mises à jour) (MyISAM). |
| COMMENT | Un commentaire pour votre table (60 caractères). |
| MAX_ROWS | Nombre de lignes maximum que vous pensez stocker dans la table. |
| MIN_ROWS | Nombre de minimum lignes que vous pensez stocker dans la table. |
| PACK_KEYS | Passez 1 si vous voulez un index plus compact. Cela rend les mises à jour plus lente, mais les lectures plus rapides (MyISAM, ISAM). |
| PASSWORD | Chiffre le fichier .frm avec un mot de passe. Cette option ne fait rien du tout pour la version standard de MySQL. |
| DELAY_KEY_WRITE | Passez 1 si vous voulez attendre la fermeture de la table pour mettre à jour les index. |
| ROW_FORMAT | Definit la méthode de stockage des lignes (réservé pour le futur). Actuellement, cette option fonctionne uniquement avec des tables MySAM qui supportent le DYNAMIC et FIXED en format de ligne. Format de table MyISAM . |

Lorsque vous utilisez une table MyISAM, MySQL utilise le produit max_rows * avg_row_longueur pour décider de la taille de la table. Si vous ne spécifiez pas ces options, la taille maximum sera 4Go (ou 2Go si votre système d'exploitation ne supporte que les tables de 2 Go). La raison de cette option est le choix des tailles de pointeurs d'index : plus la table sera petite, plus les index seront petits, et rapides à lire.

Si vous n'utilisez pas l'option PACK_KEYS , l'option par défaut est de ne compacter que les chaînes, et pas les nombres. Si vous passez PACK_KEYS=1 , les nombres seront aussi compactés.

Lorsque vous compactez des clés binaires numériques, MySQL utilisera la compression par préfixe. Cela signifie que vous n'y aurez vraiment intérêt, que si beaucoup de nombres sont identiques. La compression par préfixe utilise un octet de plus pour chaque clé, pour

indiquer le nombre d'octets de la clé courante, identique à la clé précédente (notez que le pointeur de ligne est stocké au format bigendian (les premiers bits ont le plus de poids), pour améliorer la taux de compression). Cela signifie que si vous avez plusieurs clés de la même valeurs sur des lignes consécutives, les clés ne prendront que 2 octets (y compris le pointeur de ligne). Faites vous−même la comparaison avec la méthode standard, où la clé suivante occupe `storage_size_for_key + pointer_size` (généralement 4). D'un autre coté, si toutes vos clés sont totalement différente, vous perdre un autre octet par clé (si la clé ne peut avoir de valeur `NULL` : dans ce cas, la taille de la clé compactée sera stockée dans le même octet qui indique si la clé est `NULL` .)

- Si vous spécifiez une clause `SELECT` dans une commande `CREATE STATEMENT` , MySQL créera de nouveaux champs pour tous les éléments du `SELECT` . Par exemple:

```
mysql> CREATE TABLE test (a INT NOT NULL AUTO_INCREMENT,
    ->        PRIMARY KEY (a), KEY(b))
    ->        TYPE=MyISAM SELECT b,c FROM test2;
```

Cette ligne va créer une table `MySAM` de 3 colonnes. Notez que cette table sera automatiquement supprimée si une erreur survient durant la copie des données dans la table. Voyez cet exemple:

```
mysql> SELECT * FROM foo;
+---+
| n |
+---+
| 1 |
+---+

mysql> CREATE TABLE bar (m INT) SELECT n FROM foo;
Query OK, 1 row affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM bar;
+------+---+
| m    | n |
+------+---+
| NULL | 1 |
+------+---+
1 row in set (0.00 sec)
```

Pour chaque ligne dans la table `foo` , une ligne est insérée dans `bar` avec la valeur de `foo` et les valeurs par défaut pour les nouvelles colonnes.

`CREATE TABLE ... SELECT` ne créera pas d'index automatiquement pour vous. Cela est intentionnel, pour rendre la commande aussi flexible que possible. Si vous voulez avoir des index dans la table créée, vous devez le spécifier avant la commande `SELECT` :

```
mysql> CREATE TABLE bar (UNIQUE (n)) SELECT n FROM foo;
```

Si une erreur survient durant la copie des données dans la table, elle sera automatiquement effacée.Pour d'assurer que les journaux des modifications ou les journaux binaires puissent être utilisés pour re−créer les tables originales, MySQL n'autorise pas les insertions concurrents pendant `CREATE TABLE ... SELECT` .

- L'option `RAID_TYPE` vous aidera à passer outre la limite de 2Go/4Go pour les fichiers MyISAM (mais pas le fichier d'index), sur les systèmes d'exploitation qui ne supportent pas les grands fichiers. Il est possible d'accélérer le goulet d'étranglement des I/O en disposant les répertoires `RAID` sur différents disques physiques. `RAID_TYPE` fonctionne sur tous les OS, à condition d'avoir configuré MySQL avec `--with-raid` . Actuellement, le seul `RAID_TYPE` autorisé est `STRIPED` ( `1` et `RAID0` sont des redirections vers celui ci).

Si vous spécifiez `RAID_TYPE=STRIPED` pour une table MyISAM, MyISAM va créer des sous−dossiers `RAID_CHUNKS` nommés 00, 01, 02 dans le dossier de la base de données. Dans chacun de ces dossiers, MyISAM va créer un fichier nom_de_table.MYD. Lorsqu'il écrira dans le fichier, le gestionnaire `RAID` placera les `RAID_CHUNKSIZE` *1024 premiers octets dans le premier fichier, et les `RAID_CHUNKSIZE` *1024 octets suivants dans le fichier suivant.

- UNION sert lorsque vous voulez utiliser un ensemble de tables comme une seule table. Cela ne fonctionne qu'avec les tables MERGE . Tables MERGE .

  Actuellement, vous devez avoir les droits de SELECT , UPDATE , et DELETE sur ces tables pour les consolider en une seule table MERGE . Toutes les tables doivent être de la même base que la table consolidée.
- Si vous voulez insérer des données dans une table MERGE , vous devez spécifier avec INSERT_METHOD . Tables MERGE . Cette option a été introduite dans MySQL 4.0.0.
- Dans la table ainsi créée, la clé primaire PRIMARY sera placée en premier, suivie des clés uniques et des clés standard. Cela aide l'optimiseur MySQL a utiliser les clés dans l'ordre de priorité, et à détecter les clés doubles.
- En utilisant DATA DIRECTORY="directory" ou INDEX DIRECTORY="directory" vous pouvez spécifier ou le gestionnaire de la table doit mettre la table et son index. Notez que le chemin doit être complet. Pas de chemin relatif. Cela fonctionne uniquement dans les tables MyISAM en MySQL 4.0, quand vous avez pas utilisé l'option --skip-symlink . Utiliser les liens symboliques pour les tables .

### 6.5.3.1 Modification automatique du type de colonnes

Dans certains cas, MySQL change automatiquement la spécification d'une colonne fournie dans la commande CREATE TABLE . (Cela peut aussi arriver avec ALTER TABLE ) :

- Les colonnes VARCHAR avec une taille inférieure à quatre (4) sont changées en CHAR .
- Si l'une des colonnes d'une table est de taille variable, toute la ligne est, par conséquent, de taille variable. Ainsi, si une ligne contient une colonne de taille variable ( VARCHAR , TEXT ou BLOB ) toutes les colonnes CHAR de plus de trois caractères sont transoformées en VARCHAR . Cela ne change en rien la façon dont vous utilisez les colonnes. Pour MySQL, VARCHAR est simplement une autre façon de stocker les caractères. MySQL effectue cette conversion car cela économise de la place, et rend les calculs sur les tables plus rapides. Types de tables MySQL .
- La taille d'affichage de TIMESTAMP doit être un nombre pair et être compris entre 2 et 14. (2, 4, 6, 8, 10, 12 ou 14). Si vous spécifiez une taille plus grande que 14, ou inférieure à 2, celle−ci sera tranformée en 14. Les valeurs impaires sont rammenées à la valeur pair supérieure la plus proche.
- # Vous ne pouvez pas stocker de valeur littérale NULL dans une colonne de type TIMESTAMP . Cette valeur sera remplacée par la date et l'heure courante. De ce fait, les attributs NULL et NOT NULL n'ont pas de sens pour ces colonnes et sont ignorés. DESCRIBE nom_de_table indiquera toujours que la colonne TIMESTAMP accepte les valeurs NULL .
- MySQL change certains type de colonnes utilisés par d'autres serveurs SQL en types MySQL. Types de colonnes issues d'autres bases de données .

Si vous voulez voir si MySQL a utilisé un autre type que celui que vous avez spécifié, utilisez la commande DESCRIBE nom_de_table , après votre création ou modification de structure de table. Certain types de colonnes peuvent être modifiés si vous compressez une table en utilisant l'utilitaire myisampack .

## 6.5.4 Syntaxe de ALTER TABLE

```
ALTER [IGNORE] TABLE nom_de_table alter_spec [, alter_spec ...]

alter_specification:
       ADD [COLUMN] create_definition [FIRST | AFTER nom_colonne ]
 ou    ADD [COLUMN] (create_definition, create_definition,...)
 ou    ADD INDEX [nom_index] (index_nom_colonne,...)
 ou    ADD PRIMARY KEY (index_nom_colonne,...)
```

```
ou    ADD UNIQUE [nom_index] (index_nom_colonne,...)
ou    ADD FULLTEXT [nom_index] (index_nom_colonne,...)
ou    ADD [CONSTRAINT symbol] FOREIGN KEY [nom_index] (nom_colonne_index,...)
          [reference_definition]
ou    ALTER [COLUMN] nom_colonne &#123;SET DEFAULT literal | DROP DEFAULT&#125;
ou    CHANGE [COLUMN] ancien_nom_colonne create_definition
             [FIRST | AFTER nom_de_colonne]
ou    MODIFY [COLUMN] create_definition [FIRST | AFTER nom_colonne]
ou    DROP [COLUMN] nom_colonne
ou    DROP PRIMARY KEY
ou    DROP INDEX nom_index
ou    DISABLE KEYS
ou    ENABLE KEYS
ou    RENAME [TO] nouveau_nom_de_table
ou    ORDER BY col
ou    table_options
```

`ALTER TABLE` vous permet de changer la structure d'une table existante. Par exemple, vous pouvez ajouter ou supprimer des colonnes, des index, changer le type des colonnes existantes, renommer ces colonnes, ou la table elle−même. Vous pouvez de même changer le commentaire sur la table, ou le type de celle−ci. Syntaxe de `CREATE TABLE` .Si vous utilisez `ALTER TABLE` pour modifier les spécifications d'une colonne mais que `DESCRIBE nom_de_table` vous indique que cette colonne n'a pas été modifiée, il est possible que MySQL ait ignoré vos modifications pour une des raisons décrite dans Silent column changes . Par exemple, si vous essayez de changer une colonne de type `VARCHAR` en `CHAR` , MySQL continuera d'utiliser `VARCHAR` si la table contient d'autres colonnes de taille variable.

`ALTER TABLE` effectue une copie temporaire de la table originale. Les modifications sont faites sur cette copie, puis l'original est effacée, et enfin la copie est renommée pour remplacer l'originale. Cette méthode permet de rediriger toutes les commandes automatiquement vers la nouvelle table sans pertes. Durant l'exécution de `ALTER TABLE` , la table originale est lisible par d'autres clients. Les modifications et insertions sont reportées jusqu'à ce que la nouvelle table soit prête.

Notez que si vous utilisez une autre option que `RENAME` avec `ALTER TABLE` , MySQL créera toujours une table temporaire, même si les données n'ont pas besoin d'être copiées (comme quand vous changez le nom d'une colonne). Nous avons prévu de corriger cela dans les versions suivantes, mais comme la commande `ALTER TABLE` n'est pas utilisée très souvent, cette correction ne fait pas partie de nos priorités.

Pour les tables `MyISAM` , vous pouvez accélérer la recréation de l'index (qui est la plus lente de tout le processus) en choisissant une valeur élevée pour la variable `myisam_sort_buffer_size` .

- Pour utiliser `ALTER TABLE` , vous devez avoir les droits `ALTER` , `INSERT` , et `CREATE` sur la table.
- `IGNORE` est une extension MySQL pour ANSI SQL92. Cette option contrôle la façon dont `ALTER TABLE` fonctionne s'il y a des duplications sur une clef unique de la nouvelle table. Si `IGNORE` n'est pas spécifiée, la copie est annulée et la table originale est restaurée. Si `IGNORE` est spécifiée, les lignes contenant les éléments doublons de la table seront effacées, hormis la première, qui sera conservée.
- Vous pouvez effectuer plusieurs opérations de `ADD` , `ALTER` , `DROP` , et `CHANGE` dans une même commande `ALTER TABLE` . C'est une extension de MySQL à la norme ANSI SQL92, qui n'autorise qu'une seule modification par commande `ALTER TABLE` .
- `CHANGE nom_colonne` , `DROP nom_colonne` , et `DROP INDEX` sont des extensions de MySQL à la norme ANSI SQL92.
- `MODIFY` est une extension Oracle à `ALTER TABLE` .
- Le mot optionnel `COLUMN` est purement de la fioriture et peut être ignoré.
- Si vous utilisez `ALTER TABLE nom_de_table RENAME TO nouveau_nom` sans autre option, MySQL va simplement renommer les fichiers qui correspondent à la table `nom_de_table` . Il n'y a pas de création de fichier temporaire. Syntaxe de `RENAME TABLE` .
- La définition `create_definition` utilise la même syntaxe pour les clauses `ADD` et `CHANGE` que dans `CREATE TABLE` . Notez que cette syntaxe inclut le nom de la colonne, et pas seulement son type Syntaxe de `CREATE TABLE` .

- Vous pouvez renommer une colonne avec la syntaxe `CHANGE ancien_nom_de_colonne create_definition` . Pour cela, indiquez l'ancien nom de la colonne, puis le nouveau nom et son type courant. Par exemple, pour renommer une colonne de type `INTEGER` , de `a` en `b` , vous pouvez faire ceci :
```
mysql> ALTER TABLE t1 CHANGE a b INTEGER;
```
  Si vous ne voulez changer que le type de la colonne, avec la clause `CHANGE` vous devrez redonner le nom de la colonne. Par exemple :
```
mysql> ALTER TABLE t1 CHANGE b b BIGINT NOT NULL;
```
  Cependant, à partir de la version 3.22.16a de MySQL, vous pouvez aussi utiliser la clause `MODIFY` pour changer le type d'une colonne sans la renommer :
```
mysql> ALTER TABLE t1 MODIFY b BIGINT NOT NULL;
```
- Si vous utilisez les clauses `CHANGE` ou `MODIFY` pour réduire la taille d'une colonne qui comportait un index sur une partie de la colonne (par exemple, si vous aviez un index sur 10 caractères d'une colonne de type `VARCHAR` ), vous ne pouvez pas rendre la colonne plus petite que le nombre de caractères indexés.
- Quand vous changez le type d'une colonne avec `CHANGE` ou `MODIFY` , MySQL essaye de convertir les données au niveau type dans la mesure du possible.
- A partir de la version 3.22 de MySQL, vous pouvez utiliser `FIRST` ou `ADD ... AFTER nom_colonne` pour ajouter la colonne à un endroit spécifique dans la table. Par défaut, la colonne est ajoutée à la fin. A partir de la version 4.0.1, vous pouvez aussi utiliser les mots clés `FIRST` et `AFTER` avec `CHANGE` ou `MODIFY` .
- `ALTER COLUMN` spécifie une nouvelle valeur par défaut pour une colonne ou enlève l'ancienne. si l'ancienne valeur est effacée et que la colonne peut être `NULL` , la nouvelle valeur par défaut sera `NULL` . Si la colonne ne peut être `NULL` , MySQL assigne une valeur par défaut, comme défini dans Syntaxe de `CREATE TABLE` .
- `DROP INDEX` supprime un index. C'est une extension MySQL à la norme ANSI SQL92. Syntaxe de `DROP INDEX` .
- Si des colonnes sont effacées d'une table, ces colonnes sont aussi supprimés des index dont elles font partie. Si toutes les colonnes qui forment un index sont effacées, l'index lui même est supprimé.
- Si une table ne comporte qu'une seule colonne, La colonne ne peut être supprimée. Si vous voulez effacer la table, utilisez la commande `DROP TABLE` .
- `DROP PRIMARY KEY` supprime la clef primaire. Si cette clef n'existe pas, cette commande effacera le premier index `UNIQUE` de la table. (MySQL marque la première clef `UNIQUE` en tant que `PRIMARY KEY` si aucune `PRIMARY KEY` n'a été spécifiée explicitement.)

  Si vous ajoutez un `UNIQUE INDEX` ou `PRIMARY KEY` à une table, c'est enregistré avant les index non−`UNIQUE` pour que MySQL puisse détecter les valeurs dupliquées aussi vite que possible.
- `ORDER BY` vous permet de créer une nouvelle table tout en ordonnant les lignes par défaut. Notez que cet ordre ne sera pas conservé après les prochaines insertions et modifications. Dans certains cas, cela aide MySQL si les colonnes sont dans l'ordre dans lequel vous allez trier les valeurs. Cette option n'est vraiment utile que si vous savez à l'avance dans quel ordre vous effectuerez les tris : vous y gagnerez alors en performances.
- Si vous utilisez `ALTER TABLE` sur une table `MyISAM` , tous les index non−uniques sont créés par des opérations séparées. (comme dans `REPAIR` ). Cela devrait rendre `ALTER TABLE` plus rapide quand vous avez beaucoup d'index.
- Depuis la version **4.0** de MySQL, la fonctionnalité ci−dessus peut être activée explicitement. `ALTER TABLE ... DISABLE KEYS` force MySQL à ne plus mettre à jour les index non−uniques pour les tables au format `MyISAM` . `ALTER TABLE ... ENABLE KEYS` doit alors être utilisé pour recréer les index manquants. Comme MySQL le fait avec un algorithme spécial qui est plus rapide que le fait d'insérer les clefs une par une, désactiver les clefs peut vous faire gagner en performances.
- Avec la fonction `mysql_info()` de l'API C, vous pouvez savoir combien d'enregistrements ont été copiés, et (quand `IGNORE` est spécifié) combien d'enregistrements ont été éffacés à cause de la clef unique.
- Les clauses `FOREIGN KEY` , `CHECK` , et `REFERENCES` ne font rien pour le moment, à part pour les tables `InnoDB` qui supportent la commande `ADD CONSTRAINT FOREIGN KEY (...) REFERENCES ... (...)` . Notez que `InnoDB`

ne permet pas la spécification de nom_index . Tables InnoDB . La syntaxe pour les autres types de tables est fournie pour assurer la compatibilité, rendre le port du code à partir d'autres serveurs SQL plus facile et faire fonctionner les applications qui créent des tables avec des références. Différences entre MySQL et ANSI SQL92 .

Voilà un exemple qui montre quelques utilisations de ALTER TABLE . On commence par une table t1 créée comme suit :

```
mysql> CREATE TABLE t1 (a INTEGER,b CHAR(10));
```

Pour renommer la table de t1 à t2 :

```
mysql> ALTER TABLE t1 RENAME t2;
```

Pour changer une colonne a de INTEGER en TINYINT NOT NULL (en laissant le même nom), et pour changer une colonne b de CHAR(10) à CHAR(20) et la renommant de b en c :

```
mysql> ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);
```

Pour ajouter une nouvelle colonne TIMESTAMP nommée d :

```
mysql> ALTER TABLE t2 ADD d TIMESTAMP;
```

Pour ajouter un index sur une colonne d , et rendre la colonne a la clef primaire :

```
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);
```

Pour effacer la colonne c :

```
mysql> ALTER TABLE t2 DROP COLUMN c;
```

Pour ajouter une nouvelle colonne AUTO_INCREMENT nommée c :

```
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
          ADD INDEX (c);
```

Notez que nous avons indexé c , car les colonnes AUTO_INCREMENT doivent être indexées, et que nous définissons aussi c en tant que NOT NULL , car les colonnes indexées ne peuvent être NULL .Quand vous ajoutez une colonne AUTO_INCREMENT , les valeurs de la colonne sont remplies automatiquement pour vous. Vous pouvez choisir la valeur de départ pour l'indexation en utilisant SET INSERT_ID=# avant ALTER TABLE ou en utilisant l'option AUTO_INCREMENT = # de la table. Syntaxe de SET .

Avec les tables de type MyISAM , si vous ne changez pas la colonne AUTO_INCREMENT , l'indice d'auto−incrémentation ne sera pas affecté. Si vous effacez une colonne AUTO_INCREMENT puis en ajoutez une autre, l'indexation recommencera à partir de 1.

Problèmes avec ALTER TABLE . .

## 6.5.5 Syntaxe de RENAME TABLE

```
RENAME TABLE nom_de_table TO nouveau_nom_de_table[, nom_de_table2 TO nouveau_nom_de_table2,...]
```

Le changement de nom se fait atomiquement ce qui signifie qu'aucun autre processus ne peut accéder la table tant que l'opération est en cours. Cela rend possible de remplacer une vielle table avec une table vide :

```
 CREATE TABLE nouvelle_table (...);
RENAME TABLE ancienne_table TO backup_table, nouvelle_table TO ancienne_table;
```

L'opération s'effectue de gauche à droite ce qui signifie que si vous voulez échanger deux noms de tables, vous devez :

```
RENAME TABLE ancienne_table    TO backup_table,
            nouvelle_table    TO ancienne_table,
            backup_table TO nouvelle_table;
```

Si les deux bases de données sont sur le même disque, vous pouvez renommer à travers les bases :

```
RENAME TABLE bdd_courante.nom_de_table TO autre_bdd.nom_de_table;
```

Quand vous exécutez RENAME , vous ne pouvez avoir aucune transaction active ou une table protégée en mode écriture. Vous devez avoir les privilèges ALTER et DROP sur l'ancienne table, et les privilèges CREATE et INSERT sur la nouvelle.

Si MySQL rencontre des erreurs dans un renommage multiple, il remettra les noms changés à leurs valeurs d'origine pour revenir à l'état d'origine.

`RENAME TABLE` a été ajouté à la version 3.23.23 de MySQL.

## 6.5.6 Syntaxe de DROP TABLE

```
DROP [TEMPORARY] TABLE [IF EXISTS] nom_de_table [, nom_de_table2,...] [RESTRICT | CASCADE]
```

`DROP TABLE` supprime une ou plusieurs tables. Toutes les données et la structure de la tables sont **perdues** , alors soyez **prudents** avec cette commande !Depuis la version 3.22 de MySQL, vous pouvez utiliser le mot réservé `IF EXISTS` pour éviter l'affichage des erreurs pour les tables qui n'existent pas. `SHOW WARNINGS | ERRORS` .

`RESTRICT` et `CASCADE` sont autorisés pour faciliter le port. Pour le moment, elles ne font rien.

**Note** : `DROP TABLE` va automatiquement valider les transactions actives (hormis si vous utilisez la version 4.1 et le mot clé `TEMPORARY` ).

L'option `TEMPORARY` est ignorée en 4.0. En 4.1, cette option fonctionne comme suit :

- Détruit uniquement les tables temporaires.
- Ne termine pas les transactions en cours.
- Aucun droits d'accès n'est vérifié.

`TEMPORARY` est pour le moment ignoré; Dans un futur proche, il servira à s'assurer qu'on efface vraiment une table temporaire.

## 6.5.7 Syntaxe de CREATE INDEX

```
CREATE [UNIQUE|FULLTEXT] INDEX index_name
      ON tbl_name (col_name[(length)],... )
```

La requête `CREATE INDEX` n'effectue aucune action sur les versions de MySQL antérieures à la version 3.22. Dans les versions 3.22 et supérieures, `CREATE INDEX` est équivalent à une requête `ALTER TABLE` pour créer des index. Syntaxe de `ALTER TABLE` .

Normalement, tous les index sont créés en même temps que la table elle même avec `CREATE TABLE` . Syntaxe de `CREATE TABLE` . `CREATE INDEX` permet d'ajouter des index à une table existante.

Une liste de colonnes de la forme `(col1,col2,...)` crée un index multi−colonnes. Les valeurs de l'index sont créées en concaténant la valeur dex colonnes données.

Pour les colonnes `CHAR` et `VARCHAR` , les index peut être créés sur uniquement une partie de la colonne, avec la syntax `col_name(length)` .(Pour les colonnes `BLOB` et `TEXT` la longueur d'index est obligatoire.) La requête suivante crée un index en utilisant les 10 premiers caractères de la colonne `name` :

```
mysql> CREATE INDEX part_of_name ON customer (name(10));
```

Comme la plupart des noms ont en général des différences dans les 10 premiers caractères, l'index ne devrait pas être plus lent qu'un index créé à partir de la colonne `name` en entier. Ainsi, en n'utilisant qu'une partie de la colonne pour les index, on peut réduire la taille du fichier d'index, ce qui peut permettre d'économiser beaucoup d'espace disque, et peut aussi accélérer les opérations `INSERT` !

Il est important de savoir qu'on peut indexer une colonne qui peut avoir la valeur `NULL` ou une colonne `BLOB` / `TEXT` que si on utilise une version 3.23.2 ou supérieure de MySQL et en utilisant le type `MyISAM` .

Pour plus d'informations à propos de l'utilisation des index dans MySQL, voir Comment MySQL utilise les index .

Les index `FULLTEXT` ne peuvent indexer que des colonnes `VARCHAR` ou `TEXT` , et seulement dans les tables `MyISAM` . Les index `FULLTEXT` sont disponibles dans les versions 3.23.23 et supérieures de MySQL. Recherche full−text avec MySQL .

### 6.5.8 Syntaxe de DROP INDEX

```
DROP INDEX nom_de_l_index ON nom_de_table
```

`DROP INDEX` supprime l'index nommé `nom_de_l_index` de la table `nom_de_table` . `DROP INDEX` ne fait rien avec la version 3.22 et les précédentes. Depuis cette version, `DROP INDEX` est un alias d' `ALTER TABLE` supprimant l'index.

Syntaxe de `ALTER TABLE` .

## 6.6 Commandes de bases de l'utilisateur de MySQL

### 6.6.1 Syntaxe de USE

```
USE db_name
```

La commande `USE db_name` spécifie à MySQL d'utiliser la base `db_name` comme base par défaut pour les requêtes ne les mentionnant pas. La base choisie reste la même jusqu'à la fermeture de la session ou un nouvel appel à `USE` :

```
 mysql> USE db1;
mysql> SELECT COUNT(*) FROM ma_table;      # sélectionne à partir de db1.ma_table
mysql> USE db2;
mysql> SELECT COUNT(*) FROM ma_table;      # sélectionne à partir de db2.ma_table
```

Rendre une base de données la base courante (en utilisant `USE` ) ne vous interdit pas l'accès à d'autres tables dans d'autres bases. L'exemple suivant accède à la table `author` de la base `db1` et à la table `editor` de la base `db2` :

```
 mysql> USE db1;
mysql> SELECT author_name,editor_name FROM author,db2.editor
    ->        WHERE author.editor_id = db2.editor.editor_id;
```

La commande `USE` est fournie pour assurer la compatibilité Sybase.

### 6.6.2 Syntaxe de DESCRIBE (obtenir des informations sur les colonnes)

```
&#123;DESCRIBE | DESC&#125; nom_de_table [nom_de_colonne | wild]
```

`DESCRIBE` est un raccourçi de `SHOW COLUMNS FROM` . Lire des informations sur les bases, colonnes, tables ou index .

`DESCRIBE` fournit des informations à propos des colonnes de la table. `nom_de_colonne` peut être le nom d'une colonne ou une chaîne contenant les caractères spéciaux SQL `'%'` et `'_'` .

Si le type de colonne est différent de celui que vous pensiez avoir définit lors du `CREATE TABLE` , nottez que MySQL change le type des colonnes de temps en temps. Silent column changes .

Cette instruction est fournie pour une meilleure compatibilité avec Oracle.

L'instruction `SHOW` renvoit les mêmes informations. Syntaxe de `SHOW` .

# 6.7 Commandes relatives aux verrous et aux transactions

## 6.7.1 Syntaxe de BEGIN/COMMIT/ROLLBACK

Par défaut, MySQL est lancé en mode `autocommit` . Cela signifie que chaque modification effectué est enregistré immédiatement sur le disque par MySQL.

Si vous utilisez des tables supportant les transactions (comme `InnoDB` , `BDB` ), vous pouvez configurer MySQL en mode non– `autocommit` grâce à la commande:

```
SET AUTOCOMMIT=0
```

A partir de là, vous devez utiliser `COMMIT` pour enregistrer les modifications sur le disque ou `ROLLBACK` pour ignorer les modifications apportées depuis le début de la transaction.Si vous souhaitez sortir du mode `AUTOCOMMIT` pour une série d'opérations, vous pouvez utiliser les commandes `BEGIN` ou `BEGIN WORK` :

```
BEGIN;
SELECT &#64;A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summmary=&#64;A WHERE type=1;
COMMIT;
```

Vous devez savoir que si vous utilisez des tables ne supportant pas les transactions, les modifications seront écrites immédiatement, indépendamment de l'état du mode de `autocommit` Si vous faites un `ROLLBACK` après avoir modifié une table non transactionnelle, vous obtiendrez ( `ER_WARNING_NOT_COMPLETE_ROLLBACK` ) comme message d'alerte. Toutes les tables supportant les transactions seront restaurées, mais aucune des autres tables ne changera.

Si vous utilisez `BEGIN` ou `SET AUTOCOMMIT=0` , il est recommandé d'utiliser les "binary log" de MySQL à la place des anciens logs d'update pour les sauvegardes. Les transactions sont stockées dans les logs binaires en un seul bloc, après `COMMIT` , pour être sû que les transactions qui ont été annulées ne soient pas enregistrées. Le log binaire des mises à jour .

Les commandes suivantes déclenchent la fin des transactions (comme si vous aviez utilisé `COMMIT` avant d'exécuter la commande):

| Command | Command | Command |
|---|---|---|
| ALTER TABLE | BEGIN | CREATE INDEX |
| DROP DATABASE | DROP TABLE | RENAME TABLE |
| TRUNCATE | | |

Vous pouvez changer le niveau d'isolation des transactions avec `SET TRANSACTION ISOLATION LEVEL ...` . Syntaxe de `SET TRANSACTION` .

## 6.7.2 Syntaxe de LOCK TABLES/UNLOCK TABLES

```
LOCK TABLES nom_de_table [AS alias] &#123;READ [LOCAL] | [LOW_PRIORITY] WRITE&#125;
           [, nom_de_table [AS alias] &#123;READ [LOCAL] | [LOW_PRIORITY] WRITE&#125; ...]
...
UNLOCK TABLES
```

`LOCK TABLES` verrouille une table pour le thread courant. `UNLOCK TABLES` déverrouillera automatiquement tous

les verrous posés par le thread courant. Toutes les tables verrouillées par le thread courant sont automatiquement déverrouillées quand ce thread utilise à nouveau `LOCK TABLES` , ou quand la connexion au serveur est perdue.L'utilisation de `LOCK TABLES` dans MySQL 4.0.2 nécessite le privilège `LOCK TABLES` global et un privilège de `SELECT` sur les tables impliquées. Dans MySQL 3.23, il faut les privilèges `SELECT` , `INSERT` , `DELETE` et `UPDATE` sur les tables.

Les principales raisons d'utiliser `LOCK TABLES` sont l'émulation de transactions ou l'accélération des processus de modification de tables. Cela sera détaillé plus loin.

Si un thread obtient un verrouillage `READ` sur une table, ce thread (et tous les autres threads) peuvent uniquement accéder a cette table en lecture. Si un thread obtient un verrouillage `WRITE` sur une table, alors seul le thread qui a posé le verrou peut lire ou écrire sur cette table. Tous les autres threads sont bloqués.

La différence entre `READ LOCAL` et `READ` est que `READ LOCAL` autorise des requêtes `INSERT` non−conflictuelles à être exécutées alors que le verrou est posé. Ceci ne peut cependant pas être utilisé si vous souhaitez modifier les fichiers de la base de données en dehors de MySQL pendant que le verrou est posé.

Quand vous utilisez `LOCK TABLES` , vous devez verrouiller toutes les tables que vous allez utiliser, et vous devez utiliser les mêmes alias sur ce que vous utiliserez dans vos requêtes ! Si vous utilisez une table a plusieurs reprises dans une requête (avec des alias), vous devez verrouiller chacun des alias !

Les verrous `WRITE` ont normalement des priorités supérieures aux verrous `READ` , afin de s'assurer que les updates sont exécutés au plus vite. Cela signifie que si un thread demande un verrou `READ` et qu'un autre thread demande un verrou `WRITE` , la demande de verrou `READ` attendra que le thread `WRITE` ait abouti pour libérer le verrou. Vous pouvez utiliser le verrou `LOW_PRIORITY WRITE` pour permettre à d'autres threads d'obtenir des verrous `READ` pendant que le thread attend le verrou `WRITE` . Vous ne devriez utiliser les verrous `LOW_PRIORITY WRITE` que si vous êtes sûr qu'il y aura effectivement un moment où aucun thread ne posera de verrou `READ` .

`LOCK TABLES` fonctionne de la manière suivante :

- Trie toutes les tables à verrouiller dans un ordre défini par MySQL (l'utilisateur ne définit pas d'ordre).
- Si une table est verrouillée avec un verrou read et un verrou write, il pose le verrou write avant le read.
- Verrouille une table à la fois jusqu'à ce que le thread ait tous ses verrous.

Cette politique garantit le bon verrouillage des tables. Il faut cependant connaitre certaines choses sur ce schema :

Si vous utilisez un verrou `LOW_PRIORITY WRITE` pour une table, cela signifie seulement que MySQL attendra, pour poser ce verrou, qu'aucun autre thread ne réclame de verrou `READ` . Quand le thread aura le verrou `WRITE` et qu'il attendra que les verrous soient posés sur les autres tables de la liste, tous les autres threads attendront que le verrou `WRITE` soit libéré. Si cela devient un problème grave pour votre application, il est conseillé de convertir des tables en tables supportant les transactions.

Vous pouvez terminer un thread attendant un verrouillage de table en toute sécurité avec `KILL` . Syntaxe de `KILL` .

Il est **déconseillé** de verrouiller des tables utilisées avec `INSERT DELAYED` , car, dans ce cas, la requête `INSERT` est exécutée dans un autre thread.

Normalement, vous n'avez pas besoin de verrouiller les tables puisque chaque requête `UPDATE` est atomique : aucun autre thread ne peut interférer avec une autre requête active. Il existe cependant quelques cas où vous aurez besoin de verrouiller les tables :

- Si vous allez exécuter plusieurs requêtes sur plusieurs tables, il est préférable, d'un point de vue rapidité, de verrouiller les tables dont vous aurez besoin. L'inconvénient, bien sur, est que les autres threads ne pourront pas intervenir sur ces tables durant vos opérations, ni en extraire des informations si la table est en `WRITE` –locked.La raison pour laquelle les requêtes sont plus rapides avec `LOCK TABLES` est que MySQL ne rafraichit pas l'index des clés des tables verrouillées tant que `UNLOCK TABLES` n'est pas invoqué (normalement, le cache des clés est rafraichi après chaque requête SQL). Cela accelère les insertions, les modifications et les suppressions de données dans les tables `MyISAM` .
- Si vous utilisez un type de table dans MySQL qui ne supporte pas les transactions, vous devez utiliser `LOCK TABLES` pour vous assurez qu'aucun autre thread ne s'intercale entre un `SELECT` et un `UPDATE` . L'exemple suivant necessite `LOCK TABLES` pour s'executer en toute sécurité :

```
 mysql> LOCK TABLES trans READ, customer WRITE;
mysql> SELECT SUM(value) FROM trans WHERE customer_id=some_id;
mysql> UPDATE customer SET total_value=sum_from_previous_statement
    ->        WHERE customer_id=some_id;
mysql> UNLOCK TABLES;
```

  Sans `LOCK TABLES` , Il est possible qu'un autre thread ait inseré une nouvelle ligne dans la table `trans` entre l'execution du `SELECT` et l'exécution de la requête `UPDATE` .

L'utilisation de modifications incrémentales ( `UPDATE customer SET value=value+nouvelle_valeur` ) ou de la fonction `LAST_INSERT_ID()` permet de se passer de `LOCK TABLES` en de nombreuses occasions.

Il est aussi possible de résoudre de nombreux cas en utilisant un verrou utilisateur, avec les fonctions `GET_LOCK()` et `RELEASE_LOCK()` . Ces verrous sont stockés dans une table de hashage dans le serveur et utilisent les fonctions `pthread_mutex_lock()` et `pthread_mutex_unlock()` pour plus de vitesse. Fonctions diverses .

Voir Comment MySQL verrouille les tables pour plus de détails.

Il est possible de verrouiller tous les tables de toutes les bases avec la commande `FLUSH TABLES WITH READ LOCK` .

Syntaxe de `FLUSH` . C'est une méthode très pratique pour effectuer des sauvegardes si vous utilisez un système de fichiers qui, comme Veritas, permet de créer des instantanés.

**NOTE** : `LOCK TABLES` ne fonctionne pas avec les transactions et validera automatiquement toutes les transactions actives avant de poser verrouiller la table.

## 6.7.3 Syntaxe de SET TRANSACTION

```
 SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
&#123; READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE &#125;
```

Configuration du niveau d'isolation des transactions en général, pour la totalité de la session, ou pour la prochaine transaction.

Le comportement par défaut est de configurer le niveau d'isolation pour la transaction suivante (qui n'a pas encore été commencée) En utilisant le paramètre `GLOBAL` , on configure le niveau par défaut global pour toutes les nouvelles connections. Cette commande requiert les privilèges `SUPER` . En utilisant le paramètre `SESSION` , on configure le niveau par défaut pour toutes les prochaines transactions effectuées durant la session actuelle.

On peut configurer le niveau d'isolation global des transactions pour `mysqld` avec `--transaction-isolation=...` . Options en ligne de commande de `mysqld` .

## 6.8 Recherche en Texte−entier (Full−text) dans MySQL

Comme dans la version 3.23.23, MySQL propose l'indexation et la recherche sur l'ensemble d'un champ "text" (full−text). Les index sur le texte en entier de MySQL sont des index de type `FULLTEXT` . Les index `FULLTEXT` sont utilisés avec les tables `MyISAM` et peuvent être créés depuis des colonnes de types `CHAR` , `VARCHAR` , ou `TEXT` au moment de `CREATE TABLE` ou plus tard avec `ALTER TABLE` ou `CREATE INDEX` . Pour les enregistrements les plus grands, il sera plus rapide de charger les donnés dans une table qui n'a pas d'index `FULLTEXT` , et ensuite de créer l'index avec `ALTER TABLE` (ou `CREATE INDEX` ). L'enregistrement de données dans une table qui a déjà des index `FULLTEXT` sera plus lent.

La recherche sur un texte en entier est effectuée par la fonction `MATCH()` .

```
 mysql> CREATE TABLE articles (
    ->   id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    ->   title VARCHAR(200),
    ->   body TEXT,
    ->   FULLTEXT (title,body)
    -> );
Query OK, 0 rows affected (0.00 sec)

 mysql> INSERT INTO articles VALUES
    -> (NULL,'MySQL Tutorial', 'DBMS stands for DataBase ...'),
    -> (NULL,'How To Use MySQL Efficiently', 'After you went through a ...'),
    -> (NULL,'Optimising MySQL','In this tutorial we will show ...'),
    -> (NULL,'1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    -> (NULL,'MySQL vs. YourSQL', 'In the following database comparison ...'),
    -> (NULL,'MySQL Security', 'When configured properly, MySQL ...');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

 mysql> SELECT * FROM articles
    ->          WHERE MATCH (title,body) AGAINST ('database');
+----+-------------------+-----------------------------------------+
| id | title             | body                                    |
+----+-------------------+-----------------------------------------+
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...            |
+----+-------------------+-----------------------------------------+
2 rows in set (0.00 sec)
```

La fonction `MATCH()` effectue la recherche d'une chaîne de caractères dans une liste de textes (et dans un groupe d'une ou plusieurs colonnes utilisées pour l'index `FULLTEXT` ). La chaîne recherchée est donnée en argument à `AGAINST()` . La recherche est sans distinguer les majuscules des minuscules. Pour chaque ligne de la table, `MATCH()` retourne une valeur de pertinence, qui est une mesure de la ressemblance entre le chaîne recherchée et le texte de la ligne dans le colonne donnée dans la liste de `MATCH()` .Quand `MATCH()` est utilisé comme condition de `WHERE` (voir l'exemple suivant) les lignes retournées sont automatiquement organisées avec la pertinence la plus élevé en premier. Ces pertinences sont des nombres non négatifs en virgule flottante. Une pertinence de zéro signifie qu'il n'y a pas de similarité. La pertinence est calculé en fonction du nombre de mots dans la ligne, du nombre de mots uniques dans cette ligne, du nombre total de mots dans la liste, et du nombre de documents (lignes) qui contiennent un mot en particulier.

Il est aussi possible d'exécuter une recherche en mode booléen. Ceci est décrit dans une section ultérieure.

L'exemple précédent est une illustration élémentaire qui montre comment on utilise la fonction `MATCH()` . Les lignes sont retournées par ordre décroissant de pertinence.

L'exemple suivant montre comment récupérer la valeur de pertinence explicitement. Comme il n'y a pas de condition `WHERE` ni de condition `ORDER BY` les lignes retournées ne sont pas ordonnées.

```
 mysql> SELECT id,MATCH (title,body) AGAINST ('Tutorial') FROM articles;
+----+-----------------------------------------+
```

```
| id | MATCH (title,body) AGAINST ('Tutorial') |
+----+-----------------------------------------+
|  1 |                       0.64840710366884 |
|  2 |                                      0 |
|  3 |                       0.66266459031789 |
|  4 |                                      0 |
|  5 |                                      0 |
|  6 |                                      0 |
+----+-----------------------------------------+
6 rows in set (0.00 sec)
```

L'exemple suivant est plus complexe. La requête retourne la valeur de pertinence et organise les lignes par ordre décroissant de pertinence. Pour obtenir ce résultat, il faut spécifier `MATCH()` deux fois. Cela ne cause pas de surcharge car l'optimisateur de MySQL remarquera que les deux appels à `MATCH()` sont identiques et appellent le code de recherche sur texte plein une seule fois.

```
 mysql> SELECT id, body, MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root') AS score
    -> FROM articles WHERE MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root');
+----+-------------------------------------+-----------------+
| id | body                                | score           |
+----+-------------------------------------+-----------------+
|  4 | 1. Never run mysqld as root. 2. ... | 1.5055546709332 |
|  6 | When configured properly, MySQL ... |   1.31140957288 |
+----+-------------------------------------+-----------------+
2 rows in set (0.00 sec)
```

MySQL utilise un filtre très simple pour séparer le texte en mots. Un "mot" est n'importe quelle chaîne de caractères constituée de lettres, chiffres, `'''` et `'-'` . Tout "mot" présent dans la liste des mots à ignorés ou qui est trop court (3 caractères ou moins) est ignoré.Tous les mots corrects dans la liste et dans la requête sont pondérés en fonction de leur importance dans la liste ou la requête. de cette façon, un mot présent dans de nombreux documents aura un poids faible (et peut être même un poids nul), car il a peut d'importance dans cette requête particuliète. Au contraire, si le mot est rare, il recevra un poids fort. Le poids des mots sont alors rassemblés pour calculer la pertinence de la ligne.

Une telle technique fonctionne plus efficacement sur de grands volumes de données (en fait, elle est optimisée pour cela). Avec des toutes petites tables, la distribution des mots ne reflète par correctement leur valeur sémantique et ce modèle peut parfois produire des résultats étranges.

```
 mysql> SELECT * FROM articles WHERE MATCH (title,body) AGAINST ('MySQL');
Empty set (0.00 sec)
```

La recherche du mot `MySQL` ne donne aucun résultat dans l'exemple précédent, car il est présent dans plus de la moitié des lignes. Ainsi, il est considéré comme un mot à ignorer (un mot avec une valeur sémantique nulle). C'est le comportement le plus optimal −− un langage de requêtes ne doit pas retourner chaque ligne d'une table de 1 GB.

Un mot qui est trouvé dans la moitié des enregistrements d'une table n'est pas efficace pour trouver les document appropriés. En fait, il trouvera sûrement beaucoup de documents inappropriés à la recherche. On sait tous que cela arrive souvent lorsqu'on recherche quelque chose sur internet en utilisant un moteur de recherche. C'est en suivant ce raisonnement que ces lignes se sont vues attribuer une valeur sémantique très basse dans **ce cas particulier** .

Avec la version 4.0.1, MySQL peut aussi performer des recherches booléennes en utilisant le modificateur `IN BOOLEAN MODE` .

```
 mysql> SELECT * FROM articles WHERE MATCH (title,body)
    ->     AGAINST ('+MySQL -YourSQL' IN BOOLEAN MODE);
+----+------------------------------+-------------------------------------+
| id | title                        | body                                |
+----+------------------------------+-------------------------------------+
|  1 | MySQL Tutorial               | DBMS stands for DataBase ...        |
|  2 | How To Use MySQL Efficiently | After you went through a ...        |
|  3 | Optimising MySQL             | In this tutorial we will show ...   |
|  4 | 1001 MySQL Tricks            | 1. Never run mysqld as root. 2. ... |
|  6 | MySQL Security               | When configured properly, MySQL ... |
+----+------------------------------+-------------------------------------+
```

Cette requête a retourné toutes les lignes qui contiennent le mot `MySQL` (note : la barrière des 50% n'est pas utilisée), mais ne contiennent **pas** le mot `YourSQL` . Notez qu'une recherche en mode booléen ne trie pas automatiquement les lignes par ordre de pertinence. Vous pouvez le voir dans le résultat de la requête précédente, où la ligne avec la plus haute pertinence (celui qui contient `MySQL` deux fois) est listé en dernier. Une recherche booléenne peu aussi fonctionner sans un index `FULLTEXT` , même si cela risque de **ralentir** le processus.La recherche booléenne supporte les opérateurs suivants :

+
: Un signe plus avant un mot indique que celui−ci **doit** être présent dans chaque ligne retournée.

−
: Un signe moins avant un mot indique que celui−ci **ne doit pas** être présent dans chaque ligne retournée.

*Par défaut (quand ni moins ni plus n'est spécifié) le mot est optionnel,*
: mais la ligne qui le contient sera mise en valeur. Ce fonctionnement est celui de `MATCH() ... AGAINST()` dans le paramètre `IN BOOLEAN MODE` .

< >
: Ces deux opérateurs servent à faire varier la contribution d'un mot à la pertinence associée à une ligne. L'opérateur `<` diminue la contribution alors que `>` l'augmente. Voir l'exemple ci−dessous.

*( )*
: Les parenthèses sont utilisées pour grouper des mots dans une sous−expression.

~
: Un tilde avant un mot opère comme un opérateur de négation, ce qui rend la contribution du mot à la pertinence négative. C'est utile pour se débarrasser des mots inutiles. Une ligne contenant un mot comme cela sera moins mise en avant que les autres, mais ne sera pas exclue, ce qui serait le cas avec l'opérateur `-` .

*
: C'est l'opérateur de tronquage. Contrairement aux autres opérateurs, il doit **suivre** le mot, non le précéder.

"
: La phrase, protégée par des guillemets `"` , ne trouvera que les lignes contenant cette phrase **littéralement, comme elle a été entrée** .

et voici quelques exemples :

*apple banana*
: trouve les lignes qui contiennent au moins l'un de ces mots.

*+apple +juice*
: ... les deux mots.

*+apple macintosh*
: ... le mot ``apple'', mais elles sont plus valorisées si elles contiennent aussi ``macintosh''.

*+apple −macintosh*
: ... le mot ``apple'' mais pas le mot ``macintosh''.

*+apple +(>pie <strudel)*
: ... ``apple'' et ``pie'', ou ``apple'' et ``strudel'' (l'ordre n'importe pas), mais valorise plus ``apple pie'' que ``apple strudel''.

*apple**
: ... ``apple'', ``apples'', ``applesauce'', et ``applet''.

*"some words"*
: ... ``some words of wisdom'', mais pas ``some noise words''.

## 6.8.1 Restrictions avec full−text

- Tous les paramètres de la fonction `MATCH()` doivent être des colonnes de la même table faisant partie du même index `FULLTEXT` index, sauf si `MATCH()` est en mode `BOOLEAN` .
- Les arguments de `MATCH()` doivent correspondre exactement à la liste de colonnes de certaines définitions d'index `FULLTEXT` pour la table, sauf si `MATCH()` est utilisé dans un contexte `BOOLEAN` .
- L'argument de `AGAINST()` doit être une chaîne constante.

## 6.8.2 Paramétrage précis de la recherche Full−text de MySQL

La recherche sur texte entier n'a malheureusement pas encore beaucoup de paramètres modifiables par l'utilisateur, même si l'ajout de certains apparaît très haut dans le TODO. Si vous utilisez MySQL depuis les sources ( Installer MySQL à partir des sources ), vous pouvez mieux contrôler le fonctionnement de la recherche sur texte entier.

La recherche sur texte entier a été paramétrée pour une efficacité de recherche maximale. La modification du comportement par défaut ne fera généralement que diminuer la qualité des résultats des recherches. Il ne faut pas modifier les sources de MySQL sans savoir précisément ce qu'on fait.

- La taille minimale des mots à indexer est définie dans la variable `ft_min_word_len` de MySQL. `SHOW VARIABLES` . Vous pouvez modifier cette valeur pour celle que vous préférez, puis reconstruire les index `FULLTEXT` . (Cette variable n'existe que pour la version 4.0 de MySQL)
- La liste des mots rejetés est définie dans la varaible `ft_stopword_file` . `SHOW VARIABLES` . Modifiez le selon vos goûts, recompilez MySQL et reconstruisez vos index `FULLTEXT` .
- Le taux de 50% est déterminé par la méthode de pondération choisie. Pour le désactiver, il faut changer la ligne suivante dans `myisam/ftdefs.h` :

  ```
  #define GWS_IN_USE GWS_PROB
  ```

  Par la ligne:

  ```
  #define GWS_IN_USE GWS_FREQ
  ```

  Puis recompiler MySQL. Il n'est pas nécessaire de reconstruire les index dans ce cas. **Note** : En faisant ces modifications, vous diminuez **énormément** les capacités de MySQL à fournir des valeurs pertinentes pour la fonction `MATCH()` . Si vous avez réellement besoin de faire des recherches avec ces mots courants, il est préférable de rechercher `EN MODE BOOLEEN` , lequel ne respecte pas le taux de 50%.
- Parfois le gestionnaire du moteur de recherche voudrait changer les opérateurs utilisés pour les recherches booléennes sur des textes entiers. Ceux ci sont définis dans la variable `ft_boolean_syntax` . `SHOW VARIABLES` . Cependant, cette variable n'est pas modifiable, sa valeur est fixée dans `myisam/ft_static.c` .

Pour les modifications qui nécessitent une reconstruction des index `FULLTEXT` , la méthode la plus simple pour les tables MyISAM est l'opération suivante, qui reconstruit les fichiers d'index :

```
mysql> REPAIR TABLE nom_de_table USE_FRM;
```

## 6.8.3 A faire dans la recherche Full−text

- Rendre toutes les opérations avec l'index `FULLTEXT` **plus rapides** .
- Opérateurs de proximité
- Support de listes de mots à toujours indexer ("always−index words"). Ceux ci pourraient être n'importe quelle chaîne de caractères que l'utilisateur voudrais traiter comme des mots: par exemple "C++", "AS/400", "TCP/IP", etc.
- Support de la recherche full−text sur les tables `MERGE` .

- Support des jeux de caractères multi−octets.
- Rendre la liste des mots ignorés dépendante de la langue des données.
- Stemming (dépendante de la langue des données, bien sûr).
- Pré−parseur générique pour les UDF fournies par l'utilisateur.
- Rendre le modèle plus flexible (en ajoutant des valeurs paramétrables pour `FULLTEXT` dans `CREATE/ALTER TABLE` ).

# 6.9 Cache de requêtes MySQL

Depuis la version 4.0.1, le `MySQL server` bénéficie d'un `cache de requêtes` . En fait, le cache sauvegrade le texte d'une requête `SELECT` avec le résultat qui a été envoyé au client. Si une requête identique est appelée par la suite, le serveur retournera le résultat à partir du cache plutôt que d'analyser puis exécuter la requête à nouveau.

**NOTE** : Le cache de requêtes ne retourne pas de données périmées. A chaques fois que les données sont modifiées, les entrées correspondantes dans le cache sont effacées.

La cache de requêtes est très utile dans des applications où (quelques) tables ne changent pas souvent et que vous avez les mêmes requêtes qui s'y répétent. C'est une solution typique pour les webmestres dont les sites utilisent du contenu dynamique.

Voici quelques performances du cache de requêtes. (Ces résultats on été générés en utilisant la suite benchmark MySQL sur un Linux Alpha 2 x 500 MHz avec 2GB RAM et un cache de requêtes de 64MB) :

- Si toutes les requêtes que vous effectuez sont simples (comme séléctionner un champ d'une table n'en contenant qu'un) mais diffèrent d'une manière que toutes les requêtes ne peuvent être cachées, le gain lors de l'utilisation du cache est de 13%. Cela peut être considéré comme le pire des cas. En réalité, les requêtes sont plus compliquées que notre exemple le gain est donc plus petit.
- Les recherches sur une colonne dans une table n'en contenant qu'une sont 238% plus rapides. Cela peut être considéré comme le gain minimal à attendre pour une requête cachée.
- Si vous ne voulez pas utiliser le cache de requêtes paramétrez `query_cache_size` à zéro. En désactivant le cache de requête, il n'y a aucune surcharge apparente. (le cache de requêtes peut être désactivé à l'aide de l'option de configuration `−−without−query−cache` )

## 6.9.1 Comment fonctionne le cache de requêtes

Les requêtes sont comparées avant d'être analysées :

```
SELECT * FROM table
```

et

```
Select * from table
```

sont considérées comme des requêtes différentes par le cache, ce qui fait que les requêtes doivent être les mêmes (caractère à caractère) pour être considérées comme identiques. De plus, les requêtes peuvent être considérées comme différentes si par exemple, un client utilise un nouveau format de protocole de communication ou un jeu de caractères différent d'un autre client.

Les requêtes qui utilisent différentes bases de données, différentes versions de protocole ou différents jeux de caractères par défaut sont considérées comme différentes et mises en cache séparément.

Le cache ne fonctionne pas pour les requêtes de type `SELECT CALC_ROWS ...` et `SELECT FOUND_ROWS() ...` car le nombre de lignes retournées est aussi mis en cache.

Si une table change ( `INSERT` , `UPDATE` , `DELETE` , `TRUNCATE` , `ALTER` ou `DROP TABLE|DATABASE` ), alors toutes les requêtes mises en cache qui utilisaient cette table deviennent obsolètes et en sont retirées.

Les tables transactionnelles `InnoDB` qui ont été modifiées seront rendues obsolètes lorsqu'un `COMMIT` sera exécuté.Une requête ne peut être mise en cache si elle contient l'une des fonctions suivantes :

| Fonction | Fonction | Fonction |
|---|---|---|
| Fonctions définies par l'utilisateur | CONNECTION_ID | FOUND_ROWS |
| GET_LOCK | RELEASE_LOCK | LOAD_FILE |
| MASTER_POS_WAIT | NOW | SYSDATE |
| CURRENT_TIMESTAMP | CURDATE | CURRENT_DATE |
| CURTIME | CURRENT_TIME | DATABASE |
| ENCRYPT (avec un seul paramètre) | LAST_INSERT_ID | RAND |
| UNIX_TIMESTAMP (sans paramètres) | USER | BENCHMARK |

Une requête ne peut être mise en cache non plus, si elle utilise une variable définie par l'utilisateur, si elle opère sur les tables système de MySQL, est de la forme `SELECT ... IN SHARE MODE` ou de la forme `SELECT * FROM AUTOINCREMENT_FIELD IS NULL` (pour obtenir l'index du champ `auto-increment` – utilisé par ODBC).

Par contre, `FOUND ROWS()` retournera une valeur correcte, même si la requête précédente à utilisé le cache.

Dans les cas où la requête n'utilise aucune table, en utilise une temporaire, ou que l'utilisateur à un droit particulier sur l'une des tables concernées, celle ci ne sera pas mise en cache.

Avant chaque lecture à partir du cache, MySQL vérifie que l'utilisateur à le droit de lecture (SELECT) sur toutes les bases et les tables concernées. Si ce n'est pas le cas, le cache ne sera pas utilisé.

## 6.9.2 Configuration du cache de requêtes

Le cache de requête ajoute quelques variables système `MySQL` liées à `mysqld` qui peuvent être spécifiées dans un fichier de configuration, en ligne de commande lors du démarrage de `mysqld` .

- `query_cache_limit` Ne pas cacher les résultats dont la taille est supérieure à cette valeur. (1 Mo par défaut).
- `query_cache_size` La mémoire allouée pour la mise en cache des requêtes. Si ce paramètre est à 0, le cache de requêtes est désactivé (valeur par défaut).
- `query_cache_type` Peut être (valeurs numériques seulement) :

| Option | Description |
|---|---|
| 0 | (OFF, ne pas utiliser le cache) |
| 1 | (ON, mettre en cache tous les résultats à part les requêtes SELECT SQL_NO_CACHE ... ) |
| 2 | (DEMAND, mettre en cache seulement les résultats des requêtes SELECT SQL_CACHE ... ) |

La configuratoin du cache de requête peut être changée durant la connexion. La syntaxe est la suivante :

```
QUERY_CACHE_TYPE = OFF | ON | DEMAND QUERY_CACHE_TYPE = 0 | 1 | 2
```

| Option | Description |
|---|---|

| | |
|---|---|
| 0 ou OFF | Ne pas utiliser le cache. |
| 1 ou ON | mettre en cache tous les résultats à part les requêtes SELECT SQL_NO_CACHE ... . |
| 2 ou DEMAND | mettre en cache seulement les résultat des requêtes SELECT SQL_CACHE ... . |

## 6.9.3 Options relatives au cache de requêtes dans un SELECT

Il y a deux options relatives au cache de requêtes qui peuvent être utilisées dans une requête `SELECT` :

| Option | Description |
|---|---|
| SQL_CACHE | Si QUERY_CACHE_TYPE est DEMAND , permet à la requête d'être mise en cache. Si QUERY_CACHE_TYPE est ON , ceci est la valeur par défaut. Si QUERY_CACHE_TYPE est OFF , ne fait rien. |
| SQL_NO_CACHE | Rend cette requête non-cachable, ne permet pas à cette requête d'être mise en cache. |

## 6.9.4 Status du cache de requêtes et maintenance

Avec la commande `FLUSH QUERY CACHE` vous pouvez défragmenter le cache de requêtes pour mieux en utiliser la mémoire. Cette commande n'effacera aucune requête du cache. `FLUSH TABLES` défragmente aussi le cache de requêtes.

La commande `RESET QUERY CACHE` efface tous les résultats de requêtes du cache.

Vous pouvez visualiser les performances du cache de requêtes avec `SHOW STATUS` :

| Variable | Description |
|---|---|
| Qcache_queries_in_cache | Nombre de requêtes mises en cache. |
| Qcache_inserts | Nombre de requêtes ajoutées au cache. |
| Qcache_hits | Nombre d'appel au cache. |
| Qcache_lowmem_prunes | Nombre de requêtes ôtées du cache pour cause de mémoire insuffisante. |
| Qcache_not_cached | Nombre de requêtes qui n'ont pas été mises en cache. (not cachable, or due to QUERY_CACHE_TYPE ). |
| Qcache_free_memory | Quantité de mémoire libre pour le cache de requêtes. |
| Qcache_total_blocks | Nombre total de blocs dans le cache de requêtes. |
| Qcache_free_blocks | Nombre de blocs à mémoire libre dans le cache de requêtes. |

Nombre total de requêtes = `Qcache_inserts` + `Qcache_hits` + `Qcache_not_cached` .

Le cache de requêtes utilise des blocs de longueur variable, ce qui fait que `Qcache_total_blocks` et `Qcache_free_blocks` peuvent indiquer une fragmentation de la mémoire du cache. Après un appel à `FLUSH QUERY CACHE` un seul (grand) bloc libre subsiste.

Note : Chaque requête a besoin au minimum de deux blocs (un pour le texte de la requête et un autre, ou plus, pour le résultat). De même, chaque table utilisée par une requête a besoin d'un bloc, mais si deux ou plusieurs requêtes utilisent la même table, seul un bloc a besoin d'être alloué.

Vous pouvez aussi utiliser la variable `Qcache_lowmem_prunes` pour ajuster la taille du cache de requêtes.

# 7 Types de tables MySQL

Depuis la version 3.23.6 de MySQL, vous pouvez choisir entre trois formats basique de tables ( `ISAM` , `HEAP` et `MyISAM` ). Les nouvelles versions de MySQL peuvent supporter d'autres types de tables ( `InnoDB` , ou `BDB` ), cela dépend de comment vous l'avez compilé.

Lorsque vous créez une nouvelle table,vous pouvez dire à MySQL quel type de table il doit utiliser pour celle−ci. MySQL créera toujours un fichier `.frm` pour stocker les définitions de la table et des colonnes. Selon le type de table, les index et les données seront stockés dans d'autres fichiers.

Notez que pour utiliser les tables `InnoDB` vous devez au moins utiliser l'option de démarrage `innodb_data_file_path` . Options de démarrage `InnoDB` .

Le type de table par défaut de MySQL est `MyISAM` . Si vous essayez d'utiliser un type de table qui n'est pas compilée ou activée, MySQL créera à la place une table de type `MyISAM` . C'est une fonctionnalité très utile quand vous voulez copier des tables entre différents serveurs SQL qui ne supportent pas les mêmes types de tables (comme copier des tables vers un esclave qui est optimisé pour la vitesse en ne supportant pas les tables transactionnelles). Ce changement de table automatique peut toutefois induire en erreur les nouveaux utilisateurs de MySQL. Nous allons introduire des messages d'avertissement dans MySQL 4.0 et les afficher lors des transtypages automatiques des tables.

Vous pouvez changer les types de tables en utilisant la commande `ALTER TABLE` . Syntaxe de `ALTER TABLE` .

Notez que MySQL supporte deux différents types de tables : tables transactionnelles ( `InnoDB` et `BDB` ) et tables non−transactionnelles ( `HEAP` , `ISAM` , `MERGE` , et `MyISAM` ).

Les avantages des tables transactionnelles (TST) sont :

- Plus sûr. Même si MySQL crashe ou que vous avez un problème matériel, vous pouvez récupérer vos données, soit par un recouvrement automatique, soit à partir d'une sauvegarde combinée avec le log des transactions.
- Vous pouvez combiner plusieurs commandes et les accepter toutes d'un seul coup avec la commande `COMMIT` .
- Vous pouvez utiliser `ROLLBACK` pour ignorer vos modifications (si vous n'êtes pas en mode auto−commit).
- Si une mise à jour échoue, tout vos changements seront annulés. (Avec les tables NTST tous les changements opérés sont permanants)
- Gère mieux les accès concurrents si la table recoit simultanément plusieurs lectures.

Avantages des tables non−transactionnelles (NTST) :

- Plus rapides puisqu'il n'y a pas de traitement des transactions.
- Utilisent moins d'espace disque puisqu'il n'y a pas de traitement des transavctions.
- Utilisent moins de mémoires pour les mises à jour.

Vous pouvez combiner les tables TST et NTST dans la même requête pour obtenir le meilleur des deux types.

## 7.1 Tables MyISAM

MyISAM est le type par défaut de table en MySQL version 3.23. Il est basé sur ISAM et ajoute de nombreuses extensions pratiques.

L'index est stocké dans un fichier avec l'extension .MYI (MYIndex), et les données sont stockées dans un fichier avec l'extension .MYD (MYData). Vous pouvez vérifier et réparer les tables MyISAM avec myisamchk . Utilisation de myisamchk pour résoudre un crash . Vous pouvez compresser les tables MyISAM avec myisampack , pour gagner de l'espace disque. myisampack , le générateur de tables MySQL compressées en lecture seule .

Voici les nouveautés des tables MyISAM :

- Il y a un sémaphore dans le fichier MyISAM qui indique si la table a été correctement fermée. Si mysqld est lancé avec l'option --myisam-recover , les tables MyISAM vont automatiquement être vérifiées et réparées, si elles n'ont pas été correctement refermées.
- Vous pouvez insérer de nouvelles lignes dans une table qui n'a aucun bloc vide dans le fichier de données, en même temps que d'autres threads lisent le fichier de données (insertion simultanée). Un bloc vide peut provenir d'une modification de ligne à format dynamique (les données sont maintenant plus petites). Lorsque tous les blocs vide sont à nouveau utilisés, les insertions suivantes peuvent être simultanées.
- Support des grands fichiers (63 bits) sur les systèmes de fichiers et les systèmes d'exploitation qui supportent les grands fichiers.
- Toutes les données sont stockées avec l'octet de poids faible en premier. Cela rend les données indépendantes de la machine et du système d'exploitation. La seule condition est que la machine utilise des entiers complétés à deux (comme toutes les machines depuis les 20 dernières années), et le format IEEE pour les nombres à virgule flottante (aussi répandu dans les machine mainstream). Le seul domaine où les machines ne supporteront pas la compatibilité binaire sont les systèmes embarqués (car ils ont parfois des processeurs spéciaux).Il n'est pas pénalisant de stocker les données avec l'octet de poids faible en premier. Les octets d'une ligne de table sont généralement non alignés, et cela ne prend pas beaucoup de ressources de lire des octets non alignés en ordre ou en ordre inverse. Le code qui lit les données dans une colonne n'est pas gourmand en temps, comparé à d'autres opérations.
- Toutes les clés numériques sont stockées avec l'octet de poids fort en premier, pour améliorer la compression.
- La gestion interne des colonnes AUTO_INCREMENT . MyISAM va automatiquement modifier cette valeur lors d'une insertion ou d'une modification. La valeur courante d' AUTO_INCREMENT peut être modifiée avec myisamchk . Cela va rendre les colonnes AUTO_INCREMENT plus rapide (au moins 10%) et les anciens nombres ne seront pas réutilisés, comme avec les vieilles tables ISAM . Notez que lorsque une clé AUTO_INCREMENT est définie à la fin d'une clé multiple, le vieux comportement est toujours présent.
- Lorsqu'insérée dans un ordre trié (comme lorsque vous utilisez une colonne de type AUTO_INCREMENT ), l'arbre des clé sera scindé, pour que noeud principal ne contienne qu'une clé. Cela va améliorer l'utilisation d'espace dans l'arbre des clés.
- BLOB et TEXT peuvent être indexés.
- Les valeurs NULL sont autorisées dans une colonne indexée. Elles prennent 0 à 1 octets par clé.
- La taille maximale d'une clé est de 500 octets par défaut (cela peut être modifié en recompilant). Dans le cas des clés plus grandes que 250 octets, un bloc de clé plus grand que le bloc de 1024 octets est utilisé.
- Le nombre maximal de clé par table est par défaut de 32. Cela peut être agrandi à 64, sans recompiler, avec myisamchk .
- myisamchk va marquer les tables comme vérifiée si il est exécuté avec l'option --update-state . myisamchk --fast va uniquement vérifier les tables qui n'ont pas cette marque.
- myisamchk -a stocke les statistique pour les parties de clés (et non plus pour les clés complètes,

comme en ISAM ).
- Des lignes à format dynamique seront bien moins fragmentée lorsque vous mêlez des insertions et des modifications. Cela est fait automatiquement en combinant des blocs effacés adjacent, et en étendant ces blocs au prochain bloc vide.
- myisampack peut compresser des colonnes BLOB et VARCHAR .
- Vous pouvez mettre les fichiers d'index et de données dans différents dossiers pour gagner de la vitesse (avec l'option DATA/INDEX DIRECTORY="path" de CREATE TABLE ). Syntaxe de CREATE TABLE .

MyISAM supporte aussi les fonctionnalités suivantes, dont MySQL pourra profiter sous peu :

- Support du vrai type VARCHAR ; une colonne VARCHAR commence avec une taille, stockée sur 2 octets.
- Les tables ayant des colonnes VARCHAR peuvent avoir un format de lignes fixe ou dynamique.
- VARCHAR et CHAR peuvent prendre jusqu'à 64ko. Tous les segments de clé ont leur propre définition de langue. Cela permettra à MySQL d'avoir des définitions de langue différentes pour chaque colonne.
- Un index de hachage peut être utilisé pour les index de type UNIQUE . Cela vous permettra d'avoir l'attribut UNIQUE sur n'importe quelle combinaison de colonne de votre table (vous ne pouvez pas faire de recherche sur un tel index UNIQUE ).

Notez que les fichiers d'index sont généralement plus plus petit avec les tables MyISAM qu'avec les tables ISAM . Cela signifie que MyISAM va normalement utiliser moins de ressources systèmes que ISAM , mais aura besoin de plus de processeur lors de l'insertion dans un index compressé.Les options suivantes de mysqld peuvent être utilisée pour modifier le comportement des tables MyISAM . SHOW VARIABLES .

| Option | Description |
|---|---|
| --myisam-recover=# | Réparation automatique des tables crashées. |
| -O myisam_sort_buffer_size=# | Buffer utilisé lors de la réparation des tables. |
| --delay-key-write=ALL | Ne pas écrire les buffers de clés entre deux écriture dans une table MyISAM . |
| -O myisam_max_extra_sort_file_size=# | Utilisé pour aider MySQL à décider quand utiliser la méthode du cache d'index, lente mais sûre. **Notez** que ce paramètre était donné en méga-octets avant la version 4.0.3 et en octets depuis cette version. |
| -O myisam_max_sort_file_size=# | Ne pas utiliser la méthode de tri rapide pour créer l'index, si le fichier temporaire dépasse la taille indiquée ici. **Notez** que ce paramètre était donné en méga-octets avant la version 4.0.3 et en octets depuis cette version. |
| -O bulk_insert_buffer_size=# | Taille du cache d'arbre lors d'insertion massives. **Notez** que c'est une limite **par thread** ! |

La réparation automatique est activée si vous démarrez mysqld avec l'option --myisam-recover=# . Options en ligne de commande de mysqld . Lors de l'ouverture, la marque de fermeture de la table est vérifiée, ou le compteur d'ouverture de la table n'est pas 0 et vous fonctionnez avec l'option --skip-external-locking . Si l'un des deux ci-dessus est vrai, la procédure suivante s'applique :

- La table est vérifiée.
- Si une erreur est trouvée, une réparation rapide est tentée (avec tri et sans recréation du fichier de données).
- Si la réparation rapide échoue à cause d'une erreur dans le fichier de données, (par exemple, une erreur de clé dupliquée), nous essayons à nouveau, mais cette fois ci, avec recréation du fichier de données.
- Si cette réparation échoue, on essaie encore une fois avec la méthode traditionnelle de réparation (écriture des lignes une à une, sans tri), qui doit réparer la table de toutes ses erreurs, avec peu d'espace disque...

Si la réparation n'a pas été capable de retrouver toutes les lignes d'une requête précédente, et que vous ne spécifiez pas l'option FORCE à myisam-recover , alors la correction automatique va s'arrêter avec l'erreur suivante dans le fichier d'erreurs :

```
Error: Couldn't repair table: test.g00pages
```

Dans ce cas, si vous utilisez l'option FORCE , vous obtiendrez le message suivant dans le fichier d'erreurs :

```
Warning: Found 344 of 354 rows when repairing ./test/g00pages
```

Notez que si vous exécutez une restauration automatique avec une option automatique de `BACKUP` , vous devriez avoir un script `cron` qui va automatiquement copier les fichiers avec l'extension `tablename-datetime.BAK` depuis le dossier de données vers une solution de sauvegarde. Options en ligne de commande de `mysqld` .

## 7.1.1 Espace requis pour les clefs

MySQL supporte plusieurs types d'index, mais le type normal est ISAM ou MyISAM. Ils utilisent un index B−tree, et vous pouvez avoir une approximation de la taille du fichier d'index en faisant la somme de `(longueur_clef+4)/0.67` pour toutes les clefs. (Cela est le pire des cas où les clefs sont insérées dans l'ordre et qu'aucune n'est compressée.

Les index de chaînes de caractères sont compressés par rapport aux espaces. Si la première partie de l'index est une chaîne, son préfixe sera aussi compressé. La compression des espaces rend le fichier d'index plus petit que ce que nous avions calculé précédement si la colonne chaîne possède beaucoup d'espaces invisibles en début et fin de chaîne ou est une colonne `VARCHAR` qui n'est pas toujours pleinement utilisée. La compression des préfixes est utilisée sur les clefs qui commencent par un chaîne de caractères. La compression des préfixes s'il y a plusieurs chaînes avec des préfixes identiques.

Dans les tables `MyISAM` , vous pouvez aussi compresser les nombres en spécifiant `PACK_KEYS=1` lors de la création de la table. Cela vous aidera lorsque vous aurez plusieurs clefs de types entier qui auront un préfixe identique et que les nombres seront classé par ordre décroissant des grands octets.

## 7.1.2 Formats de table MyISAM

`MyISAM` supporte 3 différent types de tables. Deux des trois sont choisis automatiquement selon le type de colonne que vous utilisez. Le troisième, tables compressées, ne peut être crée qu'avec l'outil `myisampack` .

Quand vous créez une table avec `CREATE` ou en modifiez la structure avec `ALTER` vous pouvez, pour les tables n'ayant pas de champs `BLOB` forcer le type de table en `DYNAMIC` ou `FIXED` avec l'option `ROW_FORMAT=#` des tables. Bientôt, vous pourrez compresser/décompresser les tables en spécifiant `ROW_FORMAT=compressed | default` à `ALTER TABLE` . Syntaxe de `CREATE TABLE` .

### 7.1.2.1 Caractéristiques des tables statiques (taille fixée)

Ceci est le format par défaut. Il est utilisé lorsque la table ne contient pas de colonnes de type `VARCHAR` , `BLOB` , ou `TEXT` .

Ce format est le plus simple et le plus sûr. C'est aussi le format sur disque le plus rapide. La vitesse vient de la facilité avec laquelle les données peuvent être trouvées sur le disque. La recherche de quelque chose avec un index et un format statique est très simple. Multipliez juste le nombre de lignes par la longueur des lignes.

De même, lors du scannage d'une table, il est très facile de lire un nombre constant d'enregistrements avec chaque lecture du disque.

La sécurité est mise en évidence si votre ordinateur crashe lors de l'écriture dans un fichier de taille fixée MyISAM, dans ce cas, `myisamchk` peur facilement trouver où commence et finit chaque ligne. Il

peut donc retrouver tous les enregistrements à part celui dont l'écriture a été interrompue. Notez qu'avec MySQL tous les index peuvent toujours être reconstruits :

- Toutes les colonnes `CHAR` , `NUMERIC` , et `DECIMAL` sont complétées par des espaces jusqu'à atteindre la longueur totale de la colonne.
- Très rapide.
- Facile à mettre en cache.
- Facile à reconstruire après un crash, car les enregistrements sont localisés dans des positions fixées.
- N'a pas à être réorganisé (avec `myisamchk` ) sauf si un grand nombre de lignes est effacé et que vous voulez retourner l'espace libéré au système d'exploitation.
- Requière usuellement plus d'espace disque que les tables dynamiques.

#### 7.1.2.2 Caractéristiques des tables à format de ligne dynamiques

Ce format est utilisé avec les tables qui contiennent des colonnes de type `VARCHAR` , `BLOB` ou `TEXT` , ou si la table a été créée avec l'option `ROW_FORMAT=dynamic` .

Ce format est un peu plus complexe, car chaque ligne doit avoir un entête pour indiquer sa longueur. Une ligne peut aussi être répartie sur plusieurs blocs, lorsqu'elle est agrandie lors d'une modification.

Vous pouvez utiliser la commande SQL `OPTIMIZE table` ou shell `myisamchk` pour défragmenter une table. Si vous avez des données statiques que vous modifiez souvent dans la même table, avec des colonnes `VARCHAR` ou `BLOB` , il peut être une bonne idée de placer des colonnes dans une autre table, pour éviter la fragmentation :

- Toutes les colonnes de type chaîne sont dynamiques (hormis celle qui sont de taille inférieure à 4).
- Chaque ligne est précédée d'un octet qui indique quelles sont les lignes vides ( `''` , bit à 1) et celle qui le ne sont pas (bit à 0). Une colonne vide n'est pas la même choses qu'une colonne qui contient `NULL` . Si une colonne a une taille de zéro après avoir supprimé les espaces finaux, ou un nombre a une valeur de zéro, il est marqué dans cet octet, et la colonne sera ignorée sur le disque. Les chaînes non vides sont sauvées avec un octet de plus pour y stocker la taille.
- Ce format prend généralement moins de place que des tables à format fixe.
- Chaque ligne consomme autant d'espace que nécessaire. Si une ligne devient trop grande, elle sera coupée en blocs et écrites dans le fichier de données. Cela engendre la fragmentation du fichier de données.
- Si vous modifiez une ligne avec des informations qui excèdent la capacité courante de la ligne, la ligne sera fragmentée. Dans ce cas, vous pouvez avoir à exécuter la commande `myisamchk -r` de temps en temps pour améliorer les performances. Utilisez `myisamchk -ei tbl_name` pour obtenir des statistiques.
- Ce format de table n'est pas toujours facile à reconstituer après un crash, car une ligne peut êtr fragmentée en de nombreux blocs, et un fragment peut manquer.
- La taille d'une ligne de format variable se calcule avec :

```
 3
+ (nombre de colonnes + 7) / 8
+ (nombre de colonnes de tailles chars)
+ taille compactée des colonnes numériques
+ taille des chaînes
+ (nombre de colonne de valeur NULL + 7) / 8
```

  Il y a un aussi un supplément de 6 octets pour chaque lien. Une ligne de format dynamique utilise un lien à chaque fois qu'une modification cause un agrandissement de la ligne. Chaque nouveau bloc lié fait au moins 20 octets, pour que le prochain agrandissement utilise aussi ce bloc. Si ce n'est pas le cas, un nouveau bloc sera lié, avec un autre coût de 6

octets. Vous pouvez vérifier le nombre de liens dans une table avec la commande `myisamchk -ed` . Tous les liens sont supprimés avec la commande `myisamchk -r` .

#### 7.1.2.3 Caractéristiques des tables compressées

C'est un type en lecture seule qui est généré avec l'outil optionnel `myisampack` ( `pack_isam` pour les tables `ISAM` ) :

- Toutes les distributions MySQL, même celles qui existaient avant que MySQL ne passe sous la licence GPL, peuvent lire des tables qui ont été compréssées avec `myisampack` .
- Les tables compréssées prennent très peu d'espace disque. Cela amoindri l'espace requis ce qui est fort utile lors de l'utilisation de petits disques (comme les CD−ROM).
- Chaque ligne est compressée séparement (optimisation des accès). L'entête d'un enregistrement est fixé (1−3 octets) selon le plus grand enregistrement dans la table. Chaque colonne est compressée différement. Quelques un des types de compressions sont :
    - Il y a usuellement une table Huffman différente pour chaque colonne.
    - Compression des espaces en suffixe.
    - Compression des espaces en préfixe.
    - Les nombres avec la valeur `0` sont stockés en utilisant 1 octet.
    - Si les valeurs dans une colonne de type entier ont un petit intervalle, la colonne est stockée en utilisant le type le plus petit possible. Par exemple, une colonne `BIGINT` (8 octets) peut être stocké en tant que colonne `TINYINT` (1 octet) si toutes les valeurs sont entre `0` et `255` .
    - Si une colonne n'a qu'un petit éventail de valeurs, son type est changé en `ENUM` .
    - Une colonne peut utiliser une combinaison des compressions précédentes.
- Peut gérer les enregistrements de tailles fixes ou variables.
- Peut être décompressée avec `myisamchk` .

### 7.1.3 Problèmes avec les tables MyISAM

Le format de fichier que MySQL utilise pour stocker les données a été testé à l'extrême, mais il y a toujours des circonstances qui peuvent corrompre les tables d'une base de données.

#### 7.1.3.1 Tables MyISAM corrompues

Même si le format des tables MyISAM est relativement sûr (tous les changements sont écrits avant que la requête SQL ne retourne quoi que ce soit), vous pouvez quand même vous trouver face à des tables corrompues si l'une des choses suivantes arrive :

- Le processus `mysqld` est tué au milieu d'une écriture.
- Arrêt inattendu de la machine (par exemple, coupure de courant).
- Un problème matériel.
- Vous utilisez un programme externe (comme myisamchk) sur une table active.
- Un bogue logiciel dans le code de MySQL ou de MyISAM.

Les symptômes typiques d'une table corrompue sont :

- Vous obtenez l'erreur `Incorrect key file for table: '...'. Try to repair it` pendant la sélection de données à partir de cette table.
- Les requêtes ne trouvent pas de lignes dans la table ou retournent des données

incomplètes.

Vous pouvez vérifier l'état d'une table avec la commande `CHECK TABLE` . Syntaxe de `CHECK TABLE` .

Vous pouvez réparer une table corrompue avec `REPAIR TABLE` . Syntaxe de `REPAIR TABLE` . Vous pouvez aussi réparer une table, lorsque `mysqld` ne fonctionne pas, avec la commande `myisamchk` . `myisamchk syntax` .

Si vos tables sont souvent corrompues, vous devez essayez de trouver d'où vient le problème ! Que faire si MySQL crashe constamment .

Dans ce cas, la chose la plus importante à savoir est, si la table est corrompue, si le serveur `mysqld` s'est interrompu. (cela peut être facilement vérifié en regardant s'il y a une entrée récente `restarted` `mysqld` dans le fichier d'erreurs de mysqld). Si ce n'est pas le cas, vous devez essayer d'effectuer une série de tests. Faire une batterie de tests lorsque vous faites face à un problème de table corrompue .

#### 7.1.3.2 Clients is using or hasn't closed the table properly

Chaque fichier `MyISAM .MYI` possède un compteur dans l'entête qui peut être utilisé pour savoir si une table a été fermée Proprement.

Si vous obtenez l'avertissement suivant de la part de `CHECK TABLE` ou `myisamchk` :

```
# clients is using or hasn't closed the table properly
```

cela signifie que le compteur n'est plus synchrone. Cela ne signifie Pas que la table est corrompue, mais que vous devez au moins effectuer une vérification sur la table pour vous assurer de son bon fonctionnement.Le compteur fonctionne de la façon suivante :

- La première fois qu'une table est mise à jour dans MySQL, un compteur dans l'entête du fichier est incrémenté.
- Le compteur ne change pas pour les mises à jours suivantes.
- Lors de la fermeture de la dernière instance d'une table (à cause d'un `FLUSH` ou qu'il n'y a plus de place dans le cache de la table) le compteur est decrémenté si la table n'a pas été mise à jour.
- Lorsque vous réparez la table ou vérifiez quel est en bon état, le compteur est remis à zéro.
- Pour éviter les problèmes d'interactions avec d'autres processus qui peuvent vérifier la table, le compteur n'est pas décrémenté à la fermeture si sa valeur était zéro.

En d'autres termes, les seuls moyens d'obtenir ce genre d'erreur sont :

- Les tables `MyISAM` sont copiés sans `LOCK` et `FLUSH TABLES` .
- MySQL a planté entre une mise à jour et la fermeture finale. (Notez que la table peut encore être bonne, vu que MySQL écrit toujours pour tout entre deux requêtes.)
- quelqu'un a exécuté `myisamchk --recover` ou `myisamchk --update-state` sur une table qui était utilisée par `mysqld` .
- Plusieurs serveurs `mysqld` utilisent la table et l'un d'eux a exécuté dessus un `REPAIR` ou un `CHECK` pendant qu'elle était utilisée par un autre serveur. Dans ce cas là, l'utilisation de `CHECK` n'est pas très grave (même si vous obtiendrez des avertissements sur les autres serveurs), mais `REPAIR` doit être évitée vu qu'elle remplace actuellement le fichier de données part un nouveau, ce qui n'est pas signalé aux autres serveurs.

# 7.2 Tables assemblées MERGE

Les tables `MERGE` sont nouvelles depuis MySQL version 3.23.25. Le code est toujours en phase gamma, mais il est déjà raisonnablement stable.

Une table `MERGE` (aussi connue sous le nom de `MRG_MyISAM` , ou table assemblée) est un regroupement de tables `MyISAM` identiques, qui peuvent être utilisées ensemble, comme une seule. Nous ne pouvez faire que des commandes `SELECT` , `DELETE` et `UPDATE` dans ces tables. Si vous effacez (avec `DROP` ) la table `MERGE` , vous ne faîtes qu'annuler le rassemblement `MERGE` .

Notez que la commande `DELETE FROM merge_table` utilisée sans la clause `WHERE` va seulement effacer le rassemblement de tables, et non pas les lignes dans les tables. Nous envisageons de corriger cela en version 4.1.

Avec des tables identiques, nous voulons dire que toutes les tables sont créées avec les mêmes colonnes et index. Vous ne pouvez pas rassembler des tables dans lesquelles les colonnes sont définies différemment, n'ont pas exactement le même nombre de colonne, ou ont des index dans un autre ordre. Cependant, certaines des tables peuvent être compressées avec `myisampack` . `myisampack` , le générateur de tables MySQL compressées en lecture seule .

Lorsque vous créez une table `MERGE` , vous allez obtenir un fichier `.frm` , de définition de table, et un fichier de liste de tables `.MRG` . Le fichier `.MRG` contient simplement la liste des fichiers d'index (extension `.MYI` ) qui doivent être utilisés comme un seul et même. Toutes les tables utilisées doivent être dans la même base que la table `MERGE` elle−même.

Pour le moment, vous avez simplement besoin des droits de `SELECT` , `UPDATE` et `DELETE` sur les tables que vous avez rassemblé dans la table `MERGE` .

Les tables `MERGE` peuvent vous aider dans les situations suivantes :

- Gérer facilement un jeu de table de log. Par exemple, vous pourriez placer les données de chaque mois dans un fichier séparé, en compresser certains avec `myisampack` puis créer une table `MERGE` pour les utiliser.
- Vous donner plus de vitesse. Vous pouvez répartir les grandes tables en lecture seule dans différentes parties du disque. Une table `MERGE` bâtie de cette façon peut être plus rapide qu'une grosse table (vous pouvez aussi et bien sûr, utiliser un système RAID pour arriver aux mêmes avantages).
- Effectuer des recherches plus efficaces. Si vous savez exactement ce que vous recherchez, vous pouvez faire des recherches dans une seule des tables individuelles pour les recherches, et utiliser la table `MERGE` pour les autres opérations. Vous pouvez même avoir de nombreuses tables `MERGE` actives, qui partagent les même fichiers.
- Des réparations plus efficaces. Il est plus facile de réparer les fichiers individuels qui sont rassemblés dans une table `MERGE` que de réparer une grande table.
- Fusion instantanée de plusieurs fichiers en un seul. Une table `MERGE` utilise les index des tables individuelles. Il n'y a pas besoin de gérer un seul index. Cela rend les tables `MERGE` très rapides à faire ou défaire. Notez que vous devez spécifier les définitions de clés lorsque vous créez la table `MERGE` !
- Si vous avez un jeu de table que vous rassemblez dans une grande à la demande ou pour un traitement batch, vous devriez utiliser une table `MERGE` . C'est bien plus rapide, et cela va vous faire économiser de l'espace disque.
- Contourner les limitations de taille du système d'exploitation.
- Vous pouvez créer un alias ou un synonyme pour une table, en utilisant simplement `MERGE` sur une seule. Il n'y a pas de coûts particulier en performance (hormis quelques appels de fonctions indirects, et des `memcpy()` avant chaque lecture).

Les inconvénients des tables de type MERGE sont :

- Vous devez utiliser des tables MyISAM identiques pour faire une table MERGE .
- REPLACE ne fonctionne pas.
- MERGE utilise plus de pointeurs de fichiers. Si vous utilisez une table MERGE qui couvre 10 tables et que 10 utilisateurs l'utilisent, vous consommez 10*10 + 10 pointeurs de fichiers (10 fichiers de données, et 10 utilisateurs avec 10 fichiers d'index).
- Les lectures de clés sont plus lente. Lorsque vous faîtes une lecture sur une clé, le gestionnaire MERGE doit faire une lecture dans tous les fichiers d'index des tables sous−jacentes, pour vérifier lequel est le plus proche de la valeur recherchée. Si vous faîtes une lecture du type "lit le suivant", le gestionnaire de table assemblée doit rechercher dans tous les buffers de clés pour la trouver. Uniquement lorsqu'un buffer clé est complet, doit il lire le prochain bloc. Cela rend l'accès aux clés MERGE bien plus lent que les recherches eq_ref , mais pas aussi lent que les recherches de type ref . Syntaxe de EXPLAIN (Obtenir des informations SELECT ) .
- Vous ne pouvez pas faire de commande DROP TABLE , ALTER TABLE , DELETE FROM table_name sans clause WHERE , REPAIR TABLE , TRUNCATE TABLE , OPTIMIZE TABLE ou ANALYZE TABLE sur aucune des tables qui sont couvertes par une table MERGE qui est "ouverte". Si vous faîtes cela, la table MERGE peut toujours pointeur sur la table originale, et vous allez obtenir des résultats inattendus. Le plus simple palliatif est d'utiliser la commande FLUSH TABLES , pour s'assurer qu'aucune des tables MERGE ne reste ouverte.

Lorsque vous créez une table MERGE , vous devez spécifier la liste des tables que vous allez utiliser, avec l'option UNION(list−of−tables) . Optionellement, grâce à l'option INSERT_METHOD , vous pouvez spécifier si vous voulez que la table MERGE ajoute les nouvelles lignes dans la première ou la dernière table de l' UNION . Si vous spécifiez pas INSERT_METHOD ou si vous spécifiez NO , alors les commandes INSERT donnée à la table MERGE retourneront une erreur.L'exemple suivant vous montre comme utiliser les tables MERGE :

```
 CREATE TABLE t1 (a INT AUTO_INCREMENT PRIMARY KEY, message CHAR(20));
CREATE TABLE t2 (a INT AUTO_INCREMENT PRIMARY KEY, message CHAR(20));
INSERT INTO t1 (message) VALUES ("Testing"),("table"),("t1");
INSERT INTO t2 (message) VALUES ("Testing"),("table"),("t2");
CREATE TABLE total (a INT AUTO_INCREMENT PRIMARY KEY, message CHAR(20))
            TYPE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
```

Notez que vous pouvez aussi manipuler le fichier .MRG directement, hors du serveur MySQL :

```
 shell> cd /mysql−data−directory/current−database
shell> ls −1 t1.MYI t2.MYI > total.MRG
shell> mysqladmin flush−tables
```

Maintenant, vous pouvez faire des commandes comme :

```
 mysql> SELECT * FROM total;
+---+---------+
| a | message |
+---+---------+
| 1 | Testing |
| 2 | table   |
| 3 | t1      |
| 1 | Testing |
| 2 | table   |
| 3 | t2      |
+---+---------+
```

Notez que la colonne a , bien que déclarée PRIMARY KEY , n'est pas vraiment unique, car les tables MERGE ne peuvent pas garantir l'unicité entre les tables sous −jacente à la MyISAM .

Pour redéfinir une table MERGE , vous pouvez faire ceci :

- DROP la table, puis recréez la.
- Utilisez ALTER TABLE table_name UNION(...)
- Modifiez le fichier .MRG et utilisez la commande FLUSH TABLE sur la table MERGE et toutes les tables sous−jacentes, pour forcer le gestionnaire à relire la définition.

### 7.2.1 Problèmes avec les tables MERGE

Voici une liste des problèmes connus avec les tables de type `MERGE` :

- Une table `MERGE` peut pas pas supporter de contrainte de type `UNIQUE` sur toute la table. Lorsque vous faites une insertion, les données vont dans la première ou la dernière table (suivant la méthode d'insertion `INSERT_METHOD=xxx` ) et cette table `MyISAM` s'assure que les données sont uniques, mais rien n'est fait pour vérifier l'unicité auprès des autres tables `MyISAM` tables.
- `DELETE FROM merge_table` utilisé sans clause `WHERE` va uniquement détruire la table assemblée, mais ne va pas toucher les tables sous−jacentes.
- `RENAME TABLE` utilisé sur une table de type `MERGE` peut corrompre la table. Cela sera corrigé en MySQL 4.0.x.
- La création d'une table de type `MERGE` ne vérifie pas si les tables sous−jacentes sont compatibles. Si vous utilisez une table `MERGE` de cette façon, vous devriez rencontrer des problèmes très étranges.
- Si vous utilisez la commande `ALTER TABLE` pour ajouter un index de type `UNIQUE` à une table qui est utilisée dans une table assemblée `MERGE` , puis que vous utilisez `ALTER TABLE` pour ajouter un index normal dans la table `MERGE` , l'ordre des clé sera différent suivant les tables, si jamais il y avait une vieille clé non unique. Ceci est dû au fait que `ALTER TABLE` place les clés `UNIQUE` avant les clés normales, pour être capable de détecter les doublons le plus tôt possible.
- L'optimiseur d'intervalle ne peut pas encore utiliser les tables `MERGE` efficacement, et il produit parfois des jointures sub−optimales. Cela sera corrigé en MySQL 4.0.x.
- `DROP TABLE` sur une table qui est utilisé par une table `MERGE` ne fonctionne pas sous Windows car le gestionnaire de `MERGE` garde les connexions vers les tables cachées sous la couche MySQL. Comme Windows ne vous permet pas d'effacer une table qui est ouverte, vous devez d'abord fermer toute les tables `MERGE` (avec la commande `FLUSH TABLES` ) ou effacer la table `MERGE` avant de pouvoir effacer la table désirée. Nous allons corriger lorsque nous introduirons les vues. `VIEW` s.

## 7.3 Tables ISAM

Vous pouvez aussi utiliser le type de tables désapprouvé `ISAM` . Il disparaîtra bientôt (probablement dans MySQL 5.0) car `MyISAM` est une meilleure implémentation de la même chose. `ISAM` utilise un index `B−tree` . L'index est stocké dans un fichier portant l'extension `.ISM` , et les données sont enregistrées dans un fichier avec l'extension `.ISD` . Vous pouvez vérifier/réparer les tables `ISAM` avec l'utilitaire `isamchk` . Utilisation de `myisamchk` pour résoudre un crash .

`ISAM` possède les fonctionnalités/propriétés suivantes :

- Clefs compressées et de tailles fixes
- Enregistrements de taille fixée ou dynamique
- 16 clefs avec 16 parties de clefs/clefs
- Taille maximale de la clef 256 (défaut)
- Les données sont enregistrées au format machine; c'est rapide, mais c'est dépendant de la machine/système d'exploitation.

La plupart des choses vraies pour les tables `MyISAM` le sont pour les tables `ISAM` . Tables `MyISAM` . La différence majeure comparées aux tables `MyISAM` sont :

- Les tables `ISAM` ne sont pas binairement portables entre les plate−formes/systèmes d'exploitations.
- Ne peut gérer les tables > 4G.
- Ne supporte que la compression des préfixes sur les chaînes de caractères.

- Limites de clefs plus petites.
- Les tables dynamiques sont plus fragmentées.
- Les tables sont compressées avec `pack_isam` plutôt qu'avec `myisampack` .

Si vous voulez convertir vos tables `ISAM` en `MyISAM` pour pouvoir utiliser des utilitaires tels que `mysqlcheck` , utilisez la commande `ALTER TABLE` :

```
mysql> ALTER TABLE nom_de_table TYPE = MYISAM;
```

Les versions embarquées de MySQL ne supportent pas les tables `ISAM` .

# 7.4 Tables HEAP

Les tables `HEAP` utilisent un index de hachage, et sont stockées en mémoire. Elles sont très rapides, mais si MySQL crashe, vous perdrez toutes vos données. Les tables `HEAP` sont très pratiques pour être des tables temporaires!

Les tables `HEAP` internes MySQL utilisent des hachages 100% dynamiques sans zones de débordement. Il n'y a pas d'espace nécessaire pour des listes libres. Les tables `HEAP` n'ont aussi aucun problème d'effacement et d'insertions, qui sont le lot commun des tables de hachage :

```
 mysql> CREATE TABLE test TYPE=HEAP SELECT ip,SUM(downloads) AS down
    ->                   FROM log_table GROUP BY ip;
mysql> SELECT COUNT(ip),AVG(down) FROM test;
mysql> DROP TABLE test;
```

Voici quelques conseils concernant les tables `HEAP` :

- Vous devez toujours penser à spécifier le nombre maximal de lignes de la table avec `MAX_ROWS` dans la commande de création `CREATE` pour vous assurer de ne pas accidentellement consommer toute la mémoire.
- Les index ne peuvent être utilisés qu'avec les opérateurs `=` et `<=>` (mais ils sont très rapides).
- Les tables `HEAP` ne peuvent utiliser que des clés entières pour rechercher une ligne. Par comparaisons, les tables `MyISAM` peuvent utiliser un préfixe de clé pour trouver des lignes.
- Les tables `HEAP` utilisent un format de ligne fixe.
- `HEAP` ne supporte pas les colonnes de type `BLOB` / `TEXT` .
- `HEAP` ne supporte pas les colonnes de type `AUTO_INCREMENT` .
- Avant MySQL 4.0.2, `HEAP` ne supportait les index sur les valeurs `NULL` .
- Vous pouvez utiliser des clés non−uniques avec les tables `HEAP` (ce qui n'est pas courant pour les tables de hachage).
- Les tables `HEAP` sont partagées entre tous les clients (comme une autre table).
- Vous ne pouvez pas recherche la prochaine ligne (c'est à dire, utiliser l'index pour optimiser la clause `ORDER BY` ).
- Les données pour les tables `HEAP` sont alloués par petits blocs. Les tables sont 100% dynamiques (en insertion). Aucune zone de débordement ou d'espace de clé supplémentaire n'est nécessaire. Les lignes effacées sont placées dans une liste, prêtes à être réutilisées.
- Vous avez besoin de suffisamment de mémoire pour accepter toutes les tables `HEAP` que vous allez utiliser simultanément.
- Pour libérer de la mémoire, vous devez exécuter la commande `DELETE FROM heap_table` , `TRUNCATE heap_table` ou `DROP TABLE heap_table` .
- MySQL ne peut pas calculer approximativement le nombre de lignes entre deux valeurs (ce qui est utilisé par l'optimiseur pour savoir quelle méthode utiliser). Cela peut affecter certaines requêtes, si vous changez une table `MyISAM` en table `HEAP` .
- Pour vous assurer que vous ne faîtes rien de trop téméraire, vous ne pouvez pas créer de table `HEAP` plus grande que `max_heap_table_size` .

La mémoire nécessaire pour les tables HEAP sont :

```
 SUM_OVER_ALL_KEYS(max_length_of_key + sizeof(char*) * 2)
+ ALIGN(length_of_row+1, sizeof(char*))
```

sizeof(char*) vaut 4 sur les machines 32 bits et 8 sur une machine 64 bits.

# 7.5 Tables InnoDB

## 7.5.1 Présentation des tables InnoDB

InnoDB fournit à MySQL un gestionnaire de table transactionnelle (compatible ACID ), avec validation (commits), annulations (rollback) et capacités de restauration après crash. InnoDB utilise un verrouillage de lignes, et fournit des lectures cohérentes comme Oracle, sans verrous. Ces fonctionnalités accroissent les possibilités d'utilisation simultanées des tables, et les performances. Il n'y a pas de problème de queue de verrous avec InnoDB , car les verrous de lignes utilisent très peu de place. Les tables InnoDB sont les premières tables MySQL qui supportent les contraintes de clés étrangères ( FOREIGN KEY ).

InnoDB a été conçu pour maximiser les performances lors du traitement de grandes quantités de données. Son efficacité processeur n'est égalée par aucun autre moteur de base de données.

Techniquement, InnoDB est un gestionnaire de table placé sous MySQL. InnoDB dispose de son propre buffer pour mettre en cache les données et les index en mémoire centrale. InnoDB stocke les tables et index dans un espace de table, qui peut être réparti dans plusieurs fichiers. Ceci diffère des tables comme, par exemple, MyISAM où chaque table est stockée dans un fichier différent. Les tables InnoDB peuvent prendre n'importe quelle taille, même sur les systèmes d'exploitation dont la limite est de 2 Go par fichier.

Vous pouvez trouver les dernières informations sur InnoDB à http://www.innodb.com/ . La dernière version du manuel InnoDB est toujours disponible la−bas, et vous pouvez aussi commander des licences commerciales et du support InnoDB .

InnoDB est actuellement (Octobre 2001) utilisé en production dans plusieurs sites où de grandes capacités de stockages et des performances accrues sont nécessaires. Le fameux site web Slashdot.org utilise InnoDB . Mytrix, Inc. stocke plus de 1 To de données dans une base InnoDB , et un autre site gère une moyenne de 800 insertions/modifications par secondes avec InnoDB .

Les tables InnoDB sont inclues dans la distribution source de MySQL depuis la version 3.23.34a et sont activées dans le binaire MySQL−Max. Pour Windows, les binaires de MySQL−Max sont disponibles dans la distribution standard.

Si vous avez téléchargé une version binaire de MySQL qui inclut le support de InnoDB , suivez simplement les instructions du manuel MySQL pour installer une version binaire. Is vous avez déjà la version MySQL−3.23 installée, alors le moyen de plus simple est d'installer le serveur mysqld avec l'exécutable correspondant de la distribution −Max. MySQL et MySQL −Max ne diffèrent que par leur exécutable. Installing a MySQL Binary Distribution . mysqld−max , la version étendue du serveur mysqld .

Pour compiler MySQL avec le support InnoDB , téléchargez MySQL−3.23.34a ou plus récent depuis http://www.mysql.com/ et configurez MySQL avec l'option --with−innodb . Voyez le manuel MySQL sur l'installation d'une distribution source. Installer MySQL à partir des sources .

```
cd /chemin/vers/source/de/mysql-3.23.37
./configure --with-innodb
```

Pour utiliser InnoDB , vous devez spécifier les options de démarrage InnoDB dans le fichier my.cnf ou my.ini . La méthode minimale pour modifier ces options est d'ajouter à la section [mysqld] , la ligne

```
innodb_data_file_path=ibdata:30M
```

mais pour obtenir de meilleures performances, vous devez spécifier les options telles que recommandé. Options de démarrage InnoDB .

InnoDB est sous licence GNU GPL License Version 2 (de Juin 1991). Dans la distribution source de MySQL, InnoDB apparaît comme un sous-dossier.

## 7.5.2 Options de démarrage InnoDB

Pour utiliser les tables InnoDB en MySQL-Max-3.23, vous devez spécifier des paramètres de configuration dans la section [mysqld] du fichier de configuration my.cnf , ou optionnelle, dans le fichier my.ini sous Windows.

Au minimum, en 3.23, vous devez spécifier innodb_data_file_path pour spécifier les noms et tailles de fichiers de données. Si vous décidez de ne pas mentionner innodb_data_home_dir dans my.cnf , le comportement par défaut est de créer ces fichiers dans le dossier de données de MySQL. Si vous ne spécifiez pas innodb_data_home_dir sous la forme d'une chaîne vide, vous pouvez donner un chemin absolu jusqu'au stockage de vos données dans innodb_data_file_path . En MySQL 4.0, vous n'avez même pas à spécifier l'option innodb_data_file_path : le comportement par défaut est de créer un fichier de données auto-croissant de 10 Mo appelé ibdata1 dans le dossier de données de MySQL. En MySQL 4.0.0 et 4.0.1, le fichier de données était de 64 Mo et de taille fixe.

Si vous ne voulez pas utiliser les tables InnoDB , vous pouvez ajouter l'option skip-innodb dans le fichier d'options de MySQL.

Mais pour obtenir de bonnes performances, vous devez explicitement choisir les paramètres InnoDB listés dans les exemples suivants : Depuis les versions 3.23.50 et 4.0.2, InnoDB fait que le dernier fichier spécifié dans l'option innodb_data_file_path peut être auto-croissant ( **auto-extending** ). La syntaxe pour la ligne innodb_data_file_path est alors la suivante :

```
pathtodatafile:sizespecification;pathtodatafile:sizespecification;...
...  ;pathtodatafile:sizespecification[:autoextend[:max:sizespecification]]
```

Si vous spécifiez le dernier fichier avec l'option autoextend , InnoDB va augmenter la taille du dernier fichier de données jusqu'à ce qu'il n'y ait plus de place dans l'espace de table. Les incréments se feront par bloc de 8 Mo. Par exemple :

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:100M:autoextend
```

indique à InnoDB de créer un fichier de données unique, de taille initiale de 100 Mo et qui sera agrandi de 8Mo jusqu'à ce qu'il n'y ait plus de place. Si le disque se remplit, vous placerez le prochain fichier de données sur un autre disque. Vous devez alors regarder la taille du fichier de données ibdata1 , arrondir sa taille au Mo précédent (multiple de 1024 * 1024 octets (= 1 Mo)) et spécifier la taille du fichier ibdata1 explicitement dans l'option innodb_data_file_path .Après cela, vous pouvez spécifier un autre fichier de données :

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:988M;/disk2/ibdata2:50M:autoextend
```

Soyez prudents avec les systèmes de fichiers où la taille maximale de fichier est de 2 Go! InnoDB n'est pas capable de détecter la taille maximale de fichier pour votre système d'exploitation. Sur d'autres systèmes, vous devrez spécifier la taille maximale du fichier :

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:100M:autoextend:max:2000M
```

**Exemple de fichier** my.cnf **simple** Supposons que vous avez un serveur avec 128 Mo de RAM et un disque dur. Voici un exemple de configuration de fichier my.cnf ou my.ini pour InnoDB . Nous supposons

que vous exécutez MySQL−Max−3.23.50 ou plus récent, ou MySQL−4.0.2 ou plus récent.

Cet exemple devrait convenir à une majorité d'utilisateurs, Unix et Windows, qui ne souhaitent pas répartir leur fichiers de données `InnoDB` et leurs logs sur plusieurs disques. Cette configuration crée un fichier de données auto−croissant, appelé `ibdata1` et deux fichiers de log `InnoDB ib_logfile0` et `ib_logfile1` dans le dossier de données MySQL (typiquement `/mysql/data` ). De plus, le petit fichier d'archive `InnoDB ib_arch_log_0000000000` sera placé dans `datadir` .

```
 [mysqld]
# Vous pouvez placer d'autres options MYSQL ici
# ...
#                                 Le fichier de données doivent contenir
#                                 vos données et index.
#                                 Assurez vous que vous avez l'espace disque
#                                 nécessaire.
innodb_data_file_path = ibdata1:10M:autoextend
#                                 Utilisez un buffer de taille
#                                 50 à 80 % de votre mémoire serveur
set−variable = innodb_buffer_pool_size=70M
set−variable = innodb_additional_mem_pool_size=10M
#                                 Utiisez un fichier de log de taille
#                                 25 % du buffer mémoire
set−variable = innodb_log_file_size=20M
set−variable = innodb_log_buffer_size=8M
#                                 Utilisez  ..flush_log_at_trx_commit
#                                 à 0 si vous pouvez accepter de perdre
#                                 quelques transactions
 innodb_flush_log_at_trx_commit=1
```

**Vérifiez que le serveur MySQL a les droits de créer ces fichiers** dans le `datadir` .

Notez que le fichier de données doit être inférieure à 2Go sur certains systèmes d'exploitation. La taille combinée des fichiers de log doit être inférieure à 4Go. La taille combinée des fichiers de données doit être inférieure à 10Go.

Lorsque vous créez pour la première fois une base de données `InnoDB` , il est mieux de lancer le serveur depuis la commande en ligne. `InnoDB` va afficher des informations sur la création de la base, et vous verrez commence ça se passe. Voyez la section plus bas, pour une illustration. Par exemple, sous Windows, vous pouvez démarrer `mysqld−max.exe` avec :

```
chemin−jusqu−a−mysqld>mysqld−max −−console
```

**Oz mettre le fichier `my.cnf` ou `my.ini` sous Windows?** Les règles sous Windows sont les suivantes :

- Un seul des deux fichiers `my.cnf` ou `my.ini` doit être créé.
- The `my.cnf` doit être placé dans le dossier racine du disque `C:` .
- Le fichier `my.ini` doit être placé dans le dossier `WINDIR` , e.g, `C:\WINDOWS` ou `C:\WINNT` . Vous pouvez utiliser la commande `SET` de MS−DOS pour afficher la valeur de `WINDIR` .
- Si votre PC utilise un gestionnaire de démarrage où le `C:` n'est pas votre disque de démarrage, alors votre seule option est d'utilise le fichier `my.ini` .

**Oz placer les fichiers d'options sous Unix?** Sous Unix, `mysqld` lit les options dans les fichiers suivants, si ils existent, et dans cet ordre :

- `/etc/my.cnf` Options globales.
- `COMPILATION_DATADIR/my.cnf` Options spécifiques au serveur.
- `defaults−extra−file` Le fichier spécifié avec `−−defaults−extra−file=...` .
- `~/.my.cnf` User−specific options.

`COMPILATION_DATADIR` est le dossier de données de MySQL qui a été spécifié lors de l'utilisation du script `./configure` , avant la compilation de `mysqld` . (typiquement, `/usr/local/mysql/data` pour une installation binaire, ou `/usr/local/var` pour une installation source).Si vous n'êtes pas sûr des chemins où `mysqld` lit les données `my.cnf` et `my.ini` , vous pouvez indiquer le chemin, avec la première option du serveur : `mysqld −−defaults−file=your_path_to_my_cnf` .

InnoDB forme le chemin de dossier en concaténant innodb_data_home_dir avec les noms de fichiers, ou le chemin de innodb_data_file_path , en ajoutant les slash nécessaires. Si le mot clé innodb_data_home_dir n'est pas mentionné dans my.cnf , la valeur par défaut est 'point' ./ , ce qui signifie le dossier de données de MySQL.

**Exemple de fichier** my.cnf **avancé** Supposons que vous avez un serveur Linux avec 2 Go de RAM et trois disques de 60 Go (situés dans les dossiers / , /dr2 et /dr3 . Voici ci−dessous un exemple de configuration possible pour my.cnf , de InnoDB .

**Notez que** InnoDB **ne crée pas de dossier : vous devez le créer vous même.** Utilisez la commande Unix ou MS−DOS mkdir pour créer les répertoires de données et de logs.

```
 [mysqld]
# Vous pouvez placer d'autres options MYSQL ici
# ...
innodb_data_home_dir =
#                                 Le fichier de données doivent contenir
#                                 vos données et index.
#                                 Assurez vous que vous avez l'espace disque
#                                 nécessaire.
innodb_data_file_path = /ibdata/ibdata1:2000M;/dr2/ibdata/ibdata2:2000M:autoextend
#                                 Utilisez un buffer de taille
#                                 50 à 80 % de votre mémoire serveur
#                                 mais assurez vous sous Linux que l'utilisation
#                                 totale est inférieure à 2Go
set−variable = innodb_buffer_pool_size=1G
set−variable = innodb_additional_mem_pool_size=20M
innodb_log_group_home_dir = /dr3/iblogs
#                                 .._log_arch_dir doit être le même que
 #                                 .._log_group_home_dir
innodb_log_arch_dir = /dr3/iblogs
set−variable = innodb_log_files_in_group=3
#                                 Utiisez un fichier de log de taille
#                                 15 % du buffer mémoire
set−variable = innodb_log_file_size=150M
set−variable = innodb_log_buffer_size=8M
#                                 Utilisez  ..flush_log_at_trx_commit
#                                 à 0 si vous pouvez accepter de perdre
#                                 quelques transactions
 innodb_flush_log_at_trx_commit=1
set−variable = innodb_lock_wait_timeout=50
#innodb_flush_method=fdatasync
#set−variable = innodb_thread_concurrency=5
```

Notez que nous avons placé deux fichier de données sur des disques différents. InnoDB va remplir l'espace de tables jusqu'au maximum. Dans certains cas, les performances seront améliorées si les données ne sont pas toutes placées sur le même disque physique. Placer les fichiers de log dans des disques séparés est souvent une bonne chose. Vous pouvez aussi utiliser des **partitions de disques brutes** (raw devices) comme fichier de données. Sur certains Unix, cela va accélérer les accès disques. Voyez le manuel de InnoDB , pour savoir comment les spécifier dans my.cnf .

**Attention** : en Linux x86, vous devez être très prudent, et **ne pas utiliser trop de mémoire** . glibc va autoriser les processus à dépasser la pile de thread, et votre système va crasher. Cela représente un risque réel si la valeur de

```
 innodb_buffer_pool_size + key_buffer +
max_connections * (sort_buffer + read_buffer_size) + max_connections * 2 MB
```

est proche de 2 Go ou excède 2 Go. Chaque thread va utiliser une pile (souvent 2Mo, mais les exécutables MySQL uniquement 256 ko) et dans le pire des scénarios, sort_buffer + read_buffer_size de mémoire supplémentaire. **Comment optimiser d'autres paramètres du serveur** mysqld **?** Les valeurs qui conviennent à la majorité des utilisateurs sont :

```
 skip−locking
set−variable = max_connections=200
set−variable = read_buffer_size=1M
set−variable = sort_buffer=1M
#                                 key_buffer vaut de 5 à 50%
#                                 de la RAM disponible, suivant l'utilisation des
#                                 tables MyISAM, mais garder
#                                 key_buffer + InnoDB
```

```
#                                en deça de < 80% de votre RAM
set-variable = key_buffer=...
```

Notez que certains paramètres sont données au format numérique dans `my.cnf` : `set-variable = innodb... = 123` , d'autres (chaînes et booléens) sont données dans un autre format : `innodb_... = ...` . Les significations des paramètres de configuration sont les suivantes :

| Option | Description |
|---|---|
| innodb_data_home_dir | La partie commune du chemin de tous les fichiers de données InnoDB . Si vous ne mentionnez pas cette option dans my.cnf , la valeur par défaut sera celle du dossier de données MySQL. Vous pouvez aussi spécifier une chaîne vide, et dans ce cas, les chemins spécifiés dans innodb_data_file_path seront des chemins absolus. |
| innodb_data_file_path | Chemin individuel vers les fichiers de données, et leur taill. Le chemin complet de chaque fichier de données est créé en concaténant innodb_data_home_dir avec les chemins spécifiés ici. La taille du fichier est spécifiée en méga-octets, ce qui explique la présence du 'M' après les spécifications ci-dessus. Depuis la version 3.23.44, vous pouvez donner au fichier une taille supérieure à 4 Go sur les systèmes d'exploitation qui acceptent les gros fichiers. Sur certains systèmes, la taille doit être inférieure à 2 Go. La somme des tailles des fichiers doit faire au moins 10 Mo. |
| innodb_mirrored_log_groups | Nombre de copies identiques de groupe de log que nous conservons. Actuellement, cette valeur doit être au minimum de 1. |
| innodb_log_group_home_dir | Le dossier pour les fichiers de logs. |
| innodb_log_files_in_group | Nombre de fichier de logs dans un groupe. InnoDB écrit les logs de manière circulaire. Une valeur de 3 est recommandée ici. |
| innodb_log_file_size | Taille de chaque fichier de log dans un groupe de log, exprimé en méga-octets. Les valeurs pratiques vont de 1Mo à une fraction de la taille du buffer de log (1 / le nombre de logs, en fait). Plus la taille est grande, moins de points de contrôles seront utilisés, réduisant les accès disques. La taille combinée des logs doit être inférieure à 4 Go sur les systèmes 32 bits. |
| innodb_log_buffer_size | La taille du buffer que InnoDB utilise pour écrire les log dans les fichiers de logs, sur le disque. Les valeurs utiles vont de 1 Mo à 8 Mo. Un grand buffer de log permet aux grandes transactions de s'exécuter sans avoir à écrire de données dans le fichier de log jusqu'à la validation. Par conséquent, si vous avez de grandes transactions, augmenter cette taille va réduire les accès disques. |
| innodb_flush_log_at_trx_commit | Normalement, cette option vaut 1, ce qui signifie que lors de la validation de la transaction, les logs sont écrits sur le disque, et les modifications faites par la transaction deviennent permanentes, et survivront un crash de base. Si vous souhaitez réduire la sécurité de vos données, et que vous exécutez de petites transactions, vous pouvez donner une valeur de 0 à cette option, pour réduire les accès disques. |
| innodb_log_arch_dir | Le dossier où les logs complétés doivent être archivés, si nous utilisons l'archivage de logs. La valeur de ce paramètre doit être actuellement la même que la valeur de innodb_log_group_home_dir . |
| innodb_log_archive | Cette valeur doit être actuellement de 0. Au moment de la restauration de données à partir d'une sauvegarde, à l'aide des log binaires de MySQL, il n'y a actuellement pas besoin d'archiver les fichiers de log InnoDB . |
| innodb_buffer_pool_size | La taille de buffer mémoire que InnoDB utiliser pour mettre en cache les données et les index de tables. Plus cette valeur est grand, et moins vous ferez d'accès disques. Sur un serveur dédiés, vous pouvez monter cette valeur jusqu'à 80% de la mémoire physique de la machine. Ne lui donnez pas une valeur trop grande, car cela peut engendrer l'utilisation de mémoire sur le disque par votre serveur. |
| innodb_additional_mem_pool_size | La taille du buffer mémoire d' InnoDB , pour ses dictionnaires d'informations, et ses structures internes de données. Une valeur pratique est 2Mo, mais plus vous aurez de tables dans votre application, plus vous devrez augmenter cette valeur. Si InnoDB est à court de mémoire, il va allouer de la mémoire auprès du système, et écrire des messages dans le fichier de logs MySQL. |
| innodb_file_io_threads | Nombre de pointeurs de fichier de InnoDB . Normalement, cette valeur doit être de 4, mais sur des disques Windows, les accès peuvent être améliorés en augmentant cette valeur. |
| innodb_lock_wait_timeout | Le délai d'expiration des transactions InnoDB , en cas de blocage de verrou, avant d'annuler. InnoDB détecte automatiquement les blocages de verrous et annule alors les transactions. Si vous utilisez la commande LOCK TABLES , ou un autre gestionnaire de table transactionnelles que InnoDB dans la même transaction, un blocage de verrou peut survenir, et InnoDB ne pourra pas le détecter. Ce délai est donc pratique pour résoudre ces situations. |
| innodb_flush_method | (Disponible depuis 3.23.40 et plus récent) La valeur par défaut pour cette option est fdatasync . Une autre option est O_DSYNC . |

## 7.5.3 Créer des bases InnoDB

Supposons que vous avez installé MySQL et que vous avez édité le fichier `my.cnf` de façon à ce qu'il contiennent les paramètres de configuration nécessaires de `InnoDB` . Avant de démarrer MySQL, vous devez vérifier que les dossiers que vous avez spécifié pour les fichiers de données `InnoDB` et les fichiers de logs existent, et que vous avez des accès suffisants dans ces dossiers. `InnoDB` ne peut pas créer de dossiers, uniquement des fichiers. Vérifiez aussi que vous avez d'espace disque pour les données et les logs.

Lorsque vous démarrez MySQL, `InnoDB` va commencer à créer vos fichiers de données et vos fichiers de log. InnDB va afficher ceci :

```
 ~/mysqlm/sql > mysqld
InnoDB: The first specified datafile /home/heikki/data/ibdata1
did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file /home/heikki/data/ibdata1 size to 134217728
InnoDB: Database physically writes the file full: wait...
InnoDB: datafile /home/heikki/data/ibdata2 did not exist:
new to be created
InnoDB: Setting file /home/heikki/data/ibdata2 size to 262144000
InnoDB: Database physically writes the file full: wait...
```

```
InnoDB: Log file /home/heikki/data/logs/ib_logfile0 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile0 size to 5242880
InnoDB: Log file /home/heikki/data/logs/ib_logfile1 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile1 size to 5242880
InnoDB: Log file /home/heikki/data/logs/ib_logfile2 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile2 size to 5242880
InnoDB: Started
mysqld: ready for connections
```

Une nouvelle base de données `InnoDB` a été créée. Vous pouvez vous connecter au serveur MySQL avec votre client MySQL habituel, comme `mysql` . Lorsque vous arrêtez le serveur MySQL avec `mysqladmin shutdown` , `InnoDB` va afficher :

```
 010321 18:33:34  mysqld: Normal shutdown
010321 18:33:34  mysqld: Shutdown Complete
InnoDB: Starting shutdown...
InnoDB: Shutdown completed
```

Vous pouvez observer vos fichiers de données et de logs, et vous apercevrez les fichiers créés. Le dossier de log va aussi contenir un petit fichier appelé `ib_arch_log_0000000000` . Ce fichier est le résultat de la création de base, à partir duquel `InnoDB` a désactivé l'archivage des logs. Lorsque MySQL va être redémarré, l'affichage sera :

```
 ~/mysqlm/sql > mysqld
InnoDB: Started
mysqld: ready for connections
```

#### 7.5.3.1 Si quelque chose se passe mal à la création de la base de données

Si `InnoDB` renvoie une erreur de système d'exploitation lors d'une opération sur fichier, habituellement le problème est l'un des suivants :

- Vous n'avez pas crée les dossiers de données ou de logs `InnoDB` .
- `mysqld` n'a pas le droit de créer des fichiers dans ces dossiers.
- `mysqld` ne lit pas le bon fichier `my.cnf` ou `my.ini` , et donc ne voit pas les options que vous spécifiez.
- Le disque ou l'espace disque alloué est plein.
- Vous avez crée un sous−dossier dont le nom est le même que celui d'un fichier de données que vous avez spécifié.
- Il y a une erreur de syntaxe dans `innodb_data_home_dir` ou `innodb_data_file_path` .

Si quelque chose se passe mal lors de la création d'une base de données `InnoDB` , vous devez effacer tous les fichiers créés par `InnoDB` . Cela inclut tous les fichiers de données, tous les journaux, les archives. Dans le cas où vous avez déjà crées des tables `InnoDB` , effacez aussi les fichiers `.frm` concernés dans le dossier de données de MySQL. Vous pourrez alors essayer une nouvelle création de base de données `InnoDB` .

### 7.5.4 Créer des tables InnoDB

Supposons que vous avez démarré le client MySQL avec la commande `mysql test` . Pour créer une table au format `InnoDB` vous devez spécifier le type `TYPE = InnoDB` lors de la création de table, dans la commande SQL :

```
 CREATE TABLE CUSTOMER (A INT, B CHAR (20), INDEX (A)) TYPE = InnoDB;
```

Cette commande SQL va créer une table et un index sur la colonne `A` dans la base `InnoDB` constituée par les fichiers de données que vous avez spécifié dans le fichier de configuration `my.cnf` . De plus, MySQL va créer un fichier `CUSTOMER.frm` dans le dossier de données de MySQL `test` . En interne, `InnoDB` va ajouter une entrée dans son propre dictionnaire de données une entrée pour la table `'test/CUSTOMER'` . De cette façon, vous pouvez créer plusieurs table avec le même nom de `CUSTOMER` , mais dans d'autres bases MySQL, et les noms de seront pas en conflit avec `InnoDB` .Vous pouvez demander la quantité

d'espace disponible dans l'espace de tables InnoDB avec la commande de statut de MySQL pour toutes les tables de type TYPE = InnoDB . La quantité d'espace disponible apparaît dans la section de commentaire de la commande SHOW . Par exemple :

```
SHOW TABLE STATUS FROM test LIKE 'CUSTOMER'
```

Notez que les statistiques que SHOW vous donne sur les tables InnoDB ne sont que des approximations : elles sont utilisées pour les optimisations SQL par MySQL. Les tailles réservées d'index et de table, exprimées en octets, sont précises. **Convertir une table MyISAM en InnoDB**

InnoDB n'a pas d'optimisation particulière pour la création d'un fichier externe d'index. Donc, exporter pour réimporter les données n'apporte pas de gain de performances. La méthode la plus rapide pour donner à une table le format InnoDB est de faire les insertions directement dans la table InnoDB , avec la commande ALTER TABLE ... TYPE=INNODB , ou de créer une table vide InnoDB avec les mêmes caractéristiques, et d'insérer les lignes avec la commande INSERT INTO ... SELECT * FROM ... .

Pour avoir un meilleur contrôle sur le processus d'insertion, c'est une bonne idée de faire les insertions par portions de table :

```
INSERT INTO newtable SELECT * FROM oldtable
  WHERE votre_cle > borne1 AND votre_cle <= borne2;
```

Une fois que toutes les données ont été insérées, vous pouvez renommer les tables.Durant la conversion des grandes tables, vous devriez utiliser le buffer InnoDB en lui donnant une grande taille pour réduire les accès disques. Ne lui donnez pas une taille supérieure à 80% de votre mémoire physique. Vous devriez aussi avoir de grand fichiers de log et un buffer de log important.

Assurez vous que vous avez toujours de l'espace pour les données : les tables InnoDB prennent bien plus de place que les tables MyISAM. Si une commande ALTER TABLE rencontre un problème d'espace disque, elle va s'annuler, et cela va prendre des heures si le disque est plein. Lors des insertions, InnoDB utilise un buffer d'insertion pour fusionner les index par groupe. Cela économise beaucoup d'accès disques. Lors d'une annulation, aucun mécanisme de ce type n'est utilisé, et l'annulation peut prendre jusqu'à 30 fois la durée de l'insertion.

Dans le cas d'une annulation, si vous n'avez pas de données importante dans votre base, il est mieux de terminer le processus et de détruire les données InnoDB et les fichiers de log, ainsi que le fichier de table .frm , et de recommencer, plutôt que d'attendre les millions d'accès disques.

**Contraintes de clé étrangères**

Depuis la version 3.23.43b, InnoDB supporte les contraîntes de clé étrangères. InnoDB est le premier géstionnaire de tables MySQL qui permet de définir des contraîntes de clé étrangères pour assurer la cohérence de vos données.La syntaxe d'une clé étrangère avec InnoDB :

```
[CONSTRAINT symbol] FOREIGN KEY (index_col_name, ...)
                REFERENCES table_name (index_col_name, ...)
                [ON DELETE CASCADE | ON DELETE SET NULL]
```

Les deux tables doivent être de type InnoDB et **il doit y avoir un index dans lesquelles les colonnes de clé étrangère et de clé référencée sont en premier** . InnoDB ne crée pas automatiquement des index pour les clés étrangères et les clé référencées : vous devez le faire vous même.Les colonnes correspondantes dans la clé étrangère doivent être de même type interne à InnDB, de façon à ce qu'elle puissent être comparées sans conversion. La **taille et le signe des types entiers doivent être les mêmes** . La taille des chaînes n'a pas besoin d'être les mêmes.

Depuis la version 3.23.50, vous pouvez aussi associer la clause ON DELETE CASCADE ou ON DELETE SET NULL avec les contraintes de clé étrangère.

Si ON DELETE CASCADE est spécifié, et qu'une ligne de la table parente est effacée, alors InnoDB va

automatiquement effacer la lgine dans la table fille, dont la clé étrangère est égale à la clé référencée dans la table parente. Si `ON DELETE SET NULL` est spécifié, les lignes de la table fille sont automatiquement modifiée de façon à ce que les colonnes dans la clé étrangère prennent automatiquement la valeur SQL de NULL.

Depuis la version 3.23.50, `InnoDB` ne vérifie pas les contraintes de clé étrangère pour ces clés étrangères, ou les clés qui font référence à la valeur NULL.

Depuis la version 3.23.50, `InnoDB` vous permet d'utiliser les guillemets américains (`) autour des noms de tables et colonnes dans la clause `FOREIGN KEY ... REFERENCES ...` mais `InnoDB` ne prend pas encore en compte l'option `lower_case_table_names` que vous pouvez spécifier dans le fichier `my.cnf` . Un exemple :

```
 CREATE TABLE parent(id INT NOT NULL, PRIMARY KEY (id)) TYPE=INNODB;
CREATE TABLE child(id INT, parent_id INT, INDEX par_ind (parent_id),
             FOREIGN KEY (parent_id) REFERENCES parent(id)
             ON DELETE SET NULL
) TYPE=INNODB;
```

Si MySQL retourne une erreur numéro 1005 à la commande `CREATE TABLE` , et que l'erreur fait référence à au numéro 150, alors la création de table a échoué car la clé étrangère est mal formée. De même, si une commande `ALTER TABLE` échoue avec un numéro d'erreur de 150, cela signifie que la définition de la clé étrangère a été mal formée dans la table.Depuis la version 3.23.50, `InnoDB` vous permet d'ajouter une nouvelle clé étrangère, comme ceci :

```
 ALTER TABLE yourtablename
ADD [CONSTRAINT symbol] FOREIGN KEY (...) REFERENCES anothertablename(...)
```

N'oubliez pas de commencer par créer vos index. En `InnoDB` versions inférieures à 3.23.50, `ALTER TABLE` ou `CREATE INDEX` ne doivent pas être utilisée avec les tables qui ont une clé étrangère, ou qui sont référencées dans une clé étrangère : toute commande `ALTER TABLE` supprime les contraintes de clé étrangère définie dans la table. Vous de devez pas utiliser la commande `ALTER TABLE` dans la table référencée non plus, mais utilisez plutôt `DROP TABLE` et `CREATE TABLE` pour modifier le schéma. Lorsque MySQL fait un `ALTER TABLE` il peut utiliser en interne la commande `RENAME TABLE` , et cela va poser un problème à la clé étrangère à laquelle la table se réfère. Une commande `CREATE INDEX` est traitée par MySQL comme une commande `ALTER TABLE` , et les mêmes restrictions s'appliquent.

Lorsqu'il fait des vérifications de clé étrangères, `InnoDB` pose des verrous de lignes sur les lignes filles et mères qu'il doit utiliser. `InnoDB` vérifie les contraintes immédiatement : la vérification n'est pas reportée à la validation de la transaction.

`InnoDB` vous permet d'effacer n'importe quelle table, même si cela casse la cohérence d'une clé étrangère, qui fait référence à la table. Lorsque vous effacez une table, les contraintes qui en faisaient parties sont aussi effacées.

Si vous recréez une table qui a été effacée, elle doit avoir une définition qui satisfont les contraintes de clé étrangères. Elle doit avoir les bons noms et types de colonnes, ainsi que les index nécessaires. Si ces conditions ne sont pas remplies, MySQL retourne une erreur numéro 1005, et fait référence au numéro de message 150.

Depuis la version 3.23.50, `InnoDB` retourne la définition de contrainte de clé étrangère lorsque vous appelez la commande

```
 SHOW CREATE TABLE yourtablename
```

De plus, `mysqldump` produit la définition correcte pour les tables dans l'export, et n'oublie pas les clés étrangères.Vous pouvez aussi lister les clés étrangères pour une table `T` avec

```
 SHOW TABLE STATUS FROM yourdatabasename LIKE 'T'
```

Les contraintes de clé sont listées dans les commentaires de la table.

## 7.5.5 Ajouter et retirer des données et des logs InnoDB

Depuis la version 3.23.50 et 4.0.2, vous pouvez spécifier le dernier fichier de données InnoDB dans le fichier autoextend . Alternativement, vous pouvez augmenter votre espace de données en spécifiant d'autres fichiers de données. Pour cela, vous devez arrêter le serveur MYSQL, éditer le fichier my.cnf pour y ajouter un nouveau nom de fichier dans innodb_data_file_path , puis redémarrer le serveur MySQL.

Actuellement, vous ne pouvez pas retirer de fichier de données à InnoDB . Pour réduire la taille de votre base de données, vous devez utiliser l'utilitaire mysqldump pour exporter toutes vos données de tables, créer une nouvelle base, et réimporter toutes vos données dans cette nouvelle base.

Si vous voulez changer le nombre ou la taille de vos fichiers de log InnoDB , vous devez éteindre le serveur MYSQL et vous assurer qu'il s'est arrété sans erreur. Puis, copiez les anciens fichiers de log dans une archive, car si vous rencontrez un problème ultérieurement, vous en aurez besoin pour restaurer votre base. Effacer les anciens fichiers de log du dossier de logs, éditez le fichier my.cnf , et redémarrez le serveur MySQL. InnoDB vous indiquera au démarrage qu'il va créer de nouveaux fichiers de log.

## 7.5.6 Sauver et restaurer une base InnoDB

Le secret de la gestion de bases de données sereine réside dans les sauvegardes régulières.

InnoDB Hot Backup est un outil de sauvegarde que vous pouvez utiliser pour faire des sauvegarde des bases de données InnoDB , lorsqu'elles sont utilisées. InnoDB Hot Backup ne vous impose pas de stopper le serveur, et ne pose aucun verrou ou ne perturbe votre utilisation normale des tables. InnoDB Hot Backup est un outils supplémentaire, qui n,est pas libre, et qui n'est pas inclus dans la distribution standard de MySQL. Voyez le site web de InnoDB Hot Backup http://www.innodb.com/hotbackup.html pour plus de détails et des captures d'écran.

Si vous êtes capables d'arréter votre serveur MySQL, alors pour faire une sauvegarde binaire de vos bases de données, vous devez suivre les instructions suivantes :

- Arrêter votre serveur MySQL et assurez vous qu'il s'arrête sans erreur.
- Copiez tous les fichiers de données dans une archive.
- Copiez tous les fichiers d'historique de InnoDB dans une archive.
- Copiez votre fichier de configuration my.cnf dans une archive.
- Copiez tous les fichiers .frm de vos tables InnoDB dans une archive.

En plus de réaliser des copies des fichiers binaires tels que décrit ci−dessus, il est recommandé de réaliser des sauvegardes textuelles de vos tables avec mysqldump . La raison est qu'un fichier binaire peut être corrompu sans que vous ne vous en rendiez compte. Les tables sauvées dans un fichier texte sont stockées dans un format humainement lisible, bien plus simple qu'un fichier de base. Il est alors facile de répérer une corruption des données dans ce fichier, et comme le format de fichier est simple, il est facile de le corriger.

C'est une bonne idée de prendre des copies de vos tables en même temps que vous faites une sauvegarde binaire. Vous devez éteindre tous les clients de votre base pour obtenir un bilan cohérent de vos tables. Puis, vous pouvez faire les sauvegardes binaires, et vous aurez ainsi une sauvegarde cohérente de votre base, en deux formats.

Pour être capable de restaurer des données de votre base InnoDB jusqu'à présent, à partir des sauvegardes binaires ci−dessus, vous devez faire tourner le serveur MySQL en ayant activé le log général et l'archivage des logs. Ici, par "log général", nous considérons le mécanisme général de

log de MySQL, qui est indépendant des logs InnoDB .

Pour reconstruire une table après le crash d'un serveur MySQL, la seule chose que vous devez faire est de le redémarrer. InnoDB va automatiquement vérifier les fichiers d'historiques, et effectuer une mise à jour des données. InnoDB va automatiquement annuler les transactions non validées, qui étaient en cours au moment du crash. Durant la restauration de la table, InnoDB va afficher des données comme celles−ci :

```
 ~/mysqlm/sql > mysqld
InnoDB: Database was not shut down normally.
InnoDB: Starting recovery from log files...
InnoDB: Starting log scan based on checkpoint at
InnoDB: log sequence number 0 13674004
InnoDB: Doing recovery: scanned up to log sequence number 0 13739520
InnoDB: Doing recovery: scanned up to log sequence number 0 13805056
InnoDB: Doing recovery: scanned up to log sequence number 0 13870592
InnoDB: Doing recovery: scanned up to log sequence number 0 13936128
...
InnoDB: Doing recovery: scanned up to log sequence number 0 20555264
InnoDB: Doing recovery: scanned up to log sequence number 0 20620800
InnoDB: Doing recovery: scanned up to log sequence number 0 20664692
InnoDB: 1 uncommitted transaction(s) which must be rolled back
InnoDB: Starting rollback of uncommitted transactions
InnoDB: Rolling back trx no 16745
InnoDB: Rolling back of trx no 16745 completed
InnoDB: Rollback of uncommitted transactions completed
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Apply batch completed
InnoDB: Started
mysqld: ready for connections
```

Si votre base se corrompt ou que votre disque est en panne, vous devez faire une restauration depuis une sauvegarde. En cas de corruption, vous devez trouver d'abord une sauvegarde qui n'est pas corrompue. A partir de cette sauvegarde, effectuer la restauration avec les fichiers d'historique généraux de MySQL, suivant les instructions du manuel MySQL.

### 7.5.6.1 Points de contrôle

InnoDB utilise un mécanisme de contrôle appelé points de contrôle flou. InnoDB va écrire des blocs de données modifiées depuis un buffer vers le disque par petits paquets : il n'y a pas besoin de tout écrire en une seule fois, ce qui en général, conduit à l'arrêt du traitement des autres instructions pour quelques instants.

Durant une restauration de base, InnoDB recherche un point de contrôle écrit dans les fichiers de log. Il sait que toutes les modifications de la base placées avant ce point de contrôle sont aussi présentes sur le disque de la base. Puis, InnoDB analyse le fichier de log, et applique les modifications qui ont eu lieu depuis le point de contrôle.

InnoDB écrit dans les fichiers de log en mode circulaire. Toutes les modifications validées qui font que le buffer de MySQL est différent de la version sur le disque doivent être disponibles dans les fichiers de log, au cas où InnoDB aurait besoin pour une restauration. Cela signifie que lorsque InnoDB commence à réutiliser le fichier d'historique, il doit commencer par s'assurer que le disque a reçu les modifications qui sont dans le buffer. En d'autres termes, InnoDB doit placer un point de contrôle, et souvent, cela se traduit par l'écriture de données du buffer sur le disque.

Ceci explique pourquoi utiliser de grands fichiers de log peut éviter des accès disques pour les points de contrôle. Il est recommandé d'utiliser une taille de fichier d'historique aussi grande que le buffer, voire même plus grande. L'inconvénient des grands fichiers est que la restauration dure alors plus longtemps, puisqu'il y a plus de modifications à appliquer.

## 7.5.7 Transférer une base de données InnoDB vers une autre machine

Les fichiers de données et de logs de InnoDB sont compatibles en mode binaire sur toutes les plate−formes si le format des nombre à virgule flottante est le même. Vous pouvez déplacer une base de données InnoDB en copiant tous les fichiers concernés, que nous avons déjà listés dans la section précédente traitant de la sauvegarde d'une base de données. Si les formats des nombres à virgules flottantes sont différents mais que vous n'avez pas utilisé les types de données `FLOAT` ou `DOUBLE` dans vos tables alors la procédure est là même : copiez juste les fichiers concernés. Si les formats sont différents et que vous utilisez de tels types de données, vous devez utiliser `mysqldump` et `mysqlimport` pour transférer les tables.

Un bon moyen d'avoir de bonnes performances est de couper le mode auto−commit quand vous importez des données dans votre base de données, en supposant que votre tablespace possède assez d'espace pour la grande partie d'annulations (rollback) que la grande transaction importée génère. Ne faites le commit qu'après avoir importé une table entière, ou un segment de table.

## 7.5.8 Modèle transactionnel de InnoDB

Le modèle transactionnel d' `InnoDB` a pour but de combiner les avantages des bases de données multi−version aux verrouillages traditionnels en deux phases. `InnoDB` fait un verrouillage de ligne, et exécute les requêtes par défaut avec des lectures cohérentes non bloquante, de la même façon qu'Oracle. Les verrous `InnoDB` sont stockés de manière efficace, pour que l'escalade de transaction ne soit pas nécessaire : typiquement, plusieurs utilisateurs sont autorisés à verrouiller toutes les lignes dans une base, ou un sous ensemble aléatoire de ligne, sans que `InnoDB` ne soit à court de mémoire.

Avec `InnoDB` , toutes les activités des utilisateurs sont placées dans des transactions. Si le mode de validation automatique de MySQL est utilisé, chaque requête SQL représente une transaction. MySQL démarre toujours une nouvelle connexion avec de mode activé.

Si le mode de validation automatique est désactivé, avec la commande `SET AUTOCOMMIT = 0` , alors nous considérons que l'utilisateur a toujours une transaction ouverte. Si il utilise les commandes `COMMIT` ou `ROLLBACK` , cela va terminer la transaction courante, et en démarrer une nouvelle. Les deux commandes vont libérer les verrous posés durant la transaction courante. Une validation `COMMIT` signifie que les modifications faites durant la transaction sont rendus permanents, et accessibles aux autres utilisateurs. Une annulation `ROLLBACK` , d'un autre coté, annule toutes les modifications faites durant la transaction.

Si la connexion est en mode `AUTOCOMMIT = 1` , alors l'utilisateur peut réaliser une transaction à requêtes multiples en commençant par la commande `BEGIN` et en terminant la transaction par `COMMIT` ou `ROLLBACK` .

En termes de niveau d'isolations de transaction SQL−1992, le mode par défaut d `InnoDB` est `REPEATABLE READ` . Depuis la version 4.0.5, `InnoDB` offre 4 niveaux d'isolation de transaction différents, tels que décrits dans le standard SQL 1992. Vous pouvez choisir le niveau d'isolation par défaut du serveur `[mysqld]` dans le fichier de configuration `my.cnf` :

```
transaction−isolation = &#123;READ−UNCOMMITTED | READ−COMMITTED
                        | REPEATABLE−READ | SERIALIZABLE&#125;
```

Un utilisateur peut modifier le niveau d'isolation d'une transaction ou des nouvelles connexions avec la commande suivante :

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL
                      &#123;READ UNCOMMITTED | READ COMMITTED
                       | REPEATABLE READ | SERIALIZABLE&#125;
```

Notez qu'il n'y a pas de tirets dans les noms de niveaux de la syntaxe SQL. Si vous spécifiez le mot réservé `GLOBAL` dans la commande ci−dessus, cela va configurer les valeurs pour les nouvelles connexions, mais ne modifiera pas les valeurs des connexions déjà ouvertes. Tout utilisateur est

libre de changer le niveau d'isolation de sa session, même durant la transaction. Dans les versions inférieure à 3.23.50, SET TRANSACTION n'avait aucun effet sur les tables InnoDB . Dans les versions inférieures à 4.0.5, seuls REPEATABLE READ et SERIALIZABLE étaient disponibles.

Vous pouvez connaître les valeurs de niveaux d'isolations global et courant avec la commande :

```
 SELECT &#64;&#64;global.tx_isolation;
SELECT &#64;&#64;tx_isolation;
```

Durant le verrouillage de ligne, InnoDB utilise le verrouillage de clé suivante. Cela signifie qu'en plus des lignes de l'index, InnoDB verrouille aussi l'espace 'avant' un index, pour éviter les insertions d'un autre utilisateur. Un verrou de ligne suivante signifie le verrouillage de l'index et de l'intervalle entre cet index et le précédent. Un verrouillage d'espacement indique un verrouillage de l'espace entre deux index.

Voici une description plus détaillée des différents niveaux d'isolation des transactions de InnoDB :

- READ UNCOMMITTED , qui est aussi appelé la 'lecture incohérente' : les commandes SELECT non bloquantes sont exécutées sans vérifier la présence de versions antérieures de la ligne : par conséquent, ces lectures ne sont pas cohérentes avec ce niveau d'isolation. Sinon, ce niveau fonctionne exactement comme READ COMMITTED .
- READ COMMITTED Le niveau d'isolation qui se rapproche de celui d'Oracle. TOutes les commandes SELECT ... FOR UPDATE et SELECT ... LOCK IN SHARE MODE ne verrouille que les lignes d'index, et non pas les espaces qui les précèdent, et de cette manière, ils autorisent les insertions entre deux lignes verrouillées. Les commandes UPDATE et DELETE qui utilisent un index unique avec une condition de recherche, ne vont verrouiller que les index trouvés, et non pas les espaces précédent ces verrous. Mais avec les commandes UPDATE et DELETE en mode intervalle, InnoDB va poser des verrous de clé suivante, ou verrous d'espacement, et bloquer les insertions d'autres utilisateurs dans ces espaces. Ceci est nécessaire pour éviter la présence de lignes 'fantômes', qui sont néfastes pour la réplication. **Les lectures cohérentes** fonctionnent comme sous Oracle : chaque lecture cohérente, même durant une transaction, fixe et lit une version récente de données.
- REPEATABLE READ est le niveau d'isolation par défaut d' InnoDB . SELECT ... FOR UPDATE , SELECT ... LOCK IN SHARE MODE , UPDATE , et DELETE qui utilisent un index unique comme condition de recherche, ne vont verrouiller que la ligne d'index trouvée, et non pas l'espace qui la prépare. Sinon, ces opérations vont employer le verrouillage de clés suivante, pour bloquer les insertions parasites. En **lectures cohérentes** il y a une différence importante avec le niveau d'isolation précédent : avec ce niveau, toutes les lectures faites dans une même transaction sont faîtes avec la version établit lors de la première lecture. Cette convention signifie que si vous faites plusieurs lectures SELECT dans une même transaction, ces SELECT seront cohérents entre eux.
- SERIALIZABLE Ce niveau ressemble au précédent, mais toutes les lectures SELECT s sont implicitement converties en SELECT ... LOCK IN SHARE MODE .

### 7.5.8.1 Lecture cohérente

Une lecture cohérente signifie que InnoDB utilise son système de multi−versionage pour présenter à une requête, une photo de la base à un moment donné. La requête va alors voir les différentses modifications apportées par les transactions qui ont eu lieu avant cette date, et masquera les transactions ont eu lieu depuis, ou n'ont pas été archivées. L'exception à cette règle est que la requête verra les modification apportées la requête qui a émis cette commande.

Si vous utilisez le niveau d'isolation REPEATABLE READ , alors les lectures cohérentes dans une même transaction liront le même bilan. Vous pouvez obtenir un bilan plus récent pour vos requêtes en archivant la requête courante, et en démarrant une autre.

Les lectures cohérentes sont le mode par défaut de traitement des commandes SELECT par InnoDB avec les niveaux d'isolation READ COMMITTED et REPEATABLE READ . Une lecture cohérente ne pose aucun verrou sur les tables auxquelles elle accède, et par conséquent, les autres utilisateurs peuvent librement modifier ces tables en même temps qu'une lecture cohérente est exécutée.

### 7.5.8.2 Verrous de lecture

Une lecture cohérente n'est pas toujours pratique, dans certaines circonstances. Supposons que vous voulez ajouter une ligne dans votre table CHILD , et vous assurer que l'enfant a déjà un parent dans la table PARENT .

Supposez que vous utilisiez une lecture cohérente, pour lire la table PARENT , et que vous découvrez le parent de l'enfant dans cette table. Pouvez vous ajouter tranquillement la ligne fille dans la table CHILD ? Non, car il peut arriver que durant ce temps, un autre utilisateur a effacé la ligne parente dans la table PARENT , et vous n'en êtes pas conscient.

La solution est d'exécuter la commande SELECT en mode verrouillage, avec LOCK IN SHARE MODE .

```
SELECT * FROM PARENT WHERE NAME = 'Jones' LOCK IN SHARE MODE;
```

Effectuer une lecture en mode partagé signifie que vous allons lire les dernières données disponibles, et que nous allons poser un verrou sur les lignes que nous lisons. Si les dernières données appartiennent à une transaction non validée d'un autre utilisateur, nous allons attendre que ce soit fait. Un verrou partagé évite que les autres utilisateurs ne modifient ou n'effacent la ligne que nous lisons. Après que nous ayons obtenu le nom du parent, nous pouvons tranquillement ajouter le nom du fils dans la table CHILD , et valider notre transaction. Cet exemple montre comment implenter l'intégrité référentielle dans votre application.

Ajoutons un autre exemple : nous avons un champs compteur dans la table CHILD_CODES que nous utilisons pour assigner un identifiant unique à chaque enfant que nous ajoutons dans la table CHILD . Evidemment, en utilisant une lecture cohérente ou une lecture partagée pour lire la valeur courante du compteur n'est pas une bonne idée, car deux utilisateurs de la base peuvent simultanément lire la même valeur de compteur, et nous allons obtenir une erreur de clé doublon lorsque nous ajouterons le second des deux fils.

Dans ce cas, il y a deux bonnes méthodes pour implémenter la lecture et l'incrémentation du compteur : (1) modifiez le compteur d'une unité, et lisez le après cela ou (2) lisez le compteur d'abord, avec un verrou en mode FOR UPDATE , puis incrémentez le :

```
 SELECT COUNTER_FIELD FROM CHILD_CODES FOR UPDATE;
UPDATE CHILD_CODES SET COUNTER_FIELD = COUNTER_FIELD + 1;
```

Une commande SELECT ... FOR UPDATE va lire les dernières données disponibles pour chaque ligne, et pose un verrou dessus en même tant qu'il lit. De cette façon, il pose le même verrou que la commande UPDATE .

### 7.5.8.3 Verrou de clé suivante : éviter le problème des lignes fantômes

Avec le verrouillage de ligne, InnoDB utilise un algorithme appelé le verrouillage de la clé suivante. InnoDB fait un verrouillage de telle sorte que lorsqu'il fait une recherche ou un scan d'index, il pose des verrous partagés ou exclusifs sur les lignes d'index qu'il rencontre. Par conséquent, les verrous de lignes sont plus exactement appelés des verrous d'index.

Les verrous que InnoDB posent affectent aussi l'espace qui le sépare de la ligne suivante. Si un utilisateur a un verrou partagé ou exclusif sur une ligne L dans un index, alors un autre utilisateur ne

peut faire d'insertion immédiatement avant la ligne L, dans l'ordre de l'index. Ce verrouillage est fait pour éviter le problème de la ligne fantôme. Supposons que je veuille lire et verrouiller tous les enfants ayant un identifiant supérieur à 100 dans la table `CHILD` , puis modifier certains champs des lignes ainsi identifiées :

```
SELECT * FROM CHILD WHERE ID > 100 FOR UPDATE;
```

Supposons qu'il y ait un index sur la table `CHILD` , sur la colonne `ID` . Notre requête va scanner l'index à partir de la première ligne où `ID` est plus grand que 100. Maintenant, si le verrou posé sur les lignes d'index n'empêche pas l'utilisation des intervalles entre les lignes d'index, un nouvel enfant peut être inséré dans la table durant la lecture. Et maintenant, si ma transaction exécute la commande :

```
SELECT * FROM CHILD WHERE ID > 100 FOR UPDATE;
```

je vais trouver un nouvel enfant dans le résultat de ma requête. Ceci va à l'encontre du principe d'isolation des transactions : une transaction doit être capable de s'exécuter sans que les lectures soient affectées durant ce temps. Si vous considérons un intervalle de données, alors la nouvelle ligne 'fantôme' va casser le principe d'isolation.

Lorsque `InnoDB` scanne un index, il va aussi verrouille l'espace après la dernière ligne de l'index. C'est ce qui est arrivé dans l'exemple ci−dessus : le verrou posé par `InnoDB` va éviter qu'une insertion n'intervienne dans la table où `ID` serait plus grand que 100.

Vous pouvez utiliser le verrouillage de la clé suivant pour implémenter des vérifications d'unicité dans votre application : si vous lisez des données en mode partagé, et que vous ne voyez pas de duplicata de la ligne que vous allez insérer, alors vous pouvez l'insérer sans problème, en sachant que le verrou de clé suivante va vous garantir que durant ce temps, personne ne pourra insérer de ligne, qui déboucherait sur un duplicata de la votre. Le verrou de clé suivante permet de verrouiller aussi la non−existence de ligne dans votre table.

#### 7.5.8.4 Les verrous posés par différentes requêtes SQL avec InnoDB

- `SELECT ... FROM ...` : ceci est une lecture cohérente, qui lit un bilan de la base, et ne pose aucun verrou.
- `SELECT ... FROM ... LOCK IN SHARE MODE` : pose un verrou partagé sur la prochaine clé sur tous les index que la lecture rencontre.
- `SELECT ... FROM ... FOR UPDATE` : pose un verrou exclusif sur la prochaine clé sur tous les index que la lecture rencontre.
- `INSERT INTO ... VALUES (...)` : pose un verrou exclusif sur la ligne insérée. Notez que ce verrou n'est pas un verrou de clé, et il n'empêche pas les autres utilisateurs d'insérer des lignes. Si une erreur de clé double apparaît, un verrou sera posé partagé sera posé sur la ligne doublon.
- `INSERT INTO T SELECT ... FROM S WHERE ...` : pose un verrou exclusif sur chaque ligne inséré dans `T` . Effectue la recherche sur `S` sous la forme d'une lecture cohérente, mais pose un verrou partagé sur l'index de prochaine clé de `S` si MySQL a activé le log. `InnoDB` doit poser un verrou dans cette dernière situation, car en cas d'exécution des instructions dans une phase de restauration, toutes les requêtes doivent être exécutées dans le même ordre.
- `CREATE TABLE ... SELECT ...` effectue une commande `SELECT` sous la forme d'une lecture cohérente, ou avec des verrous partagés, comme précédemment.
- `REPLACE` est similaire à une insertion, si il n'y a pas de collision sur la clé unique. Sinon, une verrou exclusif sur l'index de prochaine clé est posé sur la ligne qui sera modifiée.
- `UPDATE ... SET ... WHERE ...` : pose un verrou exclusif sur l'index de prochaine clé, à chaque ligne que la recherche trouve.
- `DELETE FROM ... WHERE ...` : pose un verrou exclusif sur l'index de prochaine clé à chaque ligne que la recherche trouve.
- Si la contrainte de `FOREIGN KEY` est définie sur une table, toute insertion, modification ou effacement qui requiert la vérification de la contrainte va poser un verrou de ligne sur la ligne

dont il doit vérifier la contrainte. De plus, dans certains cas où la contrainte échoue, InnoDB pose ces verrous.

- LOCK TABLES ... : pose un verrou de table. L'implémentation de la couche MySQL pose ce verrou. La détection automatique des blocages de InnoDB ne peut détecter les blocages lorsque de tels verrous sont posés. Voyez la section suivante. De plus, comme MySQL ne connaît pas le verrouillage de lignes, il est possible que vous posiez un verrou sur une table où un autre utilisateur a déjà posé un verrou. Mais cela ne pose pas de problème quant à l'intégrité de la requête. Restrictions sur les tables InnoDB .

### 7.5.8.5 Détection des blocages et annulation

InnoDB détecte automatiquement les blocages de transactions et annule une ou plusieurs transactions pour l'éviter. Depuis la version 4.0.5, InnoDB va essayer d'annuler les petites transactions. La taille de la transaction est déterminée par le nombre de lignes qu'elle a inséré, modifié ou effacé. Avant la version 4.0.5, InnoDB annulait toujours la transaction qui avait posé le dernier verrou avant le blocage, c'est à dire, un cycle dans le graphe des transactions.

InnoDB ne peut pas détecter les blocages causés par la commande MySQL LOCK TABLES , ou si un verrou est posé par un autre gestionnaire de table que InnoDB . Vous devez résoudre ces situations avec l'option innodb_lock_wait_timeout du fichier de configuration my.cnf .

Lorsque InnoDB effectue une annulation de transaction, tous les verrous de cette transaction sont libérés. Cependant, si une commande SQL est annulée pour cause d'erreur, certains verrous de la transaction peuvent être conservés. Deci est dû au fait que InnoDB enregistre les verrous dans un format qui ne permet pas de savoir qui l'a posé.

### 7.5.8.6 Un exemple de lecture cohérente avec InnoDB

Supposons que vous exécutez MySQL avec le niveau d'isolation des transactions de REPEATABLE READ . Lorsque vous demandez une lecture cohérente avec une commande SELECT ordinaire, InnoDB va donner à votre transaction une date jalon, en fonction de laquelle votre requête va voir la base. Ainsi, si une transaction B efface une ligne après l'assignation de ce jalon, vous ne verrez pas cette ligne. De même pour une insertion ou une modification.

Vous pouvez faire avancer votre date jalon en validant votre transaction et en exécutant un autre SELECT .

Cela s'appelle le contrôle simultané multi−versions.

```
                    User A                 User B

                SET AUTOCOMMIT=0;      SET AUTOCOMMIT=0;
time
|             SELECT * FROM t;
|             empty set
|                                   INSERT INTO t VALUES (1, 2);
|
v             SELECT * FROM t;
              empty set
                                   COMMIT;

               SELECT * FROM t;
              empty set;

               COMMIT;

               SELECT * FROM t;
              ---------------------
              |    1    |    2    |
              ---------------------
```

De cette façon, A voir la ligne insérée par B uniquement lorsque B a validé son insertion, et que A a validé sa propre transaction, de façon à ce que la date jalon soit plus récente que la validation de B.Si vous voulez avoir une vision aussi ``fraîche'' que possible de votre base, vous devez utiliser le verrou en lecture :

```
SELECT * FROM t LOCK IN SHARE MODE;
```

### 7.5.8.7 Comment gérer les blocages de verrous?

Les blocages de verrous sont un problème classique des bases de données transactionnelles, mais ils ne sont pas dangereux, à moins qu'ils ne se répètent si souvent que vous ne puissiez pas exécuter tranquillement certaines transactions. Normalement, vous devriez écrire vos applications de manière à ce qu'elles soient prêtes à tenter à nouveau une transaction si la transaction est annulée pour cause de blocage.

`InnoDB` utilise un verrouillage de lignes automatique. Vous pouvez obtenir des blocages sur une ligne, même si votre transactions ne fait que modifier ou insérer une seule ligne. Cela est dû au fait que les opérations ne sont pas réellement 'atomiques' : elles posent automatiquement des verrous (éventuellement plusieurs) sur les lignes d'index de l'enregistrement concerné.

Vous pouvez gérer ces blocages et réduire leur nombre avec les trucs suivants :

- Utilisez la commande `SHOW INNODB STATUS` avec MySQL version supérieure à 3.23.52 et 4.0.3, pour déterminer la cause du dernier blocage. Cela peut vous aider à optimiser votre application pour les éviter.
- Soyez toujours prêts à tenter une nouvelle fois une transaction si elle échoue à cause d'un blocage. Les verrous ne sont pas dangereux. Essayez juste une autre fois.
- Validez souvent vos transactions. Les petites transactions sont moins sujettes aux blocages.
- Si vous utilisez des verrous en lectures avec `SELECT ... FOR UPDATE` ou `... LOCK IN SHARE MODE` , essayez d'utiliser un niveau d'isolation plus bas comme `READ COMMITTED` .
- Accédez à vos tables et lignes dans un ordre fixé. Les transactions vont alors former des queues, et non pas des blocages.
- Ajoutez de bons index à vos tables. Vos requêtes devront scanner moins souvent les tables, et poseront donc moins de verrous. Utilisez `EXPLAIN SELECT` pour déterminer si MySQL choisit les bons index pour vos requêtes.
- Limitez votre utilisation des verrous. Si vous pouvez vous permettre de faire retourner à une commande `SELECT` des données un peu anciennes, n'ajoutez pas la clause `FOR UPDATE` ou `LOCK IN SHARE MODE` . Utiliser le niveau d'isolation `READ COMMITTED` est bon ici, car chaque lecture cohérente dans la même transaction lira avec des données aussi fraîches que possible à chaque fois.
- En dernier recours, vous pouvez forcer les verrous avec la commande : `LOCK TABLES t1 WRITE, t2 READ, ... ; [faire quelquechose avec les tables t1 et t2]; UNLOCK TABLES` . Les verrous de niveau de table forcent les transactions à se mettre en ligne, et les blocages sont évités. Notez que `LOCK TABLES` démarre implicitement une transaction, tout comme `BEGIN` , et `UNLOCK TABLES` termine une transaction avec un `COMMIT` .
- Une dernière solution est de créer un sémaphore auxiliaire sous la forme d'une table avec une seule ligne. Chaque transaction modifie cette table avant d'accéder aux autres tables. Dans ce cas, toutes les transactions se font en ordre séquentiel. Notez que dans cette configuration, même l'algorithme `InnoDB` de détection des blocages fonctionne, car le sémaphore est un verrou de ligne. Avec les verrous de niveau de table de MySQL, nous devons nous résoudre à une méthode de délai d'expiration pour résoudre un verrou.

### 7.5.8.8 Conseils sur l'amélioration des performances InnoDB

**1.** Si l'outil Unix `top` ou si le `Task Manager` de Windows montre que l'utilisation du CPU, lors de la charge de travail, est inférieure à 70%, votre calcul est probablement limité par les disques. Vous faites

peut être trop d'écriture de transaction, ou le tampon de traitement ("buffer pool") est peut être trop petit. Augmenter la taille du tampon peut aider, mais il ne faut pas qu'il dépasse 80% de la mémoire physique.

**2.** Regrouper les modifications dans une seule transaction. `InnoDB` doit Wrap several modifications into one transaction. `InnoDB` must flush the log to disk at each transaction commit, if that transaction made modifications to the database. Since the rotation speed of a disk is typically at most 167 revolutions/second, that constrains the number of commits to the same 167/second if the disk does not fool the operating system.

**3.** If you can afford the loss of some latest committed transactions, you can set the `my.cnf` parameter `innodb_flush_log_at_trx_commit` to zero. `InnoDB` tries to flush the log anyway once in a second, though the flush is not guaranteed.

**4.** Make your log files big, even as big as the buffer pool. When `InnoDB` has written the log files full, it has to write the modified contents of the buffer pool to disk in a checkpoint. Small log files will cause many unnecessary disk writes. The drawback in big log files is that recovery time will be longer.

**5.** Also the log buffer should be quite big, say 8 MB.

**6.** (Relevant from 3.23.39 up.) In some versions of Linux and Unix, flushing files to disk with the Unix `fdatasync` and other similar methods is surprisingly slow. The default method `InnoDB` uses is the `fdatasync` function. If you are not satisfied with the database write performance, you may try setting `innodb_flush_method` in `my.cnf` to `O_DSYNC` , though O_DSYNC seems to be slower on most systems.

**7.** In importing data to `InnoDB` , make sure that MySQL does not have `autocommit=1` on. Then every insert requires a log flush to disk. Put before your plain SQL import file line

```
SET AUTOCOMMIT=0;
```

and after it

```
COMMIT;
```

If you use the `mysqldump` option `--opt` , you will get dump files which are fast to import also to an `InnoDB` table, even without wrapping them to the above `SET AUTOCOMMIT=0; ... COMMIT;` wrappers.

**8.** Beware of big rollbacks of mass inserts: `InnoDB` uses the insert buffer to save disk I/O in inserts, but in a corresponding rollback no such mechanism is used. A disk−bound rollback can take 30 times the time of the corresponding insert. Killing the database process will not help because the rollback will start again at the database startup. The only way to get rid of a runaway rollback is to increase the buffer pool so that the rollback becomes CPU−bound and runs fast, or delete the whole `InnoDB` database.

**9.** Beware also of other big disk−bound operations. Use `DROP TABLE` or `TRUNCATE` (from MySQL−4.0 up) to empty a table, not `DELETE FROM yourtable` .

**10.** Use the multi−line `INSERT` to reduce communication overhead between the client and the server if you need to insert many rows:

```
INSERT INTO yourtable VALUES (1, 2), (5, 5);
```

This tip is of course valid for inserts into any table type, not just `InnoDB` .

### 7.5.8.9 Le moniteur InnoDB

Depuis la version 3.23.41, InnoDB inclut le moniteur InnoDB (InnoDB Monitor) qui affiche des informations sur l'état interne des tables InnoDB. Lorsqu'il est activé, le moniteur InnoDB va faire afficher des données au serveur MySQL `mysqld` sur la sortie standard (note : le client MySQL ne va rien afficher du tout), toutes les 15 secondes. Ces données sont très intéressantes pour optimiser les performances. Sous Windows, vous devez démarrer `mysqld-max` depuis la ligne de commande avec les options `--standalone --console` pour diriger le résultat de l'affichage à la console MS-DOS.

Il existe aussi `innodb_lock_monitor` qui affiche les mêmes informations que `innodb_monitor` , mais qui indique aussi les verrous posés par les transactions.

Les informations affichées inclut des données sur :

- les attentes de verrous par les transactions,
- Les attentes de sémaphores pour les threads,
- les requêtes en attente d'accès aux fichiers,
- les statistiques sur les buffers, et
- les statistiques sur les purges et buffers d'insertion sur le thread principal InnoDB.

Vous pouvez démarrer le moniteur InnoDB avec la commande SQL suivante :

```
CREATE TABLE innodb_monitor(a int) type = innodb;
```

et le stopper avec

```
DROP TABLE innodb_monitor;
```

La syntaxe `CREATE TABLE` est simplement un moyen de passer une commande à une table InnoDB, via l'analyseur MySQL : la table créée n'est pas importante pour le moniteur InnoDB. Si vous interrompez le serveur lorsque le moniteur fonctionne, et que vous voulez redémarrer le moniteur, vous devez alors effacer la table avant de pouvoir exécuter la même commande `CREATE TABLE` pour démarrer le moniteur. Cette syntaxe est susceptible de changer dans le futur.

Un exemple de résultat du moniteur InnoDB :

```
 ================================
010809 18:45:06 INNODB MONITOR OUTPUT
================================
--------------------------
LOCKS HELD BY TRANSACTIONS
--------------------------
LOCK INFO:
Number of locks in the record hash table 1294
LOCKS FOR TRANSACTION ID 0 579342744
TABLE LOCK table test/mytable trx id 0 582333343 lock_mode IX

 RECORD LOCKS space id 0 page no 12758 n bits 104 table test/mytable index
PRIMARY trx id 0 582333343 lock_mode X
Record lock, heap no 2 PHYSICAL RECORD: n_fields 74; 1-byte offs FALSE;
info bits 0
 0: len 4; hex 0001a801; asc ;; 1: len 6; hex 000022b5b39f; asc ";;
 2: len 7; hex 000002001e03ec; asc ;; 3: len 4; hex 00000001;
...
-----------------------------------------------
CURRENT SEMAPHORES RESERVED AND SEMAPHORE WAITS
-----------------------------------------------
SYNC INFO:
Sorry, cannot give mutex list info in non-debug version!
Sorry, cannot give rw-lock list info in non-debug version!
-------------------------------------------------------
SYNC ARRAY INFO: reservation count 6041054, signal count 2913432
4a239430 waited for by thread 49627477 op. S-LOCK file NOT KNOWN line 0
Mut ex 0 sp 5530989 r 62038708 sys 2155035;
rws 0 8257574 8025336; rwx 0 1121090 1848344
--------------------------------------------------------
CURRENT PENDING FILE I/O'S
--------------------------
Pending normal aio reads:
Reserved slot, messages 40157658 4a4a40b8
Reserved slot, messages 40157658 4a477e28
...
Reserved slot, messages 40157658 4a4424a8
```

```
Reserved slot, messages 40157658 4a39ea38
Total of 36 reserved aio slots
Pending aio writes:
Total of 0 reserved aio slots
Pending insert buffer aio reads:
Total of 0 reserved aio slots
Pending log writes or reads:
Reserved slot, messages 40158c98 40157f98
Total of 1 reserved aio slots
Pending synchronous reads or writes:
Total of 0 reserved aio slots
-----------
BUFFER POOL
-----------
LRU list length 8034
Free list length 0
Flush list length 999
Buffer pool size in pages 8192
Pending reads 39
Pending writes: LRU 0, flush list 0, single page 0
Pages read 31383918, created 51310, written 2985115
----------------------------
END OF INNODB MONITOR OUTPUT
============================
010809 18:45:22 InnoDB starts purge
010809 18:45:22 InnoDB purged 0 pages
```

Quelques notes sur le résultat :

- Si la section LOCKS HELD BY TRANSACTIONS rapporte des attentes de verrous, alors votre application a des problèmes de rétention de verrous. Le rapport peut vous aider à trouver les blocages de verrous, et les transactions bloquantes.
- La section SYNC INFO va vous indiquer les sémaphores réservés si vous compilez InnoDB avec la constante `UNIV_SYNC_DEBUG` définie dans le fichier `univ.i` .
- La section SYNC ARRAY INFO rapporte les threads en attente d'un sémaphore, et les statistiques d'attentes pour un mutex ou un verrou en lecture/écriture. Les longues attentes peuvent être dues à des requêtes paralèlles, ou des problèmes avec la programmation des threads.
- La section CURRENT PENDING FILE I/O'S list les demandes d'accès aux fichiers. Un nombre important indique que le disque est un facteur limitant dans votre installation.
- La section BUFFER POOL vous donne des statistiques sur les pages qui ont été lues et écrites. Vous pouvez calculer depuis ces nombres combien d'accès disques vos requêtes réalisent actuellement.

## 7.5.9 Implémentation du multi−versionnage

Comme `InnoDB` est une base de donnes multi−versionnée, elle doit conserver des informations des vieilles versions des lignes dans l'espace de tables. Cette information est stockée dans la structure de données que nous appelons un segment d'annulation, d'après la structure similaire d'Oracle.

En interne, `InnoDB` ajoute deux champs à chaque ligne stockée dans la base. Un champ de 6 octets note l'identifiant de la dernière transaction qui a inséré ou modifier la ligne. De plus, un effacement est traité en interne comme une modification, où un bit spécial dans la ligne sert à marquer la ligne comme effacée. Chaque ligne contient aussi une colonne de 7 octets appelé un pointeur d'annulation. Ce pointeur fait référence à une ligne dans un fichier d'annulation qui contient les informations nécessaires pour reconstruire le contenu de la ligne qui a été modifiée.

`InnoDB` utilise les informations dans le segment d'annulation pour effectuer les opérations d'annulation nécessaires dans une annulation de transaction. Il utilise aussi ces informations pour reconstruire les anciennes versions d'une ligne lors d'une lecture cohérente.

Le log d'annulation dans le segment d'annulation est divisé entre les insertions et les modifications. Le log d'insertion n'est nécessaire que lors des annulations de transactions, et peut être vidé dès que la requête est validée. Le log de modification est utilisé lors des lectures cohérentes, et les

informations y sont supprimées une fois que toutes les transactions qui font une lecture cohérente sont terminées.

Vous devez vous rappeler que valider vos transactions régulièrement, même ces transactions qui ne font que des lectures cohérentes. Sinon, `InnoDB` ne peut pas vider le log de modification, et le segment d'annulation va grossir énormément.

La taille physique d'une ligne du log d'annulation dans le segment d'annulation est typiquement plus petite que la ligne correspondante insérée ou modifiée. Vous pouvez utiliser ces informations pour calculer l'espace nécessaire à vos segments d'annulation.

Dans nos schémas de multi−versionnage, une ligne n'est pas physiquement supprimée de la table immédiatement lorsque vous l'effacez avec une requête SQL. Uniquement lorsque `InnoDB` va supprimer les lignes du log de modification, il vaut aussi supprimer physiquement la ligne, et les index. Cette opération d'effacement est appelée une purge, et elle est plutôt rapide, et aussi rapide que la requête SQL elle−même.

## 7.5.10 Structures de tables et d'index

MySQL enregistre la structure de table dans le fichier `.frm` , du dossier de données. Mais les tables InnoDB ont aussi leur propre entrée dans les tables internes InnoDB. Lorsque MySQL efface une table ou une base, il efface le ou les fichiers `.frm` et aussi les lignes correspondantes dans les tables d'administration InnoDB. C'est la raison qui fait que vous ne pouvez pas déplacer les tables InnoDB entre les bases simplement en déplaçant le fichier `.frm` , et pourquoi `DROP DATABASE` ne fonctionnait pas sous InnoDB en MySQL versions <= 3.23.43.

Chaque table InnoDB a un index spécial appelé un index en grappe, où les données des lignes sont enregistrées. Si vous définissez une clé primaire pour votre table `PRIMARY KEY` , alors l'index de la clé primaire de la table sera un index en grappe.

Si vous ne définissez pas de clé primaire pour votre table, InnoDB va générer un index en grappe, ou les lignes sont ordonnées dans l'ordre des identifiants que InnoDB assigne aux lignes de la table. L'identifiant de ligne vaut 6 octets, et s'accroît au fur et à mesure que les lignes sont ajoutées. Les lignes sont alors ordonnées dans leur ordre d'insertion.

Accéder à une ligne via l'index en grappe est rapide, car la ligne de données sera dans la même page que l'index. Dans de nombreuses bases, les données sont traditionnellement stockées dans un autre endroit. Si la table est grand, l'index en grappe économise de nombreux accès disques, comparativement aux solutions traditionnelles.

Les lignes des index qui ne sont pas en grappe (ce sont les index secondaires) dans InnoDB, contiennent la valeur de la clé primaire de la ligne. InnoDB utilise cette clé primaire pour rechercher la valeur de la ligne dans l'index en grappe. Notez que si la clé primaire est longue, les index secondaires utiliseront plus de place.

### 7.5.10.1 Structure physique d'un index

Tous les index de InnoDB sont des B−trees où les lignes d'index sont stockées dans un noeud terminal de l'arbre. La taille par défaut d'une page d'index est de 16ko. Lorsque de nouvelles lignes sont insérées, InnoDB essaie de laisser 1 / 16 de la page de libre pour les prochaines insertions et modifications dans les lignes d'index

Si les lignes d'index sont insérées dans un ordre séquentiel (croissant ou décroissant), les pages d'index résultantes seront environs pleines à 15/16. Si les lignes sont insérées dans un ordre aléatoire, les pages seront pleines de 1/2 à 15/16. Si le taux de remplissage d'une page d'index tombe à 1/2, InnoDB va essayer de contracter l'arbre d'index pour libérer la page.

### 7.5.10.2 Bufferisation des insertions

Une situation courante dans les applications de base de données apparaît lorsque la clé primaire est un identifiant unique, et que les nouvelles lignes sont insérées dans un ordre ascendant. Par conséquent, les insertions dans l'index en grappe ne nécessitent pas de lectures aléatoires dans le disque.

D'un autre coté, les index secondaires sont généralement non−uniques, et les insertions surviennent dans un ordre aléatoire. Cela causerait de nombreux accès disques aléatoire, si InnoDB de disposait pas d'un mécanisme spécial.

Si une ligne doit être insérée dans un index secondaire non unique, InnoDB vérifie si la page d'index fait partie du buffer. Dans ce cas, InnoDB va faire directement l'insertion dans une structure de buffer destinée à l'insertion. Le buffer d'insertion est conservé petit, pour qu'il reste dans le buffer général, et les insertions sont faites très vite.

Le buffer d'insertion est périodiquement fusionné avec l'arbre d'index secondaires dans la base. Souvent, nous fusionnons plusieurs insertions dans la même page de l'arbre d'index, et donc, nous économisons des accès disques. Il a été mesuré que les insertions sont jusqu'à 15 fois plus rapides de cette façon.

### 7.5.10.3 Index hash adaptatifs

Si une base de données est suffisamment petite pour tenir en mémoire, alors le plus rapide pour faire des requêtes est d'utiliser les index hash. InnoDB a un mécanisme automatique pour surveiller les recherches utilisant les index d'une table, et si InnoDB remarque que la requête pourrait profiter d'un index hash, un tel index est automatiquement constitué.

Mais notez que les index hash sont toujours bâtis à partir d'un index B−tree existant. InnoDB peut bâtir un index hash sur un préfixe de taille arbitraire de clé B−tree, suivant le modèle de recherche que InnoDB remarque dans l'index B−tree. Un index hash peut être partiel : il n'est pas obligatoire que tout l'index B−tree soit mis en cache dans le pool. InnoDB va bâtir des index hash à la demande pour les tables dont les index sont souvent sollicités.

En un sens, grâce au mécanisme d'index hash adaptatif, InnoDB s'adapte tout seul à la mémoire interne, et se rapproche des architectures de bases en mémoire vive.

### 7.5.10.4 Structure physique d'une ligne

- Chaque ligne d'index de InnoDB contient un entête de 6 octets. L'entête est utilisé pour lier des lignes consécutives ensembles, et aussi pour le verrouillage de ligne.
- Les lignes dans un index en grappe contient des champs pour toutes les colonnes définies par l'utilisateur. De plus, il y a un champ de 6 octets pour l'identification de transaction, et un champs de 7 octets pour le pointeur d'annulation.
- Si l'utilisateur n'a pas défini de clé primaire pour la table, chaque ligne de l'index en grappe contient aussi une colonne supplémentaire de 6 octets, qui sert d'identification.

- Chaque ligne d'index secondaire contient aussi les champs définis pour la clé de l'index en grappe.
- Une ligne contient aussi un pointeur pour chaque champs de la ligne. Si la taille totale des champs représentent moins de 128 octets, alors le pointeur fait 1 octets, sinon 2.

**Comment une colonne de type auto−increment fonctionne avec InnoDB**

Après le démarrage de la base, lorsqu'un utilise faire un premier insert dans la table `T` où une colonne de type auto−increment a été définie, et que l'utilisateur ne fournit pas de valeur pour la colonne, alors InnoDB exécute la requête `SELECT MAX(auto−inc−column) FROM T` , et assigne la valeur incrémentée de 1 dans la colonne, et dans le compteur d'auto−incrémente de la table. Nous disons alors que le compteur de la table `T` a été initialisé.

InnoDB suit la même procédure dans l'initialisation du compteur d'une nouvelle table.

Notez que si l'utilisateur spécifie la valeur de 0 dans l'insertion, InnoDB le traite comme une valeur non définie.

Une fois que le compteur de table a été initialisé, si l'utilisateur insère une colonne dans laquelle il spécifie explicitement une valeur, et que la valeur est plus grande que la valeur du compteur, alors le compteur prend cette valeur spécifiée. Si l'utilisateur ne fournit pas de valeur, alors InnoDB incrémente le compteur de 1, et assigne la nouvelle valeur à la colonne.

Le mécanisme auto−incrément, lorsque vous assignez une valeur au compteur, ignore le verrouillage et la gestion de transaction. Par conséquent, vous pouvez avoir des trous dans le numéro de séquence, si vous annulez une transaction qui a modifié la valeur du compteur.

Le comportement des colonnes auto−incrément n'est pas définie si un utilisateur fourni un nombre négatif pour la colonne, ou si la valeur est plus grande que le plus grand des entiers que peut stocker la colonne.

## 7.5.11 Gestion de l'espace fichiers et des entrées/sorties disque

### 7.5.11.1 Accès disques

Lors des accès disque, `InnoDB` utilise un accès asynchrone. Sous Windows NT, il utilise le mode natif asynchrone, fourni par le système d'exploitation. Sous Unix, `InnoDB` utilise un mode asynchrone simulé, intégré à `InnoDB` : `InnoDB` crée un nombre de threads d'accès disques pour prendre en charge les opérations disques, comme les lectures. Dans les prochaines versions, nous allons ajouter le support des accès disques simulés à Windows NT et les accès natifs à Unix.

Sous Windows NT, `InnoDB` utilise des accès non bufferisés. Cela signifie que les pages disques que `InnoDB` lit ou écrit ne sont pas mis en cache par le système d'exploitation. Cela économise de la mémoire.

Depuis la version 3.23.41, `InnoDB` utilise une nouvelle technique de flush de fichier, appelée doublewrite. Elle apporte de la sécurité lors de la restauration après crash du système d'exploitation, ou un problème électrique, et améliore les performances sous Unix, pour plusieurs distributions, en réduisant le besoin de synchronisation.

Doublewrite (Double écriture, en français) signifie que `InnoDB` , avant d'écrire les pages dans le fichier de données, les écrits dans une zone continue d'espace, appelée un buffer de double écriture. Une

fois que cette écriture est faite, que le buffer de double écriture a été vidé, InnoDB écrit les pages à leur destination finale. Si le système d'exploitation crashe entre temps, InnoDB va trouver une bonne copie des données dans le buffer, lors de la restauration.

Depuis la version 3.23.41, vous pouvez aussi utiliser une partition brute de disque comme fichier de données, même si cela n'a pas été testé pour le moment. Lorsque vous crée un nouveau fichier de données, vous devez ajouter le mot clé newraw immédiatement après la taille du fichier de données, dans innodb_data_file_path . La partition doit être supérieure ou égale à cette taille. Notez que 1M pour InnoDB est 1024 x 1024 octets, alors que les spécifications de disques utilisent souvent la convention de 1 Mo vaut 1000 000 octets (un million).

```
innodb_data_file_path=hdd1:5Gnewraw;hdd2:2Gnewraw
```

Lorsque vous démarrez la base à nouveau, vous **devez** changer le mot clé à raw . Sinon, InnoDB va écrire sur votre partition!

```
innodb_data_file_path=hdd1:5Graw;hdd2:2Graw
```

En utilisant un disque brut, vous pouvez réaliser des opérations non bufferisée sous Unix.

InnoDB dispose de deux heuristiques de lectures : lecture séquentielle et lecture aléatoire. En mode séquentiel, InnoDB remarque que l'accès à un segment de la table est séquentiel. Il va alors anticiper, et faire des lectures de plusieurs pages dans la base. En mode aléatoire, InnoDB note qu'une partie de la table est lue dans son entier. Il va alors anticiper, et faire automatiquement les dernières lectures.

### 7.5.11.2 Gestion de l'espace fichier

Les fichiers de données que vous définissez dans le fichier de configuration forment l'espace de données InnoDB . Les fichiers sont simplement concaténés pour former un espace de données, et il n'y a pas de parallélisme utilisé. Actuellement, vous ne pouvez pas spécifier où l'espace est alloué pour vos tables, hormis en utilisant la méthode suivante : pour chaque nouvel espace de données créé, InnoDB va allouer de l'espace en partant depuis la fin.

L'espace de tables est constitué de pages de taille 16 ko. Les pages sont groupées en ensembles de 64 pages consécutives. Les 'fichiers' dans un espace de tables sont appelés des segments en InnoDB . Le nom du segment d'annulation est un peu trompeur, car il contient en fait de nombreux segments dans l'espace de table.

Pour chaque index de InnoDB , nous allons créer deux segments : un pour les noeuds non terminaux du B tree, et un autre pour les noeuds terminaux. L'idée ici est d'améliorer la séquence des noeuds terminaux, qui contiennent les données.

Lorsqu'un segment croit dans un espace de table, InnoDB alloue les 32 premières pages spécifiquement pour un segment. Après cela, InnoDB commence à allouer des extensions au segment. InnoDB peut ajouter aux grands segments jusqu'à 4 extension en même temps, pour améliorer la séquence de données.

Certaines pages dans l'espace de table contiennent des cartes des autres pages, et donc, quelques extensions dans un espace de table InnoDB ne pourront pas être allouées en tant que segment, mais comme pages individuelles.

Lorsque vous exécutez une commande SHOW TABLE STATUS FROM ... LIKE ... pour demander quel est l'espace libre dans la table, InnoDB va rapporter les extensions qui sont encore totalement libre. InnoDB se réserve toujours quelques extensions pour la gestion interne. Ces extensions réservées ne sont pas inclues dans l'espace libre.

Lorsque vous effacez des donnes d'une table, `InnoDB` va contracter les index B−tree correspondant. Suivant les effacements qui vont libérer des pages individuelles ou des extensions, de l'espace sera rendu aux autres tables. Effacer une table ou en effacer toutes les lignes va rendre obligatoirement de l'espace aux autres tables, mais souvenez vous que les lignes effacées ne pourront être physiquement effacée qu'après une opération de purge, si aucune transaction n'en a plus besoin.

#### 7.5.11.3 Défragmentation des tables

S'il y a plusieurs insertions et suppressions dans les index d'une table, les index peuvent devenir fragmentés. Par fragmentation, nous voulons dire que l'ordre physique de la page d'index n'est pas proche de l'ordre alphabétique des enregistrements dans les pages, ou qu'il y a plusieurs pages non−utilisées dans le block de 64 pages qui ont été allouées à l'index.

Une manière d'accélérer les index est d'extraire périodiquement les données de la table dans un fichier avec `mysqldump` , d'effacer la table puis de la recréer.

Une autre manière de défragmenter consiste à exécuter un `ALTER` sur le type de la table pour le changer en `MyISAM` puis de le repasser en `InnoDB` .

Notez qu'un table `MyISAM` doit pouvoir tenir sur un seul fichier de votre système d'exploitation.

Si les insertions dans un index sont toujours ascendantes et que les lignes supprimées le sont à la fin du fichier, alors l'algorithme de gestion de l'espace fichiers de `InnoDB` garantit qu'aucune fragmentation n'aura lieu dans l'index.

## 7.5.12 Gestion des erreurs

La gestion des erreurs avec `InnoDB` n'est pas toujours la même que celle spécifiée par les standards ANSI SQL. En accord avec les standards ANSI, toute erreur survenant durant une commande SQL devrait provoquer l'annulation de celle−ci. `InnoDB` n'annule qu'une partie de la commande de temps en temps, ou la totalité de la transaction. La liste suivante spécifie la gestion des erreurs de `InnoDB` .

- Si vous dépassez l'espace d'un fichier dans l'espace des tables, vous obtiendrez l'erreur MySQL `'Table is full'` et `InnoDB` annulera la requête SQL.
- Un deadlock de transaction ou un dépassement de temps dans un attente de verrouillage fait annuler toute la transaction à `InnoDB` .
- Une erreur de clef dupliquée annule l'insertion de la ligne concernée, même dans une requête du genre `INSERT INTO ... SELECT ...` . Cela fera probablement en sorte que la commande SQL sera annulée si vous n'avez pas spécifié l'option `IGNORE` dans votre requête.
- Une erreur 'row too long' annule la commande SQL.
- Les autres erreurs sont la plupart du temps détectées au niveau de la couche MySQL du code et annulent la commande SQL correspondante.

## 7.5.13 Restrictions sur les tables InnoDB

- **Attention** : **NE convertissez PAS** un système de tabel MyISAM en InnoDB! Cela n'est pas supporté. Si vous faîtes cela, MySQL ne va pas redémarrer jusqu'à ce que vous restauriez vos données avec une vieille sauvegarde, ou que vous regénériez ces tables avec le script `mysql_install_db` .
- `SHOW TABLE STATUS` ne donne pas de statistiques précises sur les tables InnoDB, hormis pour la taille physique réservée. Le nombre de lignes est seulement une approximation utilisée en

optimisation SQL.

- Si vous essayez de créer un index unique sur un préfixe d'une colonne, vous allez obtenir une erreur :

```
CREATE TABLE T (A CHAR(20), B INT, UNIQUE (A(5))) TYPE = InnoDB;
```

  Si vous créez un index non−unique sur un préfixe de colonne, InnoDB va créer un index pour toute la colonne.
- `INSERT DELAYED` n'est pas supportés par les tables InnoDB.
- La commande MySQL `LOCK TABLES` ne reconnait pas le verrouillage de ligne InnoDB réalisé dans les commandes SQL achevées : cela signifie que vous pouvez poser un verrou sur une table même si il existe une transaction qui a été posée par un autre utilisateur. Par conséquent, votre opération doit attendre que les autres tables soient libres, et elle peut aussi entrer en conflit avec une autre requête. De plus, un blocage de verrous est possible mais il ne met pas en danger l'intégrité des transactions, car le verrou de ligne posé par InnoDB se charge toujours de l'intégrité. Enfin, un verrou de table évite aux autre transactions de poser un verrou de ligne (en conflit avec le mode de verrous) sur la table.
- Vous ne pouvez pas avoir une clé sur une colonne de type `BLOB` et `TEXT` .
- Une table ne peut pas recevoir plus de 1000 colonnes.
- `DELETE FROM TABLE` ne regénère pas la table, mais au lieu de cela, il efface les lignes une à une, ce qui est bien plus lent. Dans les prochaines versions, MySQL va pouvoir utiliser la commande `TRUNCATE` qui est très rapide.
- La taille de page par défaut est de InnoDB est de 16 ko. En recompilant le code, vous pouvez lui changer la valeur de 8 ko à 64 ko. La taille maximale de ligne est légèrement inférieure à la moitié de la taille d'une page de base dans les versions inférieure à 3.23.40 de InnoDB. Depuis la versions source 3.23.41 BLOB et TEXT sont autorisé à être plus petits que 4 Go, la taille totale de ligne doit aussi être de moins de 4 Go. InnoDB ne stocke par les fichiers dont le taille est inférieure à 128 octets dans une page séparée. Après que InnoDB a modifié la ligne en stockant les grandes colonnes dans une page séparée, le reste de la taille de la chaîen doit être plus petite que la moitiée de la page de base. La taille maximale de clé est de 7000 octets.
- Sur certains systèmes d'exploitation, le fichier de données est limité à 2 Go. La taille combinée des fichiers de log doit être inférieure à 4 Go.
- La taille maximale d'un espace de tables est de 4 millions de pages. C'est aussi la taille maximale pour une table. La taille minimale pour une table est de 10 Mo.

## 7.5.14 Historique de l'évolution InnoDB

Note : la traduction de cette section va commencer au moment du démarrage de la traduction générale. Nous mettrons en version française les nouvelles entrées. Les anciennes versions seront traduites en fonction du temps disponible.

### MySQL/ `InnoDB` −4.0.5, November 18, 2002

- `InnoDB` supporte désormais le niveau d'isolation de transaction transaction READ COMMITTED et READ UNCOMMITTED. READ COMMITTED émule plus excactement Oracle et rend le portage des applications depuis Oracle bien plus faciles.
- La résolution des blocages de verrou est devenue sélective : nous essayons de choisir les transactions qui ont modifié le moins de ligne jusqu'à présent.
- La définition des FOREIGN KEY prend en compte l'option `lower_case_table_names` de `my.cnf` .
- La ocmmande SHOW CREATE TABLE n'affiche pas le nom de la base dans une définition de FOREIGN KEY si la table identifié est dans la même base que celle de la table principale.
- `InnoDB` fait un texte de cohérence pour la plupart des pages d'index avant de les écrire dans le fichier de données.

- Si vous utilisez innodb_force_recovery > 0, InnoDB essaie de passer outre les lignest et pages d'index corrompus, lors d'une commande SELECT * FROM dans une table. Cela aide lors de l'export.
- InnoDB utilise à nouveau les accès disques asynchrones sans buffer avec Windows 2000 et XP; seulement des accès disques avec simulation d'asynchronisme non bufferisé avec NT, 95/98/ME.
- Correction de bogue : l'estimateur de taille InnoDB exagérait grandement la taille d'un intervalle court d'index, si les chemins jusqu'au point de fin dans l'arbre d'index était relié à la racine. Cela conduisait à des scans de tables inutiles. La correction sera reportée à la version 3.23.54.
- Correction d'un bogue présent dans les versions 3.23.52, 4.0.3, 4.0.4 : Le démarrage d' InnoDB pouvait prendre beaucoup de temps, et même crasher sur Windows 95/98/ME.
- Correction d'un bogue : le verrou AUTO−INC était conservé jusqu'à la fin de la transaction si il avait été donné après une attete. Cela conduisait à des blocages inutiles.
- Correction d'un bogue : Si SHOW INNODB STATUS, innodb_monitor, ou innodb_lock_monitor devait afficher plusieurs centaines de transactions en un seul rapport, le résultat était tronqué, InnoDB s'arrêtait, affichait une erreur dans le log d'erreurs : log many waits for a mutex created at srv0srv.c, line 1621.
- Correction d'un bogue : SHOW INNODB STATUS sous Unix annoncait toujours une taille de lectures de fichier moyenne de 0 octets.
- Correction d'un bogue potentiel en 4.0.4: InnoDB fait désormais des clauses ORDER BY ... DESC comme MyISAM .
- Correction d'un bogue : DROP TABLE pour causer un crash ou un arrêt si il y avait des annulations simultanées sur la même table. La correction sera appliquée à la version 3.23 si cela apparaît comme un véritable problème.
- Correction d'un bogue : ORDER BY pouvait échouer, si vous n'aviez pas de clé primaire sur la table, mais que vous aviez défini plusieurs index, parmi lesquels au moins un index UNIQUE, et toutes les colonnes étaient NOT NULL.
- Correction d'un bogue : une attente de verrou expirée sur une connexion avec la commande ON DELETE CASCADE pouvait conduire à la corruption de la base.
- Correction d'un bogue : si une commande SELECT était faite avec une clé unique sur un index primaire, et que la recherche trouvait une ligne marquée pour l'effacement, InnoDB pouvait retourner la ligne suivante.
- Bug identifié : en 4.0.4, deux bogues ont été introduits avec les colonnes de type AUTO_INCREMENT . REPLACE peut faire que le compteur est corrompu, d'une unité sous la vraie valeur. Un blocage de verrous ou une expiration de pose de verrou peut causer le même problème. Cela sera corrigé en 4.0.6.

**MySQL/InnoDB−3.23.53, October 9, 2002**

- We again use unbuffered disk i/o to data files in Windows. Win XP and Win 2000 read performance seems to be very poor with normal i/o.
- Tuned range estimator so that index range scans are preferred over full index scans.
- Allow dropping and creating a table even if InnoDB _force_recovery is set. One can use this to drop a table which would cause a crash in rollback or purge, or if a failed table import causes a runaway rollback in recovery.
- Fixed a bogue present in 3.23.52, 4.0.3, 4.0.4: InnoDB startup could take very long or even crash on some Win 95/98/ME computers.
- Correction d'un bogue : fast shutdown (which is the default) sometimes was slowed down by purge and insert buffer merge.
- Correction d'un bogue : doing a big SELECT from a table where no rows were visible in a consistent read could cause a very long (> 600 seconds) semaphore wait in btr0cur.c line 310.
- Correction d'un bogue : the AUTO−INC lock was held to the end of the transaction if it was

granted after a lock wait. This could cause unnecessary deadlocks.
- Correction d'un bogue : if you created a temporary table inside LOCK TABLES, and used that temporary table, that caused an assertion failure in ha_innobase.cc.
- Correction d'un bogue : if SHOW INNODB STATUS, innodb_monitor, or innodb_lock_monitor had to print several hundred transactions in one report, and the output became truncated, InnoDB would hang, printing to the error log many waits for a mutex created at srv0srv.c, line 1621.
- Correction d'un bogue : SHOW INNODB STATUS on Unix always reported average file read size as 0 bytes.

**MySQL/InnoDB−4.0.4, October 2, 2002**

- We again use unbuffered disk i/o in Windows. Win XP and Win 2000 read performance seems to be very poor with normal i/o.
- Increased the max key length of InnoDB tables from 500 to 1024 bytes.
- Increased the table comment field in SHOW TABLE STATUS so that up to 16000 characters of foreign key definitions can be printed there.
- The auto−increment counter is no longer incremented if an insert of a row immediately fails in an error.
- Allow dropping and creating a table even if innodb_force_recovery is set. One can use this to drop a table which would cause a crash in rollback or purge, or if a failed table import causes a runaway rollback in recovery.
- Correction d'un bogue : Using ORDER BY primarykey DESC in 4.0.3 causes an assertion failure in btr0pcur.c, line 203.
- Correction d'un bogue : fast shutdown (which is the default) sometimes was slowed down by purge and insert buffer merge.
- Correction d'un bogue : doing a big SELECT from a table where no rows were visible in a consistent read could cause a very long (> 600 seconds) semaphore wait in btr0cur.c line 310.
- Correction d'un bogue : if the MySQL query cache was used, it did not get invalidated by a modification done by ON DELETE CASCADE or ...SET NULL.
- Correction d'un bogue : if you created a temporary table inside LOCK TABLES, and used that temporary table, that caused an assertion failure in ha_innodb.cc.
- Correction d'un bogue : if you set innodb_flush_log_at_trx_commit to 1, SHOW VARIABLES would show its value as 16 million.

**MySQL/InnoDB−4.0.3, August 28, 2002**

- Removed unnecessary deadlocks when inserts have to wait for a locking read, update, or delete to release its next−key lock.
- The MySQL HANDLER SQL commands now work also for InnoDB type tables. InnoDB does the HANDLER reads always as consistent reads. HANDLER is a direct access path to read individual indexes of tables. In some cases HANDLER can be used as a substitute of server−side cursors.
- Fixed a bogue in 4.0.2: even a simple insert could crash the AIX version.
- Correction d'un bogue : if you used in a table name characters whose code is > 127, in DROP TABLE InnoDB could assert on line 155 of pars0sym.c.
- Compilation from source now provides a working version both on HP−UX−11 and HP−UX−10.20. The source of 4.0.2 worked only on 11, and the source of 3.23.52 only on 10.20.
- Correction d'un bogue : if compiled on 64−bit Solaris, InnoDB produced a bus error at startup.

**MySQL/InnoDB−3.23.52, August 16, 2002**

- The feature set of 3.23 will be frozen from this version on. New features will go the the 4.0 branch, and only bogue fixes will be made to the 3.23 branch.
- Many CPU−bound join queries now run faster. On Windows also many other CPU−bound queries run faster.
- A new SQL command SHOW INNODB STATUS returns the output of the InnoDB Monitor to the client. The InnoDB Monitor now prints detailed info on the latest detected deadlock.
- InnoDB made the SQL query optimizer to avoid too much index−only range scans and choose full table scans instead. This is now fixed.
- "BEGIN" and "COMMIT" are now added in the binlog around transactions. The MySQL replication now respects transaction borders: a user will no longer see half transactions in replication slaves.
- A replication slave now prints in crash recovery the last master binlog position it was able to recover to.
- A new setting innodb_flush_log_at_trx_commit=2 makes InnoDB to write the log to the operating system file cache at each commit. This is almost as fast as the setting innodb_flush_log_at_trx_commit=0, and the setting 2 also has the nice feature that in a crash where the operating system does not crash, no committed transaction is lost. If the operating system crashes or there is a power outage, then the setting 2 is no safer than the setting 0.
- Added checksum fields to log blocks.
- SET FOREIGN_KEY_CHECKS=0 helps in importing tables in an arbitrary order which does not respect the foreign key rules.
- SET UNIQUE_CHECKS=0 speeds up table imports into InnoDB if you have UNIQUE constraints on secondary indexes.
- SHOW TABLE STATUS now lists also possible ON DELETE CASCADE or ON DELETE SET NULL in the comment field of the table.
- When CHECK TABLE is run on any InnoDB type table, it now checks also the adaptive hash index for all tables.
- If you defined ON DELETE CASCADE or SET NULL and updated the referenced key in the parent row, InnoDB deleted or updated the child row. This is now changed to conform to SQL−92: you get the error 'Cannot delete parent row'.
- Improved the auto−increment algorithm: now the first insert or SHOW TABLE STATUS initializes the auto−inc counter for the table. This removes almost all surprising deadlocks caused by SHOW TABLE STATUS.
- Aligned some buffers used in reading and writing to data files. This allows using unbuffered raw devices as data files in Linux.
- Correction d'un bogue : If you updated the primary key of a table so that only the case of characters changed, that could cause assertion failures, mostly in page0page.ic line 515.
- Correction d'un bogue : If you delete or update a row referenced in a foreign key constraint and the foreign key check has to wait for a lock, then the check may report an erroneous result. This affects also the ON DELETE... operation.
- Correction d'un bogue : A deadlock or a lock wait timeout error in InnoDB causes InnoDB to roll back the whole transaction, but MySQL could still write the earlier SQL statements to the binlog, even though InnoDB rolled them back. This could, for example, cause replicated databases to get out−of−sync.
- Correction d'un bogue : If the database happened to crash in the middle of a commit, then the recovery might leak tablespace pages.
- Correction d'un bogue : If you specified a non−latin1 character set in my.cnf, then, in contrary to what is stated in the manual, in a foreign key constraint a string type column had to have the same length specification in the referencing table and the referenced table.
- Correction d'un bogue : DROP TABLE or DROP DATABASE could fail if there simultaneously was a CREATE TABLE running.
- Correction d'un bogue : If you configured the buffer pool bigger than 2 GB in a 32−bit computer, InnoDB would assert in buf0buf.ic line 214.

- Correction d'un bogue : on 64−bit computers updating rows which contained the SQL NULL in some column could cause the undo log and the ordinary log to become corrupt.
- Correction d'un bogue : innodb_log_monitor caused a hang if it suppressed lock prints for a page.
- Correction d'un bogue : in the HP−UX−10.20 version mutexes would leak and cause race conditions and crashes in any part of InnoDB code.
- Correction d'un bogue : if you ran in the AUTOCOMMIT mode, executed a SELECT, and immediately after that a RENAME TABLE, then RENAME would fail and MySQL would complain about error 1192.
- Correction d'un bogue : if compiled on 64−bit Solaris, InnoDB produced a bus error at startup.

**MySQL/InnoDB−4.0.2, July 10, 2002**

- InnoDB is essentially the same as InnoDB−3.23.51.
- If no innodb_data_file_path is specified, InnoDB at the database creation now creates a 10 MB auto−extending data file ibdata1 to the datadir of MySQL. In 4.0.1 the file was 64 MB and not auto−extending.

**MySQL/InnoDB−3.23.51, June 12, 2002**

- Correction d'un bogue : a join could result in a seg fault in copying of a BLOB or TEXT column if some of the BLOB or TEXT columns in the table contained SQL NULL values.
- Correction d'un bogue : if you added self−referential foreign key constraints with ON DELETE CASCADE to tables and a row deletion caused InnoDB to attempt the deletion of the same row twice because of a cascading delete, then you got an assertion failure.
- Correction d'un bogue : if you use MySQL 'user level locks' and close a connection, then InnoDB may assert in ha_innobase.cc, line 302.

**MySQL/InnoDB−3.23.50, April 23, 2002**

- InnoDB now supports an auto−extending last data file. You do not need to preallocate the whole data file at the database startup.
- Made several changes to facilitate the use of the InnoDB Hot Backup tool. It is a separate non−free tool you can use to take online backups of your database without shutting down the server or setting any locks.
- If you want to run the InnoDB Hot Backup tool on an auto−extending data file you have to upgrade it to version ibbackup−0.35.
- The log scan phase in crash recovery will now run much faster.
- Starting from this server version, the hot backup tool truncates unused ends in the backup InnoDB data files.
- To allow the hot backup tool to work, on Windows we no longer use unbuffered i/o or native async i/o; instead we use the same simulated async i/o as on Unix.
- You can now define the ON DELETE CASCADE or ON DELETE SET NULL clause on foreign keys.
- FOREIGN KEY constraints now survive ALTER TABLE and CREATE INDEX.
- We suppress the FOREIGN KEY check if any of the column values in the foreign key or referenced key to be checked is the SQL NULL. This is compatible with Oracle, for example.
- SHOW CREATE TABLE now lists also foreign key constraints. Also mysqldump no longer forgets about foreign keys in table definitions.
- You can now add a new foreign key constraint with ALTER TABLE ... ADD CONSTRAINT FOREIGN KEY (...) REFERENCES ... (...).
- FOREIGN KEY definitions now allow backquotes around table and column names.
- MySQL command SET TRANSACTION ISOLATION LEVEL ... has now the following effect on InnoDB tables: if a transaction is defined as SERIALIZABLE then InnoDB conceptually adds

LOCK IN SHARE MODE to all consistent reads. If a transaction is defined to have any other isolation level, then InnoDB obeys its default locking strategy which is REPEATABLE READ.
- SHOW TABLE STATUS no longer sets an x−lock at the end of an auto−increment index if the auto−increment counter has already been initialized. This removes in almost all cases the surprising deadlocks caused by SHOW TABLE STATUS.
- Correction d'un bogue : in a CREATE TABLE statement the string 'foreign' followed by a non−space character confused the FOREIGN KEY parser and caused table creation to fail with errno 150.

**MySQL/InnoDB−3.23.49, February 17, 2002**

- Correction d'un bogue : if you called DROP DATABASE for a database on which there simultaneously were running queries, the MySQL server could crash or hang. Crashes fixed, but a full fix has to wait some changes in the MySQL layer of code.
- Correction d'un bogue : on Windows one had to put the database name in lower case for DROP DATABASE to work. Fixed in 3.23.49: case no longer matters on Windows. On Unix the database name remains case−sensitive.
- Correction d'un bogue : if one defined a non−latin1 character set as the default character set, then definition of foreign key constraints could fail in an assertion failure in dict0crea.c, reporting an internal error 17.

**MySQL/InnoDB−3.23.48, February 9, 2002**

- Tuned the SQL optimizer to favor more often index searches over table scans.
- Fixed a performance problem when several large SELECT queries are run concurrently on a multiprocessor Linux computer. Large CPU−bound SELECT queries will now also generally run faster on all platforms.
- If MySQL binlogging is used, InnoDB now prints after crash recovery the latest MySQL binlog file name and the position in that file (= byte offset) InnoDB was able to recover to. This is useful, for example, when resynchronizing a master and a slave database in replication.
- Added better error messages to help in installation problems.
- One can now recover also MySQL temporary tables which have become orphaned inside the InnoDB tablespace.
- InnoDB now prevents a FOREIGN KEY declaration where the signedness is not the same in the referencing and referenced integer columns.
- Correction d'un bogue : calling SHOW CREATE TABLE or SHOW TABLE STATUS could cause memory corruption and make mysqld to crash. Especially at risk was mysqldump, because it calls frequently SHOW CREATE TABLE.
- Correction d'un bogue : if on Unix you did an ALTER TABLE to an InnoDB table and simultaneously did queries to it, mysqld could crash with an assertion failure in row0row.c, line 474.
- Correction d'un bogue : if inserts to several tables containing an auto−inc column were wrapped inside one LOCK TABLES, InnoDB asserted in lock0lock.c.
- In 3.23.47 we allowed several NULLS in a UNIQUE secondary index. But CHECK TABLE was not relaxed: it reports the table as corrupt. CHECK TABLE no longer complains in this situation.
- Correction d'un bogue : on Sparc and other high−endian processors SHOW VARIABLES showed innodb_flush_log_at_trx_commit and other boolean−valued startup parameters always OFF even if they were switched on.
- Correction d'un bogue : if you ran mysqld−max−nt as a service on Windows NT/2000, the service shutdown did not always wait long enough for the InnoDB shutdown to finish.

**MySQL/InnoDB−3.23.47, December 28, 2001**

- Recovery happens now faster, especially in a lightly loaded system, because background checkpointing has been made more frequent.
- InnoDB allows now several similar key values in a UNIQUE secondary index if those values contain SQL NULLs. Thus the convention is now the same as in MyISAM tables.
- InnoDB gives a better row count estimate for a table which contains BLOBs.
- In a FOREIGN KEY constraint InnoDB is now case−insensitive to column names, and in Windows also to table names.
- InnoDB allows a FOREIGN KEY column of CHAR type to refer to a column of VARCHAR type, and vice versa. MySQL silently changes the type of some columns between CHAR and VARCHAR, and these silent changes do not hinder FOREIGN KEY declaration any more.
- Recovery has been made more resilient to corruption of log files.
- Unnecessary statistics calculation has been removed from queries which generate a temporary table. Some ORDER BY and DISTINCT queries will now run much faster.
- MySQL now knows that the table scan of an InnoDB table is done through the primary key. This will save a sort in some ORDER BY queries.
- The maximum key length of InnoDB tables is again restricted to 500 bytes. The MySQL interpreter is not able to handle longer keys.
- The default value of innodb_lock_wait_timeout was changed from infinite to 50 seconds, the default value of innodb_file_io_threads from 9 to 4.

**MySQL/InnoDB−4.0.1, December 23, 2001**

- InnoDB is the same as in 3.23.47.
- In 4.0.0 the MySQL interpreter did not know the syntax LOCK IN SHARE MODE. This has been fixed.
- In 4.0.0 multi−table delete did not work for transactional tables. This has been fixed.

**MySQL/InnoDB−3.23.46, November 30, 2001**

- This is the same as 3.23.45.

**MySQL/InnoDB−3.23.45, November 23, 2001**

- This is a boguefix release.
- In versions 3.23.42−.44 when creating a table on Windows you have to use lower case letters in the database name to be able to access the table. Fixed in 3.23.45.
- InnoDB now flushes stdout and stderr every 10 seconds: if these are redirected to files, the file contents can be better viewed with an editor.
- Fixed an assertion failure in .44, in trx0trx.c, line 178 when you drop a table which has the .frm file but does not exist inside InnoDB .
- Fixed a bogue in the insert buffer. The insert buffer tree could get into an inconsistent state, causing a crash, and also crashing the recovery. This bogue could appear especially in large table imports or alterations.
- Fixed a bogue in recovery: InnoDB could go into an infinite loop constantly printing a warning message that it cannot find free blocks from the buffer pool.
- Correction d'un bogue : when you created a temporary table of the InnoDB type, and then used ALTER TABLE to it, the MySQL server could crash.
- Prevented creation of MySQL system tables 'mysql.user', 'mysql.host', or 'mysql.db', in the InnoDB type.
- Fixed a bogue which can cause an assertion failure in 3.23.44 in srv0srv.c, line 1728.

**MySQL/InnoDB−3.23.44, November 2, 2001**

- You can define foreign key constraints on InnoDB tables. An example: FOREIGN KEY (col1)

REFERENCES table2(col2).
- You can create > 4 GB data files in those file systems that allow it.
- Improved InnoDB monitors, including a new innodb_table_monitor which allows you to print the contents of the InnoDB internal data dictionary.
- DROP DATABASE will now work also for InnoDB tables.
- Accent characters in the default character set latin1 will be ordered according to the MySQL ordering.<br> NOTE: if you are using latin1 and have inserted characters whose code is > 127 to an indexed CHAR column, you should run CHECK TABLE on your table when you upgrade to 3.23.43, and drop and reimport the table if CHECK TABLE reports an error!
- InnoDB will calculate better table cardinality estimates.
- Change in deadlock resolution: in .43 a deadlock rolls back only the SQL statement, in .44 it will roll back the whole transaction.
- Deadlock, lock wait timeout, and foreign key constraint violations (no parent row, child rows exist) now return native MySQL error codes 1213, 1205, 1216, 1217, respectively.
- A new my.cnf parameter innodb_thread_concurrency helps in performance tuning in high concurrency environments.
- A new my.cnf option innodb_force_recovery will help you in dumping tables from a corrupted database.
- A new my.cnf option innodb_fast_shutdown will speed up shutdown. Normally InnoDB does a full purge and an insert buffer merge at shutdown.
- Raised maximum key length to 7000 bytes from a previous limit of 500 bytes.
- Fixed a bogue in replication of auto−inc columns with multiline inserts.
- Fixed a bogue when the case of letters changes in an update of an indexed secondary column.
- Fixed a hang when there are > 24 data files.
- Fixed a crash when MAX(col) is selected from an empty table, and col is a not the first column in a multi−column index.
- Fixed a bogue in purge which could cause crashes.

**MySQL/InnoDB−3.23.43, October 4, 2001**

- This is essentially the same as InnoDB−3.23.42.

**MySQL/InnoDB−3.23.42, September 9, 2001**

- Fixed a bogue which corrupted the table if the primary key of a > 8000−byte row was updated.
- There are now 3 types of InnoDB Monitors: innodb_monitor, innodb_lock_monitor, and innodb_tablespace_monitor. innodb_monitor now prints also buffer pool hit rate and the total number of rows inserted, updated, deleted, read.
- Fixed a bogue in RENAME TABLE.
- Fixed a bogue in replication with an auto−increment column.

**MySQL/InnoDB−3.23.41, August 13, 2001**

- Support for < 4 GB rows. The previous limit was 8000 bytes.
- Use the doublewrite file flush method.
- Raw disk partitions supported as data files.
- InnoDB Monitor.
- Several hang bogues fixed and an ORDER BY bogue ('Sort aborted') fixed.

**MySQL/InnoDB−3.23.40, July 16, 2001**

- Only a few rare bogues fixed.

**MySQL/InnoDB−3.23.39, June 13, 2001**

- CHECK TABLE now works for InnoDB tables.
- A new my.cnf parameter innodb_unix_file_flush_method introduced. It can be used to tune disk write performance.
- An auto−increment column now gets new values past the transaction mechanism. This saves CPU time and eliminates transaction deadlocks in new value assignment.
- Several bogue fixes, most notably the rollback bogue in 3.23.38.

**MySQL/InnoDB−3.23.38, May 12, 2001**

- The new syntax SELECT ... LOCK IN SHARE MODE is introduced.
- InnoDB now calls fsync after every disk write and calculates a checksum for every database page it writes or reads, which will reveal disk defects.
- Several bogue fixes.

### 7.5.15 Informations de contact InnoDB

Informations de contact de Innobase Oy, producteur de InnoDB . Site web : http://www.innodb.com/ . Courrier électronique : Heikki.Tuuri@innodb.com

```
 phone: 358−9−6969 3250 (bureau) 358−40−5617367 (portable)
Innobase Oy Inc.
World Trade Center Helsinki
Aleksanterinkatu 17
P.O.Box 800
00101 Helsinki
Finland
```

## 7.6 Tables BDB ou BerkeleyDB

### 7.6.1 Vue d'ensemble des tables BDB

Le support des tables BDB est inclus par la distribution des sources de MySQL à partir de la version 3.23.34 et est activé dans le binaire MySQL−Max.

BerkeleyDB, disponible sur http://www.sleepycat.com/ a fournit à MySQL un gestionnaire de tables transactionnelles. En utilisant les tables BerkeleyDB, vos tables ont plus de chances de survivre aux crashes, et vous avez accès à COMMIT et ROLLBACK avec les transactions. La distribution des sources de MySQL fournit une distribution patchée de BDB pour lui permettre de fonctionner d'une façon plus souple avec MySQL. Vous pouvez utiliser une version non−patchée de BDB avec MySQL.

Nous travaillons chez MySQL AB en coopération étroite avec Sleepycat pour maintenir une bonne qualité d'interface MySQL/BDB.

Lorsqu'ils utiliseront les tables BDB , nous aiderons nos utilisateurs à trouver les problèmes et à créer des batteries de tests reproductibles pour tout problème ayant trait aux tables BDB . De tels tests seront aussi envoyés à Sleepycat qui nous aidera aussi à trouver et résoudre les problèmes. Vu qu'il s'agit d'une collaboration à deux niveaux, les problèmes concernant les tables BDB prendront un peu plus de temps à être résolus en comparaison avec les autres gestionnaires de tables. Toutefois, étant doné que le code de BerkeleyDB a lui−même été utilisé par plusieurs autres applications à part MySQL, nous n'envisageons pas de rencontrer de gros problèmes avec. Support offert par MySQL AB .

## 7.6.2 Installation de BDB

Si vous avez téléchargé une version binaire de MySQL qui inclus le support de BerkeleyDB, vous n'avez qu'à suivre les instructions classiques. Installing a MySQL Binary Distribution . `mysqld-max` , la version étendue du serveur `mysqld` .

Pour compiler MySQL avec le support de Berkeley DB, téléchargez la version 3.23.34 ou plus de MySQL et configurez `MySQL` avec l'option `--with-berkeley-db` . Installer MySQL à partir des sources .

```
 cd /chemin/vers/la/source/de/mysql-3.23.34
./configure --with-berkeley-db
```

Consultez le manuel fournit avec la distribution `BDB` pour des informations plus à jour.Même si Berkeley DB a fait ses preuves, l'interface MySQL est encore considérée comme étant de qualité bêta. Nous sommes actuellement entrain de travailler à le rendre stable le plus rapidement possible.

## 7.6.3 Options de démarrage BDB

Si vous utilisez `AUTOCOMMIT=0` , vos changements dans les tables `BDB` ne seront pas effectués tant que vous n'aurez pas utilisé `COMMIT` . Vous pouvez à la place utiliser `ROLLBACK` pour annuler les changements. Syntaxe des `BEGIN/COMMIT/ROLLBACK` .

Si vous utilisez `AUTOCOMMIT=1` (par défaut), vos changements seront automatiquement pris en compte. Vous pouvez démarrer une transaction étendue avec la commande SQL `BEGIN WORK` , après quoi vos changements ne seront pris en compte que si vous exécutez `COMMIT` (vous pouvez à tout moment annuler tous vos changements en exécutant `ROLLBACK` ).

Les options suivantes de `mysqld` peuvent être utilisées pour modifier le comportement des tables `BDB` :

| Option | Description |
|---|---|
| `--bdb-home=répertoire` | Répertoire de base des tables BDB . Cela doit être le même répertoire que vous avez utilisés pour `--datadir` . |
| `--bdb-lock-detect=#` | Détection des verrouillages Berkeley. ( DEFAULT , OLDEST , RANDOM , ou YOUNGEST ). |
| `--bdb-logdir=répertoire` | Répertoire des fichiers de log de Berkeley DB. |
| `--bdb-no-sync` | Ne pas vider les tampons synchroniquement. |
| `--bdb-no-recover` | Ne pas démarrer Berkeley DB en mode recouvrement. |
| `--bdb-shared-data` | Démarrer Berkeley DB en mode multi-processus (Ne pas utiliser DB_PRIVATE lors de l'initialisation de Berkeley DB) |
| `--bdb-tmpdir=répertoire` | Répertoire des fichiers temporaires de Berkeley DB. |
| `--skip-bdb` | Désactive l'utilisation des tables BDB . |
| `-O bdb_max_lock=1000` | Définit le nombre maximal de verrous. SHOW VARIABLES. |

Si vous utilisez `--skip-bdb` , MySQL n'initialisera pas la librairie Berkeley DB et cela économisera beaucoup de mémoire. Bien sûr, vous ne pouvez pas utiliser les table `BDB` si vous utilisez cette option. Si vous essayez de créer une table `BDB` , MySQL créera une table `MyISAM` à la place.

Normalement, vous devez démarrer `mysqld` sans `--bdb-no-recover` si vous avez l'intention d'utiliser des tables `BDB` . Cela peut cependant vous poser des problèmes si vous essayez de démarrer `mysqld` alors que des fichiers de log `BDB` sont corrompus. Problèmes de démarrage de MySQL .

Vous pouvez spécifier le nombre maximal de verrous avec `bdb_max_lock` (10000 par défaut) que vous pouvez activer sur une table `BDB` . Vous devez l'augmenter si vous obtenez des erreurs du type `bdb: Lock table is out of available locks` ou `Got error 12 from ...` lorsque vous avez fait de longues transactions ou quand `mysqld` doit examiner beaucoup de lignes pour calculer la requête.

Vous voudrez peut-être aussi changer les options `binlog_cache_size` et `max_binlog_cache_size` si vous utilisez de grandes transactions multi-lignes. Syntaxe des `BEGIN/COMMIT/ROLLBACK` .

## 7.6.4 Caractéristiques des tables BDB

- Pour permettre les transactions annulabes, BDB gère un fichier de log. Pour maximiser les performances, vous devriez placer ces fichiers sur un autre disque que celui de votre base, en utilisant l'option `--bdb-logdir` .
- MySQL fait un point de contrôle à chaque fois qu'un nouveau fichier de log BDB est démarré, et supprime les fichiers de logs anciens qui ne sont pas utiles. Si vous exécutez la commande `FLUSH LOGS` , vous placerez un nouveau point de contrôle pour les tables Berkeley DB.Pour la restauration après crash, vous devez utiliser les sauvegardes et le log binaire de MySQL. Sauvegardes de base de données . **Attention** : si vous effacez les anciens fichiers de log qui sont en cours d'utilisation, BDB ne sera pas capable de faire la restauration et vous risquez de perdre des données.
- MySQL requiert une clé `PRIMARY KEY` dans chaque table BDB pour être capable de faire référence aux lignes précédemment lues. Si vous n'en créez pas, MySQL va gérer une telle clé de manière cachée. La clé cachée a une taille de 5 octets, et est incrémentée à chaque nouvelle insertion.
- Si toutes les colonnes auxquelles vous accédez dans une table BDB font parties du même index dans la clé primaire, alors MySQL peut exécuter la requête sans avoir à lire la ligne elle−même. Dans une table MyISAM , ce qui précède n'est valable que si les colonnes font partie du même index.
- La clé `PRIMARY KEY` sera plus rapide que n'importe quelle autre clé, car la `PRIMARY KEY` est stockée avec les données. Comme les autres clés sont stockées sous la forme données + `PRIMARY KEY` , il est important de garder une clé `PRIMARY KEY` aussi courte que possible pour économiser de l'espace disque, et améliorer la vitesse.
- `LOCK TABLES` fonctionne avec les tables BDB sur les autres tables. Si vous n'utilisez pas le verrou `LOCK TABLE` , MySQL va poser un verrou interne multiple sur la table, pour s'assurer que la table est bien verrouillée, si un autre thread tente de poser un verrou.
- Le verrouillage interne des tables BDB est fait au niveau page.
- `SELECT COUNT(*) FROM table_name` est très lent, car les tables BDB ne maintiennent pas un compte de leur lignes dans la table.
- Scanner est plus lent qu'avec MyISAM car les tables BDB stockent les données dans un fichier B−tree et non pas dans un fichier séparé.
- L'application doit toujours être prête à gérer des cas où une modification sur une table BDB peut être annulée, ou une lecture abandonnée pour cause de blocage de verrous.
- Les clés ne sont pas compressées avec les clés précédentes, comme pour les tables ISAM et MyISAM . En d'autres termes, les informations de clés prennent un peu plus d'espace pour les tables BDB , comparativement aux tables MyISAM qui n'utilisent pas l'option `PACK_KEYS=0` .
- Il y a souvent des trous dans les tables BDB pour vous permettre d'insérer de nouvelles lignes au milieu de l'arbre de données. Cela rend les tables BDB un peu plus grandes que les tables MyISAM .
- L'optimiseur a besoin de connaître une approximation du nombre de lignes dans la table. MySQL résoud ce problème en comptant les insertions et en conservant ce compte dans un segment séparé pour chaque table BDB . Si vous ne faites pas souvent de `DELETE` ou `ROLLBACK` , ce nombre sera plutôt précis pour l'optimiseur MySQL, mais comme MySQL ne stocke ce nombre qu'à la fermeture de la table, il peut être incorrecte si MySQL crashe. Cela ne doit pas être fatal si ce nombre n'est pas à 100% correct. Vous pouvez forcer la mise à jour de ce nombre avec la commande `ANALYZE TABLE` ou `OPTIMIZE TABLE` . Syntaxe de `ANALYZE TABLE` . Syntaxe de `OPTIMIZE TABLE` .
- Si vous atteignez la capacité maximale du disque avec la table BDB , vous allez obtenir une erreur (probablement l'erreur 28), et la transaction va s'annuler. C'est un comportement différent des tables MyISAM et ISAM qui vont attendre que mysqld ait trouvé de l'espace disque avant de continuer.

## 7.6.5 Ce que nous devons corriger dans BDB dans un futur proche :

- Il est très lent d'ouvrir de nombreuses tables BDB en même temps. Si vous utilisez des tables BDB , il ne faut pas avoir un cache de table trop grand (par exemple, > 256) et vous devriez utiliser l'option `--no-auto-rehash` avec le client `mysql` . Nous envisageons de corriger cela en partie en version 4.0.
- `SHOW TABLE STATUS` ne fourni pas encore beaucoup d'informations pour les tables BDB tables.
- Optimiser les performances.
- Ne pas utiliser les verrous de pages lorsque l'on scanne les tables.

## 7.6.6 Systèmes d'exploitation supportés par BDB

Si, après avoir construit MySQL avec le support des tables BDB tables, obtenez l'erreur suivante dans le fichier de logs quand vous démarrez `mysqld` :

```
 bdb: architecture lacks fast mutexes: applications cannot be threaded
Can't init dtabases
```

Cela signifie que les tables BDB ne sont pas supportées par votre architecture. Dans ce cas, vous devez reconstruire MySQL sans le support des tables BDB .

Note : La liste suivante n'est pas complète; nous la mettrons à jour au fur et à mesure que nous recevrons des informations à ce propos.

Actuellement, nous savons que le gestionnaire BDB fonctionne avec les systèmes d'exploitation suivants :

- Linux 2.x intel
- Solaris SPARC
- Caldera (SCO) OpenServer
- Caldera (SCO) UnixWare 7.0.1

Il ne fonctionne pas sur les systèmes d'exploitations suivants :

- Linux 2.x Alpha
- Max OS X

## 7.6.7 Restrictions avec les tables BDB

Voilà les restrictions que vous pouvez rencontrer en travaillant avec les tables BDB :

- Les tables BDB enregistrent dans le fichier `.db` le chemin vers le fichier tel qu'il était lorsqu'il a été crée. (Cela est fait pour permettre de detecter les verrous dans un environnement multi-utilisateurs qui supporte les liens symboliques).Cela fait que les tables BDB ne peuvent être changées de répertoire !
- Lors de la sauvegarde de tables BDB , vous devez utiliser `mysqldump` ou effectuer des sauvegardes de tout les fichiers `nom_de_table.db` et des fichiers de log BDB . Les fichiers de log de BDB sont les fichiers dans le répertoire de base des données nommés `log.XXXXXXXXXX` (dix chiffres); Le gestionnaire des tables BDB stocke les transactions non-terminées dans les fichiers de log et requière la présence de ceux-ci lors du démarrage de `mysqld` .

## 7.6.8 Erreurs pouvant survenir lors de l'utilisation des tables BDB

- Si vous obtenez l'erreur suivante dans le fichier `hostname.err log` lors du démarrage de `mysqld` :

  ```
  bdb:  Ignoring log file: .../log.XXXXXXXXXX: unsupported log version #
  ```

  cela signifie que la nouvelle version de `BDB` ne supporte pas l'ancien format de log. Dans ce cas, vous devez effacer tous les logs `BDB` du dossier des données (les fichiers dont le nom est au format `log.XXXXXXXXXX` ) et redémarrer `mysqld` . Nous vous recommandons aussi d'exécuter un `mysqldump --opt` de vos vieilles tables `BDB` , de les effacer, puis de restaurer les copies.
- Si vous n'êtes pas en mode auto−commit et que vous effacez une table qu'un autre thread utilise, vous obtiendrez le message d'erreur suivant dans le fichier d'erreurs de MySQL :

  ```
   001119 23:43:56  bdb:  Missing log fileid entry
  001119 23:43:56  bdb:  txn_abort: Log undo failed for LSN:
                         1 3644744: Invalid
  ```

  Ce n'est pas une erreur très grave mais nous ne vous recommandons pas d'effacer vos tables si vous n'êtes pas en mode auto−commit, tant que ce problème n'est pas résolu (la solution n'est pas triviale).

# 8 Les interfaces pour MySQL

Ce chapitre décrit les interfaces fournies par plusieurs langages pour MySQL, où les obtenir, et comment les utiliser. L'interface C est la plus abordée dans ce chapitre, puisqu'elle a été développée par l'équipe de MySQL, et qu'elle sert de base à toutes les autres interfaces.

## 8.1 API PHP pour MySQL

PHP est un langage côté serveur, qui peut être utilisé pour créer des pages web dynamiques. Il fournit un support d'accès à plusieurs systèmes de bases de données, dont MySQL fait partie. PHP peut être utilisé en tant que programme séparé, ou être compilé en tant que module pour le serveur web Apache.

La distribution et documentation sont disponibles sur le site web de PHP ( http://www.php.net/ ).

### 8.1.1 Problèmes fréquents avec MySQL et PHP

- Error: "Maximum Execution Time Exceeded" C'est une limitation de PHP; Editez le fichier `php.ini` et changez le temps maximal d'exécution d'un script de 30 secondes à plus, selon vos besoins. C'est aussi une bonne idée de doubler la quantité de RAM allouée par script en la changeant à 16MB.
- Error: "Fatal error: Call to unsupported or undefined function mysql_connect() in .." Cela signifie que votre version de PHP n'est pas compilée avec le support de MySQL. Vous pouvez soit compiler un module dynamique MySQL et le charger dans PHP, ou bien recompiler PHP avec le support natif de MySQL. Ceci est décrit en détails dans le manuel PHP
- Error: "undefined reference to `uncompress'" Cela signifie que la librairie cliente est compilée avec support d'un protocole client/serveur compressé. La solution est d'ajouter `-lz` à la fin lors de la liaison avec `-lmysqlclient` .

## 8.2 API Perl pour MySQL

Cette section documente l'interface `DBI` de Perl. L'ancienne interface s'appelait `mysqlperl` . `DBI` / `DBD` est maintenant l'interface recommandée avec Perl et donc `mysqlperl` est obsolète et n'est pas documentée ici.

### 8.2.1 DBI avec DBD::mysql

`DBI` est une interface générique à plusieurs systèmes de bases de données. Cela signifie que vous pouvez écrire un script qui fonctionne parfaitement avec plusieurs systèmes différents sans y apporter aucun changement. Vous avez besoin de définir un pilote de base de données (DataBase Driver : DBD) pour chaque système. Pour MySQL, ce pilote se nomme `DBD::mysql` .

Pour plus d'informations sur le module DBI de Perl5, visitez la page web de `DBI` et lisez la documentation :

```
http://dbi.perl.org/
```

Pour plus d'informations sur la programmation orientée objets (OOP) comme la définie Perl5, voyez

la page OOP de Perl :

```
http://language.perl.com/info/documentation.html
```

Notez que si vous voulez utiliser les transactions avec Perl, vous avez besoin d'avoir au moins la version 1.2216 de `Msql-Mysql-modules` .

Le module Perl recommandé est `DBD-mysql-2.1022` ou plus récent.

Les instructions pour l'installation du support Perl de MySQL sont données dans Commentaires sur l'installation de Perl .

Si vous avez le module MySQL installé, vous pouvez trouver des informations à propos des fonctionnalités spécifiques de MySQL avec l'une de ces commandes :

```
shell> perldoc DBD/mysql
shell> perldoc mysql
```

## 8.2.2 L'interface DBI

### Méthodes portables

| Méthode | Description |
|---|---|
| connect | Etablit une connexion à un serveur de bases de données. |
| disconnect | Déconnecte du serveur de bases de données. |
| prepare | Prépare une commande SQL pour l'exécution. |
| execute | Exécute les requêtes préparées. |
| do | Prépare et exécute une commande SQL. |
| quote | Echappe une chaîne ou une valeur BLOB avant insertion. |
| fetchrow_array | Récupère la ligne suivante en tant que tableau de champs. |
| fetchrow_arrayref | Récupère la ligne suivante comme une référence de tableau de champs. |
| fetchrow_hashref | Récupère la ligne suivante en tant que référence pour un tableau associatif. |
| fetchall_arrayref | Récupère toutes les données comme un tableau de tableaux. |
| finish | Termine une requête et permet au système de libérer les ressources. |
| rows | Retourne le nombre de lignes affectées. |
| data_sources | Retourne un tableau de bases de données disponibles sur le serveur local. |
| ChopBlanks | Contrôle la suppression des espaces avec fetchrow_* . |
| NUM_OF_PARAMS | Nombre de marqueurs dans la requête préparée. |
| NULLABLE | Les colonnes qui peuvent être NULL . |
| trace | Performe un traçage pour les corrections. |

### Méthodes spécifiques à MySQL

| Méthode | Description |
|---|---|
| insertid | La denière valeur AUTO_INCREMENT . |
| is_blob | Est-ce une valeur BLOB ? |
| is_key | Est-ce une clef ? |
| is_num | Est-ce une colonne numérique ? |
| is_pri_key | Quelles colonnes sont des clefs primaires ? |
| is_not_null | Quelles colonnes NE peuvent PAS être NULL . Voir NULLABLE . |
| length | Taille maximale possible pour la colonne. |
| max_length | Taille maximale des colonnes actuellement présentes dans le résultat. |
| NAME | Noms des colonnes. |
| NUM_OF_FIELDS | Nombre de Number of fields returned. |
| table | Noms des tables dans le résultat. |
| type | Tout les types de colonnes. |

Les méthodes Perl sont décrites dans les sections suivantes. Les variables utilisée pour les valeurs de retour des méthodes ont les significations suivantes :

*$dbh*
: Gestionnaire de base de données

*$sth*
: Gestionnaire de requête

*$rc*
: Code de retour (souvent un status)

*$rv*

Valeur de retour (souvent un comptage de lignes)

### Méthodes DBI portables

*connect($data_source, $username, $password)*

Utilise la méthode `connect` pour crée une connexion de base de données à la source. La valeur doit commencer par `DBI:nom_pilote:` . Exemples d'utilisation de `connect` avec le pilote `DBD::mysql` :

```
 $dbh = DBI->connect("DBI:mysql:$database", $user, $password);
$dbh = DBI->connect("DBI:mysql:$database:$hostname",
                    $user, $password);
$dbh = DBI->connect("DBI:mysql:$database:$hostname:$port",
                    $user, $password);
```

Si le nom d'utilisateur et/ou le mot de passe ne sont pas définis, `DBI` utilises la valeur des variables d'environnements `DBI_USER` et `DBI_PASS` , respectivement. Si vous ne spécifiez pas un nom d'hôte, il est à `'localhost'` par défaut. Si vous ne spécifiez pas un numéro de port, il est mit par défaut au port par défaut de MySQL (3306).Depuis la version 1.2009 de `Msql-Mysql-modules` , la valeur de `$data_source` accepte certains modificateurs :

*mysql_read_default_file=nom_du_fichier*

Lit le fichier `filename` en tant que fichier d'options. Pour plus d'informations sur les fichiers d'options, consulter Fichier d'options `my.cnf` .

*mysql_read_default_group=nom_du_groupe*

Le groupe par défaut lors de la lecture d'un fichier d'options est normalement le groupe `[client]` . En spécifiant l'option `mysql_read_default_group` , le groupe par défaut devient le groupe `[nom_du_groupe]` .

*mysql_compression=1*

Utiliser la communication compressée entre le client et le serveur (à partir de la version 3.22.3 de MySQL).

*mysql_socket=/path/to/socket*

Spécifie le chemin vers la socket Unix qui est utilisée pour se connecter au serveur. (MySQL version 3.21.15 ou plus).

Plusieurs modificateurs peuvent être donnés, ils doivent être séparés par des point-virgules.Par exemple, si vous voulez éviter d'avoir à écrire le nom d'utilisateur et le mot de passe en dur dans un script `DBI` , vous pouvez le prendre du fichier utilisateur d'options `~/.my.cnf` en écrivant votre appel à `connect` de la façon suivante :

```
 $dbh = DBI->connect("DBI:mysql:$database"
               . ";mysql_read_default_file=$ENV&#123;HOME&#125;/.my.cnf",
               $user, $password);
```

Cet appel lira les options définies pour le groupe `[client]` dans le fichier d'options. Si vous voulez le faire aussi pour le groupe `[perl]` par exemple, vous pouvez utiliser ce qui suit :

```
 $dbh = DBI->connect("DBI:mysql:$database"
               . ";mysql_read_default_file=$ENV&#123;HOME&#125;/.my.cnf"
               . ";mysql_read_default_group=perl",
               $user, $password);
```

*disconnect*

La méthode `disconnect` déconnecte le gestionnaire de base de données du serveur. On y fait appel avant de terminer le programme.Exemple :

```
 $rc = $dbh->disconnect;
```

*prepare($statement)*

Prépare une requête SQL pour l'exécution par le serveur de base de données et retourne un gestionnaire de requête `($sth)` , que vous pouvez utiliser pour invoquer la méthode `execute` . Le plus souvent, vous gérez les requêtes `SELECT` (et ceux qui s'en rapprochent, comme `SHOW` , `DESCRIBE` , et `EXPLAIN` ) à l'aide de `prepare` et `execute` .Exemple :

```
 $sth = $dbh->prepare($statement)
    or die "Impossible de préparer $statement: $dbh->errstr\n";
```

*execute*

La méthode `execute` exécute une requête préparée. Pour les commandes qui ne se rapprochent pas de `SELECT` , `execute` retourne le nombre de lignes affectées. Si aucune ligne ne l'est, `execute` retourne `"0E0"` , que Perl traite comme zéro mais considère comme vrai (true). Si une erreur survient, `execute` retourne `undef` . Pour les requêtes `SELECT` , `execute` ne fait que démarrer la requête SQL dans la base de données; vous avez besoin d'utiliser une des méthodes `fetch_*` décrites ici pour récupérer les données.Exemple :

```
$rv = $sth->execute
         or die "ne peut exécuter la requête : $sth->errstr;
```

*do($statement)*

La méthode `do` prépare et exécute une requête SQL et retourne le nombre de lignes affectées. Si aucune ne l'est, `do` retourne `"0E0"` , que Perl traite comme zéro mais considère comme vrai (true). Cette méthode est généralement utilisée pour les requêtes qui ne se rapprochent pas de `SELECT` qui ne peuvent être préparées à l'avance (à cause d'une limitation du pilote) ou qui n'ont pas besoin d'être exécutées plus d'une fois (insertions, suppressions, etc.). Exemple :

```
$rv = $dbh->do($statement)
       or die "Ne peut exécuter la commande $statement: $dbh- >errstr\n";
```

Généralement, la commande 'do' est plus rapide (et préférable) à prepare/execute pour les requêtes ne contenant pas de paramètres.

*quote($string)*

La méthode `quote` est utilisée pour protéger les caractères spéciaux contenus dans la chaîne et ajouter les guillemets externes requis.Exemple :

```
$sql = $dbh->quote($string)
```

*fetchrow_array*

Cette méthode récupère la ligne de données suivante et la retourne en tant que tableau de valeurs de champs.Exemple :

```
while(&#64;row = $sth->fetchrow_array) &#123;
       print qw($row[0]\t$row[1]\t$row[2]\n);
&#125;
```

*fetchrow_arrayref*

Cette méthode récupère la ligne de données suivante et la retourne en tant que référence de tableau de valeurs de champs.Exemple :

```
while($row_ref = $sth->fetchrow_arrayref) &#123;
       print qw($row_ref->[0]\t$row_ref->[1]\t$row_ref->[2]\n);
&#125;
```

*fetchrow_hashref*

Cette méthode récupère la ligne de données et retourne une référence vers un tableau associatif contenant des paires nom du champ / valeur. Cette méthode n'est pas aussi efficace que l'utilisation des références sur tableaux comme expliqué plus haut.Exemple :

```
while($hash_ref = $sth->fetchrow_hashref) &#123;
       print qw($hash_ref->&#123;firstname&#125;\t$hash_ref->&#123;lastname&#125;\t\
               $hash_ref- > title&#125;\n);
&#125;
```

*fetchall_arrayref*

Cette méthode est utilisée pour faire en sorte que toutes les données (lignes) soient retournées par la requête SQL. Elle retourne une référence à un tableau de références à des tableaux pour chaque ligne. Vous pouvez accéder aux données en utilisant des boucles profondes.Exemple :

```
my $table = $sth->fetchall_arrayref
               or die "$sth->errstr\n";
my($i, $j);
for $i ( 0 .. $#&#123;$table&#125; ) &#123;
       for $j ( 0 .. $#&#123;$table->[$i]&#125; ) &#123;
               print "$table->[$i][$j]\t";
       &#125;
       print "\n";
&#125;
```

*finish*

Indique que plus aucune donnée ne sera récupérée du gestionnaire de requête. Vous appelez cette méthode pour libérer le gestionnaire de connexion et les ressources système lui étant associées.Exemple :

```
$rc = $sth->finish;
```

*rows*

Retourne le nombre de lignes changées (mises à jour, supprimées, etc.) par la dernière commande. En l'utilise généralement après les requêtes autres que `SELECT` exécutées avec la méthode `execute` .Exemple :

```
$rv = $sth->rows;
```

*NULLABLE*

Retourne une référence vers un tableau de valeurs qui indiquent si les colonnes peuvent contenir des valeurs `NULL` . Les valeurs possibles pour chaque élément sont 0 ou la chaîne vide si la colonne ne peut être `NULL` , 1 si elle peut, et 2 si l'état `NULL` de la colonne est inconnu.Exemple :

```
$null_possible = $sth->&#123;NULLABLE&#125;;
```

*NUM_OF_FIELDS*

Cet attribut indique le nombre de champs retournés par une requête `SELECT` ou `SHOW FIELDS` . Vous pouvez l'utiliser pour savoir si une requête a retourné un résultat : une valeur nulle indique que la requête n'est pas une sélection, comme `INSERT` , `DELETE` , ou `UPDATE` .Exemple :

```
$nr_of_fields = $sth->&#123;NUM_OF_FIELDS&#125;;
```

*data_sources($driver_name)*

Cette méthode retourne un tableau contenant les noms des bases de données disponibles pour le serveur MySQL sur l'hôte `'localhost'` .Exemple :

```
&#64;dbs = DBI->data_sources("mysql");
```

*ChopBlanks*

Cet attribut détermine si les méthodes `fetchrow_*` enlèvent les caractères blancs au début et à la fin des valeurs retournées.Exemple :

```
$sth->&#123;'ChopBlanks'&#125; =1;
```

*trace($trace_level)*

*trace($trace_level, $trace_filename)*

La méthode `trace` active ou désactive le traçage. Lorsqu'elle est invoquée en tant que méthode de la classe `DBI` , elle affecte le traçage à tout les gestionnaire. Lorsqu'elle est invoquée comme un gestionnaire de base de données ou de requête, elle n'affecte le traçage que pour le gestionnaire donné (et tout ses descendants). Mettre `$trace_level` à 2 fournit des informations détaillées de traçage. Le mettre à 0 désactive le traçage. Les sorties de traçages vont dans la sortie d'erreurs standards par défaut. Si `$trace_filename` est spécifié, le fichier est ouvert et le pointeur de fichier est placé à la fin de celui−ci et les sorties de **tout** les gestionnaires tracés sont écrits dans ce fichier.Exemple :

```
 DBI->trace(2);                # trace tout
DBI->trace(2,"/tmp/dbi.out"); # trace tout dans
                              # /tmp/dbi.out
$dth->trace(2);               # trace ce gestionnaire de base de données
$sth->trace(2);               # trace ce gestionnaire de requête
```

Vous pouvez aussi activer le traçage `DBI` en définissant la variable d'environnement `DBI_TRACE` . La définir en tant que valeur numérique revient à appeler `DBI->(value)` . La définir en tant que chemin revient à appeler `DBI->(2,value)` .

### Méthodes spécifiques à MySQL

Les méthodes montrées ici sont spécifiques à MySQL et ne font pas partie du standard `DBI` . Beaucoup d'entre−elles sont aujourd'hui désapprouvées : `is_blob` , `is_key` , `is_num` , `is_pri_key` , `is_not_null` , `length` , `max_length` , et `table` . Lorsque des alternatives utilisant le standard `DBI` existent, elles sont mentionnées ici :

*insertid*

Si vous utilisez la fonctionnalité `AUTO_INCREMENT` de MySQL, la nouvelle valeur auto−incrémentée sera stockée ici.Exemple :

```
$new_id = $sth->&#123;insertid&#125;;
```

Vous pouvez, en alternative, utiliser `$dbh->{'mysql_insertid'}` .

*is_blob*

Retourne une référence vers un tableau de valeurs booléennes; pour chaque élément du tableau, une valeur TRUE indique que la colonne en question est un BLOB .Exemple :

```
$keys = $sth->&#123;is_blob&#125;;
```

*is_key*

Retourne une référence vers un tableau de valeurs booléennes; pour chaque élément du tableau, une valeur TRUE indique que la colonne en question est une clef.Exemple :

```
$keys = $sth->&#123;is_key&#125;;
```

*is_num*

Retourne une référence vers un tableau de valeurs booléennes; pour chaque élément du tableau, une valeur TRUE indique que la colonne en question contient une valeur numérique.Exemple :

```
$nums = $sth->&#123;is_num&#125;;
```

*is_pri_key*

Retourne une référence vers un tableau de valeurs booléennes; pour chaque élément du tableau, une valeur TRUE indique que la colonne en question est une clef primaire.Exemple :

```
$pri_keys = $sth->&#123;is_pri_key&#125;;
```

*is_not_null*

Retourne une référence vers un tableau de valeurs booléennes; pour chaque élément du tableau, une valeur FALSE indique que la colonne peut contenir NULL values.Exemple :

```
$not_nulls = $sth->&#123;is_not_null&#125;;
```

is_not_null est désapprouvé; il est préférable d'utiliser l'attribut NULLABLE (décrit plus haut), car c'est un standard DBI.

*length*

*max_length*

Chacune de ces méthodes retourne une référence vers un tableau de tailles de colonnes. Le tableau length indique les tailles maximales de colonnes (comme déclarés lors de la création de la table). Le tableau max_length indique la plus grande taille occupée jusqu'à présent dans les colonnes de la table. Exemple :

```
 $lengths = $sth->&#123;length&#125;;
$max_lengths = $sth->&#123;max_length&#125;;
```

*NAME*

Retourne une référence vers un tableau de noms de colonnes.Exemple :

```
$names = $sth->&#123;NAME&#125;;
```

*table*

Retourne une référence vers un tableau de noms de tables.Exemple :

```
$tables = $sth->&#123;table&#125;;
```

*type*

Retourne une référence vers un tableau de types de colonnes.Exemple :

```
$types = $sth->&#123;type&#125;;
```

## 8.2.3 Plus d'informations relatives à DBI / DBD

Vous pouvez utiliser la commande perldoc pour obtenir plus d'informations à propos de DBI .

```
 perldoc DBI
perldoc DBI::FAQ
perldoc DBD::mysql
```

Vous pouvez aussi utiliser les outils pod2man , pod2html , etc., pour traduire en différents formats.Vous pouvez trouver les dernières informations relatives à DBI à l'adresse suivante : http://dbi.perl.org/ .

# 8.3 Support ODBC avec MySQL

MySQL fournit un support de ODBC via le programme MyODBC . Ce chapitre vous apprendra à installer MyODBC , et à l'utiliser. Vous trouverez aussi une liste de programmes connus pour leur utilisation de MyODBC .

## 8.3.1 Comment installer MyODBC

MyODBC 2.50 est un pilote 32−bit ODBC 2.50 avec un niveau de spécification 0 (avec le niveau 1 et 2 de proposés) pour connecter une application compatible ODBC à MySQL. MyODBC fonctionne sur Windows 9x/Me/NT/2000/XP et la plupart des plate−formes Unix. MyODBC 3.51 est une version améliorée avec les spécifications de niveau 1 de ODBC 3.5x (API noyau complète + fonctionnalités du niveau 2).

MyODBC est Open Source , et vous pouvez trouver la version la plus récente sur http://www.mysql.com/downloads/api−myodbc.html . Notez que les version 2.50.x sont licencées LGPL tandis que les versions 3.51.x sont licencées GPL .

Si vous avez des problèmes avec MyODBC et que votre programme fonctionne aussi avec OLEDB, essayez le pilote OLEDB.

Normalement, vous n'avez besoin d'installer MyODBC que sur les machines Windows. Vous avez besoin d'installer MyODBC sous Unix si vous avez un programme tel que ColdFusion qui fonctionne sur les machines Unix et utilise ODBC pour se connecter aux bases de données.

Si vous voulez installer MyODBC sur un ordinateur Unix, vous aurez aussi besoin d'un gestionnaire ODBC . MyODBC est connu pour fonctionner avec la plupart des gestionnaires ODBC d'Unix.

Pour installer MyODBC sur Windows, vous devez télécharger le fichier .zip de MyODBC approprié, le décompresser avec WinZIP ou un programme similaire et exécuter le fichier SETUP.EXE .

Sur Windows/NT/XP vous pouvez obtenir l'erreur suivante durant l'installation de MyODBC :

```
 An error occurred while copying C:\WINDOWS\SYSTEM\MFC30.DLL. Restart
Windows and try installing again (before running any applications which
use ODBC)
```

Le problème dans ce cas est qu'un autre programme utilise ODBC et du fait de l'architecture Windows, vous ne pouvez pas installer de nouveau pilote ODBC avec le programme d'installation de Microsoft ODBC. Dans la plupart des cas, vous pouvez continuer en clickant juste sur Ignore pour copier le reste des fichiers MyODBC et l'installation finale devrait fonctionner. Si ce n'est pas le cas, la solution est de redémarrer votre machine en mode ``safe mode`` (faites le en appuyant sur F8 juste avant que votre machine ne démarre Windows), installez MyODBC , et redémarrez en mode normal.

- Pour créer une connexion à un ordinateur Unix depuis un ordinateur Windows, avec une application ODBC (une qui ne supporte pas MySQL nativement), vous devez installer MyODBC sur l'ordinateur Windows.
- L'utilisateur et la machine Windows doivent avoir les droits d'accès au serveur MySQL situé sur la machine Unix. vous pouvez configurer cela avec la commande GRANT . Syntaxe de GRANT et REVOKE .
- Vous devez créer une entrée DNS ODBC comme suit :
    - ♦ Ouvrez le panneau de configuration de Windows.
    - ♦ Double−clickez sur l'icône Sources de données ODBC (32 bits).

- Clickez sur le volet User DSN.
- Clickez sur le bouton Add.
- Sélectionnez MySQL dans l'écran Create New Data Source et clickez sur le bouton Terminer.
- L'écran de configuration par défaut du pilote MySQL est affiché. Comment remplir les différents champs dans le programme d'administrateur ODBC .

- Démarrez maintenant votre application et sélectionnez le pilote ODBC avec les DSN que vous avez spécifié dans l'administrateur ODBC.

Notez que d'autres options de configuration sont présentes dans l'écran de MySQL (traçage, se connecter automatiquement, etc.), vous pouvez les essayer en cas de problèmes.

## 8.3.2 Comment remplir les différents champs dans le programme d'administrateur ODBC

Il y a trois possibilités pour indiquer le nom du serveur sous Windows95 :

- Utiliser l'adresse IP du serveur.
- Ajouter un fichier `\windows\lmhosts` avec les informations suivantes :

  ```
  ip nomhote
  ```

  Pour exemple :

  ```
  194.216.84.21 mon_nom_hote
  ```

- Configurer le PC pour qu'il utilise DNS.

Exemple de valeurs dans l' `installation ODBC` :

```
 Windows DSN name:   test
Description:        Ceci est ma base de données
MySql Database:     test
Server:             194.216.84.21
User:               monty
Password:           mon_pass
Port:
```

La valeur du champ `Windows DSN name` est n'importe quel nom qui est unique pour votre installation de ODBC sur Windows.

Vous n'avez pas besoin de spécifier des valeurs pour les champs `Server` , `User` , `Password` , et `Port` dans l'écran d'installation d'ODBC. Toutefois, si vous le faites, ces valeurs seront celles utilisées par défaut par la suite lorsque vous tenterez d'établir une connexion. Vous pourrez changez les options à ce moment là.

Si le numéro du port n'est pas donné, le port par défaut (3306) est utilisé.

Si vous spécifiez l'option `Read options from C:\my.cnf` , les groupes `client` et `odbc` seront lus à partir du fichier `C:\my.cnf` . Vous pouvez utiliser toutes les options utilisables par `mysql_options()` . `mysql_options()` .

## 8.3.3 Paramètres de connexion de MyODBC

Il est possible de spécifier les paramètres suivants à `MyODBC` dans la section `[Servername]` du fichier `ODBC.INI` ou bien, avec l'argument `InConnectionString` dans l'appel de `SQLDriverConnect()` .

| Paramètre | Valeur par défaut | Commentaire |
|---|---|---|
| user | ODBC (sous Windows) | Le nom d'utilisateur utilisé pour se connecter à MySQL. |
| server | localhost | Le nom d'hôte du serveur MySQL. |
| database | | La base de données par défaut. |

| option | 0 | Un entier avec lequel vous spécifier le mode de fonctionnement de MyODBC . Voir ci−dessous. |
|---|---|---|
| port | 3306 | Le port TCP/IP à utiliser si le server n'est pas localhost . |
| stmt | | Une commande qui sera exécutée au moment de la connexion à MySQL . |
| password | | Le mot de passe pour le serveur server , et l'utilisateur user . |
| socket | | La socket ou le pipe Windows à utiliser. |

L'argument option est utilisé pour indiquer à MyODBC que le client n'est pas 100% compatible avec ODBC. Sous Windows, il est possible de configurer cette option en activant les options dans l'écran de connexion, mais il est aussi possible de le configurer dans ce paramètre. Les options suivantes sont listées dans l'ordre dans lequel elles apparaissent dans l'écran MyODBC :

| Bit | Description |
|---|---|
| 1 | Le client ne peut gérer le fait que MyODBC retourne la taille réelle de la colonne. |
| 2 | Le client ne peut gérer que MySQL retourne le nombre de ligne affecté. Si cette option est activée, alors MySQL retourne 'found rows' à la place. Il faut avoir MySQL 3.21.14 ou plus récent pour profiter de cette fonctionnalité. |
| 4 | Ecrit un fichier de débogage dans c:\myodbc.log. Cela revient au même que le code MYSQL_DEBUG=d:t:O,c::\myodbc.log dans le fichier AUTOEXEC.BAT |
| 8 | Ne limite pas les paquets pour les résultats et les paramètres. |
| 16 | Ne pas poser de questions, même si le pilote le veut. |
| 32 | Simule un pilote ODBC 1.0 dans certains contexte. |
| 64 | Ignore l'utilisation du nom de base de données dans la syntaxe 'base.table.colonne'. |
| 128 | Force l'utilisation du gestionnaire de curseur ODBC (expérimental). |
| 256 | Désactive l'utilisation de la lecture étendue (expérimental). |
| 512 | Complète les champs CHAR jusqu'à contenance. |
| 1024 | SQLDescribeCol() retourne les noms de colonnes complets. |
| 2048 | Utilise le protocole de communication client/serveur |
| 4096 | Indique au serveur qu'il doit ignorer les espaces après les noms de fonction, et avant le '(' (nécessaire pour PowerBuilder). Cela va faire de tous les noms de fonctions des mots clés. |
| 8192 | Connexion, avec les pipes nommés, à mysqld sur un machine NT. |
| 16384 | Change les colonnes LONGLONG en colonne INT (certaines applications ne peuvent pas gérer les LONGLONG). |
| 32768 | Retourne 'user' comme Table_qualifier et Table_owner dans les tables SQLTables (expérimental) |
| 65536 | Lit les paramètres des groupes client et odbc dans my.cnf |
| 131072 | Ajoute des vérifications de sécurité (ne devrait pas être nécessaire, mais...) |

Si vous voulez combiner des options, vous devez additionner les options ci−dessus. Par exemple, activer l'option 12 (4+8), active le déboggage sans limite de paquets.

La librairie par défaut MYODBC.DLL est compilée pour des performances optimales. Si vous voulez déboguer MyODBC (par exemple, pour activer le traçage), vous devriez utiliser MYODBCD.DLL . Pour installer ce ficher, copiez MYODBCD.DLL à la place de la librairie isntallé MYODBC.DLL .

## 8.3.4 Comment reporter les problèmes avec ODBC

MyODBC a été testé avec Access, Admndemo.exe, C++−Builder, Borland Builder 4, Centura Team Developer (formerly Gupta SQL/Windows), ColdFusion (on Solaris and NT with svc pack 5), Crystal Reports, DataJunction, Delphi, ERwin, Excel, iHTML, FileMaker Pro, FoxPro, Notes 4.5/4.6, SBSS, Perl DBD−ODBC, Paradox, Powerbuilder, Powerdesigner 32 bit, VC++, et Visual Basic.

Si vous avez connaissance de n'importe quelle autre application qui utilise MyODBC , merci d'envoyer un mail à myodbc@lists.mysql.com à ce propos !

Avec quelques programmes, vous pouvez avoir l'erreur suivante : Another user has modifies the record that you have modified . Dans la plupart des cas, le problème peut être résolu en suivant l'une des méthodes suivantes :

- ajouter une clef primaire à la table s'il n'en existe pas déjà une.
- Ajouter une colonne timestamp s'il n'y en a pas déjà une.
- N'utilisez que des champs réels doubles. Certains programmes peuvent se bloquer lorsqu'ils comparent des réels simples.

Si ce qui précède ne vous aide pas, vous devez tracer `MyODBC` et essayer de voir pourquoi les choses vont de travers.

## 8.3.5 Programmes qui fonctionnent avec MyODBC

La majorité des programmes devraient fonctionner avec `MyODBC` , mais tous ceux que nous avons listés ici sont ceux que nous avons testé nous même, ou dont nous avons reçu la confirmation de la part d'utilisateurs :

***Programme***

**Commentaire**

*Access*

Pour le faire fonctionner avec Access :

Si vous utilisez Access 2000, vous devez installer la dernière Microsoft MDAC ( `Microsoft Data Access Components` ) (version 2.6 ou plus récente) depuis http://www.microsoft.com/data/ . Cela va corriger le bug suivant d'Access : lorsque vous exportez des données vers MySQL, les noms de la table et des colonnes ne sont pas spécifiés. Un autre moyen de pallier à ce bug est de passer en MyODBC version 2.50.33 et MySQL version 3.23.x, qui, ensemble, forment un correctif à ce bug!

Vous devriez aussi installer le Microsoft Jet 4.0 Service Pack 5 (SP5) disponible à http://support.microsoft.com/support/kb/articles/Q239/1/14.ASP . Cela va corriger certains cas ou les colonnes étaient marquées comme effacées ( `#deleted#` ) dans Access.

Notez que si vous utilisez MySQL version 3.22, vous devez appliquer le patch MDAC et utiliser MyODBC 2.50.32 ou 2.50.34 et plus récent pour corriger ce bug.

Pour toutes les versions d'Access, vous devez activer l'option MyODBC appelée `Return matching rows` . Pour Access 2.0, vous devez aussi activer `Simulate ODBC 1.0` .

Vous devez avoir un timestamp dans toutes les tables que vous voulez pouvoir modifier. Pour une meilleure portabilité, `TIMESTAMP(14)` ou un simple `TIMESTAMP` est recommandé au lieu de `TIMESTAMP(X)` .

Vous devez avoir une clé primaire dans la table. Sinon, les nouvelles lignes et les lignes modifiées peuvent être marquées comme effacées `#DELETED#` .

N'utilisez que les champs `DOUBLE` . Access échoue lors des comparaisons entre les champs en simple précision. Le symptôme est généralement que les nouvelles lignes ou les lignes modifiées sont marquées comme effacées ( `#DELETED#` ), ou que vous ne pouvez plus trouver les nouvelles lignes ou les lignes modifiées.

Si vous reliez des tables via MyODBC, dont l'une des colonnes est de type `BIGINT` , les résultats sont affichés comme effacés ( `#DELETED` ). La solution est :

- Avoir au moins une colonne de type `TIMESTAMP` comme type de données, et de préférence `TIMESTAMP(14)` .
- Vérifiez l'option de connexion `'Change BIGINT columns to INT'` dans l'administrateur ODBC DSN.
- Effacez le lien de table dans Access, et recréez le.

Les anciennes lignes seront toujours marquées comme `#DELETED#` , mais les nouvelles lignes seront affichées correctement.

Si vous butez toujours dans l'erreur `Another user has changed your data` après avoir ajouté une colonne `TIMESTAMP` , le truc suivant peut vous aider : N'utilisez pas la

vue de `table` au format tableur. A la place, créez un formulaire avec les champs que vous souhaitez, et utilisez ce formulaire comme vue de la table. Vous devez utiliser la propriété `DefaultValue` pour la colonne `TIMESTAMP` , avec la valeur `NOW()` . C'est une bonne idée de cacher la colonne `TIMESTAMP` à vos utilisateurs, pour qu'ils ne soient pas perturbés.
Dans certains cas, Access peut générer des requêtes SQL illégales, que MySQL ne peut comprendre. Vous pouvez corriger cela en sélectionnant `"Query|SQLSpecific|Pass−Through"` dans le menu Access.
Access sur NT va indiquer que les colonnes `BLOB` sont des `OLE OBJECTS` . Si vous voulez avoir des colonnes de type `MEMO` à la place, vous devez changer le type de la colonne en `TEXT` avec `ALTER TABLE` .
Access ne peut pas toujours gérer correctement les colonnes `DATE` . Sis vous avez des problèmes avec, changez ces colonnes en type `DATETIME` .
Si vous avec une colonne Access de type `BYTE` , Access va essayer de l'exporter sous la forme d'un `TINYINT` au lieu d'un `TINYINT UNSIGNED` . Cela va vous donner des problèmes si vous avez des valeurs supérieures à 127!

*ADO*

Lorsque vous programmez avec l'API ADO et `MyODBC` vous devez faire attention aux propriétés par défaut qui ne sont pas supportées par MySQL. Par exemple, utiliser la propriété `CursorLocation Property` comme `adUseServer` va retourner des valeurs de −1 pour `RecordCount Property` . Pour avoir la véritable valeur, vous devez utiliser la valeur `adUseClient` , comme dans le code VB ci−contre :

```
 Dim myconn As New ADODB.Connection
Dim myrs As New Recordset
Dim mySQL As String
Dim myrows As Long

 myconn.Open "DSN=MyODBCsample"
mySQL = "SELECT * from user"
myrs.Source = mySQL
Set myrs.ActiveConnection = myconn
myrs.CursorLocation = adUseClient
myrs.Open
myrows = myrs.RecordCount

 myrs.Close
myconn.Close
```

Un autre palliatif est d'utiliser la commande `SELECT COUNT(*)` , pour connaître le nombre exact de lignes.

*Active server pages (ASP)*

Vous devez utiliser l'option `Return matching rows` .

*Applications BDE*

Pour faire fonctionner ces applications, vous devez activer les options `Don't optimize column widths` et `Return matching rows` .

*Borland Builder 4*

Lorsque vous démarrez une requête, vous pouvez utiliser la propriété `Active` ou utiliser la méthode `Open` . Notez que `Active` va commencer en générant automatiquement une requête `SELECT * FROM ...` qui peut ne pas être pratique avec de grosses tables!

*ColdFusion (On Unix)*

Les informations suivantes sont reprises de la documentation ColdFusion :

Utilisez les informations suivantes pour configurer le serveur ColdFusion Server sous Linux pour utiliser le pilote unixODBC avec `MyODBC` pour les serveurs MySQL. Allaire a vérifié que `MyODBC` version 2.50.26 fonctionne avec MySQL version 3.22.27 et ColdFusion pour Linux. (toute nouvelle version doit aussi fonctionner). Vous pouvez télécharger `MyODBC` à http://www.mysql.com/downloads/api−myodbc.html

ColdFusion version 4.5.1 vous permet d'utiliser l'administrateur ColdFusion pour ajouter des serveurs MySQL. Toutefois, le pilote n'est pas inclus dans ColdFusion version 4.5.1. Auparavant, le pilote MySQL était disponible dans le menu déroulant.

Vous devez compiler et copier le pilote `MyODBC` dans `/opt/coldfusion/lib/libmyodbc.so` .

Le dossier Contrib contient le programme `mydsn-xxx.zip` qui vous permet de compiler et supprimer des fichiers d'enregistrement DSN pour MyODBC sur les applications Coldfusion.

*DataJunction*

Vous devez le modifier pour qu'il exporte des `VARCHAR` plutôt que des `ENUM` , car il exporte ces derniers d'une manière qui pose des problèmes à MySQL.

*Excel*

Fonctionne. Quelques conseils :

Si vous avez des problèmes avec les dates, essayez de les sélectionner comme des chaînes, en utilisant la fonction `CONCAT()` . Par exemple :

```
select CONCAT(rise_time), CONCAT(set_time)
   from sunrise_sunset;
```

Les valeurs lues sous forme de chaînes seront correctement relues par Excel97.

Le but de la fonction `CONCAT()` est de faire croire à ODBC que cette colonne est de type ``string''. Sans la fonction `CONCAT()` , ODBC sait que la colonne est de type heure, et Excel ne le comprendra pas.

Notez qu'il y a un bug dans Excel, car il convertit automatiquement une chaîne en heure. Cela serait bien si la source était un fichier texte, mais c'est totalement idiot si la source est une connexion ODBC qui fournit les types exacts.

*Word*

Pour lire des données depuis MySQL vers des documents Word ou Excel, vous devez utiliser le pilote `MyODBC` et l'aide Add−in Microsoft Query.Par exemple, pour créer une base de données avec une table contenant 2 colonnes de type texte :

Insérez des lignes avec le client `mysql` .
Créez un fichier DSN en utilisant le gestionnaire ODBC, par exemple, `my` pour la base ci−dessus.
Ouvrez Word.
Créez un nouveau document vide.
Utilisez la barre "Database", pressez sur le bouton d'insertion.
Pressez sur le bouton "Get Data".
Sur la droite de l'écran, choisissez le bouton "Ms Query".
Dans ce écran, créez une nouvelle source de données avec "New Data Source" et en utilisant le fichier DSN `my` .
Sélectionner une nouvelle requête.
Sélectionnez les colonnes que vous souhaitez.
Faîtes votre filtre.
Faîtes votre tri.
Sélectionnez "Return Data" de Microsoft Word.
Cliquez sur "Finish".
Cliquez sur le bouton "Insert data" et sélectionnez les lignes.
Cliquez sur "OK" et vous verrez vous lignes dans votre document Word.

*odbcadmin*

Programme de test pour ODBC.

*Delphi*

Vous devez utiliser BDE version 3.2 ou plus récent. Utilisez l'option `Don't optimize column width` lors de la connexion à MySQL.De plus, voici un code Delphi pratique, pour configurer une entrée ODBC et une entrée BDE pour `MyODBC` (l'entrée BDE requiert le BDE Alias Editor qui est gratuit sur les pages "Delphi Super Page" près de chez vous. (Merci à Bryan Brunton bryan@flesherfab.com pour cela ) :

```
fReg:= TRegistry.Create;
 fReg.OpenKey('\Software\ODBC\ODBC.INI\DocumentsFab', True);
 fReg.WriteString('Database', 'Documents');
 fReg.WriteString('Description', ' ');
 fReg.WriteString('Driver', 'C:\WINNT\System32\myodbc.dll');
 fReg.WriteString('Flag', '1');
 fReg.WriteString('Password', '');
 fReg.WriteString('Port', ' ');
 fReg.WriteString('Server', 'xmark');
 fReg.WriteString('User', 'winuser');
 fReg.OpenKey('\Software\ODBC\ODBC.INI\ODBC Data Sources', True);
 fReg.WriteString('DocumentsFab', 'MySQL');
 fReg.CloseKey;
 fReg.Free;

  Memo1.Lines.Add('DATABASE NAME=');
 Memo1.Lines.Add('USER NAME=');
 Memo1.Lines.Add('ODBC DSN=DocumentsFab');
 Memo1.Lines.Add('OPEN MODE=READ/WRITE');
 Memo1.Lines.Add('BATCH COUNT=200');
 Memo1.Lines.Add('LANGDRIVER=');
 Memo1.Lines.Add('MAX ROWS=-1');
 Memo1.Lines.Add('SCHEMA CACHE DIR=');
 Memo1.Lines.Add('SCHEMA CACHE SIZE=8');
 Memo1.Lines.Add('SCHEMA CACHE TIME=-1');
 Memo1.Lines.Add('SQLPASSTHRU MODE=SHARED AUTOCOMMIT');
 Memo1.Lines.Add('SQLQRYMODE=');
 Memo1.Lines.Add('ENABLE SCHEMA CACHE=FALSE');
 Memo1.Lines.Add('ENABLE BCD=FALSE');
 Memo1.Lines.Add('ROWSET SIZE=20');
 Memo1.Lines.Add('BLOBS TO CACHE=64');
 Memo1.Lines.Add('BLOB SIZE=32');

  AliasEditor.Add('DocumentsFab','MySQL',Memo1.Lines);
```

*C++ Builder*

Testé avec BDE version 3.0. Le seul problème connu est que lorsque la structure d'une table change, les champs de requêtes ne changent pas. BDE ne semble pas reconnaître les clés primaires, mais uniquement l'index PRIMARY, même si cela n'a jamais été un problème.

*Vision*

Vous devriez utiliser l'option `Return matching rows` .

*Visual Basic*

Pour être capable de modifier une table, vous devez définir une clé primaire.Visual Basic avec ADO ne peut pas gérer les grands entiers. Cela signifie que les requêtes comme `SHOW PROCESSLIST` ne vont pas fonctionner correctement. Le correctif consiste à ajouter l'option `OPTION=16834` dans la chaîne de connexion ODBC pour d'utiliser l'option `Change BIGINT columns to INT` de MyODBC. Vous pouvez aussi utiliser l'option `Return matching rows` .

*VisualInterDev*

Si vous rencontrez l'erreur `[Microsoft][ODBC Driver Manager] Driver does not support this parameter` , la raison est que vous avez un grand entier `BIGINT` dans votre résultat. Essayez d'utiliser l'option `Change BIGINT columns to INT` de MyODBC.

*Visual Objects*

Vous devez utiliser l'option `Don't optimize column widths` .

## 8.3.6 Comment obtenir la valeur d'une colonne AUTO_INCREMENT avec ODBC

Un problème récurrent est d'obtenir la dernière valeur générée automatiquement par une commande `INSERT` . Avec ODBC, vous pouvez procéder de cette façon (en supposons que `auto` est un champ `AUTO_INCREMENT` ):

```
 INSERT INTO foo (auto,text) VALUES(NULL,'text');
SELECT LAST_INSERT_ID();
```

Ou, si vous voulez juste insérer cette valeur dans une autre table :

```
 INSERT INTO foo (auto,text) VALUES(NULL,'text');
INSERT INTO foo2 (id,text) VALUES(LAST_INSERT_ID(),'text');
```

Comment puis−je obtenir l'identifiant unique de la dernière ligne insérée? .Pour quelques applications utilisant ODBC (du moins Delphi et Access), la requête suivante peut être utilisée pour trouver une ligne insérée dernièrement :

```
SELECT * FROM nom_de_table WHERE auto IS NULL;
```

### 8.3.7 Rapporter des problèmes avec MyODBC

Si vous rencontrez des difficultés avec `MyODBC` , commencez par faire un fichier de log avec le gestionnaire ODBC (le fichier de log que vous obtenez en demande les logs de ODBCADMIN) et un log `MyODBC` .

Pour obtenir un fichier de log `MyODBC` , vous devez faire ceci :

- Assurez vous que vous utilisez `myodbcd.dll` et non pas `myodbc.dll` . Le moyen le plus facile pour le faire est d'obtenir `myodbcd.dll` dans la distribution MyODBC et de le copier à la place de `myodbc.dll` , qui est probablement dans le dossier `C:\windows\system32` ou `C:\winnt\system32` .Notez que vous voudrez probablement récupérer votre vieux fichier `myodbc.dll` lorsque vous aurez fini de tester, car il est bien plus rapide que `myodbcd.dll` .
- Activez l'option `Trace MyODBC' dans l'écran de configuration de `MyODBC` . Le fichier de log sera écrit dans le fichier `C:\myodbc.log` .Si l'option de trace n'est pas recommandée lorsque vous retournez dans l'écran précédent, cela signifie que vous n'utilisez pas `myodbcd.dll` (voir ci−dessus).
- Démarrez votre application, et faîtes la planter.

Vérifiez le fichier de `trace MyODBC` , pour essayer de comprendre ce qui ne va pas. Vous devriez être capable de trouver les requêtes émises en recherchant la chaîne `>mysql_real_query` dans le fichier `myodbc.log` .

Vous devriez aussi essayer de dupiquer la requête dans le client `mysql` ou `admndemo` pour voir si le problème vient de MyODBC ou MySQL.

Si vous trouvez quelques chose d'incorrect, n'envoyez que les lignes pertinentes (maximum, 40 lignes) à myodbc@lists.mysql.com . N'envoyez jamais le fichier de log MyODBC ou ODBC complet!

Si vous êtes incapables de trouver une erreur, la dernière option est de faire une archive (tar ou zip) qui contienne le fichier de trace MyODBC, le fichie de log ODBC, et un fichier README qui contienne une description du problème. Vous pouvez envoyer le tout à ftp://support.mysql.com/pub/mysql/secret/ . Seuls nous, à MYSQL AB, pourront accéder à ces fichiers, et nous seront très diligents avec vos données.

Si vous pouvez créer un problème qui reproduit le problème, essayez de l'uploader aussi!

Si le programem fonctionne avec d'autres serveurs SQL, vous devriez faire un log ODBC où vous faîtes exactement la même chose sur les autres serveurs SQL.

N'oubliez jamais que plus vous nous fournissez d'explication, plus nous pourront vous aider!

## 8.4 MySQL C API

The C API code is distributed with MySQL. It is included in the `mysqlclient` library and allows C programs to access a database.

Many of the clients in the MySQL source distribution are written in C. If you are looking for examples that demonstrate how to use the C API, take a look at these clients. You can find these in the `clients` directory in the MySQL source distribution.

Most of the other client APIs (all except Java) use the `mysqlclient` library to communicate with the MySQL server. This means that, for example, you can take advantage of many of the same environment variables that are used by other client programs, because they are referenced from the library. See Scripts côté client et outils , for a list of these variables.

The client has a maximum communication buffer size. The size of the buffer that is allocated initially (16K bytes) is automatically increased up to the maximum size (the maximum is 16M). Because buffer sizes are increased only as demand warrants, simply increasing the default maximum limit does not in itself cause more resources to be used. This size check is mostly a check for erroneous queries and communication packets.

The communication buffer must be large enough to contain a single SQL statement (for client−to−server traffic) and one row of returned data (for server−to−client traffic). Each thread's communication buffer is dynamically enlarged to handle any query or row up to the maximum limit. For example, if you have `BLOB` values that contain up to 16M of data, you must have a communication buffer limit of at least 16M (in both server and client). The client's default maximum is 16M, but the default maximum in the server is 1M. You can increase this by changing the value of the `max_allowed_packet` parameter when the server is started. Choix des paramètres du serveur .

The MySQL server shrinks each communication buffer to `net_buffer_length` bytes after each query. For clients, the size of the buffer associated with a connection is not decreased until the connection is closed, at which time client memory is reclaimed.

For programming with threads, see How to Make a Threaded Client . For creating a stand−alone application which includes the "server" and "client" in the same program (and does not communicate with an external MySQL server), see libmysqld, the Embedded MySQL Server Library .

## 8.4.1 Types de données de l'API C

*MYSQL*
: Cette structure représente un gestionnaire de connexion à la base de données. Elle est utilisée dans la plupart des fonctions MySQL.

*MYSQL_RES*
: Cette structure représente le résultat d'une requête qui retourne des lignes ( `SELECT` , `SHOW` , `DESCRIBE` , `EXPLAIN` ). L'information retournée par une requête est appelée **jeu de résultats** dans le reste de cette section.

*MYSQL_ROW*
: C'est une représentation sûre pour les types d'une ligne de données. Elle est actuellement implémentée en tant que tableau de chaîne à octets comptés. (Vous ne pouvez la traiter en tant que chaîne terminée par une valeur nulle si les valeurs du champ peuvent contenir des données binaires, car de telles valeurs peuvent contenir elles−même des octets nuls.) Les lignes sont obtenues en appelant `mysql_fetch_row()` .

*MYSQL_FIELD*
: Cette structure contient des informations à propos du champ, tel que son nom, son type, et sa taille. Ses membres sont décrit en plus de détails ici. Vous pouvez obtenir une structure `MYSQL_FIELD` pour chaque champ en appelant plusieurs fois `mysql_fetch_field()` . Les valeurs des champs ne font pas partie de la structure; elles sont contenues dans une structure `MYSQL_ROW` .

*MYSQL_FIELD_OFFSET*

C'est une représentation sûre des types pour les index dans une liste de champs MySQL. (Utilisés par `mysql_field_seek()` .) Les index sont des numéros de champs, dans une ligne, commençant à zéro.

*my_ulonglong*

Le type utilisé pour le nombre de lignes et pour `mysql_affected_rows()` , `mysql_num_rows()` , et `mysql_insert_id()` . Ce type fournit une echelle allant de `0` à `1.84e19` .Sur quelques systèmes, essayer d'écrire la valeur d'un type `my_ulonglong` ne fonctionnera pas. Pour écrire une telle valeur, convertissez là en `unsigned long` et utilisez un format d'impression `%lu` . Exemple :

```
printf ("Nombre de lignes : %lu\n", (unsigned long) mysql_num_rows(result));
```

La structure `MYSQL_FIELD` contient les membres listés ici :

*char * name*

Le nom du champ, une chaîne terminée par une valeur nulle.

*char * table*

Le nom de la table contenant ce champ, s'il n'est pas calculé. Pour les champs calculés, la valeur de `table` est une chaîne vide.

*char * def*

La valeur par défaut de ce champ, en tant que chaîne terminée par une valeur nulle. Ce n'est définit que si vous utilisez `mysql_list_fields()` .

*enum enum_field_types type*

Le type du champ. La valeur de `type` peut être l'une des suivantes :

| Valeur de type | Description du type |
|---|---|
| FIELD_TYPE_TINY | Champ TINYINT |
| FIELD_TYPE_SHORT | Champ SMALLINT |
| FIELD_TYPE_LONG | Champ INTEGER |
| FIELD_TYPE_INT24 | Champ MEDIUMINT |
| FIELD_TYPE_LONGLONG | Champ BIGINT |
| FIELD_TYPE_DECIMAL | Champ DECIMAL ou NUMERIC |
| FIELD_TYPE_FLOAT | Champ FLOAT |
| FIELD_TYPE_DOUBLE | Champ DOUBLE ou REAL |
| FIELD_TYPE_TIMESTAMP | Champ TIMESTAMP |
| FIELD_TYPE_DATE | Champ DATE |
| FIELD_TYPE_TIME | Champ TIME |
| FIELD_TYPE_DATETIME | Champ DATETIME |
| FIELD_TYPE_YEAR | Champ YEAR |
| FIELD_TYPE_STRING | Champ chaîne de caractères ( CHAR ou VARCHAR ) |
| FIELD_TYPE_BLOB | Champ BLOB ou TEXT (utilisez max_length pour déterminer la taille maximale) |
| FIELD_TYPE_SET | Champ SET |
| FIELD_TYPE_ENUM | Champ ENUM |
| FIELD_TYPE_NULL | Champ de type NULL |
| FIELD_TYPE_CHAR | Désapprouvé, utilisez FIELD_TYPE_TINY à la place |

Vous pouvez utiliser la macro `IS_NUM()` pour tester si un champ est de type numérique ou non. Passez la valeur de `type` à `IS_NUM()` et elle sera évaluée à TRUE si le champ est numérique :

```
if (IS_NUM(field->type))
   printf("Le champ est numérique\n");
```

*unsigned int length*

La taille du champ, comme spécifié dans la définition de la table.

*unsigned int max_length*

La longueur maximale du champ pour le jeu de résultats (la taille de la plus longue valeur de champ actuellement dans le jeu de résultat). Si vous utilisez `mysql_store_result()` ou `mysql_list_fields()` , cela contient la longueur maximale pour le champ. Si vous utilisez `mysql_use_result()` , la valeur de cette variable est zéro.

*unsigned int flags*

Les différents attributs sous forme de bits pour le champ. La valeur de `flags` peut avoir zéro ou plusieurs de ces bits suivants activés :

| Valeur d'attribut | Description d'attribut |
|---|---|

| | |
|---|---|
| NOT_NULL_FLAG | Le champ ne peut être NULL |
| PRI_KEY_FLAG | Le champ fait parti d'une clef primaire |
| UNIQUE_KEY_FLAG | Le champ fait parti d'une clef unique |
| MULTIPLE_KEY_FLAG | Le champ fait parti d'une clef non−unique |
| UNSIGNED_FLAG | Le champ possède l'attribut UNSIGNED |
| ZEROFILL_FLAG | Le champ possède l'attribut ZEROFILL |
| BINARY_FLAG | Le champ possède l'attribut BINARY |
| AUTO_INCREMENT_FLAG | Le champ possède l'attribut AUTO_INCREMENT |
| ENUM_FLAG | Le champ est un ENUM (désapprouvé) |
| SET_FLAG | Le champ est un SET (désapprouvé) |
| BLOB_FLAG | Le champ est un BLOB ou TEXT (désapprouvé) |
| TIMESTAMP_FLAG | Le champ est un TIMESTAMP (désapprouvé) |

L'utilisation des attributs `BLOB_FLAG` , `ENUM_FLAG` , `SET_FLAG` , et `TIMESTAMP_FLAG` est désapprouvé car ils indiquent un type de champ plutôt qu'un attribut de type de champ. Il est préférable de tester `field->type` avec `FIELD_TYPE_BLOB` , `FIELD_TYPE_ENUM` , `FIELD_TYPE_SET` , ou `FIELD_TYPE_TIMESTAMP` à la place.

L'exemple suivant illustre une utilisation typique de la valeur de `flags` :

```
if (field->flags & NOT_NULL_FLAG)
   printf("Le champ ne peut être nul\n");
```

Vous pouvez utiliser les différentes macros ci−dessous pour déterminer le status booléen de la valeur de l'attribut :

| Status de l'attribut | Description |
|---|---|
| IS_NOT_NULL(flags) | Vrai si le champ est défini en tant que NOT NULL |
| IS_PRI_KEY(flags) | Vrai si le champ est une clef primaire |
| IS_BLOB(flags) | Vrai si le champ est un BLOB ou TEXT (désapprouvé; tester plutôt field->type ) |

*unsigned int decimals*

Le nombre de décimales pour les champs numériques.

## 8.4.2 Vue d'ensemble des fonctions de l'API C

Les fonctions disponibles dans l'API C sont listées ici et décrites en plus de détails dans la section suivante. C API Function Descriptions .

| Fonction | Description |
|---|---|
| **mysql_affected_rows()** | Retourne le nombre de lignes changées/effacées/insérés par le dernier UPDATE , DELETE , ou INSERT . |
| **mysql_change_user()** | Change l'utilisateur et la base de données pour une connexion ouverte. |
| **mysql_character_set_name()** | Retourne le nom du jeu de caractère de la connexion. |
| **mysql_close()** | Ferme une connexion au serveur. |
| **mysql_connect()** | Connecte à un serveur MySQL. Cette fonction est désapprouvée; utilisez mysql_real_connect() à la place. |
| **mysql_create_db()** | Crée une base de données. Cette fonction est désapprouvée, utilisez plutôt la commande SQL CREATE DATABASE . |
| **mysql_data_seek()** | Déplace le pointeur vers une ligne arbitraire dans le jeu de résultats de la requête. |
| **mysql_debug()** | Effectue un DBUG_PUSH avec la chaîne donnée. |
| **mysql_drop_db()** | Supprime une base de données. Cette fonction est désapprouvée, utilisez plutôt la commande SQL DROP DATABASE . |
| **mysql_dump_debug_info()** | Demande au serveur d'écrire les informations de débogage dans un fichier de log. |
| **mysql_eof()** | Détermine si la dernière ligne du jeu de résultats a été lue ou non. Cette fonction est désapprouvée, vous pouvez utiliser mysql_errno() ou mysql_error() à la place. |
| **mysql_errno()** | Retourne le numéro de l'erreur de la fonction appelée en dernier. |
| **mysql_error()** | Retourne le message d'erreur de la dernière fonction MySQL appelée. |
| **mysql_escape_string()** | Protège une chaîne en échappant les caractères spéciaux. |
| **mysql_fetch_field()** | Retourne le type du champ suivant dans la table. |
| **mysql_fetch_field_direct()** | Retourne le type d'une colonne, étant donné un numéro de champ. |
| **mysql_fetch_fields()** | Retourne un tableau avec toutes les structures de champs. |
| **mysql_fetch_lengths()** | Retourne la longueur de toutes les colonnes dans la ligne suivante. |
| **mysql_fetch_row()** | Récupère la ligne suivante dans le jeu de résultats. |
| **mysql_field_seek()** | Place le curseur de colonne sur une colonne précise. |
| **mysql_field_count()** | Retourne le nombre de colonnes dans le résultat pour la requête la plus récente. |
| **mysql_field_tell()** | Retourne la position du curseur de champs utilisé pour le dernier appel à mysql_fetch_field() . |
| **mysql_free_result()** | Libère la mémoire utilisée par un jeu de résultats. |

| | |
|---|---|
| **mysql_get_client_info()** | Retourne la version du client. |
| **mysql_get_host_info()** | Retourne une chaîne décrivant la connexion. |
| **mysql_get_server_version()** | Retourne le numéro de version du serveur sous forme d'entier (nouveau en 4.1). |
| **mysql_get_proto_info()** | Retourne la version du protocole utilisé par la connexion. |
| **mysql_get_server_info()** | Retourne la version du serveur. |
| **mysql_info()** | Retourne des informations à propos de la requête la plus récente. |
| **mysql_init()** | Récupère ou initialise une structure MYSQL . |
| **mysql_insert_id()** | Retourne l'identifié généré pour une colonne AUTO_INCREMENT par la dernière requête. |
| **mysql_kill()** | Termine un processus donné. |
| **mysql_list_dbs()** | Retourne les noms des bases de données répondant à une expression régulière simple. |
| **mysql_list_fields()** | Retourne les noms des champs répondants à une expression régulière simple. |
| **mysql_list_processes()** | Retourne une liste des processus courants du serveur. |
| **mysql_list_tables()** | Retourne les noms des tables répondants à une expression régulière simple. |
| **mysql_num_fields()** | Retourne le nombre de colonnes dans un jeu de résultats. |
| **mysql_num_rows()** | Retourne le nombre de lignes dans le jeu de résultats. |
| **mysql_options()** | Configure les options de connexion pour mysql_connect() . |
| **mysql_ping()** | Vérifie si la connexion au serveur a toujours lieu. Reconnecte si besoin. |
| **mysql_query()** | Exécute une requête SQL spécifée en tant que chaîne terminée par un caractère nul. |
| **mysql_real_connect()** | Connecte à un serveur MySQL. |
| **mysql_real_escape_string()** | Protège les caractères spéciaux dans une chaîne utilisable dans une requête SQL, en prenant en compte le jeu de caractères courant de la connexion. |
| **mysql_real_query()** | Exécute une requête SQL spécifiée en tant que chaîne comptée. |
| **mysql_reload()** | Demande au serveur de recharger la table des droits. |
| **mysql_row_seek()** | Déplace le pointeur vers un ligne dans le jeu de résultats, en utilisant une valeur retournée par mysql_row_tell() . |
| **mysql_row_tell()** | Retourne la position du curseur de lignes. |
| **mysql_select_db()** | Sélectionne une base de données. |
| **mysql_shutdown()** | Termine le serveur de base de données. |
| **mysql_stat()** | Retourne le statut du serveur dans une chaîne. |
| **mysql_store_result()** | Récupère le jeu de résulats complet dans le client. |
| **mysql_thread_id()** | Retourne l'identifiant du thread courant. |
| **mysql_thread_safe()** | Retourne 1 si le client est compilé en tant que thread−safe. |
| **mysql_use_result()** | Initialise une récupération ligne par ligne des résultats. |
| **mysql_commit()** | Valide une transaction (nouveau en 4.1). |
| **mysql_rollback()** | Annule une transaction (nouveau en 4.1). |
| **mysql_autocommit()** | Active et désactive le mode d'auto validation (nouveau en 4.1). |
| **mysql_more_results()** | Vérifie si il n'y a plus de résultats (nouveau en 4.1). |
| **mysql_next_result()** | Retourne/initie le prochain résultat dans une commande multi−requête (nouveau en 4.1). |

Pour vous connecter au serveur, appelez mysql_init() pour initialiser un gestionnaire de connexion, puis appelez mysql_real_connect() avec ce gestionnaire (avec d'autres informations tel que l'hôte, l'utilisateur et le mot de passe). Lors de la connexion, mysql_real_connect() définit l'option reconnect (quit fait partie de la structure MYSQL) à 1 . Cette option indique, dans le cas où une requête ne peut être exécutée à cause d'une perte de connexion, d'essayer de se reconnecter au serveur avant d'abandonner. Lorsque vous n'avez plus besoin de la connexion, appelez mysql_close() pour la clore.

Tant qu'une connexion est active, le client envoi des requêtes SQL au serveur à l'aide de mysql_query() ou mysql_real_query() . La différence entre les deux est que mysql_query() s'attend à ce que la requête soit spécifiée en tant que chaîne terminée par la chaîne nulle, tandis que mysql_real_query() attend une chaîne de longueur connue. Si la chaîne contient des données binaires (incluant l'octet nul), vous devez utiliser mysql_real_query() .

Pour chaque requête non−sélective (par exemple, INSERT , UPDATE , DELETE ), vous pouvez trouver combien de lignes ont été mises à jour (affectées) en appelant mysql_affected_rows() .

Pour les requêtes SELECT , vous récupérez les lignes sélectionnées dans un jeu de résultat. (Notez que quelques commandes ont le même comportement que SELECT , dans le sens où elle renvoient des lignes. Cela inclut SHOW , DESCRIBE , et EXPLAIN . Elles doivent être traitées de la même façon que les requêtes SELECT .)

Il y a deux façons pour un client de gérer les jeux de résultats. Une méthode consiste à récupérer le jeu de résultat en entier et en une seule fois en appelant mysql_store_result() . Cette fonction obtient toutes les lignes retournées par la requête et les stocke dans le client. La seconde méthode consiste à initialiser une récupération ligne−par−ligne du jeu de résultats en appelant mysql_use_result() . Cette fonction initie la récupération, mais ne récupère actuellement aucune ligne à partir du serveur.

Dans les deux cas, vous accédez aux ligne en appelant `mysql_fetch_row()` . Avec `mysql_store_result()` , `mysql_fetch_row()` accède aux lignes qui ont déjà été récupérées à partir du serveur. Avec `mysql_use_result()` , `mysql_fetch_row()` récupère actuellement la ligne à partir du serveur. Les informations à propos de la taille des données dans chaque ligne est disponible en appelant `mysql_fetch_lengths()` .

Après avoir fini de traiter le jeu de résultats, appelez `mysql_free_result()` pour libérer la mémoire utilisée.

Les deux mécanismes de récupération sont complémentaires. Les programmes clients doivent utiliser l'approche qui leur convient le mieux. En pratique, les clients tendent plus à utiliser `mysql_store_result()` .

Un avantage de `mysql_store_result()` est que puisque toutes les lignes ont été récupérées dans le client, vous ne pouvez pas que accéder aux lignes séquentiellement, vous pouvez revenir en arrière ou avancer dans le jeu de résultats en utilisant `mysql_data_seek()` ou `mysql_row_seek()` pour changer la position de la ligne courante dans le jeu de résultats. Vous pouvez aussi trouver le nombre total des lignes en appelant `mysql_num_rows()` . D'un autre côté, les besoins en mémoire de `mysql_store_result()` peuvent être très grands pour les grands jeux de résultats et vous aurez des chances de rencontrer des conditions out−of−memory.

Un avantage de `mysql_use_result()` est que le client a besoinde moins de mémoire pour le jeu de résultats car il utilise une ligne à la fois (et puisque il y a moins de pertes de mémoire, `mysql_use_result()` peut être plus rapide). Les inconvénients sont que vous devez récupérer chaque ligne rapidement pour éviter de bloquer le serveur, vous n'avez pas d'accès aléatoires aux lignes dans le jeu de résultats (vous ne pouvez accéder aux lignes que séquentiellement), et vous ne savez pas combien de lignes comporte le jeu de résultats tant que vous ne les avez pas toutes récupérées. De plus, vous **devez** récupérer toutes les lignes même si vous trouvez entre−temps l'informations que vous cherchiez.

L'API permet aux clients de gérer correctement les requêtes (récupérant les lignes seulement en cas de besoin) sans savoir si la requête était un `SELECT` ou non. Vous pouvez faire cela en appelant `mysql_store_result()` après chaque `mysql_query()` (ou `mysql_real_query()` ). Si l'appel au jeu de résultats réussi, la requête était un `SELECT` et vous pouvez lire les lignes. Sinon, appelez `mysql_field_count()` pour vérifier si un résultat aurait du être retourné. Si `mysql_field_count()` retourne zéro, la requête n'a pas retourné de données (cela indique que c'était un `INSERT` , `UPDATE` , `DELETE` , etc.), et ne devait pas retourner de lignes. Si `mysql_field_count()` est non−nul, la requête aurait du retourner des lignes, mais ne l'a pas fait. Cela indique que la requête était un `SELECT` qui a échoué. Reportez vous à la description de `mysql_field_count()` pour un exemple d'utilisation.

`mysql_store_result()` et `mysql_use_result()` vous permettent d'obtenir des informations à propos des champs qui constituent le jeu de résultat (le nombre de champs, leurs noms et types, etc.). Vous pouvez accéder aux informations du champ séquentiellement dans une ligne en appelant plusieurs fois `mysql_fetch_field()` , ou avec le numéro du champ dans la ligne en appelant `mysql_fetch_field_direct()` . La position courante du pointeur de champ peut être changée en appelant `mysql_field_seek()` . Changer le pointeur de champ affecte les appels suivants à `mysql_fetch_field()` . Vous pouvez aussi obtenir en une seule fois les informations sur les champs en appelant `mysql_fetch_fields()` .

Pour détecter les erreurs, MySQL fournit un accès aux informations des erreurs via les fonctions `mysql_errno()` et `mysql_error()` . Elles retournent le code de l'erreur et le message pour la dernière fonction invoquée qui aurait pu réussir ou échouer, vous permettant ainsi de déterminer les erreurs et leurs causes.

## 8.4.3 Description des fonctions de l'API C

Dans les descriptions suivantes, un paramètre ou retour de fonction `NULL` correspond au `NULL` dans le sens C du terme, et non dans le sens de la valeur `NULL` de MySQL.

Les fonctions qui retournent une valeur retournent la plupart du temps un pointeur ou un entier. Sauf en cas d'indications contraires, les fonctions retournant un pointeur retournent une valeur non−`NULL` pour indiquer un succès ou une valeur `NULL` pour indiquer une erreur, les fonctions retournant un entier retournent zéro pour indiquer un succès et une valeur non−nulle en cas d'erreur. Notez que ``non−nulle'' ne signifie rien de plus que cela. Sauf si la description de la fonction le dit, ne testez pas avec une valeur différente de zéro :

```
if (result)                  /* correct */
   ... erreur ...

if (result < 0)              /* incorrect */
   ... erreur ...

if (result == −1)            /* incorrect */
   ... erreur ...
```

Lorsqu'une fonction retourne une erreur, la section **Erreurs** du descriptif de la fonction liste les types d'erreurs possibles. Vous pouvez trouver celle qui est arrivée en appelant `mysql_errno()` . Une chaîne de caractères représentant l'erreur peut être obtenue en appelant `mysql_error()` .

### 8.4.3.1 mysql_affected_rows()

```
my_ulonglong mysql_affected_rows(MYSQL *mysql)
```

Description

Retourne le nombre de lignes modifiées par la dernière commande `UPDATE` , supprimées par la dernière commande `DELETE` ou insérée par la dernière commande `INSERT` . Peut être appelée immédiatement après `mysql_query()` pour les commandes `UPDATE` , `DELETE` , ou `INSERT` . Pour la commande `SELECT` , `mysql_affected_rows()` fonctionne comme `mysql_num_rows()` .

Valeur de retour

Un entier supérieur à zéro indique le nombre de lignes affectées ou sélectionnées. Zéro indique qu'aucun enregistrement n'a été mis à jour pour une requête `UPDATE` , qu'aucune lignes n'a correspondu à la clause `WHERE` dans la requête ou que celle ci n'a pas encore été exécutée. −1 indique que la requête a renvoyé une erreur ou que, pour une requête `SELECT` , `mysql_affected_rows()` a été appelée avant `mysql_store_result()` .

Erreurs

Aucune.

Exemple

```
 mysql_query(&mysql,"UPDATE products SET cost=cost*1.25 WHERE group=10");
printf("%ld produits mis à jour",(long) mysql_affected_rows(&mysql));
```

Si on spécifie l'option `CLIENT_FOUND_ROWS` en se connectant à `mysqld` , `mysql_affected_rows()` retournera le nombre d'enregistrements correspondants à la clause `WHERE` pour une requête `UPDATE` .Notez que quand on utilise une commande `REPLACE` , `mysql_affected_rows()` retournera 2 si le nouvel enregistrement en a remplacé un ancien. 2 en retour car dans ce cas, l'ancienne ligne a été supprimé puis la nouvelle insérée.

### 8.4.3.2 mysql_change_user()

```
my_bool mysql_change_user(MYSQL *mysql, const char *user, const char *password, const char *db)
```

Description

Change l'utilisateur et définit la base de données spécifiée par `db` en tant que base de données par défaut (courante) dans la connexion spécifiée par `mysql` . Pour les requêtes suivantes, cette base de données sera celle utilisée pour les références aux tables ne spécifiant pas explicitement une base de données.

Cette fonction a été introduite à la version 3.23.3 de MySQL.

`mysql_change_user()` échoue si l'utilisateur ne peut être authentifié ou s'il n'a pas le droit d'utiliser cette base de données. Dans ce cas, l'utilisateur et la base de données ne sont pas changés.

Le paramètre `db` peut être mis à `NULL` si vous ne voulez pas avoir de base de données par défaut.

Valeur de retour

Zéro en cas de succès. Différent de zéro si une erreur se produit.

Erreurs

Les mêmes que vous pouvez obtenir avec `mysql_real_connect()` .

*CR_COMMANDS_OUT_OF_SYNC*
: Les commandes n'ont pas été exécutées dans le bon ordre.

*CR_SERVER_GONE_ERROR*
: Le serveur MySQL ne réponds pas.

*CR_SERVER_LOST*
: La connexion au serveur a été perdue au cours la requête.

*CR_UNKNOWN_ERROR*
: Une erreur inconnue s'est produite.

*ER_UNKNOWN_COM_ERROR*
: Le serveur MySQL n'implémente pas cette commande (probablement un ancien serveur)

*ER_ACCESS_DENIED_ERROR*
: L'utilisateur ou le mot de passe étaient erronés.

*ER_BAD_DB_ERROR*
: La base de données n'existe pas.

*ER_DBACCESS_DENIED_ERROR*
: L'utilisateur n'a pas le droit d'accéder à la base de données.

*ER_WRONG_DB_NAME*
: Le nom de la base de données était trop long.

Exemple

```
 if (mysql_change_user(&mysql, "user", "password", "new_database"))
&#123;
   fprintf(stderr, "Impossible de changer d'utilisateur.  Erreur : %s\n",
           mysql_error(&mysql));
&#125;
```

### 8.4.3.3 mysql_character_set_name()

```
const char *mysql_character_set_name(MYSQL *mysql)
```

Description

Retourne le jeu de caractères par défaut de la connexion courante.

Valeur de retour

Le jeu de caractères par défaut

Erreurs

Aucune.

### 8.4.3.4 mysql_close()

```
void mysql_close(MYSQL *mysql)
```

Description Ferme la connexion ouverte précédemment. `mysql_close()` désalloue aussi le pointeur de connexion pointé par `mysql` , si celui ci avait été alloué dynamiquement par `mysql_init()` ou `mysql_connect()` .

Valeur de retour

Aucune.

Erreurs

Aucune.

### 8.4.3.5 mysql_connect()

```
MYSQL *mysql_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd)
```

Description

Cette fonction est désapprouvée. Il est préférable d'utiliser `mysql_real_connect()` à la place.

`mysql_connect()` essaye d'établir une connexion à un serveur MySQL lancé sur `host` . `mysql_connect()` doit s'achever avec succès avant que vous ne puissiez exécuter l'une des autres fonctions de l'API, à l'exception de `mysql_get_client_info()` .

La signification des paramètres est la même que pour ceux de la fonction `mysql_real_connect()` à la différence que le paramètre de connexion peut être `NULL` . Dans ce cas, l'API C alloue automatiquement une mémoire pour la structure de connexion et la libère quand vous appelez `mysql_close()` . Le désavantage de cette approche est que vous ne pouvez pas récupérer les messages d'erreur si la connexion échoue. (Pour obtenir des informations sur les erreurs à partir de `mysql_errno()` ou `mysql_error()` , vous devez fournir un pointeur `MYSQL` valide.)

Valeur de retour

La même que pour `mysql_real_connect()` .

Erreurs

Les mêmes que pour `mysql_real_connect()` .

### 8.4.3.6 mysql_create_db()

```
int mysql_create_db(MYSQL *mysql, const char *db)
```

Description Crée la base de données nommée avec le paramètre `db` .

Cette fonction est désapprouvée. Il est préférable d'utiliser `mysql_query()` pour générer une requête SQL `CREATE DATABASE` à la place.

Valeur de retour

Zéro si la base à été crée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
: Les commandes n'ont pas été exécutées dans le bon ordre.

*CR_SERVER_GONE_ERROR*
: Le serveur MySQL ne réponds pas.

*CR_SERVER_LOST*
: La connexion au serveur a été perdue au cours la requête.

*CR_UNKNOWN_ERROR*
: Une erreur inconnue s'est produite.

Exemple

```
 if(mysql_create_db(&mysql, "ma_base"))
&#123;
   fprintf(stderr, "Impossible de créer une nouvelle base de données. Erreur : %s\n",
           mysql_error(&mysql));
&#125;
```

### 8.4.3.7 mysql_data_seek()

```
void mysql_data_seek(MYSQL_RES *result, my_ulonglong offset)
```

Description Se déplace vers une ligne arbitraire d'un jeu de résultat de requête. Cela nécessite que la structure du jeu de résultat contienne la totalité du résultat de la requête, de ce fait `mysql_data_seek()` peut être utilisée en conjonction avec `mysql_store_result()` , mais pas avec `mysql_use_result()` .

L'index de la ligne doit être compris entre 0 et `mysql_num_rows(result)-1` .

Valeur de retour

Aucune.

Erreurs Aucune.

### 8.4.3.8 mysql_debug()

```
void mysql_debug(const char *debug)
```

Description Provoque un `DBUG_PUSH` avec la chaîne donnée. `mysql_debug()` utilises la librairie de débogage Fred Fish. Pour utiliser cette fonction vous devez compiler la librairie client avec le support débogage. Déboguer un serveur MySQL . Déboguer un client MySQL .

Valeur de retour Aucune.

Erreurs Aucune.

Exemple

L'appel montré ici fais générer à la librairie du client un fichier de trace dans `/tmp/client.trace` sur la machine du client :

```
mysql_debug("d:t:O,/tmp/client.trace");
```

### 8.4.3.9 mysql_drop_db()

```
int mysql_drop_db(MYSQL *mysql, const char *db)
```

Description Supprime la base de données nommée avec le paramètre `db` .

Cette fonction est désapprouvée. Il est préférable d'utiliser `mysql_query()` pour générer une requête SQL `DROP DATABASE` à la place.

Valeur de retour

Zéro si la base à été effacée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
: Les commandes n'ont pas été exécutées dans le bon ordre.

*CR_SERVER_GONE_ERROR*
: Le serveur MySQL ne réponds pas.

*CR_SERVER_LOST*
: La connexion au serveur a été perdue au cours la requête.

*CR_UNKNOWN_ERROR*
: Une erreur inconnue s'est produite.

Exemple

```
if(mysql_drop_db(&mysql, "ma_base"))
 fprintf(stderr, "Impossible de supprimer la base de données. Erreur : %s\n",
         mysql_error(&mysql));
```

### 8.4.3.10 mysql_dump_debug_info()

```
int mysql_dump_debug_info(MYSQL *mysql)
```

Description

Demande au serveur d'écrire quelques informations de débogage dans le log. Pour que cela fonctionne, il faut que l'utilisateur ait le droit `SUPER` .

Valeur de retour

Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
: Les commandes n'ont pas été exécutées dans le bon ordre.

*CR_SERVER_GONE_ERROR*
: Le serveur MySQL ne réponds pas.

*CR_SERVER_LOST*
: La connexion au serveur a été perdue au cours la requête.

*CR_UNKNOWN_ERROR*
: Une erreur inconnue s'est produite.

### 8.4.3.11 mysql_eof()

```
my_bool mysql_eof(MYSQL_RES *result)
```

Description

Cette fonction est désapprouvée. Vous pouvez utiliser `mysql_errno()` ou `mysql_error()` à la place.

`mysql_eof()` détermine si la dernière ligne d'un jeu de résultats a été lue.

Si vous obtenez un jeu de résultats suite à un appel à `mysql_store_result()` , le client reçois le jeu entier en une seule opération. Dans ce cas, un retour `NULL` de la fonction `mysql_fetch_row()` signifie toujours que la fin du jeu de résultat a été atteinte et il n'est donc pas nécessaire d'appeler `mysql_eof()` . Lors d'une utilisation avec `mysql_store_result()` , `mysql_eof()` retournera toujours true.

D'un autre côté, si vous utilisez `mysql_use_result()` pour initialiser la récupération d'un jeu de résultats, les lignes sont obtenues du serveur une par une lors des appels successifs de `mysql_fetch_row()` . Puisque une erreur peut survenir à la connexion durant ce processus, une valeur de retour `NULL` de la part de `mysql_fetch_row()` ne signifie pas nécessairement que la fin du jeu de résultats a été atteinte normalement. Dans ce cas, vous pouvez utiliser `mysql_eof()` pour déterminer ce qui est arrivé. `mysql_eof()` retourne une valeur non−nulle si la fin du jeu de résultats a été atteinte et zéro en cas d'erreur.

Historiquement, `mysql_eof()` a vu le jour avant les fonctions d'erreurs standards de MySQL `mysql_errno()` et `mysql_error()` . Puisque ces fonctions fournissent les mêmes informations, leur utilisation est préférée à `mysql_eof()` , qui est maintenant désapprouvée. (En fait, elles fournissent plus d'informations, car `mysql_eof()` ne retourne que des valeurs booléennes alors que les fonctions d'erreurs indiquent les raisons des erreurs lorsqu'elles surviennent.)

Valeur de retour

Zéro si aucune erreur n'est survenue. Autre chose dans le cas contraire.

Erreurs Aucune.

Exemple

L'exemple suivant vous montre comment vous devez utiliser `mysql_eof()` :

```
 mysql_query(&mysql,"SELECT * FROM une_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
&#123;
    // traite les données
&#125;
if(!mysql_eof(result))  // mysql_fetch_row() failed due to an error
&#123;
    fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
&#125;
```

Vous pouvez reproduire la même chose avec les fonctions d'erreurs de MySQL :

```
 mysql_query(&mysql,"SELECT * FROM une_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
&#123;
    // traite les données
&#125;
if(mysql_errno(&mysql))  // mysql_fetch_row() ne marche pas à cause d'une erreur
&#123;
    fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
&#125;
```

### 8.4.3.12 mysql_errno()

```
unsigned int mysql_errno(MYSQL *mysql)
```

Description

Pour la connexion spécifiée par `mysql` , `mysql_errno()` retourne le code de l'erreur pour l'appel le plus récent à une fonction de l'API qui peut réussir ou échouer. Un zéro en valeur de retour signifie qu'aucune erreur ne s'est produite. Les codes erreur du client sont listés dans le fichier d'entête MySQL ( `errmsg.h` ). Les codes erreur du serveur sont listés dans le fichier `mysqld_error.h` . Dans les sources de la distribution MySQL vous pouvez trouver la liste complète des messages d'erreur et le code qui leur est associé dans le fichier `Docs/mysqld_error.txt` .

Valeur de retour

Un code d'erreur. Zéro si aucune erreur n'est survenue.

Erreurs Aucune.

### 8.4.3.13 mysql_error()

```
char *mysql_error(MYSQL *mysql)
```

Description

Pour la connexion spécifiée par `mysql` , `mysql_error()` retourne le message d'erreur pour l'appel le plus récent à une fonction de l'API qui peut réussir ou échouer. Une chaîne vide ( `""` ) est retournée si aucune erreur n'est survenue. Cela signifie que les deux tests suivants sont équivalents :

```
 if(mysql_errno(&mysql))
&#123;
    // une erreur est survenue
&#125;

 if(mysql_error(&mysql)[0] != '\0')
&#123;
    // une erreur est survenue
&#125;
```

La langue des messages d'erreurs peut être changée en recompilant la librairie du client MySQL. Actuellement, vous pouvez choisir les messages d'erreur parmi un choix de plusieurs langues.

Messages d'erreurs non−anglophones . Valeur de retour

Une chaîne de caractères qui décrit l'erreur. Une chaîne vide si aucune erreur n'est survenue.

Erreurs Aucune.

### 8.4.3.14 mysql_escape_string()

Vous devez utiliser la fonction `mysql_real_escape_string()` à la place de celle ci !

Cette fonction est identique à `mysql_real_escape_string()` à l'exception faite que `mysql_real_escape_string()` prends deux identifiants de connexion comme premiers arguments et échappe la chaîne en se basant que le jeu de caractères courant. `mysql_escape_string()` ne prends pas d'identifiant de connexion et ne respecte pas le jeu de caractères courant.

### 8.4.3.15 mysql_fetch_field()

```
MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *result)
```

Description

Retourne la définition d'une colonne d'un jeu de résultats en tant que structure `MYSQL_FIELD` . Appelez cette fonction plusieurs fois pour obtenir des informations à propos de toutes les colonnes dans le jeu de résultat. `mysql_fetch_field()` retourne `NULL` quand il ne reste plus de champs.

`mysql_fetch_field()` est mis à zéro pour retourner des informations à propos du premier champ à chaque fois que vous exécutez une nouvelle requête `SELECT` . Le champ retourné par `mysql_fetch_field()` est aussi affecté par les appels à `mysql_field_seek()` .

Si vous avez appelé `mysql_query()` pour exécuter un `SELECT` sur une table mais n'avez pas appelé `mysql_store_result()` , MySQL retourne la longueur par défaut du blob (8K octets) si vous avez appelé `mysql_fetch_field()` pour obtenir la longueur d'un champ `BLOB` . (La taille 8K est choisie car MySQL ne connaît pas la longueur maximale du `BLOB` . Cela devrait être un jour configurable.) Une fois que vous avez récupéré le jeu de résultats, `field->max_length` contient la longueur de la plus grande valeur de cette colonne dans la requête spécifiée.

Valeur de retour

La structure `MYSQL_FIELD` de la colonne courante. `NULL` s'il ne reste plus de colonnes.

Erreurs Aucune.

Exemple

```
 MYSQL_FIELD *field;

 while((field = mysql_fetch_field(result)))
&#123;
    printf("nom du champ : %s\n", field->name);
&#125;
```

### 8.4.3.16 mysql_fetch_fields()

```
MYSQL_FIELD *mysql_fetch_fields(MYSQL_RES *result)
```

Description

Retourne un tableau de toutes les structures `MYSQL_FIELD` dans un jeu de résultats. Chaque structure fournit la définition de champ d'une colonne dans le jeu de résultats.

Valeur de retour

Un tableau de structures `MYSQL_FIELD` pour toutes les colonnes dans le jeu de résultat.

Erreurs Aucune.

Exemple

```
 unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *fields;

 num_fields = mysql_num_fields(result);
fields = mysql_fetch_fields(result);
for(i = 0; i < num_fields; i++)
&#123;
   printf("Le champ %u est %s\n", i, fields[i].name);
&#125;
```

### 8.4.3.17 mysql_fetch_field_direct()

```
MYSQL_FIELD *mysql_fetch_field_direct(MYSQL_RES *result, unsigned int fieldnr)
```

Description

Etant donné un numéro de champ `fieldnr` pour une colonne dans un jeu de résultats, cette fonction retourne la définition de ce champ en tant que structure `MYSQL_FIELD` . Vous pouvez utiliser cette fonction pour obtenir la définition d'une colonne choisie arbitrairement. La valeur de `fieldnr` doit varier entre 0 et `mysql_num_fields(result)-1` .

Valeur de retour

La structure `MYSQL_FIELD` pour la colonne spécifiée.

Erreurs Aucune.

Exemple

```
 unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *field;

 num_fields = mysql_num_fields(result);
for(i = 0; i < num_fields; i++)
&#123;
    field = mysql_fetch_field_direct(result, i);
    printf("La champ %u est %s\n", i, field->name);
&#125;
```

### 8.4.3.18 mysql_fetch_lengths()

```
unsigned long *mysql_fetch_lengths(MYSQL_RES *result)
```

Description

Retourne les longueurs des colonnes de la ligne courante dans le jeu de résultats. Si vous voulez copier les valeurs des champs, cette information sur la longueur est très utile pour l'optimisation, car vous pouvez éviter les appels à `strlen()` . De plus, si le jeu de résultat contient des données binaires, vous **devez** cette fonction pour déterminer la longueur des données, car `strlen()` retourne des résultats incorrects pour les champs contenant des caractères nuls.

La longueur des colonnes vides et des colonnes contenant la valeur `NULL` est zéro. Pour savoir comment distinguer ces cas, voyez la description de `mysql_fetch_row()` .

Valeur de retour

Un tableau d'entiers longs non−signés représentant la taille de chaque colonne (n'incluant pas la caractère nul de terminaison). `NULL` si une erreur se produit.

Erreurs `mysql_fetch_lengths()` n'st valide que pour la ligne courante du jeu de résultats. Cette fonction retourne `NULL` si vous l'appelez avant d'appeler `mysql_fetch_row()` ou après avoir récupéré toutes les lignes du résultat.

Exemple

```
 MYSQL_ROW row;
unsigned long *lengths;
unsigned int num_fields;
unsigned int i;

 row = mysql_fetch_row(result);
if (row)
&#123;
    num_fields = mysql_num_fields(result);
    lengths = mysql_fetch_lengths(result);
    for(i = 0; i < num_fields; i++)
    &#123;
         printf("La colonne %u a %l octets de longueur.\n", i, lengths[i]);
    &#125;
&#125;
```

### 8.4.3.19 mysql_fetch_row()

```
MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)
```

Description

Récupère la ligne suivante d'un jeu de résultats. Lorsqu'elle est utilisée après `mysql_store_result()` , `mysql_fetch_row()` retourne `NULL` quand il n'y a plus de lignes a récupérer. Lorsqu'elle est utilisée après `mysql_use_result()` , `mysql_fetch_row()` retourne `NULL` quand il n'y a plus de lignes a récupérer ou qu'une erreur est rencontrée.

Le nombre de valeurs dans la ligne est donné par `mysql_num_fields(result)` . Si `row` contient la valeur de retour d'un appel à `mysql_fetch_row()` , les pointeurs sur les valeurs sont accédées de `row[0]` à `row[mysql_num_fields(result)−1]` . Les valeurs `NULL` de la ligne sont indiquées par des pointeurs `NULL` .

La longueur de la valeur du champ dans la ligne peut être obtenue en appelant `mysql_fetch_lengths()` . Les champs vides et les champs contenant `NULL` ont tous deux une longueur égale à zéro; vous pouvez les distinguer en vérifiant le pointeur sur la valeur du champ. Si le pointeur est `NULL` , le champ est `NULL` ; sinon, le champ est vide.

Valeur de retour

Une structure `MYSQL_ROW` pour la prochaine ligne. `NULL` s'il n'y a plus de lignes a récupérer ou qu'une erreur survient.

Erreurs

*CR_SERVER_LOST*
: La connexion au serveur a été perdue durant la requête.

*CR_UNKNOWN_ERROR*
: Une erreur inconnue est survenue.

Exemple

```
 MYSQL_ROW row;
unsigned int num_fields;
unsigned int i;

 num_fields = mysql_num_fields(result);
while ((row = mysql_fetch_row(result)))
&#123;
   unsigned long *lengths;
   lengths = mysql_fetch_lengths(result);
   for(i = 0; i < num_fields; i++)
   &#123;
       printf("[%.*s] ", (int) lengths[i], row[i] ? row[i] : "NULL");
   &#125;
   printf("\n");
&#125;
```

### 8.4.3.20 mysql_field_count()

```
unsigned int mysql_field_count(MYSQL *mysql)
```

Si vous utilisez une version de MySQL plus ancienne que la 3.22.24, vous devez utiliser `unsigned int mysql_num_fields(MYSQL *mysql)` .

Description

Retourne le nombre de colonnes pour la requête la plus récente de la connexion.

L'utilisation normale de cette fonction est lorsque `mysql_store_result()` a retourné `NULL` (et que vous n'avez donc pas de pointeur sur jeu de résultats). Dans ce cas, vous pouvez appeler `mysql_field_count()` pour déterminer si `mysql_store_result()` aurait dû produire un résultat non−vide. Cela permet au programme client d'entreprendre les bonnes actions sans savoir si la requête était un `SELECT` (ou équivalent). L'exemple suivant illustre comment cela peut être fait.

Pourquoi est ce qu'après `mysql_query()` ait indiqué un résultat positif, `mysql_store_result()` retourne parfois `NULL` ? .

Valeur de retour

Un entier non−signé représentant le nombre de champs dans un jeu de résultats.

Erreurs Aucune.

Exemple

```
 MYSQL_RES *result;
unsigned int num_fields;
unsigned int num_rows;

 if (mysql_query(&mysql,query_string))
```

```
&#123;
    // erreur
&#125;
else // requête bonne, traitons les données qu'elle renvoit
&#123;
    result = mysql_store_result(&mysql);
    if (result)  // il y a des lignes
    &#123;
        num_fields = mysql_num_fields(result);
        // récupère les lignes, puis appele mysql_free_result(result)
    &#125;
    else  // mysql_store_result() n'a rien retourné; est-ce normal ?
    &#123;
        if(mysql_field_count(&mysql) == 0)
        &#123;
            // la requête ne retourne aucune donnée
            // (ce n'était pas un SELECT)
            num_rows = mysql_affected_rows(&mysql);
        &#125;
        else // mysql_store_result() aurait du retourner des données
        &#123;
            fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
        &#125;
    &#125;
&#125;
```

Une alternative est de remplacer l'appel à `mysql_field_count(` par `mysql_errno(` . Dans ce cas, vous vérifiez directement les erreurs à partir de `mysql_store_result()` plutôt qu'à partir de `mysql_field_count()` si la requête était un `SELECT` .

### 8.4.3.21 mysql_field_seek()

```
MYSQL_FIELD_OFFSET mysql_field_seek(MYSQL_RES *result, MYSQL_FIELD_OFFSET offset)
```

Description

Place le pointeur de champs à la position donnée. Le prochain appel à `mysql_fetch_field()` récupérera la définition du champ de la colonne associée à cet index.

Pour vous placer au début d'une ligne, passez 0 comme valeur d' `offset` .

Valeur de retour

La dernière valeur de l'index de champ.

Erreurs Aucune.

### 8.4.3.22 mysql_field_tell()

```
MYSQL_FIELD_OFFSET mysql_field_tell(MYSQL_RES *result)
```

Description

Retourne la position du curseur de champ utilisé pour le dernier appel à `mysql_fetch_field()` . Cette valeur peut être utilisée en argument de `mysql_field_seek()` .

Valeur de retour

L'indice courant du curseur de champ.

Erreurs Aucune.

### 8.4.3.23 mysql_free_result()

```
void mysql_free_result(MYSQL_RES *result)
```

Description

Libère la mémoire alloué à un résultat avec mysql_store_result() , mysql_use_result() , mysql_list_dbs() , etc. Quand vous n'avez plus besoin d'un jeu de résultat, vous devez libérer la mémoire qu'il occupe en appelant mysql_free_result() .

Valeur de retour Aucune.

Erreurs Aucune.

### 8.4.3.24 mysql_get_client_info()

```
char *mysql_get_client_info(void)
```

Description

Retourne une chaîne représentant la version de la librairie du client.

Valeur de retour

Une chaîne de caractères représentant la version de la librairie du client.

Erreurs Aucune.

### 8.4.3.25 mysql_get_server_version()

```
unsigned long mysql_get_server_version(MYSQL *mysql)
```

Description

Retourne le numéro de version du serveur, sous la forme d'un entier (nouveau en 4.1).

Valeurs retournées

Un nombre, qui représente le numéro de version du serveur MySQL, au format suivant :

```
main_version*10000 + minor_version *100 + sub_version
```

Par exemple, 4.1.0 est retournée comme ceci : 40100.

Ceci est pratique pour déterminer rapidement le numéro de version d'un serveur, dans un client, et connaitre ainsi ses fonctionnalités.

Erreurs Aucune.

### 8.4.3.26 mysql_get_host_info()

```
char *mysql_get_host_info(MYSQL *mysql)
```

Description

Retourne une chaîne de caractères décrivant le type de connexion actuellement utilisé, incluant le nom du serveur.

Valeur de retour

Une chaîne de caractères représentant le nom du serveur et le type de connexion.

Erreurs Aucune.

### 8.4.3.27 mysql_get_proto_info()

```
unsigned int mysql_get_proto_info(MYSQL *mysql)
```

## Description

Retourne la version du protocole utilisé par la connexion courante.

## Valeur de retour

Un entier non signé représentant la version du protocole utilisé par la connexion courante.

## Erreurs Aucune.

### 8.4.3.28 mysql_get_server_info()

```
char *mysql_get_server_info(MYSQL *mysql)
```

## Description

Retourne une chaîne représentant le numéro de version du serveur.

## Valeur de retour

Une chaîne de caractères représentant le numéro de version du serveur.

## Erreurs Aucune.

### 8.4.3.29 mysql_info()

```
char *mysql_info(MYSQL *mysql)
```

## Description

Récupère une chaîne de caractères fournissant des informations à propos de la requête exécutée le plus récemment, mais seulement pour celles listées ici. Pour les autres requêtes, mysql_info() retournera NULL . Le format de la chaîne varie selon le type de requête, comme décrit ici. Les nombres présentés sont des exemples; la chaîne retournée contiendra les informations correspondantes à vos requêtes.

```
INSERT INTO ... SELECT ...
          String format: Records: 100 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
          String format: Records: 3 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
          String format: Records: 1 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
          String format: Records: 3 Duplicates: 0 Warnings: 0
UPDATE
          String format: Rows matched: 40 Changed: 40 Warnings: 0
```

Notez que mysql_info() retourne une valeur non-nulle ( NULL ) pour les requêtes INSERT ... VALUES seulement si une liste de valeurs multiples est fournie à la requête.

## Valeur de retour

Une chaîne de caractères représentant des informations additionnelles à propos de la dernière requête exécutée. NULL si aucune information n'est disponible pour la requête.

## Erreurs Aucune.

### 8.4.3.30 mysql_init()

```
MYSQL *mysql_init(MYSQL *mysql)
```

Description

```
Alloue ou initialise un objet MYSQL convenable pour mysql_real_connect() . Si mysql est un pointeur NULL , la fonction alloue,
initialise et retourne un nouvel objet. Sinon, l'objet est initialisé et son adresse est retournée. Si mysql_init() alloue un nouvel
objet, il sera libéré quand mysql_close() sera appelée pour clore la connexion.
```

Valeur de retour

```
Un gestionnaire MYSQL* initialisé. NULL s'il n'y avait pas assez de mémoire pour allouer le nouvel objet.
```

Erreurs Si la mémoire est insuffisante, `NULL` est retourné.

### 8.4.3.31 mysql_insert_id()

```
my_ulonglong mysql_insert_id(MYSQL *mysql)
```

Description

```
Retourne l'identifiant généré pour une colonne AUTO_INCREMENT par la dernière requête. Utilisez cette commande après avoir exécuté une
requête INSERT sur une table qui contient un champ AUTO_INCREMENT .
```

Notez que `mysql_insert_id()` retourne `0` si la dernière requête n'a pas généré de valeur `AUTO_INCREMENT` . Si vous voulez garder cette valeur pour plus tard, assurez vous d'appeler `mysql_insert_id()` immédiatement après la requête ayant généré cette valeur.

```
mysql_insert_id() est mis à jour après l'exécution de requêtes INSERT et UPDATE qui génèrent une valeur AUTO_INCREMENT ou qui
definissent la valeur d'une colonne à LAST_INSERT_ID(expr) . Fonctions diverses .
```

Notez aussi que la valeur de retour de la fonction SQL `LAST_INSERT_ID()` contient toujours la valeur d' `AUTO_INCREMENT` la plus à jour. Cette valeur n'est pas remise à zéro lors de l'exécution d'autre requêtes car elle est maintenue pour le serveur.

```
Valeur de retour
```

La valeur de la colonne `AUTO_INCREMENT` qui a été mise à jour par la dernière requête. Retourne zéro si aucune requête n'avait eu lieu durant la connexion, ou si la dernière requête n'a pas mis à jour la valeur de la colonne `AUTO_INCREMENT` .

```
Erreurs Aucune.
```

### 8.4.3.32 mysql_kill()

```
int mysql_kill(MYSQL *mysql, unsigned long pid)
```

```
Description
```

Demande au serveur de terminer le thread spécifié par `pid` .

```
Valeur de retour
```

Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

```
Erreurs

          CR_COMMANDS_OUT_OF_SYNC
                    Les commandes n'ont pas été exécutées dans le bon ordre.
          CR_SERVER_GONE_ERROR
                    Le serveur MySQL ne réponds pas.
```

*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

### 8.4.3.33 mysql_list_dbs()

MYSQL_RES *mysql_list_dbs(MYSQL *mysql, const char *wild)

## Description

Retourne une jeu de résultats se composant des noms des bases de données localisées sur le serveur qui correspondent à l'expression régulière spécifié par le paramètre wild . wild peut contenir les caractères spéciaux '%' ou '_' , ou peut être un pointeur NULL pour obtenir la liste de toutes les bases de données. Utiliser mysql_list_dbs() reviens à exécuter la requête SHOW databases [LIKE wild] .

Vous devez libérer le résultat avec mysql_free_result() .

Valeur de retour

Un jeu de résultats MYSQL_RES en cas de succès. NULL si une erreur est survenue.

Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

### 8.4.3.34 mysql_list_fields()

MYSQL_RES *mysql_list_fields(MYSQL *mysql, const char *table, const char *wild)

## Description

Retourne un jeu de résultats consistant des noms de champs dans une table qui correspondent l'expression régulière simple spécifiée par la paramètre wild . wild peut contenir les caractères spéciaux '%' ou '_' , ou peut être un pointeur NULL pour correspondre à tous les champs. Utiliser mysql_list_fields() revient à exécuter la requête SHOW COLUMNS FROM nom_de_table [LIKE wild] .

Notez qu'il est recommandé d'utiliser SHOW COLUMNS FROM nom_de_table au lieu de / mysql_list_fields() .

Vous devez libérer le résultat avec mysql_free_result() .

## Valeur de retour

Un jeu de résultats MYSQL_RES en cas de succès. NULL sinon.

## Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

### 8.4.3.35 mysql_list_processes()

MYSQL_RES *mysql_list_processes(MYSQL *mysql)

Description

Retourne un jeu de résultat décrivant les threads courants du serveur. C'est le même genre d'informations renvoyé par `mysqladmin processlist` ou une requête `SHOW PROCESSLIST` .

Vous devez libérer le jeu de résultat avec mysql_free_result() .

### Valeur de retour

Un jeu de résultat MYSQL_RES en cas de succès. NULL si une erreur est survenue.

### Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

## 8.4.3.36 mysql_list_tables()

MYSQL_RES *mysql_list_tables(MYSQL *mysql, const char *wild)

Description

Retourne un jeu de résultats consistant des noms de tables dans la base de données courante qui concordent avec l'expression régulière spécifié par le paramètre `wild` . `wild` peut contenir les caractères spéciaux `'%'` ou `'_'` , ou peut être un pointeur `NULL` pour obtenir toutes les tables. Faire appel à `mysql_list_tables()` revient à exécuter la requête `SHOW tables [LIKE wild]` .

Vous devez libérer le jeu de résultats avec mysql_free_result() .

### Valeur de retour

Un jeu d'enregistrements MYSQL_RES en cas de succès. NULL en cas d'erreurs.

### Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

## 8.4.3.37 mysql_num_fields()

unsigned int mysql_num_fields(MYSQL_RES *result)

ou

unsigned int mysql_num_fields(MYSQL *mysql)

La seconde forme ne fonctionne plus à partir de la version 3.22.24 de MySQL. Pour passer un argument MYSQL* , vous devez utiliser la fonction unsigned int mysql_field_count(MYSQL *mysql) à la place.

### Description

Retourne le nombre de colonnes dans un jeu de résultats.

Notez que vous pouvez obtenir le nombre de colonnes soit à partir d'un pointeur sur résultat, soit d'un pointeur de connexion. Vous utiliserez le pointeur de connexion si `mysql_store_result()` ou

mysql_use_result() ont retournés NULL (et que donc, vous n'avez pas de pointeur sur résultat). Dans ce cas, vous pouvez appeler mysql_field_count() pour déterminer si mysql_store_result() aurait du retourner un résultat non−vide. Cela permet au client d'effectuer les bonnes actions sans savoir si la requête était un SELECT (ou équivalent). L'exemple ci−dessous montre comment cela doit être utilisé.

Pourquoi est ce qu'après mysql_query() ait indiqué un résultat positif, mysql_store_result() retourne parfois NULL ? .

## Valeur de retour

Un entier non−signé représentant le nombre de champs dans un jeu de résultats.

## Erreurs Aucune.

Exemple

```
 MYSQL_RES *result;
unsigned int num_fields;
unsigned int num_rows;

 if (mysql_query(&mysql,query_string))
&#123;
    // erreur
&#125;
else // la requête fonctionne, on s'occupe des données
&#123;
    result = mysql_store_result(&mysql);
    if (result)  // il y a des lignes
    &#123;
        num_fields = mysql_num_fields(result);
        // recupérer les lignes, puis appeler mysql_free_result(result)
    &#125;
    else  // mysql_store_result() n'a rien retourné ! pourquoi ?
    &#123;
        if (mysql_errno(&mysql))
    &#123;
            fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
    &#125;
        else if (mysql_field_count(&mysql) == 0)
        &#123;
            // la requête ne retourne pas de données
            // (ce n'etait pas un SELECT)
            num_rows = mysql_affected_rows(&mysql);
        &#125;
    &#125;
&#125;
```

Une alternative (si vous savez que votre requête aurait du retourner des résultats) est de remplacer l'appel à mysql_errno( par un test sur la nullité de mysql_field_count( . Cela n'arrive que si un problème a été rencontré.

### 8.4.3.38 mysql_num_rows()

my_ulonglong mysql_num_rows(MYSQL_RES *result)

## Description

Retourne le nombre de lignes présentes dans le résultat.

L'utilisation de mysql_num_rows() dépend de si vous utilisez mysql_store_result() ou mysql_use_result() pour retourner le jeu résultat. Si vous utilisez mysql_store_result() , mysql_num_rows() peut être appelé immédiatement. Si vous utilisez mysql_use_result() , mysql_num_rows() ne retournera pas la valeur correcte tant que toutes les lignes du résultat n'auront pas été récupérées.

Valeur de retour

## Le nombre de lignes dans le résultat.

Erreurs Aucune.

## 8.4.3.39 mysql_options()

int mysql_options(MYSQL *mysql, enum mysql_option option, const char *arg)

Description

Cette fonction peut être utilisée pour spécifier des options de connexion et modifier le comportement de la session courante. Cette fonction peut être appelée plusieurs fois pour définir plusieurs options.

mysql_options() doit être appelée après mysql_init() et avant mysql_connect() ou mysql_real_connect() .

L'argument option est l'option que vous voulez configurer; l'arguement arg est la valeur pour cette option. Si l'option est un entier, arg doit pointer sur la valeur d'un entier.

Les valeurs possibles pour les options sont :

| Option | Type de l'argument | Fonction |
|---|---|---|
| MYSQL_OPT_CONNECT_TIMEOUT | unsigned int * | Délai d'inactivité maximal permis. |
| MYSQL_OPT_COMPRESS | Non utilisé | Utiliser le protocole compressé client/serveur. |
| MYSQL_OPT_LOCAL_INFILE | pointeur optionnel sur uint | Si aucun pointeur n'est donné, ou que celui-ci pointe sur un unsigned int != 0 la commande LOAD LOCAL INFILE est activée. |
| MYSQL_OPT_NAMED_PIPE | Non utilisé | Utiliser les tunnels nommés pour se connecter au serveur MySQL sur NT. |
| MYSQL_INIT_COMMAND | char * | Commande à exécuter lors de la connexion au serveur MySQL. Sera automatiquement ré-exécutée lors des reconnexions. |
| MYSQL_READ_DEFAULT_FILE | char * | Lit les options à partir du fichier d'options nommé plutôt que de my.cnf . |
| MYSQL_READ_DEFAULT_GROUP | char * | Lit les options à partir du groupe spécifié dans le fichier d'options my.cnf ou le fichier spécifié par MYSQL_READ_DEFAULT_FILE . |

Notez que le groupe client est toujours lu si vous utilisez MYSQL_READ_DEFAULT_FILE ou MYSQL_READ_DEFAULT_GROUP .

Le groupe spécifié dans le fichier des options peut contenir les options suivantes :

| Option | Description |
|---|---|
| connect-timeout | Délai d'inactivité maximal permis en secondes. Sous Linux, ce délai est aussi utilisé lors de l'attente de la première réponse du serveur. |
| compress | Utiliser le protocole compressé client/serveur. |
| database | Se connecter à cette base de données si aucune base n'a été sélectionnée à la connexion. |
| debug | Options de debogage. |
| disable-local-infile | Interdit l'utilisation de LOAD DATA LOCAL . |
| host | Nom d'hôte par défaut. |
| init-command | Commande lors de la connexion au serveur MySQL. Sera automatiquement ré-exécutée lors des reconnexions. |
| interactive-timeout | Revient à spécifier CLIENT_INTERACTIVE à mysql_real_connect() . mysql_real_connect(). |
| local-infile[=(0\|1)] | Si aucun argument, ou argument != 0, on permet alors l'utilisation de LOAD DATA LOCAL . |
| password | Mot de passe par défaut. |
| pipe | Utiliser les tunnels nommés pour se connecter à MySQL sur NT. |
| port | Port par défaut. |
| return-found-rows | Demande à mysql_info() de retourner les lignes trouvées au lieu des lignes mises à jour lors de l'utilisation de UPDATE . |
| socket | Numéro de socket par défaut. |
| user | Utilisateur par défaut. |

Notez que timeout a été remplacé par connect-timeout , mais que timeout fonctionne encore pour le moement.

Pour plus d'informations sur les fichiers d'options, reportez vous à Fichier d'options my.cnf .

### Valeur de retour

Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

### Exemple

```
 MYSQL mysql;

 mysql_init(&mysql);
mysql_options(&mysql,MYSQL_OPT_COMPRESS,0);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"odbc");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
```

```
&#123;
    fprintf(stderr, "Impossible de se connecter à la base de données. Erreur : %s\n",
          mysql_error(&mysql));
&#125;
```

Ce qui précède demande au client d'utiliser le protocole compressé client/serveur et lit les options optionnelles de la section odbc dans le fichier my.cnf .

### 8.4.3.40 mysql_ping()

int mysql_ping(MYSQL *mysql)

Description

Vérifie si la connexion au serveur est encore assurée. Si ce n'est pas le cas, une re−connexion automatique est tentée.

Cette fonction peut être utilisée par les clients qui restent inactifs longtemps, pour vérifier que le serveur n'a pas fermé la connexion et se re−connecter si nécessaire.

#### Valeur de retour

Zéro si le serveur répond. Autre que zéro si une erreur est survenue.

#### Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

### 8.4.3.41 mysql_query()

int mysql_query(MYSQL *mysql, const char *query)

Description Exécute la requête SQL pointée par la chaîne terminée par null query . La requête doit se composer d'une seule opération. Vous ne devez pas ajouter de caractère de terminaison ( ';' ) ou \g à la fin de la requête.

`mysql_query()` ne peut être utilisée pour les requêtes contenant des données bianaires, vous devez utiliser `mysql_real_query()` à la place. (LEs données binaires peuvent contenir le caractère `'\0'` , qui est interprété comme la fin de la chaîne requête.)

Si vous voulez savoir si la requête doit retourner un jeu de résultat ou non, vous pouvez utiliser mysql_field_count() pour vérifier. mysql_field_count() .

#### Valeur de retour

Zéro si la requête a été effectuée avec succès. Différente de zéro si une erreur est survenue.

#### Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

## 8.4.3.42 mysql_real_connect()

```
MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd, const char *db, unsigned int port,
const char *unix_socket, unsigned int client_flag)
```

Description

`mysql_real_connect()` essaye de se connecter à une base de données MySQL tournant sur l'hôte. `mysql_real_connect()` doit se terminer correctement avant que vous ne puissiez aucune autre fonction de l'API, à l'exception de `mysql_get_client_info()` .

Les paramètres sont spécifiés comme suit :

- Le premier paramètre doit être l'adresse d'une structure MYSQL existante. Avant d'appeler mysql_real_connect() vous devez appeler mysql_init() pour initialiser la structure MYSQL . Vous pouvez changer un tas d'options de connexion en appelant mysql_options() . mysql_options() .
- La valeur de host peut être un nom de domaine ou une adresse IP. Si host est NULL ou égal à la chaîne "localhost" , une connexion à la machine local est essayée. Si le système supporte les sockets (Unix) ou les tunnels nommés (Windows), elles sont utilisées au lieu de TCP/IP pour se connecter au serveur.
- La paramètre user contient l'identifiant MySQL de l'utilisateur. Si user est NULL , l'utilisateur courant est sous-entendu. Avec Unix c'est l'utilisateur courant. Avec Windows ODBC, le nom de l'utilisateur courant doit être spécifié explicitement. Comment remplir les différents champs dans le programme d'administrateur ODBC .
- La paramètre passwd contient le mot de passe de user . Si passwd est NULL , seules les entrées de la table user pour l'utilisateur ayant un champ vide seront testées. Cela permet à l'administrateur de mettre en place le système de privilèges MySQL de façon à ce que les utilisateurs aient divers privilèges selon qu'ils aient spécifié ou pas de mot de passe.

  Note : N'essayez pas d'encrypter le mot de passe avant l'appel à `mysql_real_connect()` ; l'encryptage du mot de passe est gérée automatiquement par l'API du client.
- db est le nom de la base de données. Si db n'est pas NULL , la connexion changera la base par défaut en cette valeur.
- Si port est différent de 0, sa valeur sera utilisé comme port de connexion TCP/IP. Notez que le paramètre host détermine le type de la connexion.
- Si unix_socket n'est pas NULL , la chaîne spécifie la socket ou le tunnel nommé à utiliser. Notez que le paramètre host détermine le type de la connexion.
- La valeur de client_flag est habituellement 0, mais peut être la combinaison des options suivantes dans des circonstances très spéciales :

| Nom de l'option | Description |
|---|---|
| CLIENT_COMPRESS | Utilise le protocole compressé. |
| CLIENT_FOUND_ROWS | Retourne le nombre de lignes trouvées, et non de lignes affectées. |
| CLIENT_IGNORE_SPACE | Autorise les espaces après les noms de fonctions. Rend tous les noms de fonctions des mots réservés. |
| CLIENT_INTERACTIVE | Autorise interactive_timeout secondes (au lieu de wait_timeout secondes) d'innactivité avant de fermer la connexion. |
| CLIENT_NO_SCHEMA | N'autorise pas la syntaxe db_name.tbl_name.col_name . Cela est fait pour ODBC. Il fait génèrer une erreur à l'analyseur si vous utilisez cette syntaxe, ce qui peut se réveler fort utile pour la chasse aux bogues dans les programmes ODBC. |
| CLIENT_ODBC | Le client est un client ODBC. Cela rend mysqld plus accueillant vis à vis de ODBC. |
| CLIENT_SSL | Utilise SSL (protocole encrypté). |

Valeur de retour

Un gestionnaire de connexion `MYSQL*` si la connexion a réussi, `NULL` si elle a échoué. Pour une connexion à succès, la valeur de retour est la même que la valeur du premier paramètre.

Erreurs

*CR_CONN_HOST_ERROR*
: Impossible de se connecter au serveur MySQL.

*CR_CONNECTION_ERROR*
: Impossible de se connecter au serveur MySQL local.

*CR_IPSOCK_ERROR*
: Impossible de créer une socket IP.

*CR_OUT_OF_MEMORY*
: Plus de mémoire.

*CR_SOCKET_CREATE_ERROR*
: Impossible de créer une socket UNIX.

*CR_UNKNOWN_HOST*
: Impossible de trouver l'adresse IP de l'hôte.

*CR_VERSION_ERROR*
: Une disparité de protocole a résulté de la tentative de connexion à un serveur avec une librairie de client qui utilise une version différente du protocole. Cela peut arriver si vous utilisez une très vieille librairie cliente pour vous connecter à un serveur qui n'a pas été démarré avec l'option --old-protocol .

*CR_NAMEDPIPEOPEN_ERROR*
: Impossible de créer un tunnel nommé sur Windows.

*CR_NAMEDPIPEWAIT_ERROR*
: Impossible d'attendre un tunnel nommé sur Windows.

*CR_NAMEDPIPESETSTATE_ERROR*
: Impossible d'obtenir un gestionnaire de tunnel sur Windows.

*CR_SERVER_LOST*
: Si connect_timeout > 0 et qu'il a fallu plus de connect_timeout secondes pour se connecter au serveur, ou que celui-ci n'a plus répondu durant l'exécution de init-command .

Exemple

```
 MYSQL mysql;

 mysql_init(&mysql);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"your_prog_name");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
&#123;
    fprintf(stderr, "Impossible de se connecter à la base de données, erreur : %s\n",
          mysql_error(&mysql));
&#125;
```

En utilisant mysql_options() la librairie MySQL lira les sections [client] et [your_prog_name] dans le fichier my.cnf ce qui assurera le bon fonctionnement de votre programme, même si quelqu'un a configuré MySQL d'une façon non-standard.Notez que pendant la connexion, mysql_real_connect() configure l'option reconnect (partie de la structure MYSQL ) à 1 . Cette option indique, dans le cas où une requête ne peut être exécutée à cause d'une déconnexion, d'essayer de se reconnecter au serveur avant d'abandonner.

### 8.4.3.43 mysql_real_escape_string()

unsigned long mysql_real_escape_string(MYSQL *mysql, char *en, const char *de, unsigned long longueur)

Description

Cette fonction est utilisée pour créer une requête SQL légale que vous pouvez utiliser dans une commande SQL. Chaînes .

La string dans de est encodée en chaîne échappé SQL, prenom en compte le jeu de caractères de la connexion. Le résultat est placé dans en et un octet nul de terminaison est ajouté à la fin de celui-ci. Les caractères encodés sont NUL (ASCII 0), '\n' , '\r' , '\' , ''' , '"' , et Ctrl-Z ( Comment écrire les chaînes et les nombres ? ). (En fait, MySQL a seulement besoin que l'anti-slash et le guillemet utilisé pour entourer la chaîne soient échappés. Cette fonction échappe les autre caractères pour les rendre plus facile à lire dans les fichiers de log.)

La chaîne pointée par de doit avoir une taille de longueur octets. Vous devez allouer à l'espace de en au moins longueur*2+1 octets. (Dans le pire des cas, chaque caractère devra être encodé en utilisant deux octets, et vous avez besoin de place pour l'octet nul de terminaison.) Lorsque mysql_escape_string() retourne un résultat, le contenu de en sera une chaîne terminée par un caractère nul. La valeur de retour est la longueur de la chaîne encodée, n'incluant pas le caractère nul de terminaison.

Exemple

```
 char query[1000],*end;

 end = strmov(query,"INSERT INTO test_table values(");
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"C'est quoi ça",11);
*end++ = '\'';
*end++ = ',';
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"donnée binaire : \0\r\n",16);
*end++ = '\'';
*end++ = ')';

 if (mysql_real_query(&mysql,query,(unsigned int) (end - query)))
&#123;
   fprintf(stderr, "Impossible d'insérer la ligne, erreur : %s\n",
           mysql_error(&mysql));
&#125;
```

La fonction strmov() utilisée dans cet exemple est inclue dans la librairie mysqlclient et fonctionne comme strcpy() mais retourne un pointeur sur le nul de fin du premier paramètre. Valeur de retour

La longueur de la valeur passée dans to , n'incluant pas la caractère nul de fin de chaîne.

Erreurs Aucune.

### 8.4.3.44 mysql_real_query()

int mysql_real_query(MYSQL *mysql, const char *query, unsigned long length)

Description

Exécute la requête SQL pointée par query , qui doit être une chaîne de caractères de length octets de longueur. La requête ne doit contenir qu'une seule commande. Vous ne devez pas ajouter de point virgule ( ';' ) ou \g à la fin de la requête.

Vous **devez** utiliser mysql_real_query() au lieu de mysql_query() pour les requêtes qui continent des données binaires, car celles-ci peuvent contenir le caractère '\0' . De plus, mysql_real_query() est plus rapide que mysql_query() car elle n'invoque pas strlen() sur la chaîne contenant la requête.

Si vous voulez savoir si la requête est censée retourner un jeu de résultat ou non, vous pouvez utiliser `mysql_field_count()` pour vérifier cela. `mysql_field_count()` .

Valeur de retour

Zéro si la requête a été effectuée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

## 8.4.3.45 mysql_reload()

```
int mysql_reload(MYSQL *mysql)
```

### Description

Demande au serveur MySQL de recharger les tables de droits. L'utilisateur soit avoir les droits RELOAD .

Cette fonction est déconseillée. Il est préférable d'utiliser `mysql_query()` pour exécuter une requête `FLUSH PRIVILEGES` à la place.

Valeur de retour

Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

## 8.4.3.46 mysql_row_seek()

```
MYSQL_ROW_OFFSET mysql_row_seek(MYSQL_RES *result, MYSQL_ROW_OFFSET offset)
```

Description Déplace le curseur de ligne vers une ligne arbitraire dans un jeu de résultats de requête. Cela nécessite que le jeu de résultats contienne la totalité des lignes retournée par la requête, et donc, `mysql_row_seek()` ne peut être utilisée qu'en conjonction avec `mysql_store_result()` , et non avec `mysql_use_result()` .

La position doit être une valeur retournée par un appel à mysql_row_tell() ou à mysql_row_seek() . Cette valeur n'est pas un simple numéro de ligne; si vous voulez vous déplacer dans un jeu de résultats en utilisant le numéro d'une ligne, utilisez mysql_data_seek() .

### Valeur de retour

La position précédente du curseur de ligne. Cette valeur peut être passée à mysql_row_seek() .

### Erreurs Aucune.

### 8.4.3.47 mysql_row_tell()

```
MYSQL_ROW_OFFSET mysql_row_tell(MYSQL_RES *result)
```

#### Description

Retourne la position courante du pointeur de lignes pour le dernier appel à mysql_fetch_row() . Cette valeur peut être utilisée comme argument de mysql_row_seek() .

Vous ne devez utiliser mysql_row_tell() qu'après mysql_store_result() , et non après mysql_use_result() .

#### Valeur de retour

La position courante du pointeur de ligne.

#### Erreurs

Aucune.

### 8.4.3.48 mysql_select_db()

```
int mysql_select_db(MYSQL *mysql, const char *db)
```

#### Description

Rend la base de données spécifiée par db la base par défaut (courante) pour la connexion spécifiée par mysql . Pour les requêtes suivantes, cette base de données sera utilisée comme référence pour les tables dont la base de données n'a pas été spécifiée explicitement.

mysql_select_db() échoue si l'utilisateur ne peut être reconnu ayant droit d'accès à la base de données.

#### Valeur de retour

Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

#### Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
: Les commandes n'ont pas été exécutées dans le bon ordre.

*CR_SERVER_GONE_ERROR*
: Le serveur MySQL ne réponds pas.

*CR_SERVER_LOST*
: La connexion au serveur a été perdue au cours la requête.

*CR_UNKNOWN_ERROR*
: Une erreur inconnue s'est produite.

### 8.4.3.49 mysql_shutdown()

```
int mysql_shutdown(MYSQL *mysql)
```

#### Description

Demande au serveur de base de données de se terminer. L'utilisateur connecté doit avoir le droit SHUTDOWN .

#### Valeur de retour

Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

#### Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
: Les commandes n'ont pas été exécutées dans le bon ordre.

*CR_SERVER_GONE_ERROR*
: Le serveur MySQL ne réponds pas.

*CR_SERVER_LOST*
: La connexion au serveur a été perdue au cours la requête.

*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

### 8.4.3.50 mysql_stat()

```
char *mysql_stat(MYSQL *mysql)
```

## Description

Retourne une chaîne de caractères contenant des informations similaires à celle fournies par la commande mysqladmin status . Cela inclus le temps de fonctionnement en secondes et le nombre de threads en cours d'exécution, questions, rechargement, et tables ouvertes.

## Valeur de retour

Une chaîne de caractères décrivant l'état du serveur. NULL si une erreur est survenue.

## Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

### 8.4.3.51 mysql_store_result()

```
MYSQL_RES *mysql_store_result(MYSQL *mysql)
```

Description

Vous devez appeler `mysql_store_result()` ou `mysql_use_result()` pour chaque requête qui récupère des données avec succès ( `SELECT` , `SHOW` , `DESCRIBE` , `EXPLAIN` ).

Vous n'avez pas à appeler mysql_store_result() ou mysql_use_result() pour d'autres requêtes, mais cela ne posera pas de problèmes ou ne ralentira pas vos scripts si vous appelez mysql_store_result() en tout cas. Vous pouvez savoir si la requête n'a pas renvoyé de résultat en vérifiant si mysql_store_result() retourne 0 (nous verrons cela plus tard).

Si vous voulez savoir si la requête devrait renvoyer un jeu de résultats ou non, vous pouvez utiliser `mysql_field_count()` pour vérifier. `mysql_field_count()` .

mysql_store_result() lit le résultat en entier et le stocke dans le client, alloue une structure MYSQL_RES , et place le résultat dans cette structure.

`mysql_store_result()` retourne un pointeur nul si la requête n'a pas retourné un jeu de résultats (si la requête était, par exemple, un `INSERT` ).

mysql_store_result() retourne aussi un pointeur nul si la lecture à partir du jeu de résultats échoue. Vous pouvez vérifier la présence d'erreurs en regardant si mysql_error() ne retourne pas de pointeur nul, si mysql_errno() retourne <> 0, ou si mysql_field_count() retourne <> 0.

Un jeu de résultat vide est retourné si aucune ligne n'est retournée. (Un jeu de résultats vide diffère d'un pointeur nul en tant que valeur de retour.)

Une fois que vous avez appelé mysql_store_result() et obtenu un résultat qui n'est pas un pointeur nul, vous devez appeler mysql_num_rows() pour trouver combien de lignes contient le jeu de résultats.

Vous pouvez appeler `mysql_fetch_row()` pour récupèrer des lignes à partir du jeu de résultats, ou `mysql_row_seek()` et `mysql_row_tell()` pour obtenir ou changer la ligne courante dans le jeu de résultats.

Vous devez appeler mysql_free_result() une fois que vous n'avez plus besoin du résultat.

Pourquoi est ce qu'après `mysql_query()` ait indiqué un résultat positif, `mysql_store_result()` retourne parfois `NULL` ? .

Valeur de retour

Une structure de résultat `MYSQL_RES` . `NULL` si une erreur survient.

Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_OUT_OF_MEMORY*
Plus de mémoire.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

### 8.4.3.52 mysql_thread_id()

```
unsigned long mysql_thread_id(MYSQL *mysql)
```

Description

Retourne l'identifiant du thread de la connexion courante. Cette valeur peut être utilisée comme argument de `mysql_kill()` pour terminer ce thread.

Si la connexion est perdue et que vous vous reconnectez via mysql_ping() , l'identifiant du thread changera. Cela signifie que cela ne sert à rien de récupérer l'identifiant du thread et de le sauvegarder pour l'utiliser plus tard. Vous devez le récupérer quand vous en avez besoin.

Valeur de retour

L'identifiant du thread de la connexion courante.

Erreurs

Aucune.

### 8.4.3.53 mysql_use_result()

```
MYSQL_RES *mysql_use_result(MYSQL *mysql)
```

Description

Vous devez appeler `mysql_store_result()` ou `mysql_use_result()` pour chaque requête qui récupère des données avec succès ( `SELECT` , `SHOW` , `DESCRIBE` , `EXPLAIN` ).

mysql_use_result() initialise un jeu de résultats mais ne l'enregistre pas dans le client comme le fait mysql_store_result() . A la place, chaque ligne doit être récupéré manuellement à l'aide de la commande mysql_fetch_row() . Cela lit le résultat directement à partir du serveur sans l'enregistrer dans une table temporaire ou un tampon local, ce qui est plus rapide et utilise moins de mémoire que mysql_store_result() . Le client n'allouera de la mémoire que pour la ligne courante et un tampon de communication qui peut aller jusqu'à max_allowed_packet octets.

D'une autre côté, vous ne devez pas utiliser `mysql_use_result()` si vous faites beaucoup de traitements pour chaque ligne côté client, ou que le résultat est envoyé à un écran où l'utilisateur peut entrer `^S` (arrêt défilement). Cela bloquera le serveur et empêchera les autres threads de mettre à jour n'importe quelle table à partir de laquelle les données sont lues.

Lors de l'utilisation de mysql_use_result() , vous devez exécuter mysql_fetch_row() jusqu'à ce que NULL soit retourné, sinon, les lignes non retournée seront inclues dans le jeu de résultat de votre prochaine requête. L'API C donnera l'erreur Commands out of sync; you can't run this command now si vous oubliez de le faire !

Vous ne devez pas utiliser mysql_data_seek() , mysql_row_seek() , mysql_row_tell() , mysql_num_rows() , ou mysql_affected_rows() avec un résultat retourné par mysql_use_result() , de même, vous ne devez pas exécuter d'autres requêtes tant que la commande mysql_use_result() n'est pas terminée. (Toutefois, après avoir récupéré toutes les lignes, mysql_num_rows() retournera correctement le nombre de lignes récupérées.)

Vous devez appeler mysql_free_result() lorsque vous n'avez plus besoin du jeu de résultats.

### Valeur de retour

Une structure de résultat MYSQL_RES . NULL si une erreur survient.

### Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Les commandes n'ont pas été exécutées dans le bon ordre.
*CR_OUT_OF_MEMORY*
Plus de mémoire.
*CR_SERVER_GONE_ERROR*
Le serveur MySQL ne réponds pas.
*CR_SERVER_LOST*
La connexion au serveur a été perdue au cours la requête.
*CR_UNKNOWN_ERROR*
Une erreur inconnue s'est produite.

## 8.4.3.54 mysql_commit()

```
my_bool mysql_commit(MYSQL *mysql)
```

### Description

Valide la transaction courante. Disponible depuis MySQL 4.1

### Valeurs retournées

Zéro si la fonction réussit. Non−nul si une erreur survient.

### Erreurs

Aucune

## 8.4.3.55 mysql_rollback()

```
my_bool mysql_rollback(MYSQL *mysql)
```

Description

### Annule la transaction courante. Disponible avec MySQL 4.1

Valeurs retournées

### Zéro si l'annulation a réussi, et non−nul si une erreur est survenue.

Erreurs

### Aucune.

## 8.4.3.56 mysql_autocommit()

```
my_bool mysql_autocommit(MYSQL *mysql, my_bool mode)
```

### Description

Active ou désactive le mode d'auto-validation (autocommit). Si le paramètre mode vaut 1, l'auto-validation est activée. Dans le cas où il vaut 0, l'auto-validation est désactivée. Disponible depuis MySQL 4.1

### Valeurs retournées

Zéro si la fonction réussit. Non nul si une erreur survient.

### Erreurs

Aucune.

## 8.4.3.57 mysql_more_results()

my_bool mysql_more_results(MYSQL *mysql)

Description

Retourne TRUE si il y a d'autre résultats disponibles pour la requête courante, et si l'application doit appeler `mysql_next_result()` pour lire ces résultats. Disponible en MySQL 4.1

Valeurs retournées

`TRUE` si d'autres résultats existent. `FALSE` si il n'y a plus d'autres résultats disponibles.

Erreurs

Aucune.

## 8.4.3.58 mysql_next_result()

int mysql_next_result(MYSQL *mysql)

### Description

Si il existe des résultats disponibles, mysql_next_result() va lire la prochaine ligne, et retourne son statut à l'application. Disponible depuis MySQL 4.1

### Valeurs retournées

Zéro si la fonction réussit. Non-nul si une erreur survient.

### Erreurs

Aucune.

# 8.4.4 Fonctions C de commandes préparées

Depuis MySQL 4.1, vous pouvez aussi utiliser les commandes préparées en utilisant le gestionnaire 'MYSQL_STMT', qui supporte l'exécution de commandes multiples, avec liaison en entrée et sortie.

L'exécution de requêtes préparées est un moyen efficace d'exécuter une requête plus d'une fois. La requête est préparée, ou analysée une première fois. Elle est ensuite exécutée une ou plusieurs fois plus tard, en utilisant un pointeur de commande qui est retourné durant la préparation.

Un autre avantage des commandes préparées est qu'elles utilisent un protocole binaire qui rend les transferts entre le client et le serveur bien plus efficace que l'ancien protocole MySQL.

Les commandes préparées sont plus rapides que les exécutions directes de requêtes qui sont exécutées plus d'une fois, notamment parce que la requête est analysée une seule fois. Dans le cas de l'exécution directe, la requête est analysée à chaque fois. Les commandes préparées permettent de réduire le trafic réseau durant l'exécution, car seuls les données paramètres sont échangées.

## 8.4.5 C API Prepared Statements DataTypes

Les requêtes préparées utilisent principalement les deux structures `MYSQL_STMT` et `MYSQL_BIND` :

*MYSQL_STMT*

Cette structure représente un pointeur de comande sur une commande préparée. Elle sert pour toutes les fonctions liées aux commandes.

La commande est initialisée lorsque la requête est préparée en utilisant `mysql_prepare()` .

Une connexion peut avoir de multiples pointeurs de commandes, et la limite ne dépend que des ressources systèmes.

*MYSQL_BIND*

Cette structure est utilisée pour lier les paramètres avec des buffers, avec mysql_bind_param() , pour utilisation avec mysql_execute() , ainsi que pour lier les résultats avec des buffers via mysql_bind_result() m, lors de la lecture de données avec mysql_fetch() . La structure MYSQL_BIND contient les membres suivants :

*enum enum_field_types buffer_type [input]*

Le type de buffer. La valeur type doit être l'une des suivantes :

MYSQL_TYPE_TINY
MYSQL_TYPE_SHORT
MYSQL_TYPE_LONG
MYSQL_TYPE_LONGLONG
MYSQL_TYPE_FLOAT
MYSQL_TYPE_DOUBLE
MYSQL_TYPE_TIME
MYSQL_TYPE_DATE
MYSQL_TYPE_DATETIME
MYSQL_TYPE_TIMESTAMP
MYSQL_TYPE_STRING
MYSQL_TYPE_VAR_STRING
MYSQL_TYPE_TINY_BLOB
MYSQL_TYPE_MEDIUM_BLOB
MYSQL_TYPE_LONG_BLOB
MYSQL_TYPE_BLOB

*void *buffer [input/output]*

Un pointeur sur un buffer pour les données du paramètre, si il sert à fournir des données, ou un pointeur de buffer de retour, pour les données qui seront lues dans le résultat.

*unsigned long buffer_length [input]*

La taille de *buffer en octets. Pour les caractères et les données binaires C, buffer_length spécifie la taille de *buffer à utiliser comme paramètre si il est utilisé avec mysql_bind_param() , ou la taille lue dans le résultat si il est utilisé avec mysql_bind_result() .

*long *length [input/output]*

Un pointeur sur le buffer pour le paramètre de taille. Lorsqu la structure est utilisée en entrée, cet argument pointe sur un buffer qui, lorsque mysql_execute() est appelé, contient la taille de la valeur stockée dans *buffer. Ceci est ignoré, sauf pour les données de type caractère, ou binaire C.

Si la taill est un pointeur null, alors le protocole suppose que les données caractères ou binaires sont terminées par un null.

Lorsque cette structure est utilisée dans des liaisons de lecture, alors mysql_fetch() retourne la taille des données retournées.

*bool *is_null [input/output]*

Indique si le paramètre data est NULL, ou si les données lues sont NULL.

*MYSQL_TIME*

Cette structure est utilisée pour écrire et lire des données de type DATE, TIME et TIMESTAMP, directement avec le serveur. La structure MYSQL_TIME contient les membres listées ci-dessous :

| Member | Type | Description |
|---|---|---|
| year | unsigned int | Année. |
| month | unsigned int | Mois de l'année. |
| day | unsigned int | Jour du mois. |
| hour | unsigned int | Heure du jour(TIME). |
| minute | unsigned int | Minute de l'heure. |
| second | unsigned int | Seconde de la minute. |
| neg | my_bool | Un booléen qui indique si le temps est négatif. |
| second_part | unsigned long | Fraction de la second (utilisé ultérieurement) |

## 8.4.6 Présentation des API C pour les commandes préparées.

Voici les fonctions disponibles pour les commandes préparées. Elles sont listées ici et détailles plus loin. C API Prepared Statement Function Descriptions .

| Function | Description |
|---|---|
| **mysql_prepare()** | Prépare une chaîne SQL pour l'exécution. |
| **mysql_param_count()** | Retourne le nombre de paramètres d'une commande préparée. |
| **mysql_prepare_result()** | Retourne les méta informations de la requête préparée sous la forme d'un résultat. |
| **mysql_bind_param()** | Lie un buffer avec un marqueur de paramètre, dans une commande préparée. |
| **mysql_execute()** | Exécute une commande préparée. |
| **mysql_stmt_affected_rows()** | Retourne le nombre de lignes modifiées, effacées ou insérée dans la dernière requête UPDATE, DELETE ou INSERT. |
| **mysql_bind_result()** | Lie les buffers de l'application avec les colonnes d'un résultat. |
| **mysql_stmt_store_result()** | Lit tout le résultat dans le client. |
| **mysql_fetch()** | Lit la prochaine ligne de données dans le résultat, et retourne toutes les données des colonnes liées. |
| **mysql_stmt_close()** | Libère une commande préparée de la mémoire. |
| **mysql_stmt_errno()** | Retourne le numéro d'erreur de la dernière requête. |
| **mysql_stmt_error()** | Retourne le méssage d'erreur de la dernière requête. |
| **mysql_send_long_data()** | Envoie de grandes données par parties. |

Appelez mysql_prepare() pour préparer et initialiser la commande, puis appelez mysql_bind_param() pour fournir les données des paramètres, enfin appelez mysql_execute() pour éxecuter la requête. Vous pouvez répépter mysql_execute() en modificant les valeurs des paramètres des buffers respectifs via mysql_bind_param() .

Dans le cas où la requête est une commande SELECT, ou toute autre commande qui retourne un résultat, alors mysql_prepare() va aussi retourner les méta données de résultat sous la forme d'une structure MYSQL_RES avec mysql_prepare_result() .

Vous pouvez fournir les buffers de résultat avec mysql_bind_result() , pour que mysql_fetch() lise automatiquement les résultats dans les buffers. Cela est fait ligne par ligne.

Vous pouvez aussi envoyer le texte ou les données binaires par parties, au serveur, avec mysql_send_long_data() , en spécifiant l'optoin is_long_data=1 ou length=MYSQL_LONG_DATA ou −2 dans la structure MYSQL_BIND fournie avec mysql_bind_param() .

Une fois que l'exécution de la commande est terimée, elle doit être supprimée avec mysql_stmt_close pour que toute les ressources allouées soient détruites.

Etapes d'exécution :

Pour préparer et exécuter une commande, l'application :

- appelle **mysql_prepare()** et passe une chaîne contenant la commande SQL. Si la préparation réussi, mysql_prepare() retourne un pointeur de commande valide.
- Si la requête a un résultat, alors **mysql_prepare_result** retourne les méta informations de résultat.
- spécifie les valeurs de tous les paramètres de **mysql_bind_param** . Tous les paramètres doivent être fournis, sinon, cela générera une erreur, ou engendrera des résultats inattendus.
- appelle **mysql_execute()** pour exécuter la requête.
- Répète les étapes 2 et 3 autant que nécessaire, en modifiant les valeurs des paramètres, et en ré−exécutant la commande.
- Lie les buffers de données aux lignes de résultat, si la commande génère un résultat, en utilisant **mysql_bind_result()** .
- Lit les données dnas les buffers, ligne par ligne, en appelant **mysql_fetch()** jusqu'à ce qu'il n'y ait plus de lignes.
- Lorsque **mysql_prepare()** est appelé, dans le protocole client/serveur MySQL :
  - Le serveur analyse la requête et envoie le statut OK au client en lui assignant un identifiant de commande. Il renvoie aussi le nombre total de paramètres, le nombre de colonnes et des métas informations si un résultat est attendu. La syntaxe et la sémantique de la requête sont vérifiés durant cet appel.
  - Le client utilise cet identifiant de commande pour les exécutions ultérieures, pour que le serveur identifie la commande dans le pool de commandes. Désormais, le client alloue un pointeur de commande avec cet identifiant, et le retourne à l'application.
- Lorsque **mysql_execute()** est appelé, avec le protocole client/serveur MySQL :
  - Le client utilise le pointeur de comande et envoie les paramètres au serveur.
  - Le serveur identifie la commane en utilisant l'identifiant, et remplace les marqueurs de paramètres par leur valeur, puis il exécute la requête. Si cela conduit à un résultat, il est retourné au client, ou bien un statut OK, indiquant le nombre total de ligne affecté est retourné.
- Lorsque **mysql_fetch()** est appelé, dans le protocole client/serveur MySQL :
  - Le client lit les données dans le paquet, ligne par ligne, et les place dans les buffers de données, avec les conversions nécessaires. Si le type de buffer de l'application est le même que le type de champs, alors les conversions sont immédiates.

Vous pouvez lire les codes et messages d'erreurs avec les fonctions mysql_stmt_errno() et mysql_stmt_error() , respectivement.

## 8.4.7 Descriptions des fonctions C pour les requêtes préparées

Vous devez utiliser ces fonctions lorsque vous voulez préparer et exécuter des commandes.

### 8.4.7.1 mysql_prepare()

```
MYSQL_STMT * mysql_prepare(MYSQL *mysql, const char *query, unsigned long length)
```

```
Description
```

Prépare la requête SQL représentée par la chaîne terminée par le caractère null 'query'. La requête doit être une commande seule. Vous ne devez pas ajouter le point−virgule final (`;`) ou le caractère \g.

```
L'application peut inclure un ou plusieurs marqueur de paramètre dans la commande SQL. Pour inclure un marqueur de paramètre,
l'application doit intégrer un point d'interrogation ( ? ) dans la requête SQL, à la position appropriée.
```

Les marqueur ne sont valides que dans certaines parties de la requête SQL. Par exemple, ils ne sont pas autorisés dans la liste des colonnes d'une sélection, ni comme opérande d'un opérateur binaire tel que le signe égal (=), cas cela rendrait impossible la détermination du type de marqueur. En général, les marqueurs ne sont valides que dans les manipulations de données (Data Manipulation Languange, DML), et non pas dans les requêtes de définition des données (Data Defination Language, DDL).

```
Les marqueurs de paramtères sont ensuite remplis par les variables d'application, en utilisant mysql_bind_param() .
```

#### Valeurs retournées

```
MYSQL_STMT si la préparation a été réussie. NULL si une erreur est survenue.
```

#### Erreurs

```
        CR_COMMANDS_OUT_OF_SYNC
                Commands were executed in an improper order : Les commandes ont été exécutées dans un ordre incorrect.
        CR_OUT_OF_MEMORY
                Out of memory : Plus de mémoire
        CR_SERVER_GONE_ERROR
                The MySQL server has gone away : le server MySQL n'est plus accessible.
        CR_SERVER_LOST
                The connection to the server was lost during the query : La connexion au serveur a été perdue durant la requête.
        CR_UNKNOWN_ERROR
                An unkown error occured : Erreur inconnue
```

```
Si la préparation a échoué, i.e. si mysql_prepare() a retourné la valeur NULL, les erreurs sont disponibles grâce à la fonction
mysql_error() . Exemple
```

Pour une illustration de l'utilisation de `mysql_prepare()` , voyez l'exemple de la fonction `mysql_execute()` .

### 8.4.7.2 mysql_param_count()

```
unsigned int mysql_param_count(MYSQL_STMT *stmt)
```

#### Description

```
Retourne le nombre de marqueurs de paramètres présents dans la requête préparée.
```

#### Valeurs retournées

```
Un entier non signé, représentant le nombre de paramètres dans la requête.
```

#### Erreurs

None

### Exemple

Pour une illustration de la fonction mysql_param_count() , voyez l'exemple de la fonction mysql_execute() .

## 8.4.7.3 mysql_prepare_result()

MYSQL_RES *mysql_prepare_result(MYSQL_STMT *stmt)

Description

Si la fonction mysql_prepare() a généré un résultat, alors mysql_prepare_result() retourne les méta données de résultats sous la forme d'un structure MYSQL_RES , qui peut être utilisée ultérieurement pour traiter des méta informations, telles qu le nombre de champs et les informations individuelles de champs. Ce résultat peut être passé en argument à l'une des fonctions de champs suivantes, pour traiter les données :

- mysql_num_fields()
- mysql_fetch_field()
- mysql_fetch_field_direct()
- mysql_fetch_fields()
- mysql_field_count()
- mysql_field_seek()
- mysql_field_tell() and
- mysql_free_result()

### Valeurs retournées

Une structure de type MYSQL_RES . NULL si aucune méta données n'existe pour la requête préparée.

### Erreurs

*CR_OUT_OF_MEMORY*
Out of memory : plus de mémoire
*CR_UNKNOWN_ERROR*
An unknown error occured : Une erreur inconnue est survenue.

### Exemple

Pour une illustration de la fonction mysql_prepare_result() , voyez l'exemple de la fonction mysql_fetch() .

## 8.4.7.4 mysql_bind_param()

int mysql_bind_param(MYSQL_STMT *stmt, MYSQL_BIND *bind)

Description

mysql_bind_param sert à lier des données avec les marqueurs de paramètres dans une requête SQL préparée avec mysql_prepare . Il utilise la strucutre MYSQL_BIND pour fournir les données : Les types de buffers supportés sont :

- MYSQL_TYPE_TINY
- MYSQL_TYPE_SHORT
- MYSQL_TYPE_LONG
- MYSQL_TYPE_LONGLONG
- MYSQL_TYPE_FLOAT
- MYSQL_TYPE_DOUBLE
- MYSQL_TYPE_TIME
- MYSQL_TYPE_DATE
- MYSQL_TYPE_DATETIME
- MYSQL_TYPE_TIMESTAMP
- MYSQL_TYPE_STRING
- MYSQL_TYPE_VAR_STRING
- MYSQL_TYPE_TINY_BLOB
- MYSQL_TYPE_MEDIUM_BLOB
- MYSQL_TYPE_LONG_BLOB

Valeurs retournées

Zéro si la liaison a été réussie. Non null si une erreur est survenue.

Erreurs

*CR_NO_PREPARE_STMT*
No prepared statement exists : Aucune requête préparée n'existe.
*CR_NO_PARAMETERS_EXISTS*
No parameters exists to bind : Aucun paramètre à lier n'existe.
*CR_INVALID_BUFFER_USE*
Indicates if the bind is to supply the long data in chunks and if the buffer type is non string or binary : Indique si la liason attend les données par morceaux, t si le type de buffer n'est pas chaîne ou binaire.
*CR_UNSUPPORTED_PARAM_TYPE*
The conversion is not supported, possibly the buffer_type is illegal or its not from the above list of supported types. La conversion n'est pas supportée, il est possible que le type de buffer soit illégal, ou qu'il ne fasse pas partie de la liste des options possibles.
*CR_OUT_OF_MEMOR*
Out of memory : Plus de mémoire.
*CR_UNKNOWN_ERROR*
An unknown error occured : une erreur inconnue est survenue.

Example

Pour une illustration de la fonction `mysql_bind_param()` , voyez l'exemple de la fonction `mysql_execute()` .

### 8.4.7.5 mysql_execute()

```
int mysql_execute(MYSQL_STMT *stmt .
```

## Description

mysql_execute() exécute la requête préparée, associée avec le pointeur 'stmt'. Les valeurs des marqueurs de paramètres seront envoyées au serveur durant cet appel, pour que le serveur remplace les marqueurs avec les nouvelles valeurs.

Si la commande est UPDATE, DELETE ou INSERT, le nombre total de lignes changées, modifiées ou insérées est accessible avec la fonction `mysql_stmt_affected_rows` . Si la requête retourne une un résultat, alors vous devez appeler la fonction `mysql_fetch()` pour lire les données avant d'appeler tout autre fonction de traitement du résultat. Pour plus d'informations sur ocmment lire les données binaires, voyez `mysql_fetch()` . Valeurs retournées

mysql_execute() retourne les valeurs suivantes :

| Return Value | Description |
|---|---|
| 0 | Réussi |
| 1 | Une erreur est survenue. Le code d'erreur et le message d'erreur sont disponible en appelant mysql_stmt_errno() et mysql_stmt_error() . |

Erreurs

*CR_NO_PREPARE_QUERY*
No query prepared prior to execution : Aucune requête n'a été préparée.
*CR_ALL_PARAMS_NOT_BOUND*
Not all parameters data is supplied : Toutes les valeurs des paramtères n'ont pas été fournies.
*CR_COMMANDS_OUT_OF_SYNC*
Commands were executed in an improper order : les commandes ont été exécutées dans un ordre invalide.
*CR_OUT_OF_MEMORY*
Out of memory : plus de mémoire.
*CR_SERVER_GONE_ERROR*
The MySQL server has gone away : le serveur MySQL s'est éteint.
*CR_SERVER_LOST*
The connection to the server was lost during the query : la connexion a été perdue durant la requête.
*CR_UNKNOWN_ERROR*
An unknown error occurred : une erreur inconnue est survenue.

Exemple

L'exemple suivant explique l'utilisation de `mysql_prepare` , `mysql_param_count` , `mysql_bind_param` , `mysql_execute` et `mysql_stmt_affected_rows()` .

```
 MYSQL_BIND   bind[3];
MYSQL_STMT   *stmt;
ulonglong    affected_rows;
long         length;
unsigned int param_count;
int          int_data;
short        small_data;
```

```
char         str_data[50], query[255];
my_bool      is_null;

   /* Passe en mode d'auto commit */
  mysql_autocommit(mysql, 1);

  if (mysql_query(mysql,"DROP TABLE IF EXISTS test_table"))
  &#123;
    fprintf(stderr, "\n drop table failed");
    fprintf(stderr, "\n %s", mysql_error(mysql));
    exit(0);
  &#125;
  if (mysql_query(mysql,"CREATE TABLE test_table(col1 int, col2 varchar(50), \
                                                 col3 smallint,\
                                                 col4 timestamp(14))"))
  &#123;
    fprintf(stderr, "\n create table failed");
    fprintf(stderr, "\n %s", mysql_error(mysql));
    exit(0);
  &#125;

  /* Prepare une requête d'insertion de trois paramètres */
  strmov(query, "INSERT INTO test_table(col1,col2,col3) values(?,?,?)");
  if(!(stmt = mysql_prepare(mysql, query, strlen(query))))
  &#123;
    fprintf(stderr, "\n prepare, insert failed");
    fprintf(stderr, "\n %s", mysql_error(mysql));
    exit(0);
  &#125;
  fprintf(stdout, "\n prepare, insert successful");

   /* Lit le nombre de paramètres de la requête */
  param_count= mysql_param_count(stmt);

   fprintf(stdout, "\n total parameters in insert: %d", param_count);
  if (param_count != 3) /* validate parameter count */
  &#123;
    fprintf(stderr, "\n invalid parameter count returned by MySQL");
    exit(0);
  &#125;

   /* Lie les données aux paramètres */

   /* INTEGER PART */
  bind[0].buffer_type= MYSQL_TYPE_LONG;
  bind[0].buffer= (char *)&int_data;
  bind[0].is_null= 0;
  bind[0].length= 0;

   /* STRING PART */
  bind[1].buffer_type= MYSQL_TYPE_VAR_STRING;
  bind[1].buffer= (char *)str_data;
  bind[1].buffer_length= sizeof(str_data);
  bind[1].is_null= 0;
  bind[1].length= 0;

   /* SMALLINT PART */
  bind[2].buffer_type= MYSQL_TYPE_SHORT;
  bind[2].buffer= (char *)&small_data;
  bind[2].is_null= &is_null;
  bind[2].length= 0;
  is_null= 0;
   /* Lie les buffers */
  if (mysql_bind_param(stmt, bind))
  &#123;
    fprintf(stderr, "\n param bind failed");
    fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
    exit(0);
  &#125;

   /* Spécifie les données */
  int_data= 10;             /* integer */
  strcpy(str_data,"MySQL"); /* string  */

   /* INSERT SMALLINT data as NULL */
  is_null= 1;

   /* Exécute la requête */
  if (mysql_execute(stmt))
  &#123;
    fprintf(stderr, "\n execute 1 failed");
    fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
    fprintf(stderr, "\n send a bug report to bugs&#64;lists.mysql.com, by asking why this is not working ?");
    exit(0);
  &#125;

  /* Lit le nombre de lignes affectées */
  affected_rows= mysql_stmt_affected_rows(stmt);

   fprintf(stdout, "\n total affected rows: %lld", affected_rows);
  if (affected_rows != 1) /* validate affected rows */
  &#123;
```

```
   fprintf(stderr, "\n invalid affected rows by MySQL");
   exit(0);
  &#125;

  /* Ré-exécute l'insertion, en modifiant les valeurs */
  int_data= 1000;
  strcpy(str_data,"The most popular open source database");
   small_data= 1000;         /* smallint */
  is_null= 0;               /* reset NULL */

  /* Exécute l'insertion : 2eme */
  if (mysql_execute(stmt))
  &#123;
   fprintf(stderr, "\n execute 2 failed");
   fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
   exit(0);
  &#125;

  /* Lit le nombre total de lignes affectées */
  affected_rows= mysql_stmt_affected_rows(stmt);

  fprintf(stdout, "\n total affected rows: %lld", affected_rows);
  if (affected_rows != 1) /* validate affected rows */
  &#123;
   fprintf(stderr, "\n invalid affected rows by MySQL");
   exit(0);
  &#125;

  /* Ferme la requête */
  if (mysql_stmt_close(stmt))
  &#123;
   fprintf(stderr, "\n failed while closing the statement");
   fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
   exit(0);
  &#125;

  /* Efface la table */
  if (mysql_query(mysql,"DROP TABLE test_table"))
  &#123;
   fprintf(stderr, "\n drop table failed");
   fprintf(stderr, "\n %s", mysql_error(mysql));
   exit(0);
  &#125;
  fprintf(stdout, "Success, MySQL prepared statements are working!!!");
```

## 8.4.7.6 mysql_stmt_affected_rows()

```
ulonglong mysql_stmt_affected_rows(MYSQL_STMT *stmt)
```

### Description

Retourne le nombre total de ligne modifiées par la dernière commande. Cette fonction peut être appelée immédiatement après la fonction mysql_execute() pour les commandes UPDATE, DELETE ou INSERT. Pour les commandes SELECT, mysql_stmt_affected() fonctionne comme mysql_num_rows().

### Valeurs retournées

Un entier supérieur à zéro indique le nombre de ligne affectées ou lues. Zéro indique qu'aucune ligne n'a été modifiées durant une commande UPDATE, ou qu'aucune ligne n'a vérifié la clause WHERE dans la requête, ou qu'aucune requête n'a été exécuté. -1 indique que la requête a retourné une erreur, ou que, pour une requête SELECT, mysql_stmt_affected_rows() a été appelé avant mysql_fetch().

### Erreurs

None.

### Exemple

Plus une illustration de mysql_stmt_affected_rows() voyez l'exemple de mysql_execute() .

## 8.4.7.7 mysql_bind_result()

```
my_bool mysql_bind_result(MYSQL_STMT *stmt, MYSQL_BIND *bind)
```

Description

mysql_bind_result() sert à associer, (dit aussi lier) des colonnes disponibles dans un résultat avec des buffers de données. Lorsque mysql_fetch() est appelé pour lire les données, le protocole client MySQL retourne les données des colonnes liées, dans les buffers spécifiés.

Notez que toutes les colonnes doivent être liées avant d'appeler la fonction mysql_fetch() , dans le cas de liaisons avec des buffers, sinon, mysql_fetch() ignorera simplement les données. De plus, les buffers doient être suffisamment grands pour contenir les données, car le protocole ne retourne pas les données par parties.

Une colonne peut être liée ou reliée à tout moment, même après avoir lue des donées dans le résultat. La nouvelle liaison prendra effet au prochain appel de mysql_fetch() . Par exemple, suposons que l'application lie les colonnes d'un résultat, et appelle mysql_fetch() . Le protocole retourne alors les données dans les buffers. Supposons maintenant que l'application lie les colonnes vers d'autres buffers. Le protocole ne placera pas les données dans les nouveaux buffers, mais il le fera lors du prochain appel de mysql_fetch() .

Pour lier une colonne, une application doit appeler la fonction mysql_bind_result() , et indiquer le type, l'adresse et l'adresse de la taille du buffer. Les types de buffers supportés sont :

- MYSQL_TYPE_TINY
- MYSQL_TYPE_SHORT
- MYSQL_TYPE_LONG
- MYSQL_TYPE_LONGLONG
- MYSQL_TYPE_FLOAT
- MYSQL_TYPE_DOUBLE
- MYSQL_TYPE_TIME
- MYSQL_TYPE_DATE
- MYSQL_TYPE_DATETIME
- MYSQL_TYPE_TIMESTAMP
- MYSQL_TYPE_STRING
- MYSQL_TYPE_VAR_STRING
- MYSQL_TYPE_BLOB
- MYSQL_TYPE_TINY_BLOB
- MYSQL_TYPE_MEDIUM_BLOB
- MYSQL_TYPE_LONG_BLOB

Valeurs retournéesZéro si la liaison a réussi, et non−nul si une erreur est survenue.

## Errors

*CR_NO_PREPARE_STMT*
No prepared statement exists : Aucune requête préparé n'existe.

*CR_UNSUPPORTED_PARAM_TYPE*
The conversion is not supported, possibly the buffer_type is illegal or its not from the list of supported types : la conversion n'est pas supportée, possiblement parce que le type de buffer est invalide ou qu'il ne fait pas partie de la liste des types autorisés.

*CR_OUT_OF_MEMOR*
Out of memory : plus de mémoire.

*CR_UNKNOWN_ERROR*
An unknown error occured : une erreur inconnue est survenue.

Exemple

Pour une illustration de mysql_bind_result() , voyez l'exemple de la fonction mysql_fetch()

### 8.4.7.8 mysql_stmt_store_result()

```
int mysql_stmt_store_result(MYSQL_STMT *stmt)
```

## Description

Vous devez appeler la fonction mysql_stmt_store_result() pour chaque requête qui doit lire de données ( SELECT , SHOW , DESCRIBE , EXPLAIN ) et uniquement si vous voulez lire la totalité du résultat dans un buffer du client, pour que les appels suivants à mysql_fetch() retourne des données bufferisées.

Vous n'avez pas à appeler mysql_stmt_store_result() pour les requêtes suivantes, mais cela ne causera pas de ralentissement notable. Vous pouvez détecter si une requête n'a pas de résultat en vérifiant si mysql_prepare_result() retourne 0. Pour plus d'informations, voyez mysql_prepare_result() .

Valeurs retournées

Zéro si les résultats sont mis en buffer correctement, et non−nul si une erreur survient.

Errors

*CR_COMMANDS_OUT_OF_SYNC*
Commands were executed in an improper order : les commandes ont été exécutées dans un ordre invalide.
*CR_OUT_OF_MEMORY*
Out of memory : plus de mémoire.
*CR_SERVER_GONE_ERROR*
The MySQL server has gone away : le serveur MySQL s'est éteind.
*CR_SERVER_LOST*
The connection to the server was lost during the query : la connexion au serveur MySQL s'est interrompue durant la commande.
*CR_UNKNOWN_ERROR*
An unknown error occurred : une erreur inconnue est survenue.

## 8.4.7.9 mysql_fetch()

int mysql_fetch(MYSQL_STMT *stmt)

Description

mysql_fetch() retourne la ligne suivante dans le résultat. La fonction peut être appelée uniquement si le résultat existe, c'est à dire après mysql_execute() qui crée le résultat, ou après mysql_stmt_store_result() , qui est appelé après mysql_execute() pour mettre en buffer tout le résultat.

Si les lignes sont liées à des buffers avec mysql_bind_result() , la fonction retourne les données dans ces buffers pour toutes les colonnes de la ligne en cours, et les ltaills sont retournées dans le pointeur de taille.

Notez que toutes les colonnes doivent être liées par l'application.

Si les données lues contiennent la valeur NULL, alors la valeur is_null de MYSQL_BIND contiendra TRUE, 1, ou sinon, les données et leur longueur seront reoturnées dans les variables *buffer et *length , basées sur le type de buffer, spécifié par l'application. Tous les nombres ont une taille fixe, listée en octet ci-dessous :

| Type | Length |
|---|---|
| MYSQL_TYPE_TINY | 1 |
| MYSQL_TYPE_SHORT | 2 |
| MYSQL_TYPE_LONG | 4 |
| MYSQL_TYPE_FLOAT | 4 |
| MYSQL_TYPE_LONGLONG | 8 |
| MYSQL_TYPE_DOUBLE | 8 |
| MYSQL_TYPE_TIME | sizeof(MYSQL_TIME) |
| MYSQL_TYPE_DATE | sizeof(MYSQL_TIME) |
| MYSQL_TYPE_DATETIME | sizeof(MYSQL_TIME) |
| MYSQL_TYPE_TIMESTAMP | sizeof(MYSQL_TIME) |
| MYSQL_TYPE_STRING | data length |
| MYSQL_TYPE_VAR_STRING | data_length |
| MYSQL_TYPE_BLOB | data_length |
| MYSQL_TYPE_TINY_BLOB | data_length |
| MYSQL_TYPE_MEDIUM_BLOB | data_length |
| MYSQL_TYPE_LONG_BLOB | data_length |

où *data_length ne vaut rien d'autre que 'la taille réelle des donées'.

Valeurs retournées

| Return Value | Description |
|---|---|
| 0 | Réussi. Les données ont été lues, et placées dans les buffers. |
| 1 | Une erreur est survenue. Le code d'erreur et le message d'erreur sont disponibles grâce aux fonctions mysql_stmt_errno() et mysql_stmt_error() . |
| 100, MYSQL_NO_DATA | Il ne reste plus de données ou de lignes. |

Erreurs

*CR_COMMANDS_OUT_OF_SYNC*
Commands were executed in an improper order : les commandes ont été éxécutées dans un ordre invalide.
*CR_OUT_OF_MEMORY*
Out of memory : plus de mémoire.
*CR_SERVER_GONE_ERROR*
The MySQL server has gone away : Le serveur MySQL s'est éteind.
*CR_SERVER_LOST*
The connection to the server was lost during the query : la connexion au serveur a été perdue durant la requête.
*CR_UNKNOWN_ERROR*
An unknown error occurred : Une erreur inconnue est survenue.
*CR_UNSUPPORTED_PARAM_TYPE*
If the buffer type is MYSQL_TYPE_DATE,DATETIME,TIME,or TIMESTAMP; and if the field type is not DATE, TIME, DATETIME

or TIMESTAMP. Le buffer est de type MYSQL_TYPE_DATE,DATETIME,TIME ou TIMESTAMP et le type de champs n'est pas DATE, TIME, DATETIME or TIMESTAMP.

*All other unsupported conversion errors are returned from* mysql_bind_result() . Toutes les autres erreurs de conversions non supportées sont disponibles avec mysql_bind_result() .

Exemple

L'exemple ci−dessous explique l'utilisation de mysql_prepare_result , mysql_bind_result() et mysql_fetch()

```
MYSQL_STMT *stmt;
MYSQL_BIND bind[2];
MYSQL_RES  *result;
int        int_data;
long       int_length, str_length;
char       str_data[50];
my_bool    is_null[2];

   query= "SELECT col1, col2 FROM test_table WHERE col1= 10)");
  if (!(stmt= mysql_prepare(&mysql, query, strlen(query)))
  &#123;
    fprintf(stderr, "\n prepare failed");
    fprintf(stderr, "\n %s", mysql_error(&stmt));
    exit(0);
  &#125;

   /* Lit les méta informations sur les champs */
  if (!(result= mysql_prepare_result(stmt)))
  &#123;
    fprintf(stderr, "\n prepare_result failed");
    fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
    exit(0);
  &#125;

   fprintf(stdout, "Total fields: %ld", mysql_num_fields(result));

   if (mysql_num_fields(result) != 2)
  &#123;
    fprintf(stderr, "\n prepare returned invalid field count");
    exit(0);
  &#125;

   /* Exécute la requête SELECT */
  if (mysql_execute(stmt))
  &#123;
    fprintf(stderr, "\n execute failed");
    fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
    exit(0);
  &#125;

   /* Lie les résultats dans les buffers */
  bind[0].buffer_type= MYSQL_TYPE_LONG;
  bind[0].buffer= (char *)&int_data;
  bind[0].is_null= &is_null[0];
  bind[0].length= &int_length;

   bind[1].buffer_type= MYSQL_TYPE_VAR_STRING;
  bind[1].buffer= (void *)str_data;
  bind[1].is_null= &is_null[1];
  bind[1].length= &str_length;

   if (mysql_bind_result(stmt, bind))
  &#123;
    fprintf(stderr, "\n bind_result failed");
    fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
    exit(0);
  &#125;

   /* Lit les données dans les buffers */
  if (mysql_fetch(stmt))
  &#123;
    fprintf(stderr, "\n fetch failed");
    fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
    exit(0);
  &#125;

  if (is_null[0])
    fprintf(stdout, "\n Col1 data is NULL");
  else
    fprintf(stdout, "\n Col1: %d, length: %ld", int_data, int_length);

   if (is_null[1])
     fprintf(stdout, "\n Col2 data is NULL");
   else
    fprintf(stdout, "\n Col2: %s, length: %ld", str_data, str_length);


  /* Nouvel appel de mysql_fetch() */
  if (mysql_fetch(stmt) |= MYSQL_NO_DATA)
```

```
  &#123;
    fprintf(stderr, "\n fetch return more than one row);
    exit(0);
  &#125;

   /* Libère la mémoire des données du résultat */
  mysql_free_result(result);

   /* Libère la mémoire de la commande */
  if (mysql_stmt_free(stmt))
  &#123;
    fprintf(stderr, "\n failed to free the statement handle);
    fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
    exit(0);
  &#125;
```

## 8.4.7.10 mysql_send_long_data()

```
int mysql_send_long_data(MYSQL_STMT *stmt, unsigned int parameter_number, const char *data, ulong length)
```

Description

Permet à une application d'envoyer des données par parties au serveur. Cette fonction doit être utilisée pour envoyer des caractères ou du contenu binaire par partie dans une colonne (qui sera de type TEXT ou BLOB), avec un type de caractère ou de données binaires.

Le paramètre data est un pointeur sur un buffer qui contient les données pour le paramètre, représenté par parameter_number . Le paramètre length indique la quantité de données qui doit être envoyée, en octets.

### Valeurs retournées

Zéro si les données ont pu être envoyées au serveur. Non-nul si une erreur est survenue.

### Erreurs

*CR_INVALID_PARAMETER_NO*
Invalid parameter number : nombre de paramètres invalide.
*CR_COMMANDS_OUT_OF_SYNC*
Commands were executed in an improper order : Les conmmandes ont été envoyées dans un ordre invalide.
*CR_SERVER_GONE_ERROR*
The MySQL server has gone away : le serveur MySQL s'est éteind.
*CR_OUT_OF_MEMOR*
Out of memory : plus de mémoire.
*CR_UNKNOWN_ERROR*
An unknown error occured : une erreur inconnue s'est produite.

ExempleL'exemple ci-dessous explique comment envoyer des données par partie dans une colonne de type TEXT :

```
MYSQL_BIND bind[1];
long       length;

   query= "INSERT INTO test_long_data(text_column) VALUES(?)");
  if (!mysql_prepare(&mysql, query, strlen(query))
  &#123;
    fprintf(stderr, "\n prepare failed");
    fprintf(stderr, "\n %s", mysql_error(&stmt));
    exit(0);
  &#125;
   memset(bind, 0, sizeof(bind));
   bind[0].buffer_type= MYSQL_TYPE_STRING;
   bind[0].length= &length;
   bind[0].is_null= 0;

   /* Liaison des buffers */
  if (mysql_bind_param(stmt, bind))
  &#123;
    fprintf(stderr, "\n param bind failed");
    fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
    exit(0);
  &#125;

    /* Envoi des données au serveur, par parties */
   if (!mysql_send_long_data(stmt,1,"MySQL",5))
  &#123;
    fprintf(stderr, "\n send_long_data failed");
    fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
    exit(0);
  &#125;
```

```
   /* Envoi des données suivantes */
  if (mysql_send_long_data(stmt,1," − The most popular open source database",40))
 &#123;
   fprintf(stderr, "\n send_long_data failed");
   fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
   exit(0);
 &#125;

   /* Exécution de la requête */
  if (mysql_execute(stmt))
 &#123;
   fprintf(stderr, "\n mysql_execute failed");
   fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
   exit(0);
 &#125;

   Cet exemple va insérer la phrase, "MySQL − The most popular open source database"
   dans la colonne de type 'text_column'.
```

## 8.4.7.11 mysql_stmt_close()

```
my_bool mysql_stmt_close(MYSQL_STMT *)
```

Description

Termine la commande préparée. `mysql_stmt_close()` va aussi désallouer le pointeur de commande alloué par `stmt` .

Si les résultats de la requête courante sont en attente, ou non lus, ils seront annulés. Le prochain appel pourra donc être exécuté.

### Valeur retournée

Zéro si la commande a pu être terminée. Une valeur non nulle si une erreur est survenue.

### Erreurs

*CR_SERVER_GONE_ERROR*
: The MySQL server has gone away

*CR_UNKNOWN_ERROR*
: An unkown error occured

Exemple

Pour une illustration de la fonction `mysql_stmt_close()` voyez un exemple avec `mysql_execute()` .

## 8.4.7.12 mysql_stmt_errno()

```
unsigned int mysql_stmt_errno(MYSQL_STMT *stmt)
```

### Description

Pour la commande spécifiée par stmt , mysql_stmt_errno() retourne le code d'erreur de la plus récente fonction d'API appelée, qu'elle ait réussi ou échoué. Une valeur de zéro indique qu'il n'y a pas eu d'erreur. Les numéros d'erreurs clients sont listés dans le fichier d'entêtes errmsg.h. Les messages d'erreurs serveurs sont listés dans le fichier d'entêtes mysqld_error.h. Dans la distribution source de MySQLm vous pouvez trouver une liste complète des messages d'erreurs et de leur numéro, dans le fichier Docs/mysqld_error.txt

### Valeurs retournées

Un valeur représentant un code d'erreur. Zéro représente l'absence d'erreur.

### Erreurs

Aucune

### 8.4.7.13 mysql_stmt_error()

```
char *mysql_stmt_error(MYSQL_STMT *stmt)
```

```
Description
```

Pour la commande spécifiée par `stmt` , `mysql_stmt_error()` retourne le message d'erreur de la fonction d'API la plus récemment appelée, qu'elle ait réussi ou pas. Une chaîne vide ("") est retournée si aucune erreur n'est survenue. Cela signifie que les instructions suivantes sont identiques :

```
 if (mysql_stmt_errno(stmt))
&#123;
  // une erreur est survenue
&#125;

 if (mysql_stmt_error(stmt))
&#123;
  // une erreur est survenue
&#125;
```

```
La langue utilisée pour les messages d'erreurs du client MySQL peuvent être modifiées à la compilation de la librairie cliente MySQL.
Actuellement, vous pouvez choisir les message d'erreur dans plusieurs langues.
```

#### Valeurs retournées

```
Une chaîne de caractères qui décrit l'erreur. Une chaîne vide signifie qu'il n'y a pas eu d'erreur.
```

#### Erreurs

```
Aucune
```

## 8.4.8 C API pour gérer les requêtes multiples

Depuis la version 4.1, MySQL supporte l'exécution de requêtes multiples dans une seule commande. Pour cela, vous devez activer l'option client `CLIENT_MULTI_QUERIES` lors de l'ouverture de la connexion.

Par défaut, `mysql_query()` ou `mysql_real_query()` ne retournent que le statut de la première requête, et les statuts suivants peut être obtenu avec `mysql_more_results()` et `mysql_next_result()` .

```
  /* Connexion au serveur, avec l'option CLIENT_MULTI_QUERIES */
 mysql_real_query(..., CLIENT_MULTI_QUERIES);

  /* Exécution de plusieurs requêtes */
 mysql_query(mysql,"DROP TABLE IF EXISTS test_table;\
                    CREATE TABLE test_table(id int);\
                    INSERT INTO test_table VALUES(10);\
                    UPDATE test_table SET id=20 WHERE id=10;\
                    SELECT * FROM test_table;\
                    DROP TABLE test_table";
 while (mysql_more_results(mysql))
 &#123;
   /* Traitement de tous les résultats */
   mysql_next_result(mysql);
   ...
   printf("total affected rows: %lld", mysql_affected_rows(mysql));
   ...
   if ((result= mysql_store_result(mysql))
   &#123;
     /* Retourne un résultat, le traite */
   &#125;
 &#125;
```

## 8.4.9 C API de gestions des DATE, TIME et TIMESTAMP

En utilisant le nouveau protocole binaire de MySQL 4.1 et plus récent, vous pouvez envoyer et recevoir les données de type DATE, TIME et TIMESTAMP avec la structure `MYSQL_TIME` .

La structure `MYSQL_TIME` est constituée des membres suivants :

- year
- month
- day
- hour
- minute
- second
- second_part

Afin d'envoyer les données, il faut utiliser une requête préparée avec les fonctions mysql_prepare() et mysql_execute() , et vous devez lier le paramètre en utilisant le type MYSQL_TYPE_DATE afin de traiter la valeur de date, MYSQL_TYPE_TIME pour les heures, et MYSQL_TYPE_DATETIME ou MYSQL_TYPE_TIMESTAMP pour les dates complètes (datetime) ou les timestamp en utilisant mysql_bind_param() lors de l'envoi, ou mysql_bind_results() lors de la reception.

Voici un exemple simple qui insert des données de type DATE, TIME and TIMESTAMP.

```
 MYSQL_TIME  ts;
MYSQL_BIND  bind[3];
MYSQL_STMT  *stmt;

  strmov(query, "INSERT INTO test_table(date_field, time_field,
                                        timestamp_field) VALUES(?,?,?");

   stmt= mysql_prepare(mysql, query, strlen(query)));

   /* configure les trois buffers pour les trois paramètres */
  bind[0].buffer_type= MYSQL_TYPE_DATE;
  bind[0].buffer= (char *)&ts;
  bind[0].is_null= 0;
  bind[0].length= 0;
  ..
  bind[1]= bind[2]= bind[0];
  ..

   mysql_bind_param(stmt, bind);

   /* Fourni les données à envoyer dans la structure ts */
  ts.year= 2002;
  ts.month= 02;
  ts.day= 03;

   ts.hour= 10;
  ts.minute= 45;
  ts.second= 20;

   mysql_execute(stmt);
  ..
```

## 8.4.10 Description des fonctions threadées de C

Vous devez utiliser les fonctions suivantes quand vous voulez créer un client threadé. How to Make a Threaded Client .

### 8.4.10.1 my_init()

```
void my_init(void)
```

```
Description
```

Cette fonction doit être appelée une fois dans le programme avant tout appel à une fonction MySQL. Cela initialise quelques variables globales dont MySQL a besoin. Si vous utilisez une librairie client sûr pour les threads, cela appellera aussi `mysql_thread_init()` pour ce thread.

Ceci est automatiquement appelé par mysql_init() , mysql_server_init() et mysql_connect() .

Valeur de retour

Aucune.

### 8.4.10.2 mysql_thread_init()

my_bool mysql_thread_init(void)

Description

Cette fonction doit être appelée à chaque création de thread pour initialiser les variables spécifiques aux threads.

Elle est appelée automatiquement par my_init() et mysql_connect() .

Valeur de retour

Aucune.

### 8.4.10.3 mysql_thread_end()

void mysql_thread_end(void)

Description

Cette fonction doit être appelée avant pthread_exit() pour libérer la mémoire allouée part mysql_thread_init() .

Notez que cette fonction **n'est pas invoquée automatiquement** par la librairie du client. Elle doit être invoquée explicitement pour éviter les pertes de mémoire.

Valeur de retour

Aucune.

### 8.4.10.4 mysql_thread_safe()

unsigned int mysql_thread_safe(void)

Description

Cette fonction indique si le client est compilé avec le support des threads (thread−safe).

Valeur de retour

1 indique que le client est thread−safe, 0 sinon.

## 8.4.11 Description des fonctions C du serveur embarqué

Vous devez utiliser les fonctions suivantes si vous voulez permettre à votre application d'être liée avec la librairie du serveur embarqué MySQL. libmysqld, the Embedded MySQL Server Library .

Si le programme est lié avec −lmysqlclient au lieu de −lmysqld , ces fonctions ne font rien. Cela permet de choisir d'utiliser un serveur embarqué MySQL, ou un serveur tournant à part sans avoir à changer votre code.

### 8.4.11.1 mysql_server_init()

```
int mysql_server_init(int argc, char **argv, char **groups)
```

Description

Cette fonction **doit** être appelée une fois dans le programme avant d'appeler toute autre fonction MySQL. Elle démarre le serveur et initialise tout sous−système ( `mysys` , `InnoDB` , etc.) utilisé par le serveur. Si cette fonction n'est pas appelée, le programme plantera. Si vous utilisez le paquet DBUG fournit avec MySQL, vous devez exécuter cette fonction après avoir appelé `MY_INIT()` .

Les arguments argc et argv sont analogues aux arguments de main() . Le premier élément argv est ignoré (il contient le plus souvent le nom du programme). Par convenance, argc peut être 0 (zéro) si il n'y a aucun argument passé en ligne de commande pour le serveur.

La liste de mots terminée par `NULL` dans `groups` détermine les groupes dans les fichiers d'options qui seront actifs. Fichier d'options `my.cnf` . Par convenance, `groups` peut être `NULL` , dans ce cas, les groupes `[server]` et `[emedded]` sont activés.

Exemple

```
 #include <mysql.h>
#include <stdlib.h>

 static char *server_args[] = &#123;
  "ce_programme",       /* cette chaîne n'est pas utilisée */
  "−−datadir=.",
  "−−key_buffer_size=32M"
&#125;;
static char *server_groups[] = &#123;
  "embedded",
  "server",
  "this_program_SERVER",
  (char *)NULL
&#125;;

 int main(void) &#123;
  mysql_server_init(sizeof(server_args) / sizeof(char *),
                    server_args, server_groups);

   /* Utilisez les fonction de L'API MySQL ici */

   mysql_server_end();

   return EXIT_SUCCESS;
&#125;
```

Valeur de retour0 en cas de succès, 1 si une erreur survient.

### 8.4.11.2 mysql_server_end()

```
void mysql_server_end(void)
```

Description

Cette fonction **doit** être appelée une fois dans le programme après toutes les autres fonctions MySQL. Elle coupe le serveur incorporé.

Valeur de retour Aucune.

## 8.4.12 Questions courantes sur la librairie C

### 8.4.12.1 Pourquoi est ce qu'après mysql_query() ait indiqué un résultat positif, mysql_store_result() retourne parfois NULL ?

Il est possible que `mysql_store_result()` retourne `NULL` après un appel à `mysql_query()` . Quand cela arrive, cela signifie que l'une des conditions suivantes a été remplie :

- Il y a eu un problème avec malloc() (par exemple, si la taille du résultat était trop importante).
- Les données n'ont pu être lues (erreur survenue à la connexion).
- La requête n'a retourné aucun résultat (par exemple, il s'agissait d'un INSERT , UPDATE , ou d'un DELETE ).

Vous pouvez toujours vérifier si la requête devait bien fournir un résultat non vide en invoquant mysql_field_count() . Si mysql_field_count() retourne zéro, le résultat est vide et la dernière requête n'en retournait pas (par exemple, un INSERT ou un DELETE ). Si mysql_field_count() retourne un résultat non nul, la requête aurait du produire un résultat non nul. Voyez la documentation de la fonction mysql_field_count() pour plus d'exemples.Vous pouvez tester les erreurs en faisant appel à mysql_error() ou mysql_errno() .

### 8.4.12.2 Quels résultats puis−je obtenir d'une requête?

En plus des enregistrements retournés par une requête, vous pouvez obtenir les informations suivantes :

- mysql_affected_rows() retourne le nombre d'enregistrements affectés par la dernière requête INSERT , UPDATE , ou DELETE . Une exception est que si DELETE est utilisé sans clause WHERE , la table est re−créée vide, ce qui est plus rapide! Dans ce cas, mysql_affected_rows() retournera zéro comme nombre d'enregistrements affectés.
- mysql_num_rows() retourne le nombre d'enregistrements dans le résultat. Avec mysql_store_result() , mysql_num_rows() peut être utilisée dès que mysql_store_result() retourne un résultat. Avec mysql_use_result() , mysql_num_rows() ne doit être appelé qu'après avoir récupéré tous les enregistrements avec mysql_fetch_row() .
- mysql_insert_id() retourne l'ID généré par la dernière requête qui a inséré une ligne dans une table avec un index AUTO_INCREMENT . mysql_insert_id() .
- Quelques requêtes ( LOAD DATA INFILE ... , INSERT INTO ... SELECT ... , UPDATE ) retournent des informations additionnelles. Le résultat est renvoyé par mysql_info() . Regardez la documentation de mysql_info() pour plus d'informations sur le format de la chaîne retournée. mysql_info() retourne un pointeur NULL s'il n'y a pas d'informations additionelles.

### 8.4.12.3 Comment obtenir l'identifiant unique du dernier enregistrement inséré ?

Si vous insérez une ligne dans une table qui contient une colonne ayant l'attribut `AUTO_INCREMENT` , vous pouvez obtenir le dernier identifiant généré en appelant la fonction `mysql_insert_id()` .

Vous pouvez aussi récupérer cet identifiant en utilisant la fonction LAST_INSERT_ID() dans une requête que vous passez à mysql_query() .

Vous pouvez vérifier qu'un index `AUTO_INCREMENT` est utilisé en exécutant le code suivant. Cela vérifiera aussi si la requête était un `INSERT` avec un index `AUTO_INCREMENT` :

```
 if (mysql_error(&mysql)[0] == 0 &&
    mysql_num_fields(result) == 0 &&
    mysql_insert_id(&mysql) != 0)
&#123;
    used_id = mysql_insert_id(&mysql);
&#125;
```

L'identifiant généré par la dernière insertion est maintenu sur le serveur en se basant sur la connexion. Il ne sera pas changé par un autre client. Il ne changera pas non plus si vous mettez à jour une autre colonne AUTO_INCREMENT avec une valeur normale (ni NULL ni 0 ).

Si vous voulez utiliser l'identifiant généré pour une table et l'insérer dans une autre, vous pouvez utiliser les requêtes suivantes :

```
 INSERT INTO foo (auto,text)
    VALUES(NULL,'text');              # génère un identifiant en insérant NULL
INSERT INTO foo2 (id,text)
    VALUES(LAST_INSERT_ID(),'text');  # on l'utilise dans la seconde page
```

### 8.4.12.4 Problèmes lors de la liaison avec l'API C

Lors de la liaison avec l'API C, l'erreur suivante peut survenir sur quelques systèmes :

```
 gcc −g −o client test.o −L/usr/local/lib/mysql −lmysqlclient −lsocket −lnsl

 Undefined        first referenced
 symbol          in file
floor            /usr/local/lib/mysql/libmysqlclient.a(password.o)
ld: fatal: Symbol referencing errors. No output written to client
```

Si cela se produit sur votre système, vous devez inclure la librairie mathématique en ajoutant −lm à la fin de la ligne de compilation/liaison.

## 8.4.13 Compiler les clients

Si vous compilez des clients MySQL que vous avez écrits vous−mêmes, ils doivent être liés en utilisant l'option `−lmysqlclient −lz` de la commande de liaison. Vous aurez peut−être besoin de spécifier l'option `−L` pour dire au programme ou trouver les librairies. Par exemple, si la librairie est installée dans `/usr/local/mysql/lib` , utilisez `−L/usr/local/mysql/lib −lmysqlclient −lz` dans votre commande.

Pour les clients qui utilisent les fichiers d'entêtes de MySQL, vous aurez besoin de spécifier une option −I lors de leur compilation (par exemple, −I/usr/local/mysql/include ), pour que le programme puisse les trouver.

Pour rendre ce qui précède plus simple sur Unix, nous avons fourni le script `mysql_config` . `mysql_config` , lire les options de compilation d'un client .

Vous pouvez l'utiliser pour compiler un client MySQL comme ceci :

```
 CFG=/usr/local/mysql/bin/mysql_config
sh −c "gcc −o progname `$CFG −−cflags` progname.c `$CFG −−libs`"
```

sh −c est nécessaire pour s'assurer que le shell ne traitera pas le résultat de mysql_config comme un seul mot.

## 8.4.14 Comment faire un client MySQL threadé

La librairie cliente est presque compatible avec les threads. Le problème le plus important est les routines de net.c qui lisent les sockets, et qui ne sont pas compatibles avec les interruptions. Cela a été fait en imaginant que vous souhaitiez vos propres alarmes, qui pourraient interrompre une lecture trop longue. Si vous installez des gestionnaires d'interruption pour l'alarme SIGPIPE , la gestion des sockets devraient être compatible avec les threads.

Dans les anciennes versions binaires que nous distribuions sur notre site web, ( http://www.mysql.com/ ), les librairies clientes étaient normalement compilées avec l'option de compatibilité avec les threads (les exécutables Windows sont par défaut compatible avec les threads). Les nouvelles distributions binaires doivent disposer des deux librairies, compatibles ou non avec les threads.

Pour obtenir une client threadé où vous pouvez interrompre le client avec d'autres threads, mettre des délais d'expiration lors des discussions avec le serveur MySQL, vous devriez utiliser les librairies −lmysys , −lmystrings et −ldbug , ainsi que net_serv.o que le serveur utilise.

Si vous n'avez pas besoin des interruption ou des expirations, vous pouvez compiler simplement une librairie compatible avec les threads, `(mysqlclient_r)` et l'utiliser. API C pour MySQL . Dans ce cas, vous n'avez pas à vous préoccuper du fichier `net_serv.o` ou des autres librairies MySQL.

Lorsque vous utiliser un client threadé et que vous souhaitez utiliser des délais d'expiration et des interruptions, vous pouvez faire grand usage des routines du fichier thr_alarm.c . Si vous utiliser des routines issues de la librairie mysys , la seule chose à penser est de commencer par utiliser my_init() ! C API Threaded Function Descriptions .

Toutes les fonctions, hormis `mysql_real_connect()` sont compatibles avec les threads par défaut. Les notes suivantes décrivent comment compiler une librairie cliente compatible avec les threads. Les notes ci−dessous, écrites pour `mysql_real_connect()` s'appliquent aussi à `mysql_connect()` , mais comme `mysql_connect()` est obsolète, vous devriez utiliser `mysql_real_connect()` ).

Pour rendre mysql_real_connect() compatible avec les threads, vous devez recompiler la librairie cliente avec cette commande :

```
 shell> ./configure −−enable−thread−safe−client
```

Cela va créer une librairie cliente compatible avec les threads libmysqlclient_r . Supposons que votre système d'exploitation dispose d'une fonction gethostbyname_r() compatible avec les threads. Cette librairie est compatible avec les threads pour chaque connexion. Vous pouvez partager une connexion entre deux threads, avec les limitations suivantes :

- Deux threads ne peuvent pas envoyer de requêtes simultanées au serveur MySQL, sur la même connexion. En particulier, vous devez vous assurer qu'entre mysql_query() et mysql_store_result() , aucun autre thread n'utilise la même connexion.
- De nombreux threads peuvent accéder à différents résultats qui sont lus avec mysql_store_result() .
- Si vous utilisez mysql_use_result , vous devez vous assurer qu'aucun autre thread n'utilise la même connexion jusqu'à ce qu'elle soit refermée. Cependant, il vaut bien mieux pour votre client threadé qu'ils utilisent mysql_store_result() .
- Si vous voulez utiliser de multiples threads sur la même connexion, vous devez avoir un verrou mutex autour de vos fonctions mysql_query() et mysql_store_result() . Une fois que mysql_store_result() est prêt, le verrou peut être libéré et d'autres threads vont pouvoir utiliser la connexion.
- Si vous programmez avec les threads POSIX, vous pouvez utiliser les fonctions pthread_mutex_lock() et pthread_mutex_unlock() pour poser et enlever le verrou mutex.

Vous devez savoir ce qui suit si vous avez un thread qui appel une fonction MySQL qui n,a pas crée de connexion à la base MySQL :

Lorsque vous appelez `mysql_init()` ou `mysql_connect()` , MySQL va créer une variable spécifique au thread qui est utilisée par la libaririe de débogage (entre autres).

Si vous appelez une fonction MYSQL, avant que le thread n'ai appelé mysql_init() ou mysql_connect() , le thread ne va pas avoir les variables spécifiques en place, et vous risquez d'obtenir un core dump tôt ou tard.

Pour faire fonctionner le tout proprement, vous devez suivre ces étapes :

- Appeler my_init() au début du programme, si il appelle une autre fonction MySQL, avant d'appeler mysql_real_connect() .
- Appeler mysql_thread_init() dans le gestionnaire de threads avant d'appeler une autre fonction MySQL.
- Dans le thread, appelez mysql_thread_end() avant d'appeler pthread_exit() . Cela va libérer la mémoire utiliser par les variables spécifiques MySQL.

Vous pouvez rencontrer des erreurs à cause des symboles non définis lors du link de votre client avec `libmysqlclient_r` . Dans la plupart des cas, c'est parce que vous n'avez pas inclus la librairie de thread dans la ligne de compilation.

## 8.4.15 libmysqld, la librairie du serveur embarqué MySQL

### 8.4.15.1 Vue d'ensemble de la librairie du serveur embarqué MySQL

La librairie embarquée MySQL rend possible l'accès à un serveur MySQL complet, depuis une application. Le principal avantage est l'amélioration des performances, et une gestion bien plus simple des applications.

Les API sont identiques pour la version embarquée et la version client/serveur. Pour changer les anciennes applications threadées, et les faire utiliser la librairie embarquée, vous devez simplement ajouter deux appels aux fonctions suivantes :

| Fonction | Quand l'utiliser |
|---|---|
| mysql_server_init() | Doit être appelée avant toute autre fonction MySQL, et de préférence très tôt dans la fonction main() . |
| mysql_server_end() | Doit être appelée avant que votre programme ne se termine. |
| mysql_thread_init() | Doit être appelée dans chaque thread que vous créez, qui aura accès à MySQL. |
| mysql_thread_end() | Doit être appelée avant d'appeler pthread_exit() |

Puis, vous devez compiler votre code avec libmysqld.a au lieu de libmysqlclient.a .

Les fonctions ci−dessus `mysql_server_xxx` sont aussi inclues dans la librairie `libmysqlclient.a` pour vous permettre de changer facilement entre les versions de la librairie embarquée et celle de la librairie client/serveur, en compilant simplement la bonne librairie. `mysql_server_init()` .

### 8.4.15.2 Compiler des programmes avec libmysqld

Pour avoir la librairie `libmysqld` vous devez configurer MySQL avec l'option `−−with−embedded−server` .

Quand vous liez votre programme avec libmysqld , vous devez aussi inclure les librairies pthread spécifiques au système et quelques librairies que le serveur MySQL utilise. Vous pouvez obtenir la liste complète des librairies en exécutant mysql_config −−libmysqld−libs .

Les options correctes pour compiler et lier un programme thréadé doivent être utilisées, même si vous n'appelez pas directement une fonction de threads dans votre code.

### 8.4.15.3 Restrictions lors de l'utilisation du serveur embarqué MySQL

Le serveur embarqué possède les limitations suivantes :

- Pas de support pour les tables ISAM. (Ceci est principalement fait pour rendre la librairie plus petite)
- Pas de fonctions définies par l'utilisateur (UDF).
- Pas de traçage de pile lors des vidages de mémoire (core dump).
- Pas de support pour les RAID internes. (Cela n'est normalement pas requis vu que la plupart des systèmes d'exploitation supportent aujourd'hui les grands fichiers).

- Vous pouvez le configurer en tant que serveur ou maître (pas de replication).
- vous ne pouvez vous connecter au serveur embarqué à partir d'un processus externe avec les sockets ou TCP/IP.

Quelques unes de ces limitations peuvent être changée en éditant le fichier mysql_embed.h et recompilant MySQL.

### 8.4.15.4 Utilisation de fichiers d'options avec le serveur embarqué

Voici la manière recommandée d'utiliser les fichiers d'options pour que le passage des applications client/serveur vers une application où MySQL est embarqué soit plus facile. Fichier d'options `my.cnf` .

- Placez les options communes dans la section [server] . Elles seront lues par les deux versions de MySQL.
- Placez les options spécifiques au client/serveur dans la section [mysqld] .
- Placez les options spécifiques à MySQL embarqué dans la section [embedded] .
- Placez les options spécifiques aux applications dans une section [NomApplication_SERVER] .

### 8.4.15.5 Choses à faire pour le serveur embarqué (TODO)

- Nous ne fournissons pour l'instant qu'une version statique de la librairie mysqld , nous fournirons aussi une librairie partagée pour cela plus tard.
- Nous allons proposer des options pour se débarrasser de quelques parties de MySQL pour rendre la librairie plus petite.
- Il y a encore beaucoup d'optimisations de vitesse à faire.
- Les erreurs sont écrites dans stderr. Nous ajouterons une options pour spécifier un fichier pour cela.
- Nous devons changer InnoDB pour qu'il n'y ait plus tant de sorties lors de l'utilisation du serveur embarqué.

### 8.4.15.6 Un exemple simple de serveur embarqué

Ce programme exemple et son makefile devraient fonctionner sans changements sur un système Linux ou FreeBSD. Pour les autres systèmes d'exploitation, quelques petits changements seront requis. Cet exemple est là pour vous donner assez de détails pour comprendre le problème, sans avoir en tête qu'il s'agit d'une partie nécessaire d'une application réelle.

Pour essayer ccet exemple, créez un dossier `test_libmysqld` au même niveau que le dossier des sources mysql−4.0. Sauvegardez le fichier source `test_libmysqld.c` et `GNUmakefile` dans le dossier, puis exécutez GNU `make` à l'intérieur du répertoire `test_libmysqld` .

test_libmysqld.c

```
 /*
 * Un simple exemple de client, utilisant la librairie du serveur embarqué MySQL
 */

 #include <mysql.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>

 MYSQL *db_connect(const char *dbname);
void db_disconnect(MYSQL *db);
void db_do_query(MYSQL *db, const char *query);

 const char *server_groups[] = &#123;
  "test_libmysqld_SERVER", "embedded", "server", NULL
&#125;;

 int
main(int argc, char **argv)
&#123;
  MYSQL *one, *two;

   /* mysql_server_init() doit être appelée avant toute autre fonction
   * mysql.
   *
   * Vous pouvez utiliser mysql_server_init(0, NULL, NULL), et cela initialisera
   * le serveur en utilisant groups = &#123;
   *   "server", "embedded", NULL
   *  &#125;.
   *
   * Dans votre fichier $HOME/.my.cnf, vous voudrez sûrement mettre :

 [test_libmysqld_SERVER]
language = /chemin/vers/la/source/de/mysql/sql/share/english

    * Vous pouvez, bien sûr, modifier argc et argv avant de les passer
   * à cette fonction. Ou vous pouvez en créer de nouveaux de la manière
```

```
   * que vous souhaitez. Mais tout les arguments dans argv (à part
   * argv[0], qui est le nom du programme) doivent être des options valides
   * pour le serveur MySQL.
   *
   * Si vous liez ce client avec la librairie mysqlclient normale,
   * cette fonction n'est qu'un bout de code qui ne fait rien.
   */
  mysql_server_init(argc, argv, (char **)server_groups);

   one = db_connect("test");
  two = db_connect(NULL);

   db_do_query(one, "SHOW TABLE STATUS");
  db_do_query(two, "SHOW DATABASES");

   mysql_close(two);
  mysql_close(one);

   /* Cela doit être appelé après toutes les autres fonctions mysql */
  mysql_server_end();

   exit(EXIT_SUCCESS);
&#125;

 static void
die(MYSQL *db, char *fmt, ...)
&#123;
  va_list ap;
  va_start(ap, fmt);
  vfprintf(stderr, fmt, ap);
  va_end(ap);
  (void)putc('\n', stderr);
  if (db)
    db_disconnect(db);
  exit(EXIT_FAILURE);
&#125;

 MYSQL *
db_connect(const char *dbname)
&#123;
  MYSQL *db = mysql_init(NULL);
  if (!db)
    die(db, "mysql_init a échoué : pas de mémoire");
  /*
   * Notez que le client et le serveur utilisent des noms de groupes séparés.
   * Ceci est critique, car le serveur n'acceptera pas les options du client
   * et vice versa.
   */
  mysql_options(db, MYSQL_READ_DEFAULT_GROUP, "test_libmysqld_CLIENT");
  if (!mysql_real_connect(db, NULL, NULL, NULL, dbname, 0, NULL, 0))
    die(db, "mysql_real_connect a échoué : %s", mysql_error(db));

   return db;
&#125;

 void
db_disconnect(MYSQL *db)
&#123;
  mysql_close(db);
&#125;

 void
db_do_query(MYSQL *db, const char *query)
&#123;
  if (mysql_query(db, query) != 0)
    goto err;

   if (mysql_field_count(db) > 0)
  &#123;
    MYSQL_RES   *res;
    MYSQL_ROW    row, end_row;
    int num_fields;

     if (!(res = mysql_store_result(db)))
      goto err;
    num_fields = mysql_num_fields(res);
    while ((row = mysql_fetch_row(res)))
    &#123;
      (void)fputs(">> ", stdout);
      for (end_row = row + num_fields; row < end_row; ++row)
        (void)printf("%s\t", row ? (char*)*row : "NULL");
      (void)fputc('\n', stdout);
    &#125;
    (void)fputc('\n', stdout);
  &#125;
  else
    (void)printf("Lignes affectées : %lld\n", mysql_affected_rows(db));

   return;

 err:
  die(db, "db_do_query a échoué : %s [%s]", mysql_error(db), query);
```

```
&#125;
```

GNUmakefile

```
 # On suppose que MySQL est installé dans /usr/local/mysql
inc      := /usr/local/mysql/include/mysql
lib      := /usr/local/mysql/lib

 # Si vous n'avez pas encore installé MySQL, essayez plutôt ceci
#inc      := $(HOME)/mysql-4.0/include
#lib      := $(HOME)/mysql-4.0/libmysqld

 CC       := gcc
CPPFLAGS := -I$(inc) -D_THREAD_SAFE -D_REENTRANT
CFLAGS   := -g -W -Wall
LDFLAGS  := -static
# Vous pouvez changer -lmysqld en -lmysqlclient pour utiliser
# la librairie client/serveur
LDLIBS    = -L$(lib) -lmysqld -lz -lm -lcrypt

 ifneq (,$(shell grep FreeBSD /COPYRIGHT 2>/dev/null))
# FreeBSD
LDFLAGS += -pthread
else
# Linux
LDLIBS += -lpthread
endif

 # Cela fonctionne pour les programes de tests sur un simple fichier
sources := $(wildcard *.c)
objects := $(patsubst %c,%o,$(sources))
targets := $(basename $(sources))

 all: $(targets)

 clean:
    rm -f $(targets) $(objects) *.core
```

### 8.4.15.7 Licence du serveur embarqué

Le code source de MySQL est couvert par la licence GNU GPL ( GNU General Public License ). L'un des effets de cela est que tout programme qui inclut, par liaison avec `libmysqld` , le code source de MySQL doit être réalisé en tant que logiciel libre (sous une licence compatible avec la GPL).

Nous encourageons tout le monde à promouvoir le logiciel libre en réalisant du code sous la licence GPL ou une autre licence compatible. Pour ceux qui ne peuvent le faire, une autre option est d'acheter la licence commerciale pour le code MySQL chez MySQL AB . Pour plus de détails, voyez Licences MySQL .

# 8.5 Interfaces MySQL pour C++

Deux interfaces sont disponibles dans le dossier des contribution MySQL ( http://www.mysql.com/Downloads/Contrib/ ).

## 8.5.1 Borland C++

Vous pouvez compiler la source Windows de MySQL avec Borland C++ 5.02. (La source Windows n'inclut que les fichiers issus de Microsoft VC++, pour Borland C++ vous devrez créer les fichiers de projet par vous-même.)

Un problème connu avec Borland C++ est qu'il utilise un alignement de structures différent de VC++. Cela signifie que vous aurez des problèmes si vous essayez d'utiliser la librairie par défaut libmysql.dll (qui a été compilée avec VC++) avec Borland C++. Vous pouvez faire ce qui suit pour éviter ce problème.

- Vous pouvez utiliser les librairies MySQL statiques pour Borland C++ que vous trouverez sur http://www.mysql.com/downloads/os-win32.html .
- Appelez mysql_init() avec NULL comme arguement, et non une structure MYSQL pré-allouée.

# 8.6 Connectivité Java/MySQL (JDBC)

Il y a 2 pilotes JDBC supportés pour MySQL :

- MySQL Connector/J de MySQL AB, implémenté 100% Java natif. Ce produit était connu sous le nom de pilote mm.mysql . Vous pouvez télécharger MySQL Connector/J depuis l'URL http://www.mysql.com/products/connector-j/ .
- Le pilote Resin JDBC, qui est disponible sur l'URL http://www.caucho.com/projects/jdbc-mysql/index.xtp .

Pour de la documentation, consultez celle de JDBC et des pilotes pour les fonctionnalités relatives à MySQL.

## 8.7 Interface Python pour MySQL

MySQLdb fournit le support MySQL pour Python , compatible avec l'API de base de données Python version 2.0. Elle est disponible à l'URL http://sourceforge.net/projects/mysql−python/ .

## 8.8 Interface Tcl pour MySQL

MySQLtcl est une API simple pour accéder au serveur MySQL depuis le langage de programmation Tcl. Il est disponible sur http://www.xdobry.de/mysqltcl/ .

## 8.9 Couche MySQL pour Eiffel

Eiffel MySQL est une interface avec le langage de programmation Eiffel, écrit par Michael Ravits. Il est disponible sur http://efsa.sourceforge.net/archive/ravits/mysql.htm .

# 9 Etendre MySQL

## 9.1 Rouages de MySQL

Ce chapitre décrit un grand nombre de notions que vous devez connaître lorsque vous travaillez sur le code de MySQL. Si vous envisagez de contribuer au développement de MySQL, que vous voulez accéder à du code ultra récent, ou que vous souhaitez simplement vous tenir au courant du développement, suivez les instructions de la section Installer à partir des sources de l'arbre de développement . Si vous êtes intéressés par les rouages internes de MySQL, il est recommandé de vous inscrire à notre liste de diffusion interne `internals` . Cette liste est relativement calme. Pour les détails d'inscription, voyez Les listes de diffusion MySQL . Tous les développeurs de MySQL AB sont sur la liste `internals` et nous aidons ceux qui travaillent sur le code de MySQL. Utilisez cette liste pour poser des questions sur le code, et partagez vos patchs que vous voulez voir intégré au projet MySQL.

### 9.1.1 Threads MySQL

Le serveur MySQL crée les threads suivants :

- Le thread de connexion TCP/IP, qui gère toutes les demandes de connexion, et crée un nouveau thread dédié pour gérer l'identification et le traitement des requêtes SQL, pour chaque connexion.
- Sur Windows NT, il y a un thread appelé gestionnaire de pipe, qui effectue le même travail que le thread de gestion des demandes de connexion, sur les demandes de connexion par pipe.
- Le threads de signal gère tous les signaux. Ce thread gère aussi normalement les alertes et les appels à process_alarm() pour forcer les délais d'expiration des connexions qui sont inactives.
- Si mysqld est compilé avec l'option –DUSE_ALARM_THREAD , un thread dédié aux alarmes est créé. Il est uniquement utilisé sur certains systèmes où il y a des problèmes avec la fonction sigwait() ou si vous voulez utiliser la fonction thr_alarm() dans des applications qui n'ont pas de thread dédié aux signaux.
- Lors de l'utilisation de l'option ––flush_time=# , un thread dédié est créé pour vider les tables de la mémoire à intervalle régulier.
- Chaque connexion a son propre thread dédié.
- Chaque table différente sur laquelle des INSERT DELAYED sont pratiquées reçoit un thread.
- Si vous utilisez l'option ––master–host , un thread de réplication sera démarré pour lire et appliquer les modifications du maître.

mysqladmin processlist affiche uniquement les threads de connexion, ceux de INSERT DELAYED et ceux de réplication.

### 9.1.2 Suite de test de MySQL

Jusqu'à récemment, notre suite de test principale étaient basée sur des données propriétaires de client, et pour cette raison, il n'a jamais été publié. Le seul système de test public actuel est notre script `crash–me` , qui est un script Perl DBI/DBD qui se trouve dans le dossier `sql–bench` , et divers tests qui font parti du dossier `tests` . Le manque d'une suite de test publique et standardisée rend difficile à nos utilisateurs et nos développeurs les possibilités de tests de régressions. Pour corriger ce problème, nous avons créé un nouveau système de tests qui est inclus dans les distributions source et binaires, à partir de la version 3.23.29.

Le jeu de test actuel ne couvre pas toutes les situations en MySQL, mais il permet d'identifier les bogues les plus courants lors de requêtes SQL, les problèmes de librairies et aussi les problèmes de réplication. Notre but final est de lui faire couvrir 100% du code. Nous apprécions les contributions à notre suite de test. Vous pouvez notamment fournir des test qui examine certaines fonctions critiques de votre système, et qui assureront que les futures versions de MySQL le prennent en compte.

#### 9.1.2.1 Exécuter la suite de tests MySQL

Le système de tests est constitué d'un interpréteur de langage de tests ( `mysqltest` ), un script shell qui exécute tous les scripts ( `mysql–test–run` ), les cas de tests réels, écrits dans un langage spécial de tests, et leur résultats attendus. Pour exécuter ces tests sur votre système après une compilation, tapez `make test` ou `mysql–test/mysql–test–run` depuis la racine de la distribution. Si vous avez installé une

distribution binaire, `cd` jusqu'au dossier d'installation (eg. `/usr/local/mysql` ), et exécutez `scripts/mysql-test-run` . Tous les tests doivent réussir. Si ce n'est pas le cas, vous devriez essayer de trouver pourquoi, et faire un rapport de bogues à MySQL. Rapporter des bogues dans la suite de tests .

Si vous avez une copie de `mysqld` qui fonctionne sur la machine où vous voulez faire des tests, vous n'avez pas à l'arrêter, tant qu'elle n'utilise pas les ports `9306` et `9307` . Si l'un de ces ports est pris, vous devriez éditer le script `mysql-test-run` et changer les valeurs des ports des maîtres et esclaves, en les remplaçant par des ports libres.

Vous pouvez exécuter des tests individuels avec mysql-test/mysql-test-run test_name .

Si l'un des tests échoue, vous devriez tester `mysql-test-run` avec l'option `--force` pour vérifier si aucun autre test n'échoue.

### 9.1.2.2 Améliorer la suite de tests MySQL

Vous pouvez utiliser le langage de `mysqltest` pour écrire vos propres cas de tests. Malheureusement, nous n'avons pas encore écrit une documentation complète pour ce logiciel, et nous prévoyons de le faire rapidement. Vous pouvez, toutefois, utiliser les cas de tests actuels comme exemples. Les points suivants devraient vous mettre le pied à l'étrier.

- Les tests sont situé dans mysql-test/t/*.test
- Un cas de tests est constitué de commandes terminées par un ; , et est similaire aux données d'entrées du client mysql . Une commande est par défaut une commande envoyée au serveur MySQL, à moins qu'il ne soit reconnu comme une commande interne (par exemple, sleep ).
- Toutes les requêtes qui produisent des résultats, comme SELECT , SHOW , EXPLAIN , etc., doivent être précédées par @/path/to/result/file . Le fichier contient alors les résultats attendus. Un moyen simple pour générer le résultat du fichier est d'exécuter mysqltest -r < t/test-case-name.test depuis le dossier de tests mysql-test , puis d'éditer le fichier résultant, si nécessaire, pour ajuster le contenu. Dans ce cas, soyez très prudent lors de l'ajout ou la suppression de caractères invisibles : assurez vous de ne changer que du texte, ou d'effacer des lignes. Vous pouvez utiliser od -c pour vous assurer que votre éditeur n'a pas perturbé le fichier durant l'édition. Bien sur, nous espérons que vous n'aurez jamais a éditer le résultat du fichier mysqltest -r ca vous n'avez à faire cela que lorsque vous découvrez un bug.
- Pour être cohérent avec votre configuration, vous devriez placer les fichiers de résultats dans le dossier mysql-test/r et les nommer test_name.result . Si le test produit plus qu'un résultat, vous devez utiliser test_name.a.result , test_name.b.result , etc.
- Si une commande retourne une erreur, vous devez, sur la ligne de la commande, le spécifier avec --error error-number . Le numéro d'erreur peut être une liste d'erreurs possibles, séparées par des virgules ',' .
- Si vous écrivez un test de réplication, vous devez, sur la première ligne du fichier de test, ajouter le code source include/master-slave.inc; . Pour passer entre le maître et l'esclave, utilisez connection master; et connection slave; . Si vous avez besoin d'utiliser une connexion alternative, vous pouvez utiliser connection master1; pour le maître, et connection slave1; pour l'esclave.
- Si vous avez besoin d'une boucle, vous pouvez utiliser ceci :

```
 let $1=1000;
while ($1)
&#123;
 # votre requête ici
 dec $1;
&#125;
```

- Pour faire une pause entre les requêtes, utilisez la commande sleep . Elle supporte les fraction de secondes, ce qui vous permet d'utiliser sleep 1.3; , pour attendre 1,3 secondes.
- Pour exécuter l'esclave avec des options additionnelles pour votre cas de tests, ajoutez les au format ligne de commande dans mysql-test/t/test_name-slave.opt . Pour le maître, ajoutez les dans mysql-test/t/test_name-master.opt .
- Si vous avez une question sur la suite de tests, ou que vous avez un test à proposer, envoyez le par email à internals@lists.mysql.com . Comme la liste n'accepte pas les attachements, vous devriez les placer sur le serveur FTP : ftp://support.mysql.com/pub/mysql/Incoming/

### 9.1.2.3 Rapporter des bugs dans la suite de tests MySQL

Si votre versionn de MySQL ne passe pas un teste, vous devez faire ceci :

- N'envoyez pas de rapport de bug avant d'avoir étudier au maximum les raisons possibles de l'échec! Lorsque vous le faîtes, utilisez le programme mysqlbug , pour que nous puissions obtenir un maximum d'informations sur votre système et la version de MySQL . Comment rapporter des bogues ou problèmes .
- Assurez vous d'inclure le résultat de mysql-test-run , ainsi que le contenu de tous les fichiers .reject du dossier mysql-test/r .
- Si un test de la suite échoue, vérifiez si le test échoue aussi en l'exécutant seul :

```
 cd mysql-test
mysql-test-run --local test-name
```

  Si cela échoue, alors vous devriez configurer MySQL avec --with-debug et exécuter mysql-test-run avec l'option --debug . Si cela échoue aussi, envoyez le fichier de trace var/tmp/master.trace à ftp://support.mysql.com/pub/mysql/secret pour que nous puissions l'examiner. N'oubliez pas d'inclure une description complète de votre système, ainsi que de la version de l'exécutable mysqld , et de sa compilation.
- Essayez d'exécuter mysql-test-run avec l'option --force pour voir si il n'y a pas d'autres tests qui échouent.
- Si vous avez compilé MySQL vous-même, vérifiez notre manuel, ainsi que les notes de compilations pour votre plate-forme, ou bien, utilisez à la place un des exécutables que nous avons compilé pour vous, disponibles à http://www.mysql.com/downloads/ . Toutes nos versions exécutables doivent passer la suite de tests.

- Si vous obtenez une erreur, comme Result length mismatch ou Result content mismatch , cela signifie que le résultat de la suite de tests n'a pas la taille attendue. Cela peut être un bug de MySQL, ou que votre version de MySQL fournit un résultat d'une autre taille, dans certaines circonstances.

Les résultats de tests qui ont échoués sont placés dans un fichier avec le même nom de base que le fichier de test, et avec l'extension `.reject` . Si votre test échoue, faites un `diff` sur les deux fichiers. Si vous ne pouvez pas voir où ils diffèrent, examinez ces deux fichiers avec `od -c` , et vérifiez leur tailles respectives.

- Si un test échoue totalement, vous devriez vérifier les fichiers de log dans le dossier mysql-test/var/log , pour avoir des indices sur ce qui a échoué.
- Si vous avez compilé MySQL avec le débogage, vous pouvez essayer de le déboger en exécutant mysql-test-run avec --gdb et/ou --debug . Créer des fichiers de traçage .Si vous n'avez pas compilé MySQL pour le débogage, vous devriez essayer de le faire. Spécifiez simplement l'option --with-debug dans le script de configure ! Installer MySQL à partir des sources .

# 9.2 Ajouter des fonctions à MySQL

Il y a deux méthodes pour ajouter des fonctions à MySQL :

- Vous pouvez ajouter la fonction grâce à l'interface de fonctions utilisateur (UDF). Les fonctions utilisateur sont ajoutées et supprimées dynamiquement avec les commandes CREATE FUNCTION et DROP FUNCTION . Syntaxe de CREATE FUNCTION/DROP FUNCTION .
- Vous pouvez ajouter une fonction sous la forme native (intégrée) d'une fonction MySQL. Les fonctions natives sont compilées dans mysqld et sont disponibles en permanence.

Chaque méthode a ses avantages et inconvénients :

- Si vous écrivez une fonction utilisateur, vous devez installer le fichier objet en plus du serveur lui-même. Si vous compilez votre fonction dans le serveur, vous n'avez pas ce problème.
- Vous pouvez ajouter des UDF à une distribution binaire de MySQL. Les fonctions natives requièrent une modification de la distribution source.
- Si vous mettez à jour votre distribution MySQL, vous pouvez continuer à utiliser vos fonctions précédemment installées. Pour les fonctions natives, vous devez refaire les modifications du code à chaque mise à jour.

Quelque soit la méthode que vous utilisez pour ajouter de nouvelles fonctions, ces fonctions pourront être utilisées comme des fonctions natives telles que ABS() ou SOUNDEX() .

## 9.2.1 Syntaxe de CREATE FUNCTION/DROP FUNCTION

```
CREATE [AGGREGATE] FUNCTION nom_fonction RETURNS &#123;STRING|REAL|INTEGER&#125;
      SONAME nom_librairie_partagée

DROP FUNCTION nom_fonction
```

Une fonction définie par un utilisateur (UDF) est une méthode pour intégrer une fonction qui fonctionne de la même façon qu'une fonction native de MySQL, comme ABS() et CONCAT() . AGGREGATE est une nouvelle option pour MySQL version 3.23. Une fonction AGGREGATE fonctionne exactement comme une fonction native comme SUM ou COUNT() .

`CREATE FUNCTION` enregistre le nom de la fonction, le type, et le nom des librairies partagées dans la table `mysql.func` . Vous devez avoir les droits `INSERT` et `DELETE` dans la base `mysql` pour créer et supprimer les fonctions.

Toutes les fonctions actives sont rechargées chaque fois que le serveur démarre, sauf si vous démarrez mysqld avec l'option --skip-grant-tables . Dans ce cas, l'utilisation des UDF n'est pas prise en compte et les UDFs ne sont pas disponibles. (Une fonction active est une fonction qui peut être chargée avec CREATE FUNCTION et supprimée par REMOVE FUNCTION ).

Concernant l'écriture des UDFs, Ajout de nouvelles fonctions à MySQL . Pour que le mécanisme des fonctions UDF fonctionne, les fonctions doivent être écrites en C ou C++, votre système doit supporter le chargement dynamique et vous devez avoir compilé `mysqld` dynamiquement (pas statiquement).

Notez que pour faire fonctionner AGGREGATE , vous devez avoir une table mysql.func qui contient la colonne type . Si ce n'est pas le cas, vous devez exécuter le script mysql_fix_privilege_table pour résoudre ce problème.

## 9.2.2 Ajouter une nouvelle fonction définie par l'utilisateur (UDF)

Pour que le mécanisme UDF fonctionne, les fonctions doivent êtres écrites en C ou C++ et votre système doit supporter le chargement dynamique. Les sources de MySQL incluent un fichier

sql/udf_example.cc qui définit 5 nouvelles fonctions. Consultez ce fichier pour voir comment marchent les conventions d'appels des UDF.

Pour que mysqld puisse utiliser les fonctions UDF, vous devez configurer MySQL avec l'option --with-mysqld-ldflags=-rdynamic . La raison est que sur diverses plate-formes, (Linux inclus) vous pouvez charger une librairie dynamique (avec dlopen() ) depuis un programme statique lié, que vous pouvez obtenir si vous utilisez l'option --with-mysql-ldflags=-all-static . Si vous voulez utiliser une UDF qui nécessite un accès aux symboles de mysqld (comme l'exemple methaphone dans sql/udf_example.cc qui utilise default_charset_info ), vous devez lier le programme avec -rdynamic (voir man dlopen ).

Pour chaque fonction que vous voulez utiliser dans SQL, vous devez définir les fonctions correspondantes en C (ou C++). Dans la discussion ci-dessous, le nom ``xxx'' est utilisé comme un exemple de nom de fonction. Pour faire la différence entre l'usage de SQL et de C/C++, XXX() (majuscules) indique l'appel d'une fonction SQL et xxx() (minuscules) indique l'appel d'une fonction C/C++.

Les fonctions C/C++ que vous écrivez pour l'implémentation de l'interface de XXX() sont :

xxx() *(requis)*
La fonction principale. C'est là où le résultat de la fonction est calculé. La correspondance entre le type de SQL et le type retourné par votre fonction C/C++ est affiché ci-dessous :

| SQL type | C/C++ type |
|---|---|
| STRING | char * |
| INTEGER | long long |
| REAL | double |

xxx_init() *(optionnel)*
La fonction d'initialisation de xxx() . Elle peut-être utilisée pour :
Vérifier le nombre d'arguments de XXX() .
Vérifier que les arguments correspondent aux types requis ou indiquer à MySQL de contraindre des arguments aux types que vous voulez quand la fonction principale est appelée.
Allouer la mémoire requise pour la fonction principale.
Spécifier la longueur maximale de la sortie.
Spécifier (pour les fonctions REAL ) le nombre maximal de décimales.
Spécifier si le résultat peut-être NULL .

xxx_deinit() *(optionnel)*
La dés-initialisation de la fonction xxx() . Elle doit dés-allouer toute la mémoire allouée par l'initialisation de la fonction.

Quand une requête SQL fait appel à XXX() , MySQL appelle l'initialisation de la fonction xxx_init() , pour laisser exécuter n'importe quelle action exigée, telle que la vérification d'arguments ou l'allocation de mémoire.

Si xxx_init() retourne une erreur, la requête SQL est annulée avec un message d'erreur et la fonction principale et la fonction de dés-initialisation ne sont pas appelées. Autrement, la fonction principale xxx() est appelée une fois pour chaque ligne. Après que toutes les lignes aient été traitées, la fonction de dés-initialisation xxx_deinit() est appelée pour procéder aux nettoyages requis.

Pour les fonctions d'agrégat (comme SUM() ), vous pouvez également ajouter les fonctions suivantes :

xxx_reset() *(requise)*
Remet la somme à zéro et insère l'argument en tant que valeur initiale pour un nouveau groupe.

xxx_add() *(requise)*
Ajoute l'argument à l'ancienne somme.

Quand vous utilisez les UDF d'agrégat, MySQL opère comme suit :All functions must be thread-safe (not just the main function, but the initialisation and deinitialisation functions as well). This means that you are not allowed to allocate any global or static variables that change! If you need memory, you should allocate it in xxx_init() and free it in xxx_deinit() .

- Appeler xxx_init() pour laisser la fonction d'agrégat allouer la mémoire dont elle aura besoin pour stocker les résultats.
- Trier la table en accord avec la clause GROUP BY .
- Pour la première ligne dans un nouveau groupe, appeler la fonction xxx_reset() .
- Pour chaque ligne appartenant à un même groupe, appeler la fonction xxx_add() .
- Quand le groupe change ou lorsque la dernière ligne a été traitée, appeler xxx() pour obtenir le résultat de l'agrégat.
- Répéter 3-5 tant que toutes les lignes n'ont pas été traitées.
- Appeler xxx_deinit() pour libérer la mémoire allouée.

Toutes les fonctions doivent être sûrs pour les threads (pas seulement la fonction principale, mais les fonctions d'initialisation et de dés-initialisation également). Cela signifie que vous n'êtes pas autorisés à allouer une variable globale ou statique qui change ! Si vous avez besoin de mémoire, vous devez l'allouer avec la fonction xxx_init() et la libérer avec xxx_deinit() .

### 9.2.2.1 Fonctions utiliateur : appeler des fonctions simples

La fonction principale doit être déclarée comme illustrée ici. Notez que le type de retour et les paramètres diffèrent, suivant que vous voulez déclarer une fonction SQL XXX() qui retournera une STRING , un INTEGER ou un REAL dans la commande CREATE FUNCTION :

Pour les fonctions de chînes STRING :

```
char *xxx(UDF_INIT *initid, UDF_ARGS *args,
          char *result, unsigned long *length,
          char *is_null, char *error);
```

Pour les fonctions d'entiers INTEGER :

```
long long xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Pour les fonctions de nombres à virgule flottante REAL :

```
double xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Les fonctions d'initialisation et de terminaison sont déclarées comme ceci :

```
my_bool xxx_init(UDF_INIT *initid, UDF_ARGS *args, char *message);

void xxx_deinit(UDF_INIT *initid);
```

Le paramètre initid est passé aux trois fonctions. Il pointe sur une structure UDF_INIT qui est utilisée pour communiquer des informations entre les fonctions. Les membres de la structure UDF_INIT sont ceux qui sont listés ci-dessous. La fonction d'initialisation doit préparer les membres qu'elle veut, et notamment leur donner une valeur initiale (pour utiliser les valeurs par défaut, laissez le membre intact) :

*my_bool maybe_null*
: xxx_init() doit remplacer maybe_null par 1 si xxx() peut retourner NULL . La valeur par défaut est 1 si l'un des arguments n'est déclaré comme maybe_null .

*unsigned int decimals*
: Le nombre de décimales. La valeur par défaut est le nombre maximum de décimales dans l'argument passé à la fonction. Par exemple, vis la fonction reçoit 1.34 , 1.345 , and 1.3 , ce nombre sera 3, car 1.345 a 3 décimales.

*unsigned int max_length*
: La taille maximale de la chaîne résultat. La valeur par défaut dépend du type de résultat de la fonction. Pour les fonctions de chaînes, la valeur par défaut est la taille du plus grand argument. Pour les fonctions entières, la valeur est de 21 chiffres. Pour les fonctions à nombre à virgule flottante, la valeur est de 13, plus le nombre de décimales indiquées par initid->decimals . Pour les fonctions numériques, la taille inclut le signe et le séparateur décimal.

Si vous voulez retourner un BLOB, vous devez donner à ce membre la valeur de 65K ou 16M; cet espace mémoire ne sera pas alloué, mais utilisé pour décider quel type de colonne utiliser, si il y a un besoin de stockage temporaire.

*char *ptr*
: Un pointeur que la fonction peut utiliser pour ses besoins propres. Par exemple, la fonction peut utiliser initid->ptr pour transférer de la mémoire allouée entre les trois fonctions. En xxx_init() , allouez de la mémoire, et assignez la à ce pointeur :

```
initid->ptr = allocated_memory;
```

En xxx() et xxx_deinit() , utilisez initid->ptr pour exploiter ou supprimer la mémoire.

### 9.2.2.2 Appeler des fonctions utilisateurs pour les groupements

Voici une description des différentes fonctions que vous devez définir pour réaliser des calculs sur des regroupements, avec une fonction utilisateur :

```
char *xxx_reset(UDF_INIT *initid, UDF_ARGS *args,
                char *is_null, char *error);
```

Cette fonction est appelée lorsque MySQL trouve la première ligne dans un nouveau groupe. Dans cette fonction, vous devez remettre à zéro des variables internes de sommaire, puis indique le nouvel argument comme premier membre du nouveau groupe.

Dans de nombreuses situations, cela se fait en interne en remettant à zéro toutes les variables, et en appelant `xxx_add()` .

```
char *xxx_add(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Cette fonction est appelée à chaque fois qu'une ligne qui appartient à un groupe est trouvée, hormis la première ligne. Durant cette fonction, vous devez ajouter les données dans votre variable interne de sommaire.

La fonction `xxx()` doit être déclarée de manière identique à celle d'un fonction utilisateur simple. Appel des UDF .

Cette fonction est appelée lorsque toutes les lignes d'un groupe ont été traitées. Vous ne devez normalement pas accéder à la variable args ici, mais retourner votre valeur, à partir des valeurs du sommaire interne.

Tous les traitements des arguments de `xxx_reset()` et `xxx_add()` doivent être fait d'une manière similaire à celle des fonctions UDF normales. Récupération des arguments .

La gestion de la valeur retournée par xxx() doit être identique à celle d'une fonction utilisateur classique. Valeurs retournées et gestion d'erreurs .

Le pointeur argument de `is_null` et `error` sont les mêmes pour tous les appels de `xxx_reset()` , `xxx_add()` et `xxx()` . Vous pouvez utiliser ces valeur pour vous rappeler si vous avez rencontré une erreur, ou si la fonction `xxx()` doit retourner `NULL` . Notez que vous ne devez pas stocker de chaîne dans `*error` ! C'est un conteneur d'un seul octet!

is_null est remis à zéro pour chaque (avant d'appeler xxx_reset() . error n'est jamais remis à zéro.

Si `isnull` ou `error` sont modifiés après `xxx()` , alors MySQL va retourner `NULL` comme résultat de la fonction de groupement.

### 9.2.2.3 Traitement des arguments

Le paramètre args pointe sur une structure UDF_ARGS qui dispose des membre suivants :

*unsigned int arg_count*

Le nombre d'arguments. Vérifiez cette avleur dans la fonction d'initialisation, si vous voulez que votre fonction soit appelée avec un nombre particulier d'arguments. Par exemple :

```
 if (args->arg_count != 2)
&#123;
    strcpy(message,"XXX() requires two arguments");
    return 1;
&#125;
```

*enum Item_result *arg_type*

Le type de chaque argument. Les valeurs possibles pour chaque type sont STRING_RESULT , INT_RESULT et REAL_RESULT .Pour s'assurer que les arguments sont d'un type donné, et retourner une erreur dans le cas contraire, vérifiez le tableau arg_type durant la fonction d'initialisation. Par exemple :

```
 if (args->arg_type[0] != STRING_RESULT ||
    args->arg_type[1] != INT_RESULT)
&#123;
    strcpy(message,"XXX() requires a string and an integer");
    return 1;
&#125;
```

Comme alternative à l'imposition d'un type particulier pour les arguments des fonctions, vous pouvez utiliser la fonction pour qu'elle modifie le type des arguments et donne aux valeurs de arg_type le type que vous souhaitez. Cela fait que MYSQL va forcer les arguments à un type donnée, pour chaque appel de la fonction xxx() . Par exemple, pour forcer le type des deux premiers arguments en chaîne et entier, vous pouvez utiliser la fonction d'initisalition xxx_init() :

```
 args->arg_type[0] = STRING_RESULT;
args->arg_type[1] = INT_RESULT;
```

*char **args*

args->args communique les informations à la fonction d'initialisation, ainsi que la nature des arguments avec laquelle elle a été appelée. Pour un argument constant i , args->args[i] pointe sur la valeur de l'argument. Voir plus bas pour les instructions d'accès à cette valeur. Pour les valeurs non constantes, args->args[i] vaut 0 . Un argument constant est une expression qui utilise des constantes, comme 3 ou 4*7-2 ou SIN(3.14) . Un argument non-constant est une expression qui fait référence aux valeurs qui peuvent changer de ligne en ligne, par exemple des noms de colonnes ou des fonctions qui sont appelées avec des arguments non-constants.Pour chaque invocation de la fonction principale, args->args contient les arguments réels qui sont passés à la ligne qui sera traitée.

Les fonctions peuvent faire référence à un argument `i` comme ceci :

Un argument de type STRING_RESULT est donné sous la forme d'un pointeur de chaîne, plus une longueur, pour permettre la gestion des données binaires ou des données de taille arbitraire. Le contenu des chaînes est disponible avec l'expression args->args[i] et la taille de la chaîne est args->lengths[i] . Ne supposez pas que les chaînes sont terminés par le caractère nul.
Pour un argument de type INT_RESULT , vous devez transtyper la valeur args->args[i] en valeur long long :

```
 long long int_val;
int_val = *((long long*) args->args[i]);
```

Pour un argument de type REAL_RESULT , vous devez transtyper la valeur args->args[i] en valeur double :

```
 double    real_val;
real_val = *((double*) args->args[i]);
```

*unsigned long *lengths*

Pour une fonction d'initialisation, le tableau lengths indique la taille maximale des chaînes pour chaque argument. Vous ne devez pas les modifier. Pour chaque appel de la fonction principale, lengths contient la taille réelle de toutes les chaînes arguments qui sont passé pour la ligne traitée. Pour les arguments de type INT_RESULT ou REAL_RESULT , lengths contient toujours la taille maximale de l'argument (comme pour la fonction d'initialisation).

### 9.2.2.4 Valeurs de retour et gestion d'erreurs.

La fonction d'initialisation doit retourner 0 si aucune erreur ne s'est produite et 1 sinon. Si une erreur s'est produite, xxx_init() doit stocker un message se terminant par un NULL dans le paramètre message . Le message sera retourné au client. La taille du tampon du message est de MYSQL_ERRMSG_SIZE caractères, mais vous devez essayer de garder une taille de message inférieure à 80 caractères, sinon, il remplit la largeur d'un écran de terminal standard.

La valeur de retour de la fonction principale xxx() est la valeur de la fonction, pour les fonctions long long et double . Une fonction de chaîne de caractères doit retourner un pointeur vers le résultat et stocker la taille de la chaîne de caractères dans l'argument length .

Affectez cette valeur au contenu et à la longueur de la valeur retournée. Par exemple :

```
 memcpy(result, "chaîne retournée", 16);
*length = 16;
```

Le tampon result qui est passé à la fonction calc à une taille de 255 bits. Si votre résultat dépasse ceci, ne vous inquietez pas de l'allocation de mémoire pour ce résultat.Si votre fonction de chaînes de caractères a besoin de retourner une chaîne de caractères plus grande que 255 bits, vous devez allouer de l'espace pour cela avec malloc() dans votre fonction xxx_init() . Vous pouvez stocker la mémoire allouée dans le slot ptr de la structure UDF_INIT pour être ré-utilisée par les appels futurs de xxx() . Appel des UDF .

Pour indiquer une valeur de retour NULL dans la fonction principale, mettez is_null à 1 :

```
 *is_null = 1;
```

Pour indiquer une erreur retournée dans la fonction principale, mettez le paramètre error à 1 :

```
 *error = 1;
```

Si xxx() met *error à 1 pour chaque ligne, la valeur de la fonction est NULL pour la ligne en question et pour chaque ligne suivante traitée par le processus dans lequel XXX() est invoqué. ( xxx() ne sera même pas appelé pour les lignes suivantes.) **Remarque :** dans les versions antérieures à 3.22.10, vous devez définir *error et *is_null :

```
 *error = 1;
*is_null = 1;
```

## 9.2.2.5 Compiler et installer des fonctions utilisateurs

Les fichiers qui implémentent des fonctions utilisateurs doivent être compilés et installés sur le même hôte que celui du serveur. Ce processus est décrit plus bas, avec le fichier udf_example.cc qui est inclut dans les sources MysQL. Ce fichier contient les fonctions suivantes :

- metaphon() retourne la version métaphone de la chaîne en argument. C'est une technique proche du soundex, mais elle est bien plus optimisée pour l'anglais.
- myfunc_double() retourne la moyenne des codes ASCII des caractères de la chaîne passée en argument.
- myfunc_int() retourne la somme de tailles des arguments.
- sequence([const int]) retourne une séquence, commençant à partir du nombr choisit ou 1, si aucun nombre n'a été fourni.
- lookup() retourne l'adresse IP numérique d'un hôte.
- reverse_lookup() retourne le nom d'hôte pour une adresse IP. Cette fonction peut être appelée avec une chaîne au format "xxx.xxx.xxx.xxx" ou quatre nombres.

Un fichier dynamiquement chargeable doit être compilé sous la forme d'un objet partagé, grâce à une commande comme celle-ci :

```
 shell> gcc -shared -o udf_example.so myfunc.cc
```

Vous pouvez facilement trouver les options correctes pour la compilation en exécutant cette commande dans le dossier sql de votre installation source :

```
 shell> make udf_example.o
```

Vous devez exécuter une commande de compilation similaire à celle que le make affiche, sauf que vous devrez supprimer l'option -c près de la fin de la ligne, et ajouter -o udf_example.so à la fin de la ligne. Sur certains systèmes, vous devrez aussi supprimer -c de la commande).Une fois que vous compilez un objet partagés contenant des fonctions utilisateurs, vous devez les installer, et prévenir le serveur MYSQL. Compilier un objet partagé avec udf_example.cc produit un fichier qui s'appelle udf_example.so (le nom exact peut varier suivant la plate-forme). Copiez ce fichier dans l'un des dossiers utilisé par ld , tel que /usr/lib . Sur de nombreux systèmes, vous pouvez faire pointer la variable d'environnement LD_LIBRARY ou LD_LIBRARY_PATH pour qu'elle pointe dans le dossier où vous avez vos fichiers de fonctions. Le manuel de dlopen vous indiquera quelle variable utiliser sur votre système. Vous devriez indiquer cette valeur dans les options de démarrage de mysql.server et safe_mysqld , et redémarrer mysqld .

Après que la librairie ait été installée, indiquez à mysqld ces nouvelles fonctions avec ces commandes :

```
 mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_double RETURNS REAL SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_int RETURNS INTEGER SONAME "udf_example.so";
mysql> CREATE FUNCTION lookup RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION reverse_lookup
    ->        RETURNS STRING SONAME "udf_example.so";
mysql> CREATE AGGREGATE FUNCTION avgcost
    ->        RETURNS REAL SONAME "udf_example.so";
```

Les fonctions peuvent être effacées plus tard avec DROP FUNCTION :

```
 mysql> DROP FUNCTION metaphon;
mysql> DROP FUNCTION myfunc_double;
mysql> DROP FUNCTION myfunc_int;
mysql> DROP FUNCTION lookup;
mysql> DROP FUNCTION reverse_lookup;
mysql> DROP FUNCTION avgcost;
```

Les commandes CREATE FUNCTION et DROP FUNCTION modifient la table système func dans la base mysql . Le nom de la fonction, son type et le nom de la librairie partagée sont alors sauvés dans la table. Vous devez avoir les droits de INSERT et DELETE dans la base mysql

pour ajouter et effacer des fonctions.Vous ne devez pas utiliser la commande CREATE FUNCTION pour ajouter une fonction qui a déjà été créée. Si vous devez reinstaller une fonction, vous devez la supprimer avec la commande DROP FUNCTION puis la reinstaller avec CREATE FUNCTION . Vous devrez faire cela, par exemple, si vous recompilez une nouvelle version de votre fonction, pour que mysqld utilise cette nouvelle version. Sinon, le serveur va continuer à utiliser l'ancienne version.

Les fonctions actives sont rechargées à chaque fois que le serveur démarre, à moins que vous ne démarriez le serveur `mysqld` avec l'option `--skip-grant-tables` . Dans ce cas, l'initialisation des fonctions utilisateurs sont ignorées, et ces fonctions sont inutilisables. Une fonction active doit avoir été créée avec `CREATE FUNCTION` et pas supprimée avec `DROP FUNCTION` .

### 9.2.3 Ajouter de nouvelles fonctions natives

La procédure pour ajouter une nouvelle fonction native est décrite ici. Notez que vous ne pouvez ajouter de fonctions natives à une distribution binaire car la procédure implique la modifications du code source de MySQL. Vous devez compiler MySQL vous-même à partir d'une distribution des sources. Notez aussi que si vous migrez vers une autre version de MySQL (par exemple, quand une nouvelle version est réalisée), vous devrez répéter la procédure avec la nouvelle version.

Pour ajouter une nouvelle fonction native à MySQL, suivez les étapes suivantes :

- Ajoutez une ligne dans lex.h qui définit le nom de la fonction dans le tableau sql_functions[] .
- Si le prototype de la fonction est simple (elle prend zéro, un, deux ou trois arguments), vous devez spécifier dans lex.h SYM(FUNC_ARG#) (où # est le nombre d'arguments) en tant que second argument dans le tableau sql_functions[] et ajouter une fonction qui crée un objet fonction dans item_create.cc . Regardez "ABS" et create_funcs_abs() pour un exemple.Si le prototype de la fonction est compliqué (par exemple, elle prend un nombre variable d'arguments), vous devez ajouter deux lignes à sql_yacc.yy . Une qui indique le symbole préprocesseur que yacc doit définir (cela doit être ajouté au début du fichier). Puis définir les paramètres de la fonction et un ``item'' avec ces paramètres à la règle simple_expr . Pour un exemple, vérifiez toutes les occurences de ATAN dans sql_yacc.yy pour voir comment cela se fait.
- Dans item_func.h , déclarez une classe héritant de Item_num_func ou de Item_str_func , selon que vvotre fonction retourne une nombre ou un chaîne.
- Dans le fichier item_func.cc , ajoutez l'une des déclaration suivantes, selon que vous définissez une fonction numérique ou de chaîne de caractères :

```
 double   Item_func_newname::val()
longlong Item_func_newname::val_int()
String  *Item_func_newname::Str(String *str)
```

  Si vous héritez votre objet de l'un des éléments standards (comme Item_num_func ) vous n'aurez probablement qu'à définir l'une des fonctions décrites ci-dessus et laisser l'objet parent prendre soin des autres fonctions. Par exemple, la classe Item_str_func définit une fonction val() qui exécute atof() sur la valeur retournée par ::str() .
- Vous devez aussi probablement définir la fonction objet suivante :

```
 void Item_func_newname::fix_length_and_dec()
```

  Cette fonction doit au moins calculer max_length en se basant sur les arguments donnés. max_length est le nombre maximal de caractères que la fonction peut retourner. Cette fonction doit aussi définir maybe_null = 0 si la fonction principale ne peut pas retourner une valeur NULL . La fonction peut vérifier si l'un de ses arguments peut retourner NULL en vérifiant la variable maybe_null des arguments. Vous pouvez regarder Item_func_mod::fix_length_and_dec pour avoir un exemple concret.

Toutes les fonctions doivent être sûres pour les threads (en d'autres termes, n'utilisez aucune variable statique ou globale dans la fonction sans les protéger avec mutexes).

Si vous voulez retourner NULL , à partir de ::val() , ::val_int() ou ::str() vous devez mettre null_value à 1 et retourner 0.

Pour les fonctions de l'objet `::str()` , il y a d'autres considérations à prendre en compte :

- L'argument String *str fournit un tampon de chaîne qui peut être utilisé pour contenir le résultat. (Pour plus d'informations à propos du type String , regardez le fichier sql_string.h .)
- La fonction ::str() doit retourner la chaîne contenant le résultat ou (char*) 0 si celui-ci est NULL .
- Aucune des fonctions de chaînes n'essaye d'allouer de mémoire tant que ce n'est pas nécessaire !

## 9.3 Ajouter une nouvelle procédure à MySQL

Avec MySQL, vous pouvez définir une procédure en C++ qui accède et modifie les données dans la requête avant que celle-ci ne soit envoyée au client. La modification peut être faite ligne par ligne ou au niveau de `GROUP BY` .

Nous avons crée une procédure d'exemple avec la version 3.23 de MySQL pour vous montrer comment cela fonctionne.

De plus, nous vous recommandons de jeter un oeil à `mylua` . Avec cela, vous pouvez utiliser le langage LUA pour charger une dynamiquement une procédure dans `mysqld` .

## 9.3.1 La procédure Analyse

```
analyse([max elements,[max memory]])
```

Cette procédure est définie dans le fichier `sql/sql_analyse.cc` . Elle examine les résultats de vos requêtes et en retourne une analyse :

- max elements (256 par défaut) est le nombre maximal de valeurs distinctes que analyse retiendra par colonne. C'est utilisé par analyse pour vérifier si le type optimal de la colonne ne serait pas le type ENUM .
- max memory (8192 par défaut) est le maximum de mémoire que analyse doit allouer par colonne quand elle essaye de trouver toutes les valeurs distinctes.

```
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
```

## 9.3.2 Ecrire une procédure

Pour le moment, la seule documentation à ce sujet est constituée par les sources.

Vous pourrez trouver toutes les informations sur les procédures en examinant les fichiers suivants :

- sql/sql_analyse.cc
- sql/procedure.h
- sql/procedure.cc
- sql/sql_select.cc

# 10 Problèmes et erreurs communes

Ce chapitre liste quelques problèmes et erreurs communes que les utilisateurs rencontreront. Vous apprendrez à déterminer d'où vient le problème, et comment le résoudre. Vous trouverez aussi les solutions appropriées à quelques problèmes communs.

## 10.1 Comment déterminer ce qui pose problème

Lorsque vous faite face à un problème, la première chose à faire et de trouver quel programme / pièce de l'équipement pose ce problème :

- Si vous avez l'un des symptômes suivants, alors il est probable que cela soit un problème matériel (mémoire, carte mère, processeur, ou disque dur) ou un problème de noyau (kernel) :
  - Le clavier ne fonctionne pas. Cela peut être vérifié en pressant la touche Caps Lock (verrouillage majuscules). Si la lumière de Caps Lock light ne change pas, vous devez remplacer votre clavier. (Avant de le faire, redémarrez votre ordinateur après avoir vérifié les câbles du clavier.)
  - Le curseur de la souris ne bouge pas.
  - La machine ne répond pas à un ping externe.
  - D'autres programmes ne fonctionnent pas correctement.
  - Votre système a redémarré sans que vous vous y attendiez (un programme corrompu appartenant à un utilisateur ne devrait **jamais** être capable de couper votre système).

  Dans ce cas, vous devez commencer par vérifier tout vos cables et démarrer quelques outils de diagnostic pour vérifier votre matériel ! Vous devez aussi regarder s'il existe des patches, mises à jours, ou packs de services pour votre système d'exploitation qui pourraient résoudre votre problème. Vérifiez aussi que vos librairies (comme glibc) sont à jour.Il est toujours bon d'utiliser une machine avec de la mémoire ECC pour découvrir les problèmes de mémoire assez tôt !
- Si votre clavier est bloqué, vous pouvez réparer cela en vous loggant sur votre machine à partir d'une autre machine et exécutant kbd_mode −a .
- Examinez votre fichier de log système (/var/log/messages ou similaire) pour connaître les raisons de vos problèmes. Si vous pensez que le problème vient de MySQL, vous devez aussi examiner les fichiers de log de MySQL. Le log des mises à jour .
- Si vous ne pensez pas avoir de problèmes au niveau du matériel, vous devez trouver quel programme pose problème.Essayez en utilisant top , ps , taskmanager , ou des programmes similaires, pour voir quel programme utilise trop de ressources ou bloque la machine.
- Vérifiez avec top , df , ou un programme similaire si vous n'avez plus de mémoire, d'espace disque, trop de fichiers ouverts ou un problème avec une autre ressource critique.
- Si le problème vient d'un processus, vous pouvez toujours essayer de le terminer. S'il ne veut pas se terminer, c'est probablement un bogue du système d'exploitation.

Si après tout cela vous pensez encore que le problème vient du serveur MySQL ou du client MySQL, il est temps de préparer un rapport de bogue pour notre liste de diffusion ou notre équipe de support. Dans ce rapport, essayez de donner la description la plus détaillée possible du comportement du système et de ce que vous pensez qu'il se passe. Vous devez aussi mentionner pourquoi est−ce que vous pensez que le problème vient de MySQL. Prenez en considération toutes les situations décrites dans ce chapitre. Décrivez les problèmes exactement comme ils surviennent sur votre système. Utilisez la méthode 'copier/coller' pour les affichages et les messages d'erreurs provenant des programmes ou des fichiers de log.

Essayez de décrire en détail quel est le programme qui ne fonctionne pas et tous les symptômes que vous voyez ! Nous avons déjà reçu beaucoup de rapports de bogue qui disaient juste "le système ne marche pas". Cela ne nous fournit aucune information à propos du problème.

Si un programme échoue, il est toujours utile de savoir :

- Est−ce que le programme en question a causé une erreur de segmentation (core dumped)?
- Est−ce que le programme consomme toutes les ressources CPU ? Vérifiez avec top Laissez le programme fonctionne un bout de temps, il se peut qu'il soit entrain de traiter une tâche lourde.
- Si c'est le serveur mysqld qui pose problème, pouvez vous essayer un mysqladmin −u root ping ou mysqladmin −u root processlist ?
- Que dit un programme client (essayez avec mysql , par exemple) quand vous essayez de vous connecter au serveur MySQL ? Le programme se bloque−t−il ? Obtenez vous un retour quelconque ?

Lors de l'envoi d'un rapport de bogue, vous devez respecter les règles définies dans ce manuel. Poser des questions ou rapporter des bogues .

## 10.2 Erreurs communes rencontrées avec MySQL

Cette section couvre les erreurs les plus fréquement rencontrées par les utilisateurs. Vous trouverez ici une description de ces erreurs et un moyen de les corriger.

## 10.2.1 Erreur Access denied

Voir Comment le système de droits fonctionne et plus particulièrement Causes possibles de l'erreur Access denied .

## 10.2.2 Erreur MySQL server has gone away

Cette section couvre aussi l'erreur `Lost connection to server during query` .

Le plus souvent, l'erreur `MySQL server has gone away` se produit lorsque le serveur a dépassé le délai d'attente d'actions et a clos la connexion. Par défaut, le serveur clos la connexion après 8 heures si rien n'est arrivé. Vous pouvez changer cette limite en configurant le paramètre `wait_timeout` lorsque vous démarrez `mysqld` .

Une autre raison de recevoir l'erreur MySQL server has gone away est d'avoir emis un ``close'' sur votre connexion puis d'avoir essayé d'actionner une autre commande alors que la connexion était close.

Si vous avez un script, vous n'avez qu'à lancer la requête à nouveau pour que le client se reconnecte automatiquement.

Vous obtiendrez normalement les codes erreurs suivants dans ce cas (qui est indépendant du système d'exploitation) :

| Code erreur | Description |
|---|---|
| CR_SERVER_GONE_ERROR | Le client ne peut envoyer de commandes au serveur. |
| CR_SERVER_LOST | Le client n'a pas obtenu d'erreur en contactant le serveur, mais n'a pas obtenu de réponse complète à la question posée. |

vous obtiendrez aussi cette erreur si quelqu'un à terminé le processus avec kill #idprocessus# .

Vous pouvez vérifier si le serveur MySQL est encore en marche en exécutant `mysqladmin version` et examinant la date de mise en route. Si le problème est que mysqld a crashé, vous devriez vous concentrer sur la résolution du problème. Vous devez dans ce cas commencer par vérifier si emmètre la même requête fera à nouveau crasher MySQL. Que faire si MySQL crashe constamment .

Vous pouvez aussi obtenir ces erreurs si vous envoyez une requête incorrecte ou trop grande au serveur. Si mysqld reçoit un paquet trop large ou mal ordonné, il suppose que quelque chose s'est mal passé au niveau du client et ferme la connexion. Si vous avez besoin de grande requêtes (par exemple, si vous travaillez avec de grandes colonnes BLOB ) vous pouvez augmenter la taille limite des requêtes en démarrant mysqld avec l'option -O max_allowed_packet=# (1M par défaut). Le surplus de mémoire est alloué à la demande, ce qui fait que mysqld n'utilisera de la mémoire que lorsque vous emmétrez une grande requête ou qu'il aura à retourner de grandes réponses !

Si vous voulez rapporter un bogue concernant ce problème, merci d'inclure les informations suivantes :

- MySQL a-t-il planté ? (Vous pouvez le savoir en regardant le fichier hostname.err . Que faire si MySQL crashe constamment .
- Si une requête spécifique fait planter mysqld et que les tables concernées ont bien été vérifiées avec CHECK TABLE avant l'exécution, pouvez-vous faire une batterie de tests ? Faire une batterie de tests lorsque vous faites face à un problème de table corrompue .
- Quelle est la valeur de la variable wait_timeout dans le serveur MySQL ? mysqladmin variables vous donnera une réponse
- Avez-vous essayé de démarrer mysqld avec --log et vérifié si la requête apparaît bien dans le log ?

Poser des questions ou rapporter des bogues .

## 10.2.3 Erreur Can't connect to [local] MySQL server

Un client MySQL sous Unix peut se connecter au serveur `mysqld` de deux façons différentes : sockets Unix, qui se connectent via un fichier du système de fichiers ( `/tmp/mysqld.sock` par défaut) ou TCP/IP, qui se connecte via un port. Les sockets Unix sont plus rapides que TCP/IP mais ne peuvent être utilisée que pour des connexions locales. Les sockets sont utilisées si vous ne spécifiez pas de nom d'hôte ou si vous spécifiez le nom d'hôte spécial `localhost` .

Sur Windows, si le serveur `mysqld` tourne sur 9x/Me, vous ne pouvez vous connecter qu'avec TCP/IP. Si le serveur tourne sur NT/2000/XP et que mysqld a été démarré avec l'option `--enable-named-pipe` ,

vous pouvez aussi vous connectez avec un tunnel nommé. Son nom est MySQL. Si vous ne spécifiez pas un nom d'hôte lors de la connexion à mysqld , un client MySQL essayera d'abord de se connecter au tunnel nommé, et si cela ne marche pas il se connectera au port TCP/IP. Vous pouvez forcer l'utilisation des tunnels nommés sous Windows en utilisant . en tant que nom d'hôte.

L'erreur (2002) Can't connect to ... signifie généralement qu'il n'y a aucun serveur MySQL qui tourne sur la machine ou que vous utiliser un mauvais fichier de socket ou un port erroné quand vous essayez de vous connecter au serveur mysqld .

Commencez par vérifier (en utilisant ps ou le gestionnaire de tâches sous Windows) qu'il y a un processus nommé mysqld sur votre serveur ! S'il n'y en a aucun, vous devrez en démarrer un. Problèmes de démarrage de MySQL .

Si un processus mysqld est actif, vous pouvez testerle serveur avec l'une des connexions suivantes (le port et le chemin vers la socket peuvent être différents chez vous, bien sûr) :

```
 shell> mysqladmin version
shell> mysqladmin variables
shell> mysqladmin -h `hostname` version variables
shell> mysqladmin -h `hostname` --port=3306 version
shell> mysqladmin -h 'ip de votre hôte' version
shell> mysqladmin --socket=/tmp/mysql.sock version
```

Notez l'utilisation des guillemets obliques plutôt que les simples avec la commande hostname ; cela provoque la substitution de hostname par la valeur courante du nom d'hôte de la machine dans la commande mysqladmin .Voilà quelques raisons pouvant entraîner l'erreur Can't connect to local MySQL server :

- mysqld ne fonctionne pas.
- Vous utilisez un système qui utilise les pthreads MIT. Si vous utilisez un système qui n'a pas le support natif des threads, mysqld utilises le package MIT-pthreads. Operating Systems Supported by MySQL . Toutefois, toutes les versions de MIT-pthreads ne supportent pas les sockets Unix. Sur un système qui ne supporte pas les sockets vous devez toujours spécifier le nom d'hôte explicitement lors de la connexion au serveur. Utilisez cette commande pour vérifier la connexion au serveur :

  ```
  shell> mysqladmin -h `hostname` version
  ```

- Quelqu'un a effacé le fichier de socket Unix que mysqld utilise ( /tmp/mysqld.sock par défaut). Vous avez peut-être une tâche cron qui efface la socket MySQL (par exemple, une tâche qui supprime les anciens fichiers du dossier /tmp ). Vous pouvez toujours exécuter mysqladmin version et vérifier que la socket que mysqladmin tente d'utiliser existe vraiment. La solution dans ce cas est de modifier la tâche cron pour qu'elle n'efface plus mysqld.sock ou de placer la socket quelque part d'autre. Comment protéger /tmp/mysql.sock .
- Vous avez démarré mysqld avec l'option --socket=/chemin/vers/socket . Si vous changez le chemin vers la socket vous devez aussi en notifier les clients. Vous pouvez le faire en fournissant le chemin vers la socket en argument au client. Comment protéger /tmp/mysql.sock .
- Vous utilisez Linux et un thread s'est terminé (core dumped). Dans ce cas, vous devez aussi terminer les autres threads mysqld (par exemple, avec le script mysql_zap avant de pouvoir démarrer un nouveau serveur MySQL. Que faire si MySQL crashe constamment .
- Vous n'avez peut-être pas les privilèges de lecture et écriture sur le dossier contenant la socket ou sur la socket elle-même. Dans ce cas, vous devez changer les droits sur ce dossier / fichier ou redémarrer mysqld pour qu'il prenne en compte un dossier auquel vous avec accès.

Si vous obtenez l'erreur Can't connect to MySQL server on un_hôte , vous pouvez essayer ce qui suit pour trouver le problème :

- Vérifiez que le serveur fonctionne en faisant telnet votre-nom-d-hôte port-tcp-ip et pressez la touche Enter plusieurs fois. Si il y a un serveur MySQL qui tourne sur ce port, vous devriez obtenir une réponse contenant le numéro de version du serveur. Si vous obtenez une erreur proche de telnet: Unable to connect to remote host: Connection refused , c'est qu'il n'y a pas de serveur tournant sur le port donné.
- Essayez de vous connecter au démon mysqld sur la machine locale et vérifiez le port TCP/IP de la configuration de mysqld (variable port ) avec mysqladmin variables .
- Vérifiez que votre serveur mysqld n'est pas configuré avec l'option --skip-networking .

## 10.2.4 Host '...' is blocked Error

Si vous obtenez cette erreur :

```
 Host 'hostname' is blocked because of many connection errors.
Unblock with 'mysqladmin flush-hosts'
```

cela veut dire que mysqld a reçu trop de ( max_connect_errors ) tentatives de connexions à l'hôte 'hostname' qui ont été interrompus en plein milieu. Après max_connect_errors requêtes échouées, mysqld pense qu'il se passe quelque chose de mauvais (comme une attaque de la part d'un pirate), et bloque le serveur pour les prochaines connexions jusqu'à ce que quelqu'un exécute la commande mysqladmin flush-hosts .Par défaut, mysqld bloque le serveur après 10 connexions erronées. Vous pouvez facilement changer ce comportement en démarrant le serveur avec ces arguments :

```
 shell> safe_mysqld -O max_connect_errors=10000 &
```

Notez que si vous recevez ce message pour un hôte en particulier, vous devriez vous assurer qu'il n'y a pas de problèmes de connexions TCP/IP depuis cet hôte. Si vos connexions TCP/IP ne marchent pas, il ne servira à rien d'augmenter la valeur de la variable max_connect_errors !

## 10.2.5 Erreur Too many connections

Si vous obtenez l'erreur `Too many connections` en essayant de vous connecter à MySQL, cela signifie qu'il y a déjà `max_connections` clients connectés au serveur `mysqld` .

Si vous avez besoin de plus de connexion que par défaut (100), vous devez redémarrer mysqld avec une plus grande valeur pour la variable max_connections .

Notez que `mysqld` permet actuellement à ( `max_connections` +1) clients de se connecter. La dernière connexion est réservée à l'utilisateur ayant le privilège `SUPER` . En ne donnant pas ce privilège aux utilisateurs normaux (ils ne devraient pas en avoir besoin), un administrateur avec ce privilège peut se connecter et utiliser `SHOW PROCESSLIST` pour trouver ce qui pose problème. Syntaxe de `SHOW` .

Le nombre maximal de connexion MySQL dépend de la qualité de la librairie des threads sur une plate−forme donnée. Linux et Solaris devraient être capables de supporter jusqu'à 500−1000 connexion simultanées, cela dépend évidemment de la quantité de RAM que vous avez et de ce que font les clients.

## 10.2.6 Erreur Some non−transactional changed tables couldn't be rolled back

Si vous obtenez l'erreur / avertissement : `Warning: Some non−transactional changed tables couldn't be rolled back` en essayant de faire un `ROLLBACK` , cela signifie que certaines tables que vous avez utilisé dans la transaction ne supportent pas les transactions. Ces tables non−transactionneles ne seront pas affectées par la commande `ROLLBACK` .

Le cas le plus typique pour que cela arrive est que vous avez essayé de créer une table d'nu type non supporté par votre binaire mysqld . Si mysqld ne supporte pas un type de table (ou si le type de table est désactivé par une option de démarrage), il créera une table avec le type qui se rapproche le plus de celui que vous aviez choisi, probablement MyISAM .

Vous pouvez vérifier le type d'une table en faisant :

SHOW TABLE STATUS LIKE 'nom_de_table' . SHOW TABLE STATUS .

Vous pouvez vérifier les extensions que votre binaire `mysqld` supporte en faisant :

show variables like 'have_%' . SHOW VARIABLES .

## 10.2.7 Erreur Out of memory

Si vous lancez une requête et que vous obtenez l'erreur suivante :

```
 mysql: Out of memory at line 42, 'malloc.c'
mysql: needed 8136 byte (8k), memory in use: 12481367 bytes (12189k)
ERROR 2008: MySQL client ran out of memory
```

notez que cette erreur parle du client MySQL client mysql . La raison de cette erreur est simplememnt que le client n'a pas la mémoire suffisante pour stocker le résultat en entier.Pour remédier à ce problème, vérifiez d'abord que votre requête est correcte. Est−ce normal qu'elle retourne autant de lignes ? Si oui, vous pouvez utiliser mysql −−quick , qui utilise mysql_use_result() pour récupérer les résultats. Cela solicittera moins le client (mais plus le serveur).

## 10.2.8 Erreur Packet too large

Lorsqu'un client MySQL ou le serveur `mysqld` reçoit un paquet plus grand que `max_allowed_packet` octets, il provoque une erreur `Packet too large` et ferme la connexion.

En MySQL 3.23 le plus gros paquet possible est 16M (à cause des limites du protocole client/serveur). En MySQL 4.0.1 et plus, cela n'est plus limité que par la quantité de mémoire que vous avez sur votre serveur (cela va théoriquement à un maximum de 2G).

Un paquet de communication est une simple commande SQL envoyée au serveur, ou une simple ligne renvoyée au client.

Lorsqu'un client MySQL ou que le serveur `mysqld` reçoit un paquet plus grand que `max_allowed_packet` octets, il provoque une erreur `Packet too large` et ferme la connexion. avec quelques clients, vous pouvez aussi obtenir l'erreur `Lost connection to MySQL server during query` si le paquet est trop grand.

Notez que le client et le serveur ont chacun leur propre variable max_allowed_packet . Si vous voulez gérer les gros paquets, vous devrez changer cette variable côté client et côté serveur.

Il n'est pas dangereux d'augmenter cette valeur étant donné que la mémoire n'est alloué que lorsque besoin en est. Cette variable est plus une précaution pour capturer les paquets erronés qui circulent entre le client et le serveur. Elle sert aussi a vous assurer que vous n'utilisez pas accidentellement de gros paquets qui consommeront toute la mémoire.

Si vous utilisez le client mysql , vous pouvez spécifier un plus grand tampon en démarrant le client avec mysql --set-variable=max_allowed_packet=8M . Les autres clients ont différentes méthodes pour configurer cette variable. Notez que --set-variable est désapprouvée depuis MySQL 4.0, utilisez --max-allowed-packet=8M à la place. Vous pouvez utiliser le fichier d'options pour augmenter la taille de max_allowed_packet dans mysqld . Par exemple, si vous vous attendez à stocker la totalité d'un MEDIUMBLOB dans une table, vous aurez besoin de démarrer le serveur avec l'option set-variable=max_allowed_packet=16M .

Vous pouvez aussi rencontrer d'étranges problèmes avec les gros paquets si vous utilisez les grands blob, mais que vous n'avez pas donné à `mysqld` l'accès à assez de mémoire pour gérer ces requêtes. Si vous pensez être dans ce cas, essayez d'ajouter `ulimit -d 256000` au début du script `safe_mysqld` et redémarrez `mysqld` .

## 10.2.9 Erreurs de communication / Connexion annulée

A partir de MySQL 3.23.40 vous n'obtenez l'erreur Aborted connection que si vous démarrez mysqld avec --warnings .

Si vous trouvez des erreurs comme celle qui suit dans vos logs d'erreurs :

```
010301 14:38:23  Aborted connection 854 to db: 'users' user: 'josh'
```

Le log d'erreurs .

Cela signifie qu'un problème est survenu :

- Le programme client n'a pas appelé mysql_close() avant de quitter.
- Le client a été inactif plus de wait_timeout ou interactive_timeout secondes sans aucune requête. SHOW VARIABLES . SHOW VARIABLES .
- L'exécution du programme client s'est terminée soudainement au milieu d'un transfert.

Lorsque ce qui précède arrive, la variable Aborted_clients est incrémentée.

La variable serveur `Aborted_connects` est incrémentée lorsque :

- Un paquet de connexion ne contient pas la bonne information.
- L'utilisateur n'avait pas les droits d'accès à la base de données.
- L'utilisateur a utilisé un mot de passe erroné.
- Il a fallu plus de connect_timeout secondes pour obtenir un paquet de communication. SHOW VARIABLES

Notez que ce qui précède peut indiquer que quelqu'un essaye de s'introduire dans votre base de données !

Autres raisons pour les problèmes de clients echoués / connexions interrompues :

- L'utilisation du protocole Ethernet sous Linux, que le Duplex soit intégral (Full-Duplex) ou partiel (Half Duplex) avec Linux. La plupart des pilotes Ethernet de Linux ont ce bogue. Vous pouvez tester ce bogue en transférant un énorme fichier via FTP d'une machine à l'autre. Si le transfert est saccadé (succession de transfert-pause-transfert-pause) alors vous vivez le syndrôme du duplex de Linux. La seule solution est de changer le mode de Duplex sur les deux cartes réseau et Hub/Switch en duplex partiel ou duplex intégral et de faire des tests pour connaître la meilleure configuration.
- Quelques problèmes avec la librairie des threads qui causent des interruptions de lectures.
- Mauvaise configuration TCP/IP.
- Les câbles, concentrateurs ou commutateurs Ethernet déffectueux. On peut le diagnostiquer aisément en remplacement le matériel.
- max_allowed_packet est trop petit ou les requêtes ont besoin de plus de mémoire que celle que vous avez alloué à mysqld . Erreur Packet too large .

## 10.2.10 Erreur The table is full

Il y a différents cas où vous pouvez obtenir cette erreur :

- Vous utilisez une ancienne version de MySQL (avant 3.23.0) quand une table temporaire en mémoire devient plus grande que tmp_table_size octets. Pour éviter ce problème, vous pouvez utiliser l'option -O tmp_table_size=# pour faire augmenter la taille des tables temporaires à mysqld ou utiliser l'option SQL BIG_TABLES avant d'exécuter la requête qui pose problème. Syntaxe de SET .

  Vous pouvez aussi démarrer `mysqld` avec l'option `--big-tables` . Cela revient à utiliser `BIG_TABLES` pour toutes les requêtes.

  Dans la version 3.23 de MySQL, les tables temporaires en mémoire seront automatiquement changées en tables physique MyISAM après qu'elles n'aient dépassé tmp_table_size .
- Vous utilisez des tables InnoDB et avez dépassé leur taille. Dans ce cas, la solution est d'augmenter les tailles des tables.
- Vous utilisez des tables ISAM ou MyISAM sur un système d'exploitation qui ne supporte pas les fichiers de plus de 2G et vous avez atteint cette limite dans le fichier de données ou d'index.
- Vous utilisez des tables MyISAM et la taille des données ou de l'index est plus grande que celle que MySQL a alloué aux pointeurs. (Si vous ne spécifiez pas MAX_ROWS à CREATE TABLE MySQL n'allouera que des pointeurs supportant 4G de données).Vous pouvez obtenir la taille maximale des données / index en faisant :

  ```
  SHOW TABLE STATUS FROM database LIKE 'nom_de_table';
  ```

  or using myisamchk -dv database/nom_de_table .Si le problème vient de là, vous pouvez le corriger en faisant quelque chose se rapprochant de :

  ```
  ALTER TABLE nom_de_table MAX_ROWS=1000000000 AVG_ROW_LENGTH=nnn;
  ```

  Vous n'avez besoin de spécifier AVG_ROW_LENGTH que pour les tables avec des champs BLOB/TEXT car dans ce cas, MySQL ne peut optimiser l'espace requis en se basant uniquement sur le nombre de lignes.

## 10.2.11 Erreur Can't create/write to file

Si vous obtenez une erreur de ce type pour quelques requêtes :

```
Can't create/write to file '\\sqla3fe_0.ism'.
```

cela signifie que MySQL ne peut créer de fichier temporaire pour le jeu de résultats dans le dossier temporaire défini. (L'erreur précédente est typique de Windows, et le message d'erreur Unix est similaire.) La solution est de démarrer mysqld avec l'option --tmpdir=chemin ou d'ajouter à votre fichier d'options :

```
[mysqld]
tmpdir=C:/temp
```

en supposant que le dossier c:\\temp existe. Fichier d'options my.cnf .Vérifiez aussi le code erreur que vous obtenez avec perror . Une autre rasion peut être une erreur de disque saturé.

```
shell> perror 28
Error code  28:  No space left on device
```

## 10.2.12 Erreur du client Commands out of sync

Si vous obtenez l'erreur Commands out of sync; you can't run this command now , le problème vient du fait que vous appelez les fonctions dans le mauvais ordre dans votre code !

Cela peut se produire, par exemple, si vous utilisez `mysql_use_result()` et essayez d'exécuter une nouvelle requête avant d'avoir appelé `mysql_free_result()` . Cela peut aussi se produire si vous essayez d'exécuter deux requêtes qui retournent des données dans appeler `mysql_use_result()` ou `mysql_store_result()` entre les deux.

## 10.2.13 Erreur Ignoring user

Si vous obtenez l'erreur suivante :

```
Found wrong password for user: 'some_user@some_host'; ignoring user
```

cela signifie que lors du démarrage de `mysqld` ou lorsqu'il a rechargé les tables de permissions, il a trouvé une entrée dans la table `user` avec un mot de passe invalide. De ce fait, l'entrée est tout simplement ignorée par le système de droits.

Causes possibles et solutions pour ce problème :

- Vous faites peut-être tourner une nouvelle version de mysqld avec une vielle table user . Vous pouvez vérifier cela en exécutant mysqlshow mysql user pour voir si le champ du mot de passe est plus petit que 16 caractères. Si c'est le cas, vous pouvez le corriger en exécutant le script scripts/add_long_password .
- L'utilisateur a un ancien mot de passe (8 caractères) et vous n'avez pas démarré mysqld avec l'option --old-protocol . Mettez à jour le mot de passe dans la table user ou redémarrez mysqld avec --old-protocol .
- Vous avez spécifié un mot de passe dans la table user sans passer par la fonction PASSWORD() . Utilisez mysql pour mettre à jour l'utilisateur dans la table user avec un nouveau mot de passe. Assurez-vous d'utiliser la fonction PASSWORD() :

```
mysql> UPDATE user SET password=PASSWORD('votre mot de passe')
    ->             WHERE user='XXX';
```

## 10.2.14 Erreur Table 'xxx' doesn't exist

Si vous obtenez l'erreur `Table 'xxx' doesn't exist` ou `Can't find file: 'xxx' (errno: 2)` , cela signifie qu'aucune table portant le nom `xxx` n'existe dans la base de données courante.

Notez que comme MySQL utilise des dossiers et des fichiers pour stocker les bases de données et les tables, les noms sont **sensibles à la casse** ! (Sous Windows les noms ne sont pas sensibles à la casse, mais vous devez utiliser la même casse dans une même requête !)

Vous pouvez obtenir la liste des tables que vous avez dans la base de données courante avec la commande SHOW TABLES . Syntaxe de SHOW .

## 10.2.15 Erreur Can't initialize character set xxx

Si vous obtenez l'erreur suivante :

```
MySQL Connection Failed: Can't initialize character set xxx
```

Cela signifie l'une des choses suivantes :

- Le jeu de caractères est un jeu multi-octets et votre client ne les supporte pas.Dans ce cas, vous devez recompiler le client avec --with-charset=xxx ou avec --with-extra-charsets=xxx . Les options habituelles de configure .

  Tous les binaires standards de MySQL sont compilés avec `--with-extra-character-sets=complex` qui active le support de tous les jeux de caractères multi-octets. Le jeu de caractères utilisé pour les données et le stockage .
- Le jeu de caractères est un simple jeu de caractères qui n'est pas compilé dans mysqld et les fichiers de définition du jeu ne sont pas à l'endroit où le client si attend.

  Dans ce cas vous avez besoin de :

  - Recompiler le client avec le support du jeu de caractères. Les options habituelles de configure .
  - Spécifier au client où les fichiers de définition du jeu de caractères se situent. Pour beaucoup de clients, vous pouvez le faire avec l'option --character-sets-dir=chemin-vers-dossier-jeu-caractères .
  - Copier les fichiers de définition du jeu de caractères dans le dossier où le client s'attend à les trouver.

## 10.2.16 Fichier non trouvé

Si vous obtenez `ERROR '...' not found (errno: 23)` , `Can't open file: ... (errno: 24)` , ou toute autre erreur avec `errno 23` ou `errno 24` de la part de MySQL, cela signifie que vous n'avez pas alloué assez de descripteurs de fichiers à MySQL. Vous pouvez utiliser l'utilitaire `perror` pour obtenir une description de ce que veut dire l'identifiant de l'erreur :

```
 shell> perror 23
File table overflow
shell> perror 24
Too many open files
shell> perror 11
Resource temporarily unavailable
```

Le problème ici est que mysqld essaye de garder trop de fichiers ouverts en même temps. Vous pouvez soit demander à mysqld de ne pas ouvrir autant de fichiers simultanément ou augmenter le nombre de descripteurs de fichiers alloués à mysqld .Pour dire à mysqld de garder moins de fichiers ouverts en même temps, vous pouvez rendre le cache de tables plus petit en utilisant l'option -O table_cache=32 de safe_mysqld (la valeur par défaut est 64). Réduire la valeur de max_connections réduira aussi le nombre de fichiers ouverts (90 comme valeur de défaut).

Pour changer le nombre de descripteurs de fichiers alloués à `mysqld` , vous pouvez utiliser l'option `--open-files-limit=#` de `safe_mysqld` ou `-O open-files-limit=#` de `mysqld` . `SHOW VARIABLES` . La façon la plus facile de faire cela est d'ajouter cette option dans votre fichiers d'options. Fichier d'options `my.cnf` . Si vous avec une ancienne version de `mysqld` qui ne le supporte pas, vous pouvez éditer le script `safe_mysqld` . Il y a une ligne commentée `ulimit -n 256` dans le script. Vous pouvez enlever le caractère `'#'` pour décommenter cette ligne, et changer le nombre 256 pour affecter le nombre de descripteurs de fichiers alloués à `mysqld` .

ulimit (et open-files-limit ) peuvent augmenter le nombre de descripteurs de fichiers, mais seulement jusqu'à la limite imposée par le système d'exploitation. Il y a aussi une limite 'matérielle' qui ne peut être dépassée que si vous démarrez safe_mysqld ou mysqld en tant que root (souvenez-vous juste que vous devez aussi utiliser l'option --user=... dans ce cas). Si vous avez besoin de repousser les limites du système d'exploitation pour les descripteurs de fichiers disponibles pour chaque processus, consultez la documentation de votre système.

Notez que si vous démarrez le shell `tcsh` , `ulimit` ne fonctionnera pas ! `tcsh` retournera aussi des valeurs incorrectes si vous atteignez la limite courante ! Dans ce cas, vous devez démarrer `safe_mysqld` avec `sh` !

# 10.3 Notes relatives à l'installation

## 10.3.1 Problèmes lors de la liaison avec la librairie du client MySQL

Si vous liez votre programme et que vous obtenez des erreurs pour des symboles non-référencés qui commencent par `mysql_` , comme ce qui suit :

```
 /tmp/ccFKsdPa.o: In function `main':
/tmp/ccFKsdPa.o(.text+0xb): undefined reference to `mysql_init'
/tmp/ccFKsdPa.o(.text+0x31): undefined reference to `mysql_real_connect'
/tmp/ccFKsdPa.o(.text+0x57): undefined reference to `mysql_real_connect'
/tmp/ccFKsdPa.o(.text+0x69): undefined reference to `mysql_error'
/tmp/ccFKsdPa.o(.text+0x9a): undefined reference to `mysql_close'
```

vous pouvez réparer cela en ajoutant -Lchemin-vers-la-librairie-mysql -lmysqlclient **last** dans votre ligne de liaison.

Si vous obtenez une erreur `undefined reference` pour la fonction `uncompress` ou `compress` , ajoutez `-lz` **à la fin** de votre ligne de liaison et essayez à nouveau !

Si vous obtenez des erreurs undefined reference pour des fonctions qui devraient exister sur votre système, comme connect , vérifiez la page de manuel de la fonction en question, pour les librairies que vous devez ajouter à la ligne de liaison !

Si vous obtenez une erreur `undefined reference` pour des fonctions inexistantes sur votre système, ressemblant à ce qui suit :

```
 mf_format.o(.text+0x201): undefined reference to `__lxstat'
```

cela signifie que votre librairie est compilé sur un système qui n'est pas à 100% compatible avec le votre. Dans ce cas, vous devez obtenir la dernière distribution des sources de MySQL et compiler vous-mêmes. Installer MySQL à partir des sources .

Si vous essayez de faire fonctionner un programme et que vous obtenez des erreurs pour des symboles non-référencés qui commencent par `mysql_` ou une erreur disant que la librairie `mysqlclient` ne peut être trouvée, cela signifie que votre système n'arrive pas à trouver la librairie partagée `libmysqlclient.so` .

La solution est de dire à votre système de chercher les librairies partagées là où la librairie est située avec l'une des méthodes suivantes :

- Ajouter le chemin vers le dossier où se situe libmysqlclient.so dans la variable d'environnement LD_LIBRARY_PATH .
- Ajouter le chemin vers le dossier où se situe libmysqlclient.so dans la variable d'environnement LD_LIBRARY .
- Copiez le fichier libmysqlclient.so à un endroit où votre système le cherche, comme dans le dossier /lib , et mettez à jour les informations de la librairie partagée en exécutant ldconfig .

Un autre moyen de résoudre ce problème est de lier votre programme statiquement, avec -static , ou en effaçant les librairies dynamiques de MySQL avant de lier votre code. Dans le second cas vous devez vous assurer qu'aucun autre programme n'utilise les

librairies dynamiques !

## 10.3.2 Comment exécuter MySQL comme un utilisateur normal

Le serveur MySQL mysqld peut être démarré par n'importe quel utilisateur. Afin de changer l'utilisateur qui fait tourner mysqld en l'utilisateur Unix nom_utilisateur , vous devez faire ceci :

- Stoppez le serveur si il fonctionne (utilisez mysqladmin shutdown ).
- Changez le propriétaire du dossier et des fichiers de bases pour qu'il soit nom_utilisateur . Il faut que cet utilisateur ait les droits d'écriture et de lecture (vous pourriez avoir à faire cette manipulation en tant que root Unix) :

```
shell> chown -R nom_utilisateur /path/to/mysql/datadir
```

  Si les dossier ou les fichiers de données de MySQL sont des liens symboliques, vous devez vous assurer de pouvoir suivre ces lignes, et de changer les propriétaires des fichiers et dossiers sur lesquels ils pointent. L'option chown -R de chown peut ne pas suivre les liens symboliques.
- Démarrez le serveur avec l'utilisateur nom_utilisateur , ou bien, si vous utilisez MySQL version 3.22 ou plus récent, démarrez mysqld en tant que root Unix, et utilisez l'option --user=nom_utilisateur . mysqld va alors changer automatiquement d'utilisateur pour utiliser nom_utilisateur avant d'accepter les connexions.
- Pour démarrer le serveur sous le nom d'utilisateur automatiquement au moment du démarrage du système, ajouter une ligne user qui spécifie le nom de l'utilisateur que le groupe de [mysqld] est du même groupe que le fichier d'options /etc/my.cnf ou le fichier d'options my.cnf dans le dossier de données du serveur. Par exemple :

```
[mysqld]
user=nom_utilisateur
```

A ce moment, votre processus `mysqld` doit fonctionner normalement sous le nom de l'utilisateur Unix `nom_utilisateur` . Une chose n'a pas changé : les droits dans les tables de droits de MySQL. Par défaut, juste après avoir exécuté le script d'installation des tables de droits `mysql_install_db` , l'utilisateur MySQL `root` est le seul utilisateur du système avec les droits de créer et de détruire les bases. A moins que vous n'ayez changé ces droits, ils ont toujours cours. Cela ne va pas vous empêcher d'accéder à MySQL en tant que `root` MySQL, même si vous n'êtes pas connecté en tant que `root` Unix. Spécifiez simplement l'option `-u root` au programme client.

Notez qu'accéder à MySQL en tant que root , en fournissant l'option -u root en ligne de commande, **n'a rien a voir** avec MySQL qui fonctionne avec les droits de root Unix, ou d'un autre utilisateur Unix. Les droits d'accès et les noms d'utilisateurs MySQL sont complètement séparé des noms d'utilisateurs et des mots de passes Unix. Le seul rapport avec les utilisateurs Unix est que si vous ne fournissez pas l'option -u lorsque vous démarrez votre client, le client va essayer de se connecter à MySQL avec votre nom d'utilisateur Unix.

Si votre serveur Unix n'est pas sécurisé, il est recommandé de donner un mot de passe à l'utilisateur MySQL `root` dans les tables de droits. Sinon, n'importe quel utilisateur ayant un compte sur cette machine va pouvoir accéder au compte root avec l'option `mysql -u root nom_base` et faire ce qu'il veut.

## 10.3.3 Problèmes avec les permissions sur fichiers

Si vous avez des problèmes avec les droits sur fichiers, par exemple, si mysql génère l'erreur suivante lorsque vous créez une table :

```
ERROR: Can't find file: 'path/with/filename.frm' (Errcode: 13)
```

alors la variable d'environnement UMASK est peut-être mal configurée lorsque mysqld est démarré. La valeur par défaut de umask est 0660 . Vous pouvez corriger ce comportement en démarrant safe_mysqld de la façon suivante :

```
shell> UMASK=384  # = 600 en octal
shell> export UMASK
shell> /chemin/vers/safe_mysqld &
```

Par défaut, MySQL créera les dossiers des bases de données et de RAID avec 0700 comme type de permissions. Vous pouvez modifier ce comportement en définissant la variable UMASK_DIR . Si vous le faite, les nouveaux dossiers seront crées en combinant UMASK et UMASK_DIR . Par exemple, si vous voulez donner un accès de groupe à tout les nouveaux dossiers, vous pouvez faire :

```
shell> UMASK_DIR=504  # = 770 en octal
shell> export UMASK_DIR
shell> /chemin/vers/safe_mysqld &
```

A partir de la version 3.23.25, MySQL suppose que les valeurs de UMASK et UMASK_DIR sont en octal si elles commencent par un zéro. Variables d'environnement MySQL .

# 10.4 Notes relatives à l'administration

## 10.4.1 Que faire si MySQL crashe constamment ?

Toutes les versions de MySQL sont testées sur plusieurs plate−formes avant leur publication. Cela ne signifie pas qu'elles sont exemptées de bogues, cela signifie juste que si il y a des bogues, il y'en a très peu et sont durs à trouver. Si vous avez un problème, cela nous aidera toujours si vous essayez de trouver d'où vient exactement le crashage système, et vous aurez plus de chances de le voir résolu rapidement.

D'abord, vous devez essayer de trouver si le problème vient du démon mysqld qui se termine, ou s'il est lié à votre client. Vous pouvez savoir depuis combien de temps le serveur mysqld tourne en exécutant mysqladmin version . Si mysqld s'est terminé, vous trouverez sûrement la raison dans le fichier mysql−data−directory/`hostname`.err . Le log d'erreurs .

Plusieurs crashes de MySQL sont causés par des fichiers de données ou d'index corrompus. MySQL écrira les données sur le disque avec un appel système à `write()` , après chaque requête et avant d'en notifier le client. (Cela n'est pas vrai si vous utilisez `delay_key_write` , auquel cas seul les données sont écrites.) Cela signifie que les données sont intègres même si `mysqld` crashe, puisque le système d'exploitation s'assurera que les données non sorties du tampon ne sont pas enregistrée sur le disque. Vous pouvez forcer MySQL à se synchroniser avec le disque après chaque requête en démarrant `mysqld` avec `−−flush` .

Ce qui précède signifie que normalement, vous ne devriez obtenir de tables corrompues que si :

- Quelqu'un ou quelque chose a coupé mysqld ou la machine au milieu d'une mise à jour.
- Vous avez trouvé un bogue dans mysqld qui le termine au milieu d'une mise à jour.
- Quelqu'un manipule les fichiers de données ou d'index en dehors de **mysqld** sans verrouiller proprement les tables.
- Si vous faites tourner plusieurs serveurs mysqld avec les mêmes données sur un système qui ne gère pas bien les vérrous de fichiers (normalement gérés par le démon lockd ) ou que vous le faites avec −−skip−external−locking
- Vous avez un fichier de données ou d'index corrompu qui contient des données faussées ce qui amène mysqld à confusion.
- Vous avez trouvé un bogue dans le système de stockage des données. Cela parait impossible, mais sait−on jamais ? Dans ce cas, essayez de changer le type de fichier pour qu'il soit pris en charge par un autre gestionnaire de bases de données en utilisant ALTER TABLE sur une copie réparée de la table !

Parce qu'il est très difficile de savoir pourquoi quelque chose crashe, essayez d'abord de voir si les choses qui marchent pour les autres ne fonctionnent pas chez vous. Merci d'essayer les différentes choses suivantes :

- Utilisez mysqld −−log et essayez de déterminer à partir des informations du log si une requête spécifique fait planter le serveur. Plus de 95% de tous les bogues sont liés à une requête spécifique ! Normalement, c'est la dernière requête dans le fichier de log avant que MySQL n'ait redémarré. Le log général de requêtes . Si vous pouvez faire planter MySQL à plusieurs reprise avec une requête, même après avoir vérifié toutes les tables avant de l'exécuter, alors vous avez trouvé le bogue et vous devez faire un rapport de bogue pour nous en avertir ! Comment rapporter des bogues ou problèmes .
- Essayer d'effectuer une batterie de tests que nous pourrons utiliser pour reproduire le problème. Faire une batterie de tests lorsque vous faites face à un problème de table corrompue .
- Essayez d'exécuter le test inclus mysql−test et les benchmarks MySQL. Suites de tests MySQL . Ils devraient tester plutôt bien MySQL. Vous pouvez aussi ajouter ce code au benchmarks pour simuler votre application ! Les benchmarks peuvent être trouvés dans le répertoire bench dans la distribution des sources ou, pour une distribution binaire, dans le répertoire sql−bench de votre dossier d'installation MySQL.
- Essayez fork_test.pl et fork2_test.pl .
- Si vous configurez MySQL pour le débogage, il sera plus facile d'obtenir des informations à propos des erreurs possibles si quelque chose se passe mal. Reconfigurez MySQL avec l'option −−with−debug ou −−with−debug=full de configure puis recompilez. Déboguer un serveur MySQL .
- Configurer MySQL pour le débogage inclu un outil d'allocation de mémoire qui peut trouver quelques erreurs. Il fournit aussi beaucoup d'informations sur ce qui se passe.
- Avez−vous appliqué les derniers patches de votre système d'exploitation ?
- Utilisez l'option −−skip−external−locking de mysqld . Sur quelques systèmes, le gestionnaire des verrous lockd ne fonctionne pas convenablement; l'option −−skip−external−locking dit à mysqld de ne pas utiliser de pas utiliser de verrous externes. (Cela signifie que vous ne pouvez pas faire tourner deux serveurs mysqld sur les mêmes données et que vous devez faire attention si vous utilisez myisamchk , mais il peut être instructif d'essayer cette option comme test.)
- Avez−vous essayé mysqladmin −u root processlist lorsque mysqld semble fonctionner mais ne répond plus ? Quelquefois, mysqld n'est pas comateux, même si vous le croyez. Le problème peut−être que toutes les connexions sont utilisées, ou qu'il y a quelques problèmes avec les verrous internes. mysqladmin processlist devra normalement être en mesure d'effectuer une connexion même dans ce cas, et peut fournir des informations utiles à propos du nombre de connexions courantes et de leur status.
- Exécutez la commande mysqladmin −i 5 status ou mysqladmin −i 5 −r status ou dans une fenêtre séparée pour produire des statistiques pendant que vous exécutez vos autres requêtes.
- Essayez ce qui suit :
  - Démarrez mysqld à partir de gdb (ou d'un autre débogueur). Déboguer mysqld sous gdb .
  - Exécutez vos scripts de tests.
  - Affichez le traçage et les variables locales aux trois niveaux les plus bas. Avec gdb vous pouvez le faire avec les commandes suivantes lorsque mysqld a crashé à l'intérieur de gdb :

    ```
    backtrace
    info local
    up
    info local
    up
    info local
    ```

    Avec gdb vous pouvez aussi savoir quels threads existent avec info threads et en prendre un avec thread # , où # est l'identifiant du thread.
- Essayez de simuler votre application avec un script Perl pour forcer MySQL à crasher ou à avoir un comportement défectueux.
- Envoyez un rapport de bogue normal. Comment rapporter des bogues ou problèmes . Soyez le plus précis possible et donnez plus de détails que d'habitude. Puisque MySQL fonctionne pour beaucoup de personnes, il se peut que le crash résulte de quelque chose de spécifique à votre système (par exemple, une erreur liée à la particularité de vos librairies système).
- Si vous avez des problèmes avec des tables à lignes de longueurs dynamiques et que vous n'utilisez pas de colonnes BLOB/TEXT (mais seulement des colonnes VARCHAR ), vous pouvez essayer de changer tous les VARCHAR en CHAR avec ALTER TABLE . Cela forcera MySQL à utiliser des lignes de tailles fixes. Les lignes à tailles fixées prennent un peu plus d'espace, mais sont

plus tolérants aux corruptions !Le code courant des lignes dynamiques est utilisé chez MySQL AB depuis au moins 3 ans sans aucun problème, mais par nature, les lignes à longueur dynamique sont plus exposées aux erreurs, il est donc bon d'essayer ce qui précède pour voir si cela vous aide !

## 10.4.2 Comment réinitialiser un mot de passe Root oublié

Si vous n'avez jamais configuré un mot de passe `root` pour MySQL, le serveur n'en demendera jamais un pour toutes les connexions de cet utilisateur. Il est recommandé de toujours assigner un mot de passe à chaque utilisateur. Comment protéger MySQL contre les pirates .

Si vous avez configuré un mot de passe pour l'utilisateur root , mais que vous l'avez oublié, vous pouvez en choisir un nouveau en suivant la procédure suivante :

- Terminez le serveur mysqld en lui envoyant un kill (pas un kill −9 ). L'identifiant du processus est stocké dans un fichier .pid , qui est normalement situé dans le dossier des bases de données de MySQL :
```
shell> kill `cat /dossier−donnees−mysql/hote.pid`
```
  Vous devez être l'utiliateur Unix root ou l'utilisateur qui fait tourner mysqld pour pouvoir le faire.
- Redémarrez mysqld avec l'option −−skip−grant−tables .
- Choisissez un nouveau mot de passe avec la commande mysqladmin password :
```
shell> mysqladmin −u root password 'nouveaumotdepasse'
```
- Maintenant, vous pouvez soit stopper mysqld et le redémarrer normalement, ou bien juste charger les tables de privilèges avec :
```
shell> mysqladmin −h hote flush−privileges
```
- Après cela, vous devriez pouvoir vous connecter avec le nouveau mot de passe.

Vous pouvez aussi choisir le nouveau mot de passe en utilisant le client mysql :

- Stoppez et redémarrez mysqld avec l'option −−skip−grant−tables comme décrit plus haut.
- Connectez vous au serveur mysqld avec :
```
shell> mysql −u root mysql
```
- Exécutez la commande suivante dans le client mysql :
```
mysql> UPDATE user SET Password=PASSWORD('nouveaumotdepasse')
   −>             WHERE User='root';
mysql> FLUSH PRIVILEGES;
```
- Après cela, vous devriez pouvoir vous connecter avec le nouveau mot de passe.
- Vous pouvez maintenant stopper et redémarrer mysqld de façon normale.

## 10.4.3 Comment MySQL gère un disque plein

Lorsqu'il n'y a plus d'espace disque, MySQL fait ce qui suit :

- Il vérifie chaque minute pour voir s'il y a assez d'espace pour écrire la ligne courante. Si oui, il continue comme si rien ne s'était passé.
- Chaque 6 minutes, il ajoute un avertissement dans le fichier de log à propos de la condition du disque.

Pour contourner ce problème, vous pouvez effectuer les actions suivantes :

- Pour continuer, il suffit juste d'avoir assez d'espace disque pour insérer tous les enregistrements.
- Pour annuler le thread, vous devez lui envoyer un mysqladmin kill . Le thread sera annulé la prochaine fois qu'il vérifiera le disque (dans 1 minute).
- Notez que d'autres threads peuvent être en train d'attendre pour accéder à la table qui a causé le problème de disque plein. Si vous avez beaucoup de threads ``locked'', terminer le thread qui a causé la défaillance permettra aux autres threads de continuer.

Les exceptions pour le comportement suivant sont lorsque vous utilisez REPAIR ou OPTIMIZE ou lorsque les index sont créés dans un batch après l'exécution de LOAD DATA INFILE ou d'un ALTER TABLE .

Toutes les commandes précédentes risquent d'utiliser de gros fichiers temporaires, qui pourraient perturber le reste du système s'ils n'étaient pas supprimés. Si MySQL obtient une erreur de disque plein lors de l'exécution d'une des commandes précédentes, il effacera les gros fichiers temporaires et marquera la table comme crashée (à part pour `ALTER TABLE` , où l'ancienne table sera restaurée).

## 10.4.4 Où MySQL stocke les fichiers temporaires ?

MySQL utilise la valeur de la variable d'environnement `TMPDIR` comme chemin du dossier où stocker les fichiers temporaires. Si vous n'avez pas de variable `TMPDIR` , MySQL utilise alors le dossier système par défaut, qui est normalement `/tmp` ou `/usr/tmp` . Si le support qui contient votre dossier temporaire est trop petit, modifiez le script `safe_mysqld` pour configurer `TMPDIR` , et lui faire désigner un

dossier où vous aurez la place. Vous pouvez aussi configurer ce dossier avec l'option `--tmpdir` de `mysqld` .

MySQL crée tous les fichiers temporaires sous forme de fichier cachés. Cela garantit que les fichiers temporaires seront supprimés lorsque mysqld est terminé. L'inconvénient d'utiliser les fichiers cachés est que vous ne verrez pas que le dossier temporaire est gros, et qu'ils risquent de remplir votre dossier temporaire.

Lors des tris avec les clauses `ORDER BY` ou `GROUP BY` , MySQL utilise normalement deux dossiers temporaires. L'espace disque maximal nécessaire est :

```
 (taille de ce qui est trié + taille du pointeur de base)
* nombre de lignes trouvées
* 2
```

taille du pointeur de base vaut généralement 4, mais peut croître dans le futur pour les tables réellement grandes.

Pour certaines requêtes `SELECT` , MySQL crée aussi des tables temporaires SQL. Elles ne sont pas cachées, et portent un nom du type `SQL_*` .

ALTER TABLE crée une table temporaire dans le même dossier que la table originale.

## 10.4.5 Comment protéger ou changer le fichier socket /tmp/mysql.sock

Si vous avez un problème avec le fait que n'importe qui peut effacer le fichier de socket de communication MySQL `/tmp/mysql.sock` , vous pouvez, dans la plupart des versions d'Unix, protéger votre système de fichiers `/tmp` en lui assignant le bit `sticky` . Connectez vous en `root` et faite ce qui suit :

```
shell> chmod +t /tmp
```

Cela protégera votre système de fichiers /tmp de sorte que les fichiers ne pourront être effacés que par leurs propriétaires ou le super utilisateur ( root ).

Vous pouvez vérifier que le bit `sticky` est actif en exécutant `ls -ld /tmp` . Si le dernier bit de permission est `t` , il l'est.

Vous pouvez changer l'endroit où MySQL utilise / place le fichier de socket de la façon suivante :

- Spécifiez le chemin dans un fichier d'options globales ou locales. Par exemple, placez dans /etc/my.cnf :
```
 [client]
socket=chemin-vers-fichier-socket

 [mysqld]
socket=chemin-vers-fichier-socket
```
  Fichier d'options my.cnf .
- Spécifiez cela en ligne de commande à safe_mysqld et à la plupart des clients avec l'option `--socket=chemin-vers-fichier-socket` .
- Spécifiez le chemin vers la socket dans la variable d'environnement MYSQL_UNIX_PORT .
- Définissez le chemin avec l'option de configure `--with-unix-socket-path=chemin-vers-fichier-socket` . Les options habituelles de configure .

Vous pouvez vérifier que la socket fonctionne avec cette commande :

```
shell> mysqladmin --socket=/chemin/vers/socket version
```

## 10.4.6 Problèmes de fuseaux horaires

Si vous avez un problème avec SELECT NOW() qui retournerait des valeurs en GMT et non votre temps local, vous devez configurer la variable d'environnement TZ et la mettre sur votre fuseau horaire courant. Cela peut être fait pour l'environnement dans lequel le serveur fonctionne, par exemple, dans safe_mysqld ou mysql.server . Variables d'environnement MySQL .

# 10.5 Problèmes relatifs aux requêtes

## 10.5.1 Sensibilité à la casse dans les recherches

Par défaut, les recherchez de MySQL ne sont pas sensibles à la casse (cependant, il existe des jeux de caractères qui ne sont jamais insensibles à la casse, comme `czech` ).

Cela signifie que si vous recherchez avec nom_colonne LIKE 'a%' , vous aurez toutes les valeurs de la colonne qui commencent par un A ou un a . Si vous voulez que cette recherche soit sensible à la casse, utilisez par exemple INSTR(nom_colonne, "A")=1 pour vérifier un préfixe. Utilisez STRCMP(nom_colonne, "A") = 0 si la valeur de la colonne doit être exactement "A" .

Les opérations de comparaisons simples ( `>=, >, = , < , <=` , tri et groupement) sont basées sur la ``valeur de tri'' de chaque caractère. Les caractères avec la même valeur de tri (comme E, e et é) sont considérés comme le même caractère !

Dans les anciennes versions de MySQL les comparaisons avec LIKE étaient effectuées sur la majuscule de chaque caractère (E == e mais E <> é). Dans les nouvelles versions LIKE fonctionne comme les autres opérateurs de comparaison.

Si vous voulez qu'une colonne soit toujours traitée de façon sensible à la casse, déclarez là en tant que `BINARY` . Syntaxe de `CREATE TABLE` .

Si vous utilisez des donneés chinoises avec l'encodage big5, vous devez rendre toutes les colonnes de chaînes BINARY . Cela fonctionne car l'ordre de tri de l'encodage big5 est basé sur l'ordre des codes ASCII.

## 10.5.2 Problèmes avec l'utilisation des colonnes DATE

Le format d'une valeur de DATE est 'YYYY-MM-DD' . En accord avec ANSI SQL, aucun autre format n'est autorisé. Vous devez utiliser ce format dans les UPDATE et les clauses WHERE des requêtes SELECT . Par exemple :

```
mysql> SELECT * FROM nom_de_table WHERE date >= '1997-05-05';
```

MySQL convertit automatiquement une date en nombre si la date est utilisée dans un contexte numérique (et vice versa). Il est aussi assez intelligent pour permettre une forme ``relaxée'' lors des mises à jour et dans les clauses WHERE qui comparent une date et une colonne TIMESTAMP , DATE , ou DATETIME . (Forme relaxée signifie que n'importe quel caractère de ponctuation peut être utilisé en tant que séparateurs des parties. Par exemple, '1998-08-15' et '1998#08#15' sont équivalents.) MySQL peut convertir une chaîne ne contenant aucun séparateur (comme '19980815' ), en supposant qu'elle a un sens pour une date.La date spéciale '0000-00-00' peut être stockée et récupérée en tant que '0000-00-00'. Lors de l'utilisation d'une date '0000-00-00' avec MyODBC , elle sera automatiquement convertie en NULL à partir de la version 2.50.12 de MyODBC , car ODBC ne peut gérer ce type de dates.

Puisque MySQL effectue les conversions décrites plus haut, ce qui suit fonctionnera :

```
 mysql> INSERT INTO nom_de_table (idate) VALUES (19970505);
mysql> INSERT INTO nom_de_table (idate) VALUES ('19970505');
mysql> INSERT INTO nom_de_table (idate) VALUES ('97-05-05');
mysql> INSERT INTO nom_de_table (idate) VALUES ('1997.05.05');
mysql> INSERT INTO nom_de_table (idate) VALUES ('1997 05 05');
mysql> INSERT INTO nom_de_table (idate) VALUES ('0000-00-00');

 mysql> SELECT idate FROM nom_de_table WHERE idate >= '1997-05-05';
mysql> SELECT idate FROM nom_de_table WHERE idate >= 19970505;
mysql> SELECT MOD(idate,100) FROM nom_de_table WHERE idate >= 19970505;
mysql> SELECT idate FROM nom_de_table WHERE idate >= '19970505';
```

Toutefois, ce qui suit ne fonctionnera pas :

```
mysql> SELECT idate FROM nom_de_table WHERE STRCMP(idate,'19970505')=0;
```

STRCMP() est une fonction de chaînes de caractères, il convertit donc idate en une chaîne et effectue une comparaison de chaînes. Il ne convertit pas '19970505' en date et n'effectue donc pas de comparaison de dates.

Notez que MySQL vérifie très peu l'intégrité des dates. Si vous stockez une date erronée, comme `'1998-2-31'` , la date erronée sera enregistrée.

Vu que MySQL compresse les dates pour les stocker, il ne peut stocker tout format donné car il risquerait de ne pas correspondre au tampon de résultat. Les règles d'acceptations de dates sont :

- Si MySQL peut enregistrer et récupérer une date donnée, la date erronée est acceptée pour les colonnes DATE et DATETIME
- Toutes les valeurs de ours comprises entre 0 et 31 sont acceptées. Cela est fort convenable pour les applications web où vous demandez l'année, mois et jour dans trois champs textes (ou liste déroulantes) différents.
- Le champ jour ou mois peut être un zéro. Cela est convenable si vous voulez enregistrer un anniversaire dans une colonne DATE et que vous ne connaissez qu'une partie de la date.

Si la date ne peut être convertie en une valeur raisonnable, un 0 est inséré dans le champ DATE , il sera récupéré en tant que

0000-00-00 . Cela est une solution rapide et convenue vu que nous considérons que la responsabilité de la base de données est de récupérer la même date que vous aviez stocké (même si la date n'est pas correcte). Nous pensons que c'est à l'application de vérifier les dates, et non au serveur de le faire.

## 10.5.3 Problèmes avec les valeurs NULL

Le concept de la valeur `NULL` est une source de confusions pour les débutants en SQL, qui pensent souvent que `NULL` est la même chose qu'une chaîne de caractères vide `""` . Ce n'est pas le cas ! Par exemple, les deux requêtes suivantes sont complètement différentes :

```
mysql> INSERT INTO ma_table (telephone) VALUES (NULL);
mysql> INSERT INTO ma_table (telephone) VALUES ("");
```

Les deux requêtes insèrent des valeurs dans la colonne telephone , mais la première insère une valeur NULL et la seconde insère une chaîne vide. La signification de la première peut être ``le numéro de téléphone est inconnu'' et la seconde peut être considérée comme ``elle n'a pas de téléphone''.

En SQL, la valeur `NULL` est toujours false en comparaison à n'importe quelle autre valeur, même `NULL` . Une expression contenant `NULL` produit toujours un résultat `NULL` sauf si une indication contraire est présente dans la documentation des opérateurs et des fonctions impliquées dans l'expression. Toutes les colonnes de l'exemple suivant retournent `NULL` :

```
mysql> SELECT NULL,1+NULL,CONCAT('Invisible',NULL);
```

Si vous voulez trouver les colonnes dont la valeur est NULL , vous ne pouvez pas utiliser le test =NULL . La requête suivante ne retourne aucune ligne car expr = NULL est FALSE, pour n'importe quelle expression :

```
mysql> SELECT * FROM ma_table WHERE telephone = NULL;
```

Pour trouver les valeurs NULL , vous devez utiliser le test IS NULL . Ce qui suit montre comment trouver les numéros de téléphone NULL et les numéros vides :

```
mysql> SELECT * FROM ma_table WHERE telephone IS NULL;
mysql> SELECT * FROM ma_table WHERE telephone = "";
```

Notez que vous ne pouvez ajouter d'index qu'aux colonnes pouvant avoir la valeur NULL si vous utilisez la version 3.23.2 de MySQL ou plus récente avec des tables de type MyISAM ou InnoDB . Dans les versions précédentes et avec les autres types, vous devez déclarer de telles colonnes NOT NULL . Cela signifie aussi que vous ne pouvez pas insérer NULL dans les colonnes indexées.

Lors de la lecture de données avec `LOAD DATA INFILE` , les colonnes vides sont interprétées en tant que `''` . Si vous voulez une valeur `NULL` dans une colonne, vous devez utiliser `\N` dans le fichier. Le mot littéral `'NULL'` peut aussi être utilisé dans certaines circonstances. Syntaxe des `LOAD DATA INFILE` .

Lors de l'utilisation de ORDER BY , les valeurs NULL sont présentées en premier. Si vous triez dans l'ordre décroissant en utilisant DESC , les valeurs NULL sont présentées en dernier. Lors de l'utilisation de GROUP BY , toutes les valeurs NULL sont considérées comme égales.

Pour mieux gérer les valeurs `NULL` , vous pouvez utiliser les opérateurs `IS NULL` et `IS NOT NULL` et la fonction `IFNULL()` .

Pour certains types de colonnes, les valeurs NULL sont traitées spécialement. Si vous insérez NULL dans la première colonne TIMESTAMP d'une table, la date et le temps courants sont insérés. Si vous insérez NULL dans une colonne AUTO_INCREMENT , le nombre suivant de la séquence sera inséré.

## 10.5.4 Problèmes avec les alias

Vous pouvez utiliser un alias pour vous référer à une colonne dans une clause `GROUP BY` , `ORDER BY` , ou `HAVING` . Les alias peuvent aussi être utilisés pour donner de meilleurs noms aux colonnes :

```
SELECT SQRT(a*b) as rt FROM nom_de_table GROUP BY rt HAVING rt > 0;
SELECT id,COUNT(*) AS cnt FROM nom_de_table GROUP BY id HAVING cnt > 0;
SELECT id AS "Identité du client" FROM nom_de_table;
```

Notez que ANSI SQL ne vous permet pas de vous référer à un alias dans une clause WHERE . Il en est ainsi car lorsque le code de WHERE est exécuté, la valeur de la colonne ne peut pas encore être déterminée. Par exemple, la requête suivante est **illégale** :

```
SELECT id,COUNT(*) AS cnt FROM nom_de_table WHERE cnt > 0 GROUP BY id;
```

La clause WHERE est exécutée pour savoir quelles lignes devraient être incluses dans la partie GROUP BY tandis que HAVING est utilisé pour décider quelles lignes du jeu de résultats doivent être utilisées.

## 10.5.5 Effacer des lignes de tables reliées

Comme MySQL ne supporte encore ni les sous−requêtes, ni l'utilisation de plusieurs tables dans une requête `DELETE` (avant la version 4.0), vous devez utiliser l'approche suivante pour effacer des lignes de deux tables reliées :

- Sélectionnez ( SELECT ) les lignes de la table principale en vous basant sur une condition WHERE .
- Effacez ( DELETE ) les lignes de la table principale en vous basant sur la même condition.
- DELETE FROM table_liée WHERE colonne_liée IN (lignes_séléctionnées) .

Si le nombre total des caractères dans la requête avec la `colonne_liée` est supérieur à 1,048,576 (la valeur par défaut est `max_allowed_packet` , vous devez la découper en parties plus petites et exécuter plusieurs `DELETE` . Vous obtiendrez probablement les suppressions les plus rapides en n'effaçant que 100−1000 `colonne_liée` par requête si `colonne_liée` est un index. Si ce n'est pas un index, la vitesse est indépendante du nombre d'arguments dans la clause `IN` .

## 10.5.6 Résoudre les problèmes des lignes non retournées

Si vous avez une requête complexe avec beaucoup de tables et qu'elle ne retourne aucun résultat, vous devez suivre la procédure suivante pour trouver ce qui cloche dans votre requête :

- Testez la requête avec EXPLAIN et vérifiez si vous trouvez quelque chose qui vous parait fausse. Syntaxe de EXPLAIN (Obtenir des informations SELECT ) .
- Ne séléctionnez que les champs que vous utilisez dans la clause WHERE :
- Enlevez une table à la fois de la requête jusqu'à ce qu'elle retourne quelques lignes. Si les tables sont grandes, il est bon d'utiliser LIMIT 10 dans la requête.
- Exécutez un SELECT pour les colonnes qui auraient du trouver des lignes dans la dernière table supprimée de la requête.
- Si vous comparez des colonnes FLOAT ou DOUBLE avec des nombres à virgule, vous ne pouvez pas utiliser '=' . C'est un problème commun à la plupart des langages de programmation car les valeurs à virgules flottantes ne sont pas des valeurs exactes. Dans le plupart des cas, changer la colonnes FLOAT en DOUBLE corrigera cela. Problèmes lors de la comparaison de nombres à virgules flottantes .
- si vous ne pouvez toujours pas trouver ce qui ne va pas, créez un test minimal pouvant être exécuté avec mysql test < query.sql montrant votre problème. Vous pouvez créer un fichier de test avec mysqldump −−quick base tables > query.sql . Editez le fichier et supprimez quelques lignes d'insertions (s'il y'en a trop), et ajoutez votre requête de séléction à la fin du fichier.

  Vérifiez que vous avez encore le problème en faisant :

```
 shell> mysqladmin create test2
shell> mysql test2 < query.sql
```

  Envoyez le fichier de test, en utilisant mysqlbug , à mysql@lists.mysql.com .

## 10.5.7 Problèmes de comparaisons avec nombres à virgule flottante

Les nombres à virgule flottante portent souvent à confusion, car ils ne sont pas enregistrés en tant que valeurs exactes dans l'architecture de l'ordinateur. Ce qu'on voit à l'écran n'est souvent pas la valeur exacte du nombre.

Ce discours s'applique aux champs de types `FLOAT` , `DOUBLE` et `DECIMAL` .

```
 CREATE TABLE t1 (i INT, d1 DECIMAL(9,2), d2 DECIMAL(9,2));
INSERT INTO t1 VALUES (1, 101.40, 21.40), (1, −80.00, 0.00),
(2, 0.00, 0.00), (2, −13.20, 0.00), (2, 59.60, 46.40),
(2, 30.40, 30.40), (3, 37.00, 7.40), (3, −29.60, 0.00),
(4, 60.00, 15.40), (4, −10.60, 0.00), (4, −34.00, 0.00),
(5, 33.00, 0.00), (5, −25.80, 0.00), (5, 0.00, 7.20),
(6, 0.00, 0.00), (6, −51.40, 0.00);

 mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b
    −> FROM t1 GROUP BY i HAVING a <> b;
+−−−−−−+−−−−−−−−+−−−−−−−+
| i    | a      | b     |
+−−−−−−+−−−−−−−−+−−−−−−−+
|    1 |  21.40 | 21.40 |
```

```
|    2 |  76.80 | 76.80 |
|    3 |   7.40 |  7.40 |
|    4 |  15.40 | 15.40 |
|    5 |   7.20 |  7.20 |
|    6 | −51.40 |  0.00 |
+------+--------+-------+
```

Le résultat est correcte. Même si les cinq premiers enregistrements ne devraient pas passer le test à priori, ils le font sûrement car la différence entre les nombres se situe plus loin que les décimales, ou que cela dépend de l'architecture de l'ordinateur.

Le problème ne peut être résolu en utilisant ROUND() (ou une fonction similaire), car le résultat est encore un nombre à virgule flottante. Exemple :

```
mysql> SELECT i, ROUND(SUM(d1), 2) AS a, ROUND(SUM(d2), 2) AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+-------+
| i    | a      | b     |
+------+--------+-------+
|    1 |  21.40 | 21.40 |
|    2 |  76.80 | 76.80 |
|    3 |   7.40 |  7.40 |
|    4 |  15.40 | 15.40 |
|    5 |   7.20 |  7.20 |
|    6 | −51.40 |  0.00 |
+------+--------+-------+
```

Voilà ce à quoi les nombres dans le champ 'a' ressemblent :

```
mysql> SELECT i, ROUND(SUM(d1), 2)*1.0000000000000000 AS a,
    -> ROUND(SUM(d2), 2) AS b FROM t1 GROUP BY i HAVING a <> b;
+------+----------------------+-------+
| i    | a                    | b     |
+------+----------------------+-------+
|    1 |  21.3999999999999986 | 21.40 |
|    2 |  76.7999999999999972 | 76.80 |
|    3 |   7.4000000000000004 |  7.40 |
|    4 |  15.4000000000000004 | 15.40 |
|    5 |   7.2000000000000002 |  7.20 |
|    6 | −51.3999999999999986 |  0.00 |
+------+----------------------+-------+
```

Selon l'architecture de votre ordinateur, vous pouvez obtenir ou non les mêmes résultats. Chaque CPU peut évaluer les nombres à virgule flottante d'une manière différente. Par exemple, sur des machines vous pouvez obtenir les résultats 'correctes' en multipliant les deux arguments par 1, l'exemple suivant illustre cela.

## ATTENTION : NE FAITES JAMAIS CONFIANCE A CETTE METHODE DANS VOS APPLICATIONS, C'EST UN EXEMPLE DE MAUVAISE METHODES !!!

```
mysql> SELECT i, ROUND(SUM(d1), 2)*1 AS a, ROUND(SUM(d2), 2)*1 AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+------+
| i    | a      | b    |
+------+--------+------+
|    6 | −51.40 | 0.00 |
+------+--------+------+
```

La raison pour laquelle l'exemple précédent semble fonctionner est que sur la machine en particulier où le test a été effectué, l'arithmétique des nombres à virgule flottante du CPU arrondi les nombres à la même valeur, mais il n'y a aucune règle stipulant qu'un CPU doit faire cela, on ne peut donc pas s'y fier.

La façon correcte d'effectuer des comparaisons de nombre à virgule flottante est d'en premier lieu décider du degré de tolérance voulu entre les nombres puis d'effectuer la comparaison selon le nombre de tolérance. Par exemple, si nous décidons que les nombres à virgule flottante sont considérés comme égaux si ils ont la même précision au dix−millième près (0.0001), la comparaison ressemblera à cela :

```
mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a − b) > 0.0001;
+------+--------+------+
| i    | a      | b    |
+------+--------+------+
|    6 | −51.40 | 0.00 |
+------+--------+------+
1 row in set (0.00 sec)
```

Et vice versa, si vous voulez obtenir les lignes où les nombres sont les mêmes, le test sera :

```
mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a - b) < 0.0001;
+------+-------+-------+
| i    | a     | b     |
+------+-------+-------+
|    1 | 21.40 | 21.40 |
|    2 | 76.80 | 76.80 |
|    3 |  7.40 |  7.40 |
|    4 | 15.40 | 15.40 |
|    5 |  7.20 |  7.20 |
+------+-------+-------+
```

# 10.6 Questions relatives aux définitions de tables

## 10.6.1 Problèmes avec ALTER TABLE .

ALTER TABLE change une table avec le jeu de caractères courant. Si durant l'exécution d' ALTER TABLE vous obtenez une erreur de clef dupliquée, alors la cause est soit que le nouveau jeu de caractères interprète deux clefs à la même valeur ou que la table est corrompue, au quel cas vous devez exécuter REPAIR TABLE sur la table.

Si ALTER TABLE se termine avec une erreur de ce genre :

```
Error on rename of './database/name.frm' to './database/B-a.frm' (Errcode: 17)
```

le problème est peut-être que MySQL a crashé lors d'un précédent appel à ALTER TABLE et qu'il y a une ancienne table nommée A-quelquechose ou B-quelquechose qui subsiste. Dans ce cas, déplacez-vous dans le dossier des données MySQL et effacez tout les fichiers dont les noms commencent par A- ou B- . (Vous voudrez peut-être les déplacer autre part plutôt que de les effacer.)

ALTER TABLE fonctionne de la façon suivante :

- Crée une nouvelle table nommée A-xxx avec les changements voulus.
- Toutes les lignes de l'ancienne table sont copiées dans A-xxx .
- L'ancienne table est renommée B-xxx .
- A-xxx est renommée avec le nom de votre ancienne table.
- B-xxx est supprimée.

Si quelque chose se passe mal durant l'opération de changement de nom, MySQL essaye d'annuler les changements. Si quelque chose de grave se passe (cela ne devrait jamais arriver bien sûr), MySQL peut laisser l'ancienne table en tant que B-xxx , mais un simple renommage au niveau système devrait restaurer vos données.

## 10.6.2 Comment changer l'ordre des colonnes dans une table

Le but de SQL est de séparer l'application du format de stockage des données. Vous devriez toujours spécifier l'ordre dans lequel vous voulez récupérer vos données. Par exemple :

```
SELECT nom_colonne1, nom_colonne2, nom_colonne3 FROM nom_de_table;
```

retournera les colonnes dans l'ordre nom_colonne1 , nom_colonne2 , nom_colonne3 , tandis que :

```
SELECT nom_colonne1, nom_colonne3, nom_colonne2 FROM nom_de_table;
```

retournera les colonnes dans l'ordre nom_colonne1 , nom_colonne3 , nom_colonne2 .Vous ne devez **jamais** , dans une application, utiliser SELECT * et récupérer l'ordre des colonnes en vous basant sur leurs positions, car l'ordre dans lequel elles sont retournées ne peut-être garantit tout le temps. Une simple modification dans votre base de données peut endommager gravement vos applications.

Si vous voulez changer l'ordre de vos colonnes, vous pouvez faire ce qui suit :

- Créer une nouvelle table avec les colonnes dans le bon ordre.
- Exécuter INSERT INTO nouvelle_table SELECT champs-dans-nouvel-ordre FROM ancienne_table .
- Effacer ou renommer ancienne_table .
- ALTER TABLE nouvelle_table RENAME ancienne_table .

## 10.6.3 Problèmes avec les tables temporaires

Voici une liste des limitations avec TEMPORARY TABLES .

- Une table temporaire ne peut être que du type HEAP , ISAM , MyISAM ou InnoDB .
- Vous ne pouvez utiliser une table plus d'une fois dans une requête. Par exemple, ce qui suit ne marche pas :

```
mysql> SELECT * FROM temporary_table, temporary_table AS t2;
```

Nous allons corriger ce qui précède dans la version 4.0.

- Vous ne pouvez utiliser RENAME sur une table TEMPORARY . Notez que ALTER TABLE nom_origine RENAME nouveau_nom fonctionne !Nous allons corriger ce qui précède dans la version 4.0.

# 11 Contributions

De nombreux utilisateurs de MySQL ont contribués à ces utilitaires **très** pratiques.

Une liste de quelques logiciels disponibles sur le site de MySQL (ou des mirroirs) est présentée ici.

Vous pouvez aussi visiter notre liste des logiciels liés à MySQL à l'adresse suivante : http://www.mysql.com/portal/software/ . Ce site communautaire facilite vos propres contributions!

Si vous voulez compiler MySQL avec les interfaces Perl DBI / DBD , vous devriez commencer par télécharger les fichiers de Data−Dumper , DBI et Msql−Mysql−modules , et les installer. Commentaires sur l'installation de Perl .

Note : les programmes listés ci−dessous peuvent être librement téléchargés et utilisés. Ils portent le copyright de leurs propriétaires respectifs. Reportez−vous aux documentations de chacun de ces programme pour avoir plus de détails sur les licences. MySQL AB n'assume aucune responsabilité quant à l'exactitude des informations de cette page, ou de l'utilisation appropriée des logiciels qui y sont listés.

**APIs**

- Perl Modules
  - http://www.mysql.com/Downloads/Contrib/KAMXbase1.2.tar.gz Conversion de fichier .dbf en tables MySQL et réciproquement. Le module Perl a été écrit par Pratap Pereira pereira@ee.eng.ohio−state.edu , et amélioré par Kevin A. McGrail kmcgrail@digital1.peregrinehw.com . Ce convertisseur supporte les champs MEMO .
  - http://www.mysql.com/Downloads/Contrib/HandySQL−1.1.tar.gz HandySQL est un module d'accès à MySQL. Il offre une interface en C, intégrée à Perl, qui est 20% plus rapide que DBI.
- OLEDB
  - http://www.mysql.com/Downloads/Win32/MyOLEDB3.exe Installeur MyOLEDB 3.0 par SWSoft.
  - http://www.mysql.com/Downloads/Win32/mysql−oledb−3.0.0.zip Source de MyOLEDB 3.0.
  - http://www.mysql.com/Downloads/Win32/MySamples.zip Exemples et documentation de MyOLEDB.
  - http://www.mysql.com/Downloads/Win32/MyOLEDB.chm Fichiers d'aide de MyOLEDB.
  - http://www.mysql.com/Downloads/Win32/libmyodbc.zip Librairie MyODBC statique, utilisée pour compiler MyOLEDB. Basée sur du code MyODBC.
- C++
  - http://www.mysql.com/Downloads/Contrib/mysql−c++−0.02.tar.gz Librairie MySQL C++. Par Roland Haenel, rh@ginster.net .
  - http://www.mysql.com/Downloads/Contrib/MyDAO.tar.gz API MySQL C++. Par Satish spitfire@pn3.vsnl.net.in . Inspiré de la librairie de Roland Haenel's C++ et de la librairie MyC de Ed Carp.
  - http://www.mysql.com/products/mysql++/ API MySQL C++ (plus qu'une simple couche supplémentaire). Concept original de kevina@clark.net . Maintenant géré par Sinisa chez MySQL AB.
  - http://nelsonjr.homepage.com/NJrAPI/ Une couche d'abastraction de base de données C++ qui supporte MySQL.
- Delphi
  - http://www.mysql.com/Downloads/Contrib/DelphiMySQL2.zip Interface Delphi avec libmysql.dll , par bsilva@umesd.k12.or.us .
  - http://www.mysql.com/Downloads/Contrib/Udmysql.pas Une interface pour libmysql.dll, à utiliser avec Delphi. Par Reiner Sombrowsky.
  - http://www.fichtner.net/delphi/mysql.delphi.phtml Une interface Delphi avec MySQL, avec le code source. Par Matthias Fichtner.
  - http://www.productivity.org/projects/tmysql/ TmySQL , une librairie pour utiliser MySQL avec Delphi.
  - http://sourceforge.net/projects/zeoslib/ La librairie Zeos est un jeu de données et de composants de librairie pour MySql, PostgreSql, Interbase, MS SQL, Oracle, DB/2. Elle inclut aussi des outils de développement tel que Database Explorer et Database Designer.
  - http://www.mysql.com/Downloads/Contrib/Win32/SBMySQL50Share.exe Composants Shareware Delphi 5 pour MySQL.
- http://www.mysql.com/Downloads/Contrib/mysql−ruby−2.2.0.tar.gz Module MySQL pour Ruby. Par TOMITA Masahiro tommy@tmtm.org Ruby est un langage interprété orienté objet ( http://www.netlab.co.jp/ruby/ ).
- http://www.mysql.com/Downloads/Contrib/JdmMysqlDriver−0.1.0.tar.gz Un pilote VisualWorks 3.0 Smalltalk pour MySQL. Par joshmiller@earthlink.net .
- http://www.mysql.com/Downloads/Contrib/Db.py Module Python avec cache. Par gandalf@rosmail.com .
- http://www.mysql.com/Downloads/Contrib/MySQLmodule−1.4.tar.gz Interface Python pour MySQL. Par Joseph Skinner joe@earthlight.co.nz . Modifié par Joerg Senekowitsch senekow@ibm.net .
- http://www.mysql.com/Downloads/Contrib/MySQL−python−0.3.0.tar.gz MySQLdb Python est une interface avec MySQL compatible DB−API v2.0. Les transactions sont supportées si le serveur et les tables les supportent. Elle est compatible avec les threads, et dispose d'une interface de compatibilité avec les anciens codes qui sont écrit avec l'interface MySQLmodule, aujourd'hui abandonnée.
- http://www.mysql.com/Downloads/Contrib/mysql_mex_12.tar.gz Un programme d'interface pour Matlab de MathWorks. L'interface est faîte par Kimmo Uutela et John Fisher (pas par Mathworks). Vérifiez http://boojum.hut.fi/~kuutela/mysqlmex.html pour plus d'informations.
- http://www.mysql.com/Downloads/Contrib/mysqltcl−1.53.tar.gz Interface Tcl pour MySQL. Basée sur msqltcl−1.50.tar.gz . Pour la version 2.0 et plus d'informations, voyez http://www.xdobry.de/mysqltcl/ .
- http://www.mysql.com/Downloads/Contrib/MyC−0.1.tar.gz Une API proche du Visual Basic, par Ed Carp.
- http://www.mysql.com/Downloads/Contrib/Vdb−dflts−2.1.tar.gz Ceci est la nouvelle version d'un jeu de librairies utilitaires, qui fournissent une interface générique aux moteurs de bases de données, pour que votre application soit entièrement trois tiers. L'avantage est que vous pouvez facilement passer d'un serveur à l'autre en modifiant simplement un fichier, sans toucher à votre application. Par damian@cablenet.net .
- http://www.mysql.com/Downloads/Contrib/DbFramework−1.10.tar.gz DbFramework est une collection de classes pour interagir avec les bases MySQL. Les classes sont inspirées par le CDIF Data Model Subject Area. Par Paul Sharpe paul@miraclefish.com .
- http://www.mysql.com/Downloads/Contrib/pike−mysql−1.4.tar.gz Module MySQL pour pike. A utiliser avec le serveur web Roxen.
- http://www.mysql.com/Downloads/Contrib/squile.tar.gz Module pour guile qui permet à guile de travailler avec les serveurs SQL. Par Hal Roberts.
- http://www.mysql.com/Downloads/Contrib/stk−mysql.tar.gz Interface pour Stk. Stk est les widgets Tk qui utilise Scheme au lieu de Tcl. Par Terry Jones.
- http://www.mysql.com/Downloads/Contrib/eiffel−wrapper−1.0.tar.gz Module Eiffel par Michael Ravits.
- http://www.mysql.com/Downloads/Contrib/SQLmy0.06.tgz FlagShip Replaceable Database Driver (RDD) pour MySQL. Par Alejandro Fernandez Herrero. Le site de Flagship RDD est http://www.fship.com/rdds.html .
- http://www.mysql.com/Downloads/Contrib/mydsn−1.0.zip Sources et exécutables pour mydsn.dll . mydsn doit être utilisée pour compiler et supprimer le registre DSN pour le pilote MyODBC, pour les applications Coldfusion. Par Miguel Angel Solórzano.

- http://www.mysql.com/Downloads/Contrib/MySQL-ADA95_API.zip Une interface ADA95 avec l'API MySQL . Par Francois Fabien.
- http://www.mysql.com/Downloads/Contrib/MyTool-DLL_for_VB_and_MySQL.zip Une DLL avec l'API C MySQL pour Visual Basic. Par Ken Menzel kenm@icarz.com .
- http://www.mysql.com/Downloads/Contrib/MYSQLX.EXE MySQL ActiveX Object pour accéder directement au serveur MySQL depuis un serveur IIS/ASP, VB, VC++ en évitant les méthodes ODBC plus lente. Complètement adaptable, multi-threadé, avec support complet des types de données MySQL (version 2001.1.1). Par SciBit http://www.scibit.com/ .
- http://www.fastflow.it/mylua/ MyLUA home page; comment utiliser le langagge LUA pour écrire des PROCEDURE MySQL qui peuvent être exécuté.
    - http://www.mysql.com/Downloads/Contrib/lua-4.0.tar.gz LUA 4.0
    - http://www.mysql.com/Downloads/Contrib/mylua-3.23.32.1.tar.gz Patch pour MySQL 3.23.32 et LUA 4.0. Par Cristian Giussani.
- http://www.mysql.com/Downloads/Contrib/patched_myodbc.zip Patch (pour le support Omniform 4.0) pour le pilote MyODBC. Par Thomas Thaele tthaele@papenmeier.de

**Convertisseurs**

- http://www.mysql.com/Downloads/Contrib/mssql2mysql.txt Convertisseur depuis MS-SQL vers MySQL. Par Michael Kofler. La page de mssql2mysql est http://www.kofler.cc/mysql/mssql2mysql.html .
- http://www.mysql.com/Downloads/Contrib/dbf2mysql-1.14.tar.gz Convertit des fichiers .dbf en tables MySQL et vice-versa. Par Maarten Boekhold ( boekhold@cindy.et.tudelft.nl ), William Volkman et Michael Widenius. Ce convertisseur inclut le support rudimentaire des champs MEMO.
- http://www.mysql.com/Downloads/Contrib/dbf2mysql-1.13.tgz Convertisseur entre les formats .dbf et MySQL. Par Maarten Boekhold, boekhold@cindy.et.tudelft.nl et Michael Widenius. Ce convertisseur ne peut pas gérer les champs MEMO.
- http://www.mysql.com/Downloads/Contrib/dbf2mysql.zip Convertit les fichiers FoxPro .dbf en tables MySQL sous Windows. Par Alexander Eltsyn, ae@nica.ru ou ae@usa.net .
- http://www.mysql.com/Downloads/Contrib/dbf2sql.zip Programme court et simple qui vous aide à transporter les données depuis les tables foxpro vers les tables MySQL. Par Danko Josic.
- http://www.mysql.com/Downloads/Contrib/dump2h-1.20.gz Convertit le résultat de mysqldump en fichier d'entête C. Par Harry Brueckner, brueckner@mail.respublica.de .
- http://www.mysql.com/Downloads/Contrib/exportsql.txt Un script similaire à access_to_mysql.txt , hormis le fait que celui-ci est complètement configurable, gère mieux les conversions de type (y compris les champs TIMESTAMP ), fournit des alertes et des suggestions lors de la conversion, protège **tous** les caractères spéciaux dans le texte les données binaires, etc. Il va aussi convertir les tables mSQL v1 et v2, et il est gratuit pour tous. Voyez http://www.cynergi.net/exportsql/ pour la dernière version. Par Pedro Freire, support@cynergi.net . Note : ne fonctionne pas avec Access2!
- http://www.mysql.com/Downloads/Contrib/access_to_mysql.txt Copiez cette fonction dans un module Access qui a accès aux tables que vous voulez exporter. Voyez aussi exportsql . Par Brian Andrews. Note: ne fonctionne pas avec Access2!
- http://www.mysql.com/Downloads/Contrib/importsql.txt Un script qui fait le contraire de exportsql.txt . C'est à dire qu'il importe les données depuis MySQL dans la base Access via ODBC. C'est très pratique lorsqu'il est compbiné avec exportsql, car il permet d'utiliser Access pour la conception de la base, et la synchronise avec un serveur MySQL. Gratuit. VOyez http://www.netdive.com/freebies/importsql/ pour les mises à jour. Créé par Laurent Bossavit of NetDIVE. **Note** : ne fonctionne pas avec Access2!
- http://www.mysql.com/Downloads/Contrib/mdb2sql.bas Convertisseur depuis Access97 vers MySQL par Moshe Gurvich.
- http://www.mysql.com/Downloads/Contrib/msql2mysqlWrapper-1.0.tgz Une interface C de mSQL vers MySQL. Par alfred@sb.net
- http://www.mysql.com/Downloads/Contrib/sqlconv.pl Un script simple qui peut être utilisé pouar copier des champs d'une table MySQL vers une autre, en masse. En bref, vous pouvez exécuter mysqldump et le rediriger vers le script sqlconv.pl via un pipe. Le script va analyser le résultat de mysqldump et va réarranger les champs pour qu'ils puissent être réinséré dans la nouvelle table. Un exemple d'utilisation est le cas ou vous voulez créer une nouvelle table à partir d'un site sur lequel vous travaillez, mais que cette table est juste un petit peut différente (au niveau de l'ordre des champs, etc.). Par Steve Shreeve.
- http://www.mysql.com/Downloads/Contrib/oracledump Programme Perl pour convertir des bases Oracle vers MySQL. Il a le même format que mysqldump . Par Johan Andersson.
- http://www.mysql.com/Downloads/Contrib/excel2mysql Programme Perl pour importer des feuilles Excel dans MySQL. Par Stephen Hurd shurd@sk.sympatico.ca
- http://www.mysql.com/Downloads/Contrib/T2S_100.ZIP . Programme Windows qui convertit des fichiers texte en table MySQL. Par Asaf Azulay.

**Utilitaire**

- http://worldcommunity.com/opensource/utilities/mysql_backup.html MySQL Backup est un script de sauvegarde pour MySQL. Par Peter F. Brown.
- http://www.mysql.com/Downloads/Contrib/mysql_watchdog.pl Surveille le démon MySQL contre les blocages. Par Yermo Lamers, yml@yml.com .
- http://www.mysql.com/Downloads/Contrib/mysql_structure_dumper.tar.gz
- http://www.mysql.com/Downloads/Contrib/mysql_structure_dumper.tgz Affiche la structure de toutes les tables dans une base. Par Thomas Wana.
- http://www.mysql.com/Downloads/Contrib/mysqlsync . Un script Perl pour synchroniser des tables MySQL distantes, avec une copie centrale.Par Mark Jeftovic. markjr@easydns.com .
- http://www.mysql.com/Downloads/Contrib/MySQLTutor-0.2.tar.gz . MySQLTutor. Un tutoriel MySQL pour les débutants.
- http://www.mysql.com/Downloads/Contrib/MySQLDB.zip
- http://www.mysql.com/Downloads/Contrib/MySQLDB-readme.html . Une librairie COM pour MySQL par Alok Singh.
- http://www.mysql.com/Downloads/Contrib/mysql_replicate.pl Un programme Perl qui gère les réplications. Par elble@icculus.nsg.nwu.edu
- http://www.mysql.com/Downloads/Contrib/DBIx-TextIndex-0.02.tar.gz Un script Perl qui utilise l'indexation inversse pour gérer les recherches dans les grands textes. Par Daniel Koch.
- http://www.mysql.com/Downloads/Contrib/dbcheck Un script Perl qui installe des copies de sauvegarde, et exécute isamchk dessus. Par Elizabeth.
- http://www.mysql.com/Downloads/Contrib/mybackup .
- http://www.mswanson.com/mybackup (mybackup home page) Surcouche de mysqldump qui sauvegarde toutes les bases. Par Marc Swanson.
- http://www.mysql.com/Downloads/Contrib/mdu.pl.gz Affiche la place prise par une base MySQL.

# 12 Crédits

Cet annexe liste les développeurs, contributeurs, et supporters qui ont fait de MySQL est ce qu'il est aujourd'hui.

## 12.1 Développeurs chez MySQL AB

Voici les développeurs qui sont employés par MySQL AB pour travailler sur la base de données MySQL , grosso modo dans leur ordre d'embauche. Pour chaque développeur, vous trouverez une petite liste de leur tâches et de leur responsabilités, ainsi que leur réalisations. Tous les développeurs sont impliqués dans le support.

*Michael (Monty) Widenius*

Chef d'équipe et auteur principal du serveur MySQL ( mysqld ).
Nouvelles fonctions dans la librairie de chaînes.
La majorité de la librairie mysys .
Les librairies ISAM et MyISAM (les fichiers d'index B-tree avec compression d'index et différentes formats d'enregistrement).
La librairie HEAP . Un système de gestion des tables en mémoire avec un hashage dynamique efficace.
Utilisé depuis 1980 et publié en 1984.
Le programme replace (jetez y un oeil c'est carrément **COOL** !).
MyODBC , le pilote ODBC de Windows95.
La correction de bogues avec les MIT-pthreads pour qu'ils fonctionnent avec MySQL. Et aussi l'Unireg, une application à base de curses avec de nombreuses utilisations.
Le port d'outils mSQL comme msqlperl , DBD / DBI et DB2mysql .
La majorité de crash-me et les fondations des tests de performances de MySQL.

*David Axmark*

Auteur original du **manuel de référence** , incluent les améliorations de texi2html .
La mise à jour automatique du site web depuis le manuel.
Le support initial de Autoconf, Automake et Libtool.
Les licences.
Des parties de tous les fichiers textes (de nos jours, uniquement le README est encore présent. le reste est dans le manuel).
Nombreux tests des nouvelles fonctionnalités.
Notre expert légal des logiciels libres.
Le responsable des listes de diffusion (qui n'a jamais le temps de le faire correctement...).
Notre code initial pour le port (bientôt plus de 10 ans). De nos jours, seules des parties de mysys restent.
La personne que Monty appelle au milieu de la nuit lorsqu'il a réussi a faire fonctionner la nouvelle fonctionnalité!
Le chef "Open Source" (Relations avec la communauté MySQL).

*Jani Tolonen*

mysqlimport
Un grand nombre d'extension pour le client en ligne de commande.
PROCEDURE ANALYSE()

*Sinisa Milivojevic*

Compression du protocole client/serveur avec zlib .
Hashing parfait pour la phase d'analyse lexicale.
Insertions multi-lignes
Option mysqldump -e
LOAD DATA LOCAL INFILE
Option SQL_CALC_FOUND_ROWS SELECT
Option --max-user-connections=...
net_read et net_write_timeout
GRANT / REVOKE et SHOW GRANTS FOR
Nouveau protocole client/serveur pour la version 4.0
UNION en version 4.0
Traitements multi-table de DELETE / UPDATE
Tables dérivées en version 4.1
Gestion des ressources utilisateurs.
Développeur initial de l'API MySQL++ C++ API et du client MySQLGUI .

*Tonu Samuel (ancien développeur)*

Interface VIO (la base pour le protocole client/serveur chiffré ).
Système de fichier MySQL (une méthode pour utiliser la base MySQL comme un système de fichiers).
L'expression CASE .
Les fonctions MD5() et COALESCE() .
Le support RAID des tables MyISAM .

*Sasha Pachev*

Implémentation initiale de la réplication (jusqu'en version 4.0).
SHOW CREATE TABLE .
mysql-bench

*Matt Wagner*

Suite de tests MySQL.
Webmestre (jusqu'en 2002).
Coordinateur du développement.

*Miguel Solorzano*

Développement Win32 et publications.
Code du serveur sur Windows NT.
WinMySQLAdmin

*Timothy Smith (ancien développeur)*

Support des jeux de caractères dynamiques.
le script configure, les RPMs et d'autres parties du système de compilation.
Développeur initial de libmysqld , le serveur intégré.

*Sergei Golubchik*

Recherche en texte plein.
Librairie de clés pour MERGE .

*Jeremy Cole*
Relecture et édition de ce manuel.
ALTER TABLE ... ORDER BY ... .
UPDATE ... ORDER BY ... .
DELETE ... ORDER BY ... .

*Indrek Siitan*
Design et programmation de notre interface web.
Auteur de notre lettre d'actualité.

*Jorge del Conde*
MySQLCC ( MySQL Control Center )
Développement Win32
Implémentation initiale des portails du site web.

*Venu Anuganti*
MyODBC 3.51
Nouveau protocole client/serveur pour la version 4.1 (pour les requêtes préparées).

*Arjen Lentz*
Responsable du manuel de référence MySQL.
Préparation de la version imprimée chez O'Reilly.

*Alexander (Bar) Barkov, Alexey (Holyfoot) Botchkov et Ramil Kalimullin*
Données spatiales (GIS) et R-Trees en version 4.1
Unicode et jeux de caractères pour la version 4.1

*Oleksandr (Sanja) Byelkin*
Cache de requêtes en version 4.0
Implémentation des sous-requêtes en version 4.1.

*Aleksey (Walrus) Kishkin et Alexey (Ranger) Stroganov*
Design des tests de performance et analyse.
Maintenance de la suite de test MySQL.

*Zak Greant*
Porte parole Open Source, relation avec la communauté.

*Carsten Pedersen*
Le programme de certification MySQL.

*Lenz Grimmer*
Ingénierie de mise en production (compilation et publication).

*Peter Zaitsev*
Fonction SHA1() , AES_ENCRYPT() et AES_DECRYPT() .
Débogage et nettoyage de diverses fonctionnalités.

*Alexander (Salle) Keremidarski*
Support.
Débogage.

*Per-Erik Martin*
Chef de projet pour les procédures stockées et les triggers.

*Jim Winstead*
Chef développeur web.

*Mark Matthews*
Connecteur/Pilote J (Java).

*Peter Gulutzan*
Documentation des codes/algorithmes MySQL existant.

# 12.2 Contributeurs à MySQL

Bien que MySQL AB possède tous les droits du serveur MySQL et du manuel MySQL , nous voulons montrer notre reconnaissance à ceux qui ont apportés leur contribution d'une manière ou d'une autre à la distribution MySQL . Les contributeurs sont listés ici, dans un ordre aléatoire :

*Paul DuBois*
Aide constante pour rendre le manuel correct et compréhensible. Ce qui inclus la réécriture de l'anglais de Monty et David's en un anglais que tout le monde comprend.

*Gianmassimo Vigazzola qwerg@mbox.vol.it ou qwerg@tin.it*
Le portage initial sur Win32/NT.

*Kim Aldale*
Aide à la réécriture de l'anglais de Monty et David.

*Per Eric Olsson*
Pour des critiques plus ou moins constructives, et des tests sur les formats de lignes dynamiques.

*Irena Pancirov irena@mail.yacc.it*
Portage sur Win32 avec le compilateur Borland. mysqlshutdown.exe et mysqlwatch.exe

*David J. Hughes*
Pour ses efforts de constitution d'une base de données SQL partagée. Chez TcX, le prédécesseur de MySQL AB, nous avons commencé avec mSQL , mais nous nous sommes aperçus que cela ne satisfaisait pas nos besoins, et nous avons écrit une interface SQL avec notre application Unireg. Les clients mysqladmin et mysql ont été largement influencés par leur équivalent mSQL . Nous avons mis une grande partie de nos efforts à faire que la syntaxe de MySQL soit un sur ensemble de celle de mSQL . De nombreuses idées d'API ont été empruntées à mSQL pour rendre plus facile le portage des programmes depuis mSQL vers MySQL. Le logiciel MySQL ne contient aucun code extrait de mSQL . Les deux fichiers ( client/insert_test.c et client/select_test.c ) qui font partie de la distribution sont basés sur les fichiers correspondants, et sans droits de la distribution mSQL , mais sont modifiés, pour servir d'exemple aux modifications nécessaires à la conversion de code de mSQL vers MySQL. ( mSQL est la propriété de David J. Hughes).

*Fred Fish*
Pour son excellente librairie de débogage et de trace en langage C. Monty a fait un grand nombre de petites améliorations de cette librairie (en vitesse et en fonctionnalités).

*Richard A. O'Keefe*
Pour sa librairie de chaînes de caractères du domaine public.

*Henry Spencer*
Pour sa librairie d'expression régulières, utilisées dans les clauses WHERE column REGEXP regexp .

*Free Software Foundation*
De qui nous tenons l'excellent compilateur ( gcc ), la librairie libc (dont nous avons emprunté le fichier strto.c pour faire fonctionner du code sous Linux), et pour la librairie readline (pour le client mysql ).

*Free Software Foundation &The XEmacs development team*
Pour un environnement et un éditeur vraiment excellent, utilisé pratiquement par tout le monde à MySQL AB/TcX/detron.

*Patrick Lynch*
Pour nous avoir aidé à acquérir http://www.mysql.com/ .

*Fred Lindberg*

Pour avoir configuré qmail pour qu'il gère les listes de diffusions MySQLm et pour l'aide incroyable que nous avons reçu pour gérer les listes de diffusions de MySQL.

*Igor Romanenko igor@frog.kiev.ua*

mysqldump (précédemment appelé msqldump , mais porté et amélioré par Monty).

*Yuri Dario*

Pour avoir suivi et amélioré la version MySQL pour OS/2.

*Tim Bunce, Alligator Descartes*

Pour l'interface DBD (Perl).

*Tim Bunce*

Auteur de mysqlhotcopy .

*Andreas Koenig a.koenig@mind.de*

Pour l'interface Perl avec le serveur MySQL.

*Eugene Chan eugene@acenet.com.sg*

Pour avoir porté MySQL avec PHP.

*Michael J. Miller Jr. mke@terrapin.turbolift.com*

Pour le premier manuel MySQL. Et un grand nombre de corrections d'erreurs et d'orthographe dans la FAQ (qui est devenu le manuel MySQL depuis bien longtemps).

*Yan Cailin*

Premier traducteur du manuel de référence MySQL en chinois simplifié, au début de l'an 2000, sur lequel les versions Big5 et HK ( http://mysql.hitstar.com/ ) sont basées. Personal home page at linuxdb.yeah.net .

*Giovanni Maruzzelli maruzz@matrice.it*

Pour le portage de iODBC (Unix ODBC).

*Chris Provenzano*

pthreads portables au niveau utilisateur. D'après les droits : ce produit inclut un logiciel développé par Chris Provenzano, l'University of California, Berkeley et des contributeurs. Nous utilisons actuellement la version 1_60_beta6, modifiée par Monty (voir mit-pthreads/Changes-mysql ).

*Xavier Leroy Xavier.Leroy@inria.fr*

L'auteur de LinuxThreads (utilisé par le serveur MySQL sous Linux).

*Zarko Mocnik zarko.mocnik@dem.si*

Tri pour le slovène et le module cset.tar.gz qui rend plus simple l'ajout d'autres jeux de caractères.

*"TAMITO" tommy@valley.ne.jp*

Les macros de jeu de caractères _MB m et les jeux de caractères ujis et sjis.

*Joshua Chamas joshua@chamas.com*

La base des insertions concurrentes, la syntaxe de date améliorée, le débogage sous NT et les réponses sur la liste de diffusion.

*Yves Carlier Yves.Carlier@rug.ac.be*

mysqlaccess , un programme qui affiche les droits des utilisateurs.

*Rhys Jones rhys@wales.com (And GWE Technologies Limited)*

Pour JDBC, un module pour extraire des données de bases MySQL avec un client Java.

*Dr Xiaokun Kelvin ZHU X.Zhu@brad.ac.uk*

Développement du pilote JDBC et d'autres outils MySQL liés à Java.

*James Cooper pixel@organic.com*

Pour la configuration d'une archive indexée des listes de diffusion sur son site.

*Rick Mehalick Rick_Mehalick@i-o.com*

Pour xmysql , un client grahique X pour MySQL.

*Doug Sisk sisk@wix.com*

Pour les packages RPM de MySQL pour RedHat Linux.

*Diemand Alexander V. axeld@vial.ethz.ch*

Pour les packages RPM de MySQL pour RedHat Linux-Alpha.

*Antoni Pamies Olive toni@readysoft.es*

Pour les packages RPM de MySQL pour Intel et SPARC.

*Jay Bloodworth jay@pathways.sde.state.sc.us*

Pour les packages RPM de MySQL pour MySQL Version 3.21.

*Jochen Wiedmann wiedmann@neckar-alb.de*

Pour l'entretien du module Perl DBD::mysql .

*Therrien Gilbert gilbert@ican.net, Jean-Marc Pouyot jmp@scalaire.fr*

Pour les messages d'erreurs en français.

*Petr Snajdr, snajdr@pvt.net*

Pour les messages d'erreurs en tchèque.

*Jaroslaw Lewandowski jotel@itnet.com.pl*

Pour les messages d'erreurs en polonais.

*Miguel Angel Fernandez Roiz*

Pour les messages d'erreurs en espagnol.

*Roy-Magne Mo rmo@www.hivolda.no*

Pour les messages d'erreurs en norvégien, et les tests de la version 3.21.#.

*Timur I. Bakeyev root@timur.tatarstan.ru*

Pour les messages d'erreurs en russe.

*brenno@dewinter.com &Filippo Grassilli phil@hyppo.com*

Pour les messages d'erreurs en italien.

*Dirk Munzinger dirk@trinity.saar.de*

Pour les messages d'erreurs en allemand.

*Billik Stefan billik@sun.uniag.sk*

Pour les messages d'erreurs en slovène.

*Stefan Saroiu tzoompy@cs.washington.edu*

Pour les messages d'erreurs en roumain.

*Peter Feher*

Pour les messages d'erreurs en hongrois.

*Roberto M. Serqueira*

Pour les messages d'erreurs en portugais.

*Carsten H. Pedersen*

Pour les messages d'erreurs en danois.

*Arjen G. Lentz*

Pour les messages d'erreurs en hollandais, en complément d'une traduction initiale partielle, et d'un travail sur la cohérence et l'orthographe.

*David Sacerdote davids@secnet.com*

Idées pour la vérification sécuritaire des noms d'hôtes.

*Wei-Jou Chen jou@nematic.ieo.nctu.edu.tw*

Le support des caractères chinois Chinese(BIG5).

*Wei He hewei@mail.ied.ac.cn*

Un grand nombre de fonctionnalités pour le jeu de caractères Chinese(GBK).

*Zeev Suraski bourbon@netvision.net.il*

Format d'heure FROM_UNIXTIME() , fonctions ENCRYPT() et conseiller bison . Membre actif des listes de diffusion.

*Luuk de Boer luuk@wxs.nl*

Portage et amélioration de la suite de tests avec DBI / DBD . A été d'une grande aide avec le test crash-me et l'exécution des tests. Certaines améliorations de la fonction de date. Le script mysql_setpermissions .

*Jay Flaherty fty@mediapulse.com*

Grand parties de la section Perl DBI / DBD dans le manuel.

*Paul Southworth pauls@etext.org, Ray Loyzaga yar@cs.su.oz.au*

Relecture du manuel.

*Alexis Mikhailov root@medinf.chuvashia.su*

Fonctions utilisateurs (UDF); CREATE FUNCTION et DROP FUNCTION .

*Andreas F. Bobak bobak@relog.ch*
L'extension AGGREGATE des fonctions UDF.

*Ross Wakelin R.Wakelin@march.co.uk*
Aide avec InstallShield pour MySQL-Win32.

*Jethro Wright III jetman@li.net*
La librairie libmysql.dll .

*James Pereria jpereira@iafrica.com*
Mysqlmanager, un outil d'administration Win32 graphique pour MySQL.

*Curt Sampson cjs@portal.ca*
Portage des MIT-pthreads vers NetBSD/Alpha et NetBSD 1.3/i386.

*Antony T. Curtis antony.curtis@olcs.net*
Portage de la base MySQL sur OS/2.

*Martin Ramsch m.ramsch@computer.org*
Exemples dans les tutoriels du manuel MySQL.

*Steve Harvey*
Pour la sécurisation de mysqlaccess .

*Konark IA-64 Centre of Persistent Systems Private Limited*
http://www.pspl.co.in/konark/ . Aide avec le port Win64 du serveur MySQL.

*Albert Chin-A-Young.*
Modifications de la configuration pour Tru64, support des grands fichiers, et amélioration des gestionnaires TCP.

*John Birrell*
Emulation de pthread_mutex() pour OS/2.

*Benjamin Pflugmann*
Tables MERGE améliorée pour la gestion des INSERTS . Membre actif des listes de diffusion.

*Guilhem Bichot*
Correction des exposants du type DECIMAL . Auteur de mysql_tableinfo .

*Jocelyn Fournier*
Travail excellent et report d'un grand nombre de bogues (surtout dans le code des requêtes imbriquées de MySQL 4.1).

*Georg Richter*
Tests de MySQL 4.1 et trackage de bogues. Nouvelle extension mysqli (API) PHP 5.0 pour une utilisation avec MySQL 4.1 et plus.

*Marc Liyanage*
Maintenance des packages Mac OS X et informations de grande valeur en ce qui concerne la création des packages pour Mac OS X.

D'auters contributeurs, débusqueurs de bogues et testeurs : James H. Thompson, Maurizio Menghini, Wojciech Tryc, Luca Berra, Zarko Mocnik, Wim Bonis, Elmar Haneke, jehamby@lightside , psmith@BayNetworks.com , duane@connect.com.au , Ted Deppner ted@psyber.com , Mike Simons, Jaakko Hyvatti.Et bien d'autres rapports de bogues et de corrections, fournis par les amis des listes de diffusions :

Notre reconnaissance va à ceux qui nous aident à répondre sur les listes de diffusions mysql@lists.mysql.com .

*Daniel Koch dkoch@amcity.com*
Configuration Irix.

*Luuk de Boer luuk@wxs.nl*
Questions de tests de performances.

*Tim Sailer tps@users.buoy.com*
Questions sur DBD-mysql .

*Boyd Lynn Gerber gerberb@zenez.com*
Questions sur SCO.

*Richard Mehalick RM186061@shellus.com*
Questions sur xmysql et les questions d'installation simples.

*Zeev Suraski bourbon@netvision.net.il*
Questions sur le module apache module et sa configuration (log &auth), Questions sur PHP, la syntaxe SQL, et d'autres questions en général.

*Francesc Guasch frankie@citel.upc.es*
Questions générales.

*Jonathan J Smith jsmith@wtp.net*
Questions sur les spécificités des Linux, la syntaxe SQL, et d'autres aspects qui requiert un peu de travail.

*David Sklar sklar@student.net*
Utilisation de MySQL avec PHP et Perl.

*Alistair MacDonald A.MacDonald@uel.ac.uk*
Pas vraiment précisé, mais généralise, et sais gèrer les questions Linux et un peu HP-UX. Pousse l'utilisateur à exploiter mysqlbug .

*John Lyon jlyon@imag.net*
Questions sur l'installation de MySQL sur Linux, avec les paquets .rpm ou via les sources.

*Lorvid Ltd. lorvid@WOLFENET.com*
Questions simples sur les facturations, les licences, le support et les droits.

*Patrick Sherrill patrick@coconet.com*
Questions sur les interfaces ODBC et VisualC++.

*Randy Harmon rjharmon@uptimecomputers.com*
Questions sur DBD , Linux, et la syntaxe SQL.

# 12.3 Supporters de MySQL

Tandis que MySQL AB possède tous les copyrights du serveur MySQL et du manuel MySQL , nous voulons montrer notre reconnaissances aux compagnies suivantes, qui nous ont aidé à financer le developement du serveur MySQL , soit en nous payant pour developper de nouvelles fonctionnalitées, soit en nous fournissant en matériel pour le developpement du serveur MySQL .

*VA Linux / Andover.net*
Financement de la replication.

*NuSphere*
Edition du manuel MySQL.

*Stork Design studio*

Le site web MySQL utilisé entre 1998 et 2000.

*Intel*

Contribution au développement sur les plate-formes Windows et Linux.

*Compaq*

Contribution au développement sur Linux/Alpha.

*SWSoft*

Développement de la version embarquée mysqld .

*FutureQuest*

--skip-show-database

# 13 Historique des changements MySQL

Cet appendice liste les changements de version à version dans le code source de MySQL.

Nous travaillons maintenant activement sur MySQL 4.x et ne fournirons que les correctifs pour les bogues critiques de MySQL 3.23. Nous mettons à jour cette section lorsque nous ajoutons de nouvelles fonctionnalités pour que tout le monde puisse suivre le cours du développement.

Notre section TODO contient ce que nous planifions pour les versions 4.x. MySQL and The Future (The TODO) .

Notez que nous essayons de mettre à jour le manuel en même temps que nous apportons des changements à MySQL. Si vous trouvez une version mentionnée ici que vous ne pouvez retrouver dans la page des téléchargements MySQL ( http://www.mysql.com/downloads/ ), cela signifie que la version n'a pas encore été publiée !

## 13.1 Changes in release 5.0.0 (Development)

Version 5.0 is for the moment only available on for MySQL developers. It will be publicly available as soon we are confident that the base for stored procedures is sound and working.

The following changelog shows what is already done in the 5.0 tree:

- Some support for stored procedures (ANSI SQL 99 style).
- Added SELECT INTO list_of_vars, which can be of mixed, i.e., global and local type.
- Deprecated the update log : it is not supported anymore and replaced by the binary log.

## 13.2 Changes in release 4.1.x (Alpha)

Version 4.1 of the MySQL server includes many enhancements and new features: This version is available from our public BitKeeper tree (sources) and will be available as a binary shortly.

- Subqueries:
```
SELECT * FROM t1 WHERE t1.a=(SELECT t2.b FROM t2);

SELECT * FROM t1 WHERE (1,2,3) IN (SELECT a,b,c FROM t2);
```
- Derived tables:
```
SELECT t1.a FROM t1, (SELECT * FROM t2) t3 WHERE t1.a=t3.a;
```
- INSERT ... ON DUPLICATE KEY UPDATE ... syntax. This allows you to UPDATE an existing row if the insert would cause a duplicate value in a PRIMARY or UNIQUE key. ( REPLACE allows you to overwrite an existing row, which is something entirely different.) Syntaxe des INSERT .
- Extensive Unicode (UTF8) support.
- Character sets can be defined per column, table and database.
- BTREE index on HEAP tables.
- Support for OpenGIS (Geographical data).
- SHOW WARNINGS shows warnings for the last command. SHOW WARNINGS | ERRORS .
- Faster binary protocol with prepared statements and parameter binding. C API Prepared Statements .
- Multi-line queries: You can now issue several queries at once and then read the results in one go. C API Handling multiple query executions .
- Create Table: CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table LIKE table .
- Server based HELP command that can be used in the mysql command line client (and other clients) to get help for SQL commands.

For a full list of changes, please refer to the changelog sections for each individual 4.1.x release.

### 13.2.1 Changes in release 4.1.0

- On Windows, we are now using shared memory to communicate between server and client when they are running on the same machine and you are connecting to localhost .
- REPAIR of MyISAM tables now uses less temporary disk space when sorting char columns .
- DATE / DATETIME checking is now a bit stricter then before to be able to automatically distinguish between date, datetime, and time with microseconds. For example, dates of type YYYYMMDD HHMMDD are not supported anymore; One must either have separators between each DATE / TIME part or not at all.
- Server side help for all MySQL functions. One can now type help week in the mysql client and get help for the week() function.
- Added new client function: mysql_get_server_version() .
- Fixed bug in libmysqlclient that fetched field defaults
- Fixed bug in mysql.cc client when skipping comments
- Added record_in_range() method to MERGE tables to be able to choose the right index when there are many to choose from.
- Replication now works with RAND() and user variables @var .

- Allow on to change mode for ANSI_QUOTES on the fly.
- EXPLAIN SELECT now can be killed. Syntaxe de KILL .
- REPAIR TABLE now can be killed. Syntaxe de KILL .
- Allow one to specify empty key lists for USE|IGNORE|FORCE INDEX .
- DROP TEMPORARY TABLE now only drops temporary tables and doesn't end transactions.
- Added a support for UNION in derived tables
- TIMESTAMP is now returned as string of type 'YYYY-MM-DD HH:MM:DD' .
- New faster client/server protocol which supports prepared statements, bound parameters, and bound result columns, binary transfer of data, warnings.
- Added database and real table name (in case of alias) to the MYSQL_FIELD structure.
- Multi-line queries: You can now issue several queries at once and then read the results in one go.
- In CREATE TABLE foo (a int not null primary key) the PRIMARY word is now optional.
- In CREATE TABLE the attribute SERIAL is now an alias for BIGINT NOT NULL AUTO_INCREMENT UNIQUE .
- SELECT ... FROM DUAL is an alias for SELECT ... . (To be compatible with some other databases).
- If one creates a too long CHAR / VARCHAR it's now automatically changed to TEXT or BLOB ; One will get a warning in this case.
- One can specify the different BLOB / TEXT types with the syntax BLOB(length) and TEXT(length) . MySQL will automatically change it to one of the internal BLOB / TEXT types.
- CHAR BYTE is an alias for CHAR BINARY .
- VARCHARACTER is an alias for VARCHAR .
- New operators integer MOD integer and integer DIV integer .
- SERIAL DEFAULT VALUE added as an alias for AUTO_INCREMENT .
- TRUE and FALSE added as alias for 1 and 0, respectively.
- Aliases are now forced in derived tables, as per SQL-99.
- Fixed SELECT .. LIMIT 0 to return proper row count for SQL_CALC_FOUND_ROWS .
- One can specify many temporary directories to be used in a round-robin fashion with: --tmpdir=dirname1:dirname2:dirname3 .
- Subqueries: SELECT * from t1 where t1.a=(SELECT t2.b FROM t2) .
- Derived tables:

```
SELECT a.col1, b.col2
      FROM (SELECT MAX(col1) AS col1 FROM root_table) a,
      other_table b
      WHERE a.col1=b.col1;
```

- Character sets to be defined per column, table and database.
- Unicode (UTF8) support.
- BTREE index on HEAP tables.
- Faster embedded server (new internal communication protocol).
- One can add a comment per column in CREATE TABLE .
- SHOW FULL COLUMNS FROM table_name shows column comments.
- ALTER DATABASE .
- Support for GIS (Geometrical data).
- SHOW [COUNT(*)] WARNINGS shows warnings from the last command.
- One can specify a column type for a colum in CREATE TABLE ... SELECT by defining the column in the CREATE part.

```
CREATE TABLE foo (a tinyint not null) SELECT b+1 AS 'a' FROM bar;
```

- expr SOUNDS LIKE expr same as SOUNDEX(expr)=SOUNDEX(expr) .
- VARIANCE(expr) returns the variance of expr
- One can create a table from the existing table using CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table (LIKE table) . The table can be either normal or temporary.
- New options --reconnect and disable-reconnect for the mysql client, to reconnect automatically or not if the connection is lost.
- START SLAVE ( STOP SLAVE ) does not return an error anymore if the slave is already started (stopped); it returns a warning instead.
- SLAVE START and SLAVE STOP are not accepted by the query parser anymore, use START SLAVE and STOP SLAVE instead.

# 13.3 Changements dans la publication 4.0.x (Beta)

Version 4.0 of the MySQL server includes many enhancements and new features:

- The InnoDB table type is now included in the standard binaries, adding transactions, row-level locking, and foreign keys. Tables InnoDB .
- A query qache, offering vastly increased performance for many applications. By caching complete result sets, later identical queries can return instantly. Cache de requêtes MySQL .
- Improved full-text indexing with boolean mode, truncation, and phrase searching. Recherche full-text avec MySQL .
- Enhanced MERGE tables, now supporting INSERT s and AUTO_INCREMENT . Tables MERGE .
- UNION syntax in SELECT . Syntaxe de UNION .
- Multi-table DELETE statements. Syntaxe des DELETE .
- libmysqld , the embedded server library. libmysqld, the Embedded MySQL Server Library .
- Additional GRANT privilege options for even tighter control and security. Syntaxe de GRANT et REVOKE .
- Management of user resources in the GRANT system, particularly useful for ISPs and other hosting providers. Limiter les ressources utilisateur .
- Dynamic server variables, allowing configuration changes without having to take down the server. Syntaxe de SET .
- Improved replication code and features. Réplication avec MySQL .
- Numerous new functions and options.
- Changes to existing code for enhanced performance and reliability.

For a full list of changes, please refer to the changelog sections for each individual 4.0.x release.

## 13.3.1 Changes in release 4.0.13 (not released yet)

### Functionality added or changed:

- When using a non-existing table type with CREATE TABLE , first try if the default table type exists before falling back to MyISAM .
- Fixed wrong hostname for TCP/IP connections displayed in SHOW PROCESSLIST .
- Added MEMORY as an alias for HEAP .
- Renamed function rnd to my_rnd as the name was too generic and is an exported symbol in libmysqlclient (thanks to Dennis Haney for the initial patch).
- Portability fix: renamed include/dbug.h to include/my_debug.h .

- mysqldump does not silently delete the binlogs when called with --master-data or --first-slave anymore; while this behaviour was convenient for some users, others may suffer from it. Now one has to explicitely ask for this deletion with the new --delete-master-logs option.
- If the slave is configured (using for example replicate-wild-ignore-table=mysql.% ) to exclude mysql.user , mysql.host , mysql.db , mysql.tables_priv and mysql.columns_priv from replication, then GRANT and REVOKE will not be replicated.

Bugs fixed:

- Fixed bug in multi-table updates which occured whenever a temporary table, containing update data, had to be converted from HEAP to MyISAM .
- Fixed a bug when doing LOAD DATA INFILE IGNORE : when reading the binary log, mysqlbinlog and the replication code read REPLACE instead of IGNORE . This could make the slave's table become different from the master's table. Bug #218.
- Fixed a deadlock when relay_log_space_limit was set to a too small value. Bug #79.
- Fixed a bug in HAVING clause when an alias is used from the **select list** .
- Fixed a bug in MyISAM when a row is inserted into a table with a large number of NULL columns. Bug was caused by wrong calculation of the record length, as the space required for storage of NULL bits was not added to the total record length.
- Fixed a bug when SELECT **@nonexistent_variable** caused the error in client - server protocl due to net_printf() being sent to the client twice.
- Fixed a bug in setting SQL_BIG_SELECTS option.
- Fixed a bug in SHOW PROCESSLIST which only displayed a localhost in the "Host" column. This was caused by a glitch that only used current thread info instead info from the linked list of threads.
- Removed unnecessary Mac OS X helper files from server RPM (bug #144).
- Allow optimzation of multi-table-update for InnoDB tables as well.
- Fixed a bug in multi-table-updates that caused some rows to be updated several times.
- Fixed a bug in mysqldump when it was called with --master-data : the CHANGE MASTER TO commands appended to the SQL dump had wrong coordinates (bug #159).
- Fixed a bug when an updating query using USER() was replicated on the slave ; this caused segfault on the slave (bug #178). USER() is still badly replicated on the slave (it is replicated to "" ).

## 13.3.2 Changes in release 4.0.12 (15 Mar 2003: Production)

### Functionality added or changed:

- mysqld no longer reads options from world-writeable config files.
- Integer values between 9223372036854775807 and 9999999999999999999 are now regarded as unsigned longlongs, not as floats. This makes these values work similar to values between 10000000000000000000 and 18446744073709551615.
- SHOW PROCESSLIST will now include the client TCP port after the hostname to make it easier to know from which client the request originated.

Bugs fixed:

- Fixed mysqld crash on extremely small values of sort_buffer variable.
- INSERT INTO u SELECT ... FROM t was written too late to the binary log if t was very frequently updated during the execution of this query. This could cause a problem with mysqlbinlog or replication. The master must be upgraded, not the slave. (bug #136).
- Fixed checking of random part of WHERE clause (bug #142).
- Fixed a bug with multi-table updates with InnoDB tables. This bug occured as, in many cases, InnoDB tables can not be updated "on the fly", but offsets to the records have to be stored in a temporary table.
- Added missing file mysql_secure_installation to the server RPM subpackage (bug #141).
- Fixed MySQL (and myisamchk ) crash on artificially corrupted .MYI files.
- Don't allow BACKUP TABLE to overwrite existing files.
- Fixed a bug with multi-table UPDATE s when user had all privileges on the database where tables are located and there were any entries in tables_priv table, i.e. grant_option was true.
- Fixed a bug that allowed a user with table or column grants on some table, TRUNCATE any table in the same database.
- Fixed deadlock when doing LOCK TABLE followed by DROP TABLE in the same thread. In this case one could still kill the thread with KILL .
- LOAD DATA LOCAL INFILE was not properly written to the binary log (hence not properly replicated). (bug #82).
- RAND() entries were not read correctly by mysqlbinlog from the binary log which caused problems when restoring a table that was inserted with RAND() . INSERT INTO t1 VALUES(RAND()) . In replication this worked ok.
- SET SQL_LOG_BIN=0 was ignored for INSERT DELAYED queries. (bug #104).
- SHOW SLAVE STATUS reported too old positions (columns Relay_Master_Log_File and Exec_master_log_pos ) for the last executed statement from the master, if this statement was the COMMIT of a transaction. The master must be upgraded for that, not the slave. (bug #52).
- LOAD DATA INFILE was not replicated by the slave if replicate_*_table was set on the slave. (bug #86).
- After RESET SLAVE , the coordinates displayed by SHOW SLAVE STATUS looked un-reset (though they were, but only internally). (bug #70).
- Fixed query cache invalidation on LOAD DATA .
- Fixed memory leak on ANALYZE procedure with error.
- Fixed a bug in handling CHAR(0) columns that could cause wrong results from the query.
- Fixed rare bug with wrong initialization of AUTO_INCREMENT column, as a secondary column in a multi-column key ( Utiliser AUTO_INCREMENT ), when data was inserted with INSERT ... SELECT or LOAD DATA into an empty table.
- On windows, STOP SLAVE didn't stop the slave until the slave got one new command from the master (this bug has been fixed for MySQL 4.0.11 by releasing updated 4.0.11a windows packages, which include this individual fix on top of the 4.0.11 sources). (bug #69).
- Fixed a crash when no database was selected and LOAD DATA command was issued with full table name specified, including database prefix.
- Fixed a crash when shutting down replication on some platforms (e.g. Mac OS X).
- Fixed a portability bug with pthread_attr_getstacksize on HP-UX 10.20 (Patch was also included in 4.0.11a sources).
- Fixed the bigint test to not fail on some platforms (e.g. HP-UX and Tru64) due to different return values of the atof() function.
- Fixed the rpl_rotate_logs test to not fail on certain platforms (e.g. Mac OS X) due to a too long file name (changed slave-master-info.opt to .slave-mi ).

## 13.3.3 Changes in release 4.0.11 (20 Feb 2003)

### Functionality added or changed:

- NULL is now sorted **LAST** if you use ORDER BY ... DESC (as it was before before MySQL 4.0.2). This is change was required to follow ANSI SQL 99. (The original change was because we thought that ANSI SQL 99 required NULL to be always sorted at the same position, but this was wrong).
- Added START TRANSACTION (ANSI SQL-99 syntax) as alias for BEGIN . This is recommended to use instead of BEGIN to start a transaction.
- Added OLD_PASSWORD() as a synonym for PASSWORD() .
- Allow keyword ALL in group functions.
- Added support for some new INNER JOIN and JOIN syntaxes. For example, SELECT * FROM t1 INNER JOIN t2 didn't work before.
- Novell NetWare 6.0 porting effort completed, Novell patches merged into the main source tree.

Bugs fixed:

- Fixed problem with multi-table-delete and InnoDB tables.
- Fixed a problem with BLOB NOT NULL columns used with IS NULL .
- Re-added missing pre- and post(un)install scripts to the Linux RPM packages (they were missing after the renaming of the server subpackage).
- Fixed that table locks are not released with multi-table updates and deletes with InnoDB storage engine.
- Fixed bug in updating BLOB columns with long strings.
- Fixed integer-wraparound when giving big integer (>= 10 digits) to function that requires an unsigned argument, like CREATE TABLE (...) AUTO_INCREMENT=# .
- MIN(key_column) could in some cases return NULL on a column with NULL and other values.
- MIN(key_column) and MAX(key_column) could in some cases return wrong values when used in OUTER JOIN .
- MIN(key_column) and MAX(key_column) could return wrong values if one of the tables was empty.
- Fixed rare crash in compressed MyISAM tables with blobs.
- Fixed bug in using aggregate functions as argument for INTERVAL , CASE , FIELD , CONCAT_WS , ELT and MAKE_SET functions.
- When running with --lower-case-table-names (default on windows) and you had tables or databases with mixed case on disk, then executing SHOW TABLE STATUS followed with DROP DATABASE or DROP TABLE could fail with Errcode 13 .

## 13.3.4 Changes in release 4.0.10 (29 Jan 2003)

### Functionality added or changed:

- Added option --log-error[=file_name] to mysqld_safe and mysqld . This option will force all error messages to be put in a log file if the option --console is not given. On Windows --log-error is enabled as default, with a default name of host_name.err if the name is not specified.
- Changed some things from Warning: to Note: in the log files.
- The mysqld server should now compile on Netware.
- Added optimisation that if one does GROUP BY ... ORDER BY NULL then result is not sorted.
- New command-line option for mysqld to replace/disable built-in stopword list, that is used in fulltext search. SHOW VARIABLES .
- Changed default stack size from 64K to 192K; This fixes a core dump problem on RedHat 8.0 and other systems with a glibc that requires > 128K stack for gethostbyaddr() to resolve a hostname. You can fix this for earlier MySQL versions by starting mysqld with --thread-stack=192K .
- Added mysql_waitpid to the binary distribution and the MySQL-client RPM subpackage (required for mysql-test-run ).
- Renamed the main MySQL RPM package to MySQL-server . When updating from an older version, MySQL-server.rpm will simply replace MySQL.rpm .
- If a slave is configured with replicate_wild_do_table=db.% or replicate_wild_ignore_table=db.% , these rules will be applied to CREATE/DROP DATABASE too.
- Added timeout value for MASTER_POS_WAIT() .

Bugs fixed:

- Fixed initialization of the random seed for newly created threads to give a better rand() distribution from the first call.
- Fixed a bug that caused mysqld to hang when a table was opened with the HANDLER command and then dropped without being closed.
- Fixed bug in logging to binary log (which affects replication) a query that inserts a NULL in an auto_increment field and also uses LAST_INSERT_ID() .
- Fixed an unlikely bug that could cause a memory overrun when using ORDER BY constant_expression .
- Fixed a table corruption in myisamchk 's parallel repair mode.
- Fixed bug in query cache invalidation on simple table renaming.
- Fixed bug in mysqladmin --relative .
- On some 64 bit systems, show status reported a strange number for Open_files and Open_streams .
- Fixed wrong number of columns in EXPLAIN on empty table.
- Fixed bug in LEFT JOIN that caused zero rows to be returned in the case the WHERE condition was evaluated as FALSE after reading const tables. (Unlikely condition).
- FLUSH PRIVILEGES didn't correctly flush table/column privileges when mysql.tables_priv is empty.
- Fixed bug in replication when using LOAD DATA INFILE one a file that updated and auto_increment field with NULL or 0 . This bug only affected MySQL 4.0 masters (not slaves or MySQL 3.23 masters). **NOTE** : If you have a slave that has replicated a file with generated auto_increment fields then the slave data is corrupted and you should reinitialise the affected tables from the master.
- Fixed possible memory overrun when sending a blob > 16M to the client.
- Fixed wrong error message when setting a NOT NULL field to an expression that returned NULL .
- Fixed core dump bug in str LIKE "%other_str%" where str or other_str contained characters >= 128.
- Fixed bug: When executing on master LOAD DATA and InnoDB failed with table full error the binary log was corrupted.

## 13.3.5 Changes in release 4.0.9 (09 Jan 2003)

### Functionality added or changed:

- OPTIMIZE TABLE will for MyISAM tables treat all NULL values as different when calculating cardinality. This helps in optimising joins between tables where one of the tables has a lot of NULL values in a indexed column:
```
SELECT * from t1,t2 where t1.a=t2.key_with_a_lot_of_null;
```
- Added join operator FORCE INDEX (key_list) . This acts likes USE INDEX (key_list) but with the addition that a table scan is assumed to be VERY expensive. One bad thing with this is that it makes FORCE a reserved word.
- Reset internal row buffer in MyISAM after each query. This will reduce memory in the case you have a lot of big blobs in a table.

Bugs fixed:

- A security patch in 4.0.8 causes the mysqld server to die if the remote hostname can't be resolved. This is now fixed.
- Fixed crash when replication big LOAD DATA INFILE statement that caused log rotation.

## 13.3.6 Changes in release 4.0.8 (07 Jan 2003)

### Functionality added or changed:

- Default max_packet_length for libmysqld.c is now 1024*1024*1024.
- One can now specify max_allowed_packet in a file ready by mysql_options(MYSQL_READ_DEFAULT_FILE) . for clients.
- When sending a too big packet to the server with the not compressed protocol, the client now gets an error message instead of a lost connection.
- We now send big queries/result rows in bigger hunks, which should give a small speed improvement.
- Fixed some bugs with the compressed protocol for rows > 16M.
- InnoDB tables now also supports ON UPDATE CASCADE in FOREIGN KEY constraints. See the InnoDB section in the manual for the InnoDB changelog.

Bugs fixed:

- Fixed bug in ALTER TABLE with BDB tables.
- Fixed core dump bug in QUOTE() function.
- Fixed a bug in handling communication packets bigger than 16M. Unfortunately this required a protocol change; If you upgrade the server to 4.0.8 and above and have clients that uses packets >= 255*255*255 bytes (=16581375) you must also upgrade your clients to at least 4.0.8. If you don't upgrade, the clients will hang when sending a big packet.
- Fixed bug when sending blobs longer than 16M to client.
- Fixed bug in GROUP BY when used on BLOB column with NULL values.
- Fixed a bug in handling NULL s in CASE ... WHEN ...

## 13.3.7 Changes in release 4.0.7 (20 Dec 2002)

### Functionality added or changed:

- mysqlbug now also reports the compiler version used for building the binaries (if the compiler supports the option --version ).

Bugs fixed:

- Fixed compilation problems on OpenUnix and HPUX 10.20.
- Fixed some optimisation problems when compiling MySQL with -DBIG_TABLES on a 32 bit system.
- mysql_drop_db() didn't check permissions properly so anyone could drop another users database. DROP DATABASE is checked properly.

## 13.3.8 Changes in release 4.0.6 (14 Dec 2002: Gamma)

### Functionality added or changed:

- Added syntax support for CHARACTER SET xxx and CHARSET=xxx table options (to be able to read table dumps from 4.1).
- Fixed replication bug that caused the slave to loose its position in some cases when the replication log was rotated.
- Fixed that a slave will restart from the start of a transaction if it's killed in the middle of one.
- Moved the manual pages from man to man/man1 in the binary distributions.
- The default type returned by IFNULL(A,B) is now set to be the more 'general' of the types of A and B . (The order is STRING , REAL or INTEGER ).
- Moved the mysql.server startup script in the RPM packages from /etc/rc.d/init.d/mysql to /etc/init.d/mysql (which almost all current Linux distributions support for LSB compliance).
- Added Qcache_lowmem_prunes status variable (number of queries that were deleted from cache because of low memory).
- Fixed mysqlcheck so it can deal with table names containing dashes.
- Bulk insert optimisation ( SHOW VARIABLES ) is no longer used when inserting small (less than 100) number of rows.
- Optimisation added for queries like SELECT ... FROM merge_table WHERE indexed_column=constant_expr .
- Added functions LOCALTIME and LOCALTIMESTAMP as synonyms for NOW() .
- CEIL is now an alias for CEILING .
- The CURRENT_USER() function can be used to get a user@host value as it was matched in the GRANT system. Fonctions diverses .
- Fixed CHECK constraints to be compatible with ANSI SQL. This made CHECK a reserved word. (Checking of CHECK constraints is still not implemented).
- Added CAST(... as CHAR) .
- Added PostgreSQL compatible LIMIT syntax: SELECT ... LIMIT # OFFSET #
- mysql_change_user() will now reset the connection to the state of a fresh connect (Ie, ROLLBACK any active transaction, close all temporary tables, reset all user variables etc..)

Bugs fixed:

- Fixed number of found rows returned in multi table updates
- Make --lower-case-table-names default on Mac OS X as the file system is case sensitive.
- Transactions in AUTOCOMMIT=0 mode didn't rotate binary log.
- A fix for the bug in a SELECT with joined tables with ORDER BY and LIMIT clause when filesort had to be used. In that case LIMIT was applied to filesort of one of the tables, although it could not be. This fix solved problems with LEFT JOIN too.
- mysql_server_init() now makes a copy of all arguments. This fixes a problem when using the embedded server in C# program.
- Fixed buffer overrun in libmysqlclient library that allowed a malicious MySQL server to crash the client application.
- Fixed security-related bug in mysql_change_user() handling. All users are strongly recommended to upgrade to version 4.0.6.
- Fixed bug that prevented --chroot command-line option of mysqld from working.
- Fixed bug in phrase operator "..." in boolean full-text search.

- Fixed bug that caused OPTIMIZE TABLE to corrupt the table under some rare circumstances.
- Part rewrite of multi-table-update to optimise it, make it safer and more bug free.
- LOCK TABLES now works together with multi-table-update and multi-table-delete.
- --replicate-do=xxx didn't work for UPDATE commands. (Bug introduced in 4.0.0)
- Fixed shutdown problem on Mac OS X.
- Major InnoDB bugs in REPLACE, AUTO_INCREMENT, INSERT INTO ... SELECT ... were fixed. See the InnoDB changelog in the InnoDB section of the manual.

## 13.3.9 Changes in release 4.0.5 (13 Nov 2002)

### Functionality added or changed:

- Port number was added to host name (if it is known) in SHOW PROCESSLIST command
- Changed handling of last argument in WEEK() so that one can get week number according to the ISO 8601 specification. (Old code should still work).
- Fixed that INSERT DELAYED threads doesn't hang on Waiting for INSERT when one sends a SIGHUP to mysqld.
- Change that AND works according to SQL ANSI99 when it comes to NULL handling. In practice, this only affects queries where you do something like WHERE ... NOT (NULL AND 0) .
- mysqld will now resolve basedir to its full path (with realpath() ). This enables one to use relative symlinks to the MySQL installation directory. This will however cause show variables to report different directories on systems where there is a symbolic link in the path.
- Fixed that MySQL will not use index scan on index disabled with IGNORE INDEX or USE INDEX . to be ignored.
- Added --use-frm option to mysqlcheck . When used with REPAIR , it gets the table structure from the .frm file, so the table can be repaired even if the .MYI header is corrupted.
- Fixed bug in MAX() optimisation when used with JOIN and ON expressions.
- Added support for reading of MySQL 4.1 table definition files.
- BETWEEN behaviour changed ( Opérateurs de comparaison ). Now datetime_col BETWEEN timestamp AND timestamp should work as expected.
- One can create TEMPORARY MERGE tables now.
- DELETE FROM myisam_table now shrinks not only the .MYD file but also the .MYI file.
- When one uses the --open-files-limit=# option to mysqld_safe it's now passed on to mysqld .
- Changed output from EXPLAIN from 'where used' to 'Using where' to make it more in line with other output.
- Removed variable safe_show_database as it was not used anymore.
- Updated source tree to be built using automake 1.5 and libtool 1.4 .
- Fixed an inadvertently changed option ( --ignore-space ) back to the original --ignore-spaces in mysqlclient . (Both syntaxes will work).
- Don't require UPDATE privilege when using REPLACE .
- Added support for DROP TEMPORARY TABLE ... , to be used to make replication safer.
- When transactions are enabled, all commands that update temporary tables inside a BEGIN/COMMIT are now stored in the binary log on COMMIT and not stored if one does ROLLBACK . This fixes some problems with non-transactional temporary tables used inside transactions.
- Allow braces in joins in all positions. Formerly, things like SELECT * FROM (t2 LEFT JOIN t3 USING (a)), t1 worked, but not SELECT * FROM t1, (t2 LEFT JOIN t3 USING (a)) . Note that braces are simply removed, they do not change the way the join is executed.
- InnoDB now supports also isolation levels READ UNCOMMITTED and READ COMMITTED . For a detailed InnoDB changelog, Historique des changements InnoDB in this manual.

Bugs fixed:

- Fixed bug in MAX() optimisation when used with JOIN and ON expressions.
- Fixed that INSERT DELAY threads don't hang on Waiting for INSERT when one sends a SIGHUP to mysqld .
- Fixed that MySQL will not use an index scan on an index that has been disabled with IGNORE INDEX or USE INDEX .
- Corrected test for root user in mysqld_safe .
- Fixed error message issued when storage engine cannot do CHECK or REPAIR .
- Fixed rare core dump problem in complicated GROUP BY queries that didn't return any result.
- Fixed mysqlshow to work properly with wildcarded database names and with database names that contain underscores.
- Portability fixes to get MySQL to compile cleanly with Sun Forte 5.0.
- Fixed MyISAM crash when using dynamic-row tables with huge numbers of packed fields.
- Fixed query cache behaviour with BDB transactions.
- Fixed possible floating point exception in MATCH relevance calculations.
- Fixed bug in full-text search IN BOOLEAN MODE that made MATCH to return incorrect relevance value in some complex joins.
- Fixed a bug that limited MyISAM key length to a value slightly less that 500. It is exactly 500 now.
- Fixed that GROUP BY on columns that may have a NULL value doesn't always use disk based temporary tables.
- The filename argument for the --des-key-file argument to mysqld is interpreted relative to the data directory if given as a relative pathname.
- Removed a condition that temp table with index on column that can be NULL has to be MyISAM . This was okay for 3.23, but not needed in 4.*. This resulted in slowdown in many queries since 4.0.2.
- Small code improvement in multi-table updates.
- Fixed a newly introduced bug that caused ORDER BY ... LIMIT # to not return all rows.
- Fixed a bug in multi-table deletes when outer join is used on an empty table, which gets first to be deleted.
- Fixed a bug in multi-table updates when a single table is updated.
- Fixed bug that caused REPAIR TABLE and myisamchk to corrupt FULLTEXT indexes.
- Fixed bug with caching the mysql grant table database. Now queries in this database are not cached in the query cache.
- Small fix in mysqld_safe for some shells.
- Give error if a MyISAM MERGE table has more than 2 ^ 32 rows and MySQL was not compiled with -DBIG_TABLES .
- Fixed some ORDER BY ... DESC problems with InnoDB tables.

## 13.3.10 Changes in release 4.0.4 (29 Sep 2002)

- Fixed bug where GRANT / REVOKE failed if hostname was given in non-matching case.
- Don't give warning in LOAD DATA INFILE when setting a timestamp to a string value of '0' .
- Fixed bug in myisamchk -R mode.
- Fixed bug that caused mysqld to crash on REVOKE .
- Fixed bug in ORDER BY when there is a constant in the SELECT statement.
- One didn't get an error message if mysqld couldn't open the privilege tables.
- SET PASSWORD FOR ... closed the connection in case of errors (bug from 4.0.3).
- Increased max possible max_allowed_packet in mysqld to 1GB.
- Fixed bug when doing a multi-line INSERT on a table with an AUTO_INCREMENT key which was not in the first part of the key.
- Changed LOAD DATA INFILE to not recreate index if the table had rows from before.
- Fixed overrun bug when calling AES_DECRYPT() with incorrect arguments.

- --skip-ssl can now be used to disable SSL in the MySQL clients, even if one is using other SSL options in an option file or previously on the command line.
- Fixed bug in MATCH ... AGAINST( ... IN BOOLEAN MODE) used with ORDER BY .
- Added LOCK TABLES and CREATE TEMPORARY TABLES privilege on the database level. One must run the mysql_fix_privilege_tables script on old installations to activate these.
- Dans SHOW TABLE ... STATUS , les tables compressées n'apparaissaient pas comme dynamic .
- SELECT @@[global|session].var_name didn't report global | session in the result column name.
- Fixed problem in replication that FLUSH LOGS in a circular replication setup created an infinite number of binary log files. Now a rotate-binary-log command in the binary log will not cause slaves to rotate logs.
- Removed STOP EVENT from binary log when doing FLUSH LOGS .
- Disable the use of SHOW NEW MASTER FOR SLAVE as this needs to be completely changed in 4.1.
- Fixed a bug with constant expression (e.g. field of a one-row table, or field from a table, referenced by a UNIQUE key) appeared in ORDER BY part of SELECT DISTINCT .
- --log-binary=a.b.c now properly strips off .b.c .
- FLUSH LOGS removed numerical extension for all future update logs.
- GRANT ... REQUIRE didn't store the SSL information in the mysql.user table if SSL was not enabled in the server.
- GRANT ... REQUIRE NONE can now be used to remove SSL information.
- AND is now optional between REQUIRE options.
- REQUIRE option was not properly saved, which could cause strange output in SHOW GRANTS .
- Fixed that mysqld --help reports correct values for --datadir and --bind-address .
- Fixed that one can drop UDFs that didn't exist when mysqld was started.
- Fixed core dump problem with SHOW VARIABLES on some 64 bit systems (like Solaris sparc).
- Fixed a bug in my_getopt; --set-variable syntax didn't work for those options that didn't have a valid variable in my_option struct. This affected at least default-table-type option.
- Fixed a bug from 4.0.2 that caused REPAIR TABLE and myisamchk --recover to fail on tables with duplicates in a unique key.
- Fixed a bug from 4.0.3 in calculating the default field type for some functions. This affected queries of type CREATE TABLE table_name SELECT expression(),...
- Fixed bug in queries of type SELECT * FROM table-list GROUP BY ... and SELECT DISTINCT * FROM ... .
- Fixed bug with the --slow-log when logging an administrator command (like FLUSH TABLES ).
- Fixed a bug that OPTIMIZE of locked and modified table, reported table corruption.
- Fixed a bug in my_getopt in handling of special prefixes ( --skip- , --enable- ). --skip-external-locking didn't work and the bug may have affected other similar options.
- Fixed bug in checking for output file name of the tee option.
- Added some more optimisation to use index for SELECT ... FROM many_tables .. ORDER BY key limit #
- Fixed problem in SHOW OPEN TABLES when a user didn't have access permissions to one of the opened tables.

## 13.3.11 Changes in release 4.0.3 (26 Aug 2002: Beta)

- Fixed problem with configure ... --localstatedir=... .
- Cleaned up mysql.server script.
- Fixed a bug in mysqladmin shutdown when pid file was modified while mysqladmin was still waiting for the previous one to disappear. This could happen during a very quick restart and caused mysqladmin to hang until shutdown_timeout seconds had passed.
- Don't increment warnings when setting AUTO_INCREMENT columns to NULL in LOAD DATA INFILE .
- Fixed all boolean type variables/options to work with the old syntax, e.g. all of these work: --lower-case-table-names, --lower-case-table-names=1, -O lower-case-table-names=1, --set-variable=lower-case-table-names=1
- Fixed shutdown problem (SIGTERM signal handling) on Solaris. (Bug from 4.0.2).
- SHOW MASTER STATUS now returns an empty set if binary log is not enabled.
- SHOW SLAVE STATUS now returns an empty set if slave is not initialised.
- Don't update MyISAM index file on update if not strictly necessary.
- Fixed bug in SELECT DISTINCT ... FROM many_tables ORDER BY not-used-column .
- Fixed a bug with BIGINT s and quoted strings.
- Added QUOTE() function that performs SQL quoting to produce values that can be used as data values in queries.
- Changed variable DELAY_KEY_WRITE to an enum to allow one set DELAY_KEY_WRITE for all tables without taking down the server.
- Changed behaviour of IF(condition,column,NULL) so that it returns the value of the column type.
- Made safe_mysqld a symlink to mysqld_safe in binary distribution.
- Fixed security bug when having an empty database name in the user.db table.
- Fixed some problems with CREATE TABLE ... SELECT function() .
- mysqld now has the option --temp-pool enabled by default as this gives better performance with some operating systems.
- Fixed hang in CHANGE MASTER TO if the slave thread died very quickly.
- Big cleanup in replication code (less logging, better error messages, etc..)
- If the --code-file option is specified, the server calls setrlimit() to set the maximum allowed core file size to unlimited, so core files can be generated.
- Fixed bug in query cache after temporary table creation.
- Added --count=N ( -c ) option to mysqladmin , to make the program do only N iterations. To be used with --sleep ( -i ). Useful in scripts.
- Fixed bug in multi-table UPDATE : when updating a table, do_select() became confused about reading records from a cache.
- Fixed bug in multi-table UPDATE when several fields were referenced from a single table
- Fixed bug in truncating nonexisting table.
- Fixed bug in REVOKE that caused user resources to be randomly set.
- Fixed bug in GRANT for the new CREATE TEMPORARY TABLE privilege.
- Fixed bug in multi-table DELETE when tables are re-ordered in the table initialisation method and ref_lengths are of different sizes.
- Fixed two bugs in SELECT DISTINCT with large tables.
- Fixed bug in query cache initialisation with very small query cache size.
- Allow DEFAULT with INSERT statement.
- The startup parameters myisam_max_sort_file_size and myisam_max_extra_sort_file_size are now given in bytes, not megabytes.
- External system locking of MyISAM / ISAM files is now turned off by default. One can turn this on with --external-locking . (For most users this is never needed).
- Fixed core dump bug with INSERT ... SET db_name.table_name.colname='' .
- Fixed client hangup bug when using some SQL commands with wrong syntax.
- Fixed a timing bug in DROP DATABASE
- New SET [GLOBAL | SESSION] syntax to change thread-specific and global server variables at runtime.
- Added variable slave_compressed_protocol .
- Renamed variable query_cache_startup_type to query_cache_type , myisam_bulk_insert_tree_size to bulk_insert_buffer_size , record_buffer to read_buffer_size and record_rnd_buffer to record_rnd_buffer_size .
- Renamed some SQL variables, but old names will still work until 5.0. Upgrading From Version 3.23 to 4.0 .
- Renamed --skip-locking to --skip-external-locking .
- Removed unused variable query_buffer_size .
- Fixed a bug that made the pager option in the mysql client non-functional.
- Added full AUTO_INCREMENT support to MERGE tables.
- Extended LOG() function to accept an optional arbitrary base parameter. Fonctions mathématiques .
- Added LOG2() function (useful for finding out how many bits a number would require for storage).
- Added LN() natural logarithm function for compatibility with other databases. It is synonymous with LOG(X) .

# 13.3.12 Changes in release 4.0.2 (01 Jul 2002)

- Cleaned up NULL handling for default values in DESCRIBE table_name .
- Fixed truncate() to round up negative values to the nearest integer.
- Changed --chroot=path option to execute chroot() immediately after all options have been parsed.
- Don't allow database names that contain '\' .
- lower_case_table_names now also affects database names.
- Added XOR operator (logical and bitwise XOR ) with ^ as a synonym for bitwise XOR .
- Added function IS_FREE_LOCK("lock_name") . Based on code contributed by Hartmut Holzgraefe hartmut@six.de .
- Removed mysql_ssl_clear() from C API, as it was not needed.
- DECIMAL and NUMERIC types can now read exponential numbers.
- Added SHA1() function to calculate 160 bit hash value as described in RFC 3174 (Secure Hash Algorithm). This function can be considered a cryptographically more secure equivalent of MD5() . Fonctions diverses .
- Added AES_ENCRYPT() and AES_DECRYPT() functions to perform encryption according to AES standard (Rijndael). Fonctions diverses .
- Added --single-transaction option to mysqldump , allowing a consistent dump of InnoDB tables. mysqldump , exporter les structures de tables et les données .
- Fixed bug in innodb_log_group_home_dir in SHOW VARIABLES .
- Fixed a bug in optimiser with merge tables when non-unique values are used in summing up (causing crashes).
- Fixed a bug in optimiser when a range specified makes index grouping impossible (causing crashes).
- Fixed a rare bug when FULLTEXT index is present and no tables are used.
- Added privileges CREATE TEMPORARY TABLES , EXECUTE , LOCK TABLES , REPLICATION CLIENT , REPLICATION SLAVE , SHOW DATABASES and SUPER . To use these, you must have run the mysql_fix_privilege_tables script after upgrading.
- Fixed query cache align data bug.
- Fixed mutex bug in replication when reading from master fails.
- Added missing mutex in TRUNCATE TABLE ; This fixes some core dump/hangup problems when using TRUNCATE TABLE .
- Fixed bug in multi-table DELETE when optimiser uses only indices.
- Fixed that ALTER TABLE table_name RENAME new_table_name is as fast as RENAME TABLE .
- Fixed bug in GROUP BY with two or more fields, where at least one field can contain NULL values.
- Use Turbo Boyer-Moore algorithm to speed up LIKE "%keyword%" searches.
- Fixed bug in DROP DATABASE with symlink.
- Fixed crash in REPAIR ... USE_FRM .
- Fixed bug in EXPLAIN with LIMIT offset != 0 .
- Fixed bug in phrase operator "..." in boolean full-text search.
- Fixed bug that caused duplicated rows when using truncation operator * in boolean full-text search.
- Fixed bug in truncation operator of boolean full-text search (wrong results when there are only +word* s in the query).
- Fixed bug in boolean full-text search that caused a crash when an identical MATCH expression that did not use an index appeared twice.
- Query cache is now automatically disabled in mysqldump .
- Fixed problem on Windows 98 that made sending of results very slow.
- Boolean full-text search weighting scheme changed to something more reasonable.
- Fixed bug in boolean full-text search that caused MySQL to ignore queries of ft_min_word_len characters.
- Boolean full-text search now supports ``phrase searches''.
- New configure option --without-query-cache .
- Memory allocation strategy for ``root memory'' changed. Block size now grows with number of allocated blocks.
- INET_NTOA() now returns NULL if you give it an argument that is too large (greater than the value corresponding to 255.255.255.255 ).
- Fix SQL_CALC_FOUND_ROWS to work with UNION s. It will work only if the first SELECT has this option and if there is global LIMIT for the entire statement. For the moment, this requires using parentheses for individual SELECT queries within the statement.
- Fixed bug in SQL_CALC_FOUND_ROWS and LIMIT .
- Don't give an error for CREATE TABLE ...(... VARCHAR(0)) .
- Fixed SIGINT and SIGQUIT problems in mysql.cc on Linux with some glibc versions.
- Fixed bug in convert.cc , which is caused by having an incorrect net_store_length() linked in the CONVERT::store() method.
- DOUBLE and FLOAT columns now honor the UNSIGNED flag on storage.
- InnoDB now retains foreign key constraints through ALTER TABLE and CREATE/DROP INDEX .
- InnoDB now allows foreign key constraints to be added through the ALTER TABLE syntax.
- InnoDB tables can now be set to automatically grow in size (autoextend).
- Added --ignore-lines=n option to mysqlimport . This has the same effect as the IGNORE n LINES clause for LOAD DATA .
- Fixed bug in UNION with last offset being transposed to total result set.
- REPAIR ... USE_FRM added.
- Fixed that DEFAULT_SELECT_LIMIT is always imposed on UNION result set.
- Fixed that some SELECT options can appear only in the first SELECT .
- Fixed bug with LIMIT with UNION , where last select is in the braces.
- Fixed that full-text works fine with UNION operations.
- Fixed bug with indexless boolean full-text search.
- Fixed bug that sometimes appeared when full-text search was used with ``const'' tables.
- Fixed incorrect error value when doing a SELECT with an empty HEAP table.
- Use ORDER BY column DESC now sorts NULL values first. (In other words, NULL values sort first in all cases, whether or not DESC is specified.)
- Fixed bug in WHERE key_name='constant' ORDER BY key_name DESC .
- Fixed bug in SELECT DISTINCT ... ORDER BY DESC optimisation.
- Fixed bug in ... HAVING 'GROUP_FUNCTION'(xxx) IS [NOT] NULL .
- Fixed bug in truncation operator for boolean full-text search.
- Allow value of --user=# option for mysqld to be specified as a numeric user ID.
- Fixed a bug where SQL_CALC_ROWS returned an incorrect value when used with one table and ORDER BY and with InnoDB tables.
- Fixed that SELECT 0 LIMIT 0 doesn't hang thread.
- Fixed some problems with USE/IGNORE INDEX when using many keys with the same start column.
- Don't use table scan with BerkeleyDB and InnoDB tables when we can use an index that covers the whole row.
- Optimised InnoDB sort-buffer handling to take less memory.
- Fixed bug in multi-table DELETE and InnoDB tables.
- Fixed problem with TRUNCATE and InnoDB tables that produced the error Can't execute the given command because you have active locked tables or an active transaction .
- Added NO_UNSIGNED_SUBTRACTION to the set of flags that may be specified with the --sql-mode option for mysqld . It disables unsigned arithmetic rules when it comes to subtraction. (This will make MySQL 4.0 behave more closely to 3.23 with UNSIGNED columns).
- The result returned for all bit functions ( | , << , ...) is now of type unsigned integer .
- Added detection of nan values in MyISAM to make it possible to repair tables with nan in float or double columns.
- Fixed new bug in myisamchk where it didn't correctly update number of ``parts'' in the MyISAM index file.
- Changed to use autoconf 2.52 (from autoconf 2.13).
- Fixed optimisation problem where the MySQL Server was in ``preparing'' state for a long time when selecting from an empty table which had contained a lot of rows.
- Fixed bug in complicated join with const tables. This fix also improves performance a bit when referring to another table from a const table.
- First pre-version of multi-table UPDATE statement.
- Fixed bug in multi-table DELETE .
- Fixed bug in SELECT CONCAT(argument_list) ... GROUP BY 1 .
- INSERT ... SELECT did a full rollback in case of an error. Fixed so that we only roll back the last statement in the current transaction.

- Fixed bug with empty expression for boolean full-text search.
- Fixed core dump bug in updating full-text key from/to NULL .
- ODBC compatibility: Added BIT_LENGTH() function.
- Fixed core dump bug in GROUP BY BINARY column .
- Added support for NULL keys in HEAP tables.
- Use index for ORDER BY in queries of type: SELECT * FROM t WHERE key_part1=1 ORDER BY key_part1 DESC,key_part2 DESC
- Fixed bug in FLUSH QUERY CACHE .
- Added CAST() and CONVERT() functions. The CAST and CONVERT functions are nearly identical and mainly useful when you want to create a column with a specific type in a CREATE ... SELECT statement. For more information, read Fonctions de transtypage .
- CREATE ... SELECT on DATE and TIME functions now create columns of the expected type.
- Changed order in which keys are created in tables.
- Added new columns Null and Index_type to SHOW INDEX output.
- Added --no-beep and --prompt options to mysql command-line client.
- New feature: management of user resources.

```
GRANT ... WITH MAX_QUERIES_PER_HOUR N1
              MAX_UPDATES_PER_HOUR N2
              MAX_CONNECTIONS_PER_HOUR N3;
```

  Limiter les ressources utilisateur .
- Added mysql_secure_installation to the scripts/ directory.

## 13.3.13 Changes in release 4.0.1 (23 Dec 2001)

- Fixed bug when HANDLER was used with some unsupported table type.
- mysqldump now puts ALTER TABLE tbl_name DISABLE KEYS and ALTER TABLE tbl_name ENABLE KEYS in the sql dump.
- Added mysql_fix_extensions script.
- Fixed stack overrun problem with LOAD DATA FROM MASTER on OSF/1.
- Fixed shutdown problem on HP-UX.
- Added DES_ENCRYPT() and DES_DECRYPT() functions.
- Added FLUSH DES_KEY_FILE statement.
- Added --des-key-file option to mysqld .
- HEX(string) now returns the characters in string converted to hexadecimal.
- Fixed problem with GRANT when using lower_case_table_names=1 .
- Changed SELECT ... IN SHARE MODE to SELECT ... LOCK IN SHARE MODE (as in MySQL 3.23).
- A new query cache to cache results from identical SELECT queries.
- Fixed core dump bug on 64-bit machines when it got an incorrect communication packet.
- MATCH ... AGAINST(... IN BOOLEAN MODE) can now work without FULLTEXT index.
- Fixed slave to replicate from 3.23 master.
- Miscellaneous replication fixes/cleanup.
- Got shutdown to work on Mac OS X.
- Added myisam/ft_dump utility for low-level inspection of FULLTEXT indexes.
- Fixed bug in DELETE ... WHERE ... MATCH ... .
- Added support for MATCH ... AGAINST(... IN BOOLEAN MODE) . **Note: you must rebuild your tables with** ALTER TABLE tablename TYPE=MyISAM **to be able to use boolean full-text search** .
- LOCATE() and INSTR() are now case-sensitive if either argument is a binary string.
- Changed RAND() initialisation so that RAND(N) and RAND(N+1) are more distinct.
- Fixed core dump bug in UPDATE ... ORDER BY .
- Changed INSERT INTO ... SELECT to stop on errors by default.
- Ignore DATA DIRECTORY and INDEX DIRECTORY directives on Windows.
- Added boolean full-text search code. It should be considered early alpha.
- Extended MODIFY and CHANGE in ALTER TABLE to accept the FIRST and AFTER keywords.
- Indexes are now used with ORDER BY on a whole InnoDB table.

## 13.3.14 Changes in release 4.0.0 (Oct 2001: Alpha)

- Added --xml option to mysql for producing XML output.
- Added full-text variables ft_min_word_len , ft_max_word_len , and ft_max_word_len_for_sort .
- Added documentation for libmysqld , the embedded MySQL server library. Also added example programs (a mysql client and mysqltest test program) which use libmysqld .
- Removed all Gemini hooks from MySQL server.
- Removed my_thread_init() and my_thread_end() from mysql_com.h , and added mysql_thread_init() and mysql_thread_end() to mysql.h .
- Support for communication packets > 16M. In 4.0.1 we will extend MyISAM to be able to handle these.
- Secure connections (with SSL).
- Unsigned BIGINT constants now work. MIN() and MAX() now handle signed and unsigned BIGINT numbers correctly.
- New character set latin1_de which provides correct German sorting.
- STRCMP() now uses the current character set when doing comparisons, which means that the default comparison behaviour now is case-insensitive.
- TRUNCATE TABLE and DELETE FROM tbl_name are now separate functions. One bonus is that DELETE FROM tbl_name now returns the number of deleted rows, rather than zero.
- DROP DATABASE now executes a DROP TABLE on all tables in the database, which fixes a problem with InnoDB tables.
- Added support for UNION .
- Added support for multi-table DELETE operations.
- A new HANDLER interface to MyISAM tables.
- Added support for INSERT on MERGE tables. Patch from Benjamin Pflugmann.
- Changed WEEK(#,0) to match the calendar in the USA.
- COUNT(DISTINCT) is about 30% faster.
- Speed up all internal list handling.
- Speed up IS NULL , ISNULL() and some other internal primitives.
- Full-text index creation now is much faster.
- Tree-like cache to speed up bulk inserts and myisam_bulk_insert_tree_size variable.
- Searching on packed ( CHAR / VARCHAR ) keys is now much faster.
- Optimised queries of type: SELECT DISTINCT * from tbl_name ORDER by key_part1 LIMIT # .
- SHOW CREATE TABLE now shows all table attributes.
- ORDER BY ... DESC can now use keys.
- LOAD DATA FROM MASTER ``auto-magically'' sets up a slave.
- Renamed safe_mysqld to mysqld_safe to make this name more in line with other MySQL scripts/commands.
- Added support for symbolic links to MyISAM tables. Symlink handling is now enabled by default for Windows.
- Added SQL_CALC_FOUND_ROWS and FOUND_ROWS() . This makes it possible to know how many rows a query would have returned without a LIMIT clause.
- Changed output format of SHOW OPEN TABLES .
- Allow SELECT expression LIMIT ... .
- Added IDENTITY as a synonym for AUTO_INCREMENT (like Sybase).

- Added ORDER BY syntax to UPDATE and DELETE .
- SHOW INDEXES is now a synonym for SHOW INDEX .
- Added ALTER TABLE tbl_name DISABLE KEYS and ALTER TABLE tbl_name ENABLE KEYS commands.
- Allow use of IN as a synonym for FROM in SHOW commands.
- Implemented ``repair by sort'' for FULLTEXT indexes. REPAIR TABLE , ALTER TABLE , and OPTIMIZE TABLE for tables with FULLTEXT indexes are now up to 100 times faster.
- Allow ANSI SQL syntax X'hexadecimal-number' .
- Cleaned up global lock handling for FLUSH TABLES WITH READ LOCK .
- Fixed problem with DATETIME = constant in WHERE optimisation.
- Added --master-data and --no-autocommit options to mysqldump . (Thanks to Brian Aker for this.)
- Added script mysql_explain_log.sh to distribution. (Thanks to mobile.de).

# 13.4 Changes in release 3.23.x (Recent; still supported)

Please note that since release 4.0 is now production level, only critical fixes are done in the 3.23 release series. You are recommended to upgrade when possible, to take advantage of all speed and feature improvements in 4.0. Upgrading From Version 3.23 to 4.0 .

The 3.23 release has several major features that are not present in previous versions. We have added three new table types:

MyISAM
: A new ISAM library which is tuned for SQL and supports large files.

InnoDB
: A transaction-safe storage engine that supports row level locking, and many Oracle-like features.

BerkeleyDB *or* BDB
: Uses the Berkeley DB library from Sleepycat Software to implement transaction-safe tables.

Note that only MyISAM is available in the standard binary distribution.The 3.23 release also includes support for database replication between a master and many slaves, full-text indexing, and much more.

All new features are being developed in the 4.x version. Only bug fixes and minor enhancements to existing features will be added to 3.23.

The replication code and BerkeleyDB code is still not as tested and as the rest of the code, so we will probably need to do a couple of future releases of 3.23 with small fixes for this part of the code. As long as you don't use these features, you should be quite safe with MySQL 3.23!

Note that the above doesn't mean that replication or Berkeley DB don't work. We have done a lot of testing of all code, including replication and BDB without finding any problems. It only means that not as many users use this code as the rest of the code and because of this we are not yet 100% confident in this code.

## 13.4.1 Changes in release 3.23.57 (not released yet)

- Fixed 64 bit bug that affected at least AMD hammer systems.
- Fixed a bug when doing LOAD DATA INFILE IGNORE : when reading the binary log, mysqlbinlog and the replication code read REPLACE instead of IGNORE . This could make the slave's table become different from the master's table. Bug #218.
- Fixed a bug in MyISAM when a row is inserted into a table with a large number of NULL columns. Bug was caused by wrong calculation of the record length, as the space required for storage of NULL bits was not added to the total record length.
- Binary log was not locked during a TRUNCATE table_name or DELETE FROM table_name which could cause an INSERT to table_name to be written to the binary log before the TRUNCATE / DELETE command.
- Correction d'un bogue rare avec la commande UPDATE sur les tables InnoDB , où une ligne pouvait être modifiées plusieurs fois.
- Affiche une erreur pour les noms de tables et de colonne vides.
- Changed PROCEDURE ANALYSE() to report DATE instead of NEWDATE .
- Changed PROCEDURE ANALYSE(#) to restrict number of values in enum to # also for string values.
- mysqldump does not silently delete the binlogs when called with --master-data or --first-slave anymore; while this behaviour was convenient for some users, others may suffer from it. Now one has to explicitely ask for this deletion with the new --delete-master-logs option.
- Fixed a bug in mysqldump when it was called with --master-data : the CHANGE MASTER TO commands appended to the SQL dump had wrong coordinates (bug #159).

## 13.4.2 Changes in release 3.23.56 (13 Mar 2003)

- Fixed mysqld crash on extremely small values of sort_buffer variable.
- Fixed a bug in privilege system for GRANT UPDATE on column level.
- Fixed a rare bug when using a date in HAVING with GROUP BY .
- Fixed checking of random part of WHERE clause (Bug #142).
- Fixed MySQL (and myisamchk ) crash on artificially corrupted .MYI files.
- Security enhancement: mysqld no longer reads options from world-writeable config files.
- Security enhancement: mysqld and safe_mysqld now only use the first --user option specified on the command line. (Normally this comes from /etc/my.cnf )
- Security enhancement: Don't allow BACKUP TABLE to overwrite existing files.
- Fixed unlikely deadlock bug when one thread did a LOCK TABLE and another thread did a DROP TABLE . In this case one could do

a KILL on one of the threads to resolve the deadlock.
- LOAD DATA INFILE was not replicated by slave if replicate_*_table was set on the slave.
- Fixed a bug in handling CHAR(0) columns that could cause wrong results from the query.
- Fixed a bug in SHOW VARIABLES on 64-bit platforms. The bug was caused by wrong declaration of variable server_id .
- The Comment column in SHOW TABLE STATUS now reports that it can contain NULL values (which is the case for a crashed .frm file).
- Fixed the rpl_rotate_logs test to not fail on certain platforms (e.g. Mac OS X) due to a too long file name (changed slave-master-info.opt to .slave-mi ).
- Fixed a problem with BLOB NOT NULL columns used with IS NULL .
- Fixed bug in MAX() optimization in MERGE tables.
- Better RAND() initialization for new connections.
- Fixed bug with connect timeout. This bug was manifested on OS's with poll() system call, which resulted in timeout the value specified as it was executed in both select() and poll() .
- Fixed bug in SELECT * FROM table WHERE datetime1 IS NULL OR datetime2 IS NULL .
- Fixed bug in using aggregate functions as argument for INTERVAL , CASE , FIELD , CONCAT_WS , ELT and MAKE_SET functions.
- When running with --lower-case-table-names (default on windows) and you had tables or databases with mixed case on disk, then executing SHOW TABLE STATUS followed with DROP DATABASE or DROP TABLE could fail with Errcode 13 .
- Fixed bug in logging to binary log (which affects replication) a query that inserts a NULL in an auto_increment field and also uses LAST_INSERT_ID() .
- Fixed bug in mysqladmin --relative .
- On some 64 bit systems, show status reported a strange number for Open_files and Open_streams .

## 13.4.3 Changes in release 3.23.55 (23 Jan 2003)

- Fixed double free 'd pointer bug in mysql_change_user() handling, that enabled a specially hacked version of MySQL client to crash mysqld . **Note** , that one needs to login to the server by using a valid user account to be able to exploit this bug.
- Fixed bug with the --slow-log when logging an administrator command (like FLUSH TABLES ).
- Fixed bug in GROUP BY when used on BLOB column with NULL values.
- Fixed a bug in handling NULL s in CASE ... WHEN ... .
- Bugfix for --chroot ( Changes in release 3.23.54 (05 Dec 2002) ) is reverted. Unfortunately, there is no way to make it to work, without introducing backward-incompatible changes in my.cnf . Those who need --chroot functionality, should upgrade to MySQL 4.0. (The fix in the 4.0 branch did not break backward-compatibility).
- Make --lower-case-table-names default on Mac OS X as the file system is case insensitive.
- Fixed a bug in scripts/mysqld_safe.sh in NOHUP_NICENESS testing.
- Transactions in AUTOCOMMIT=0 mode didn't rotate binary log.
- Fixed a bug in scripts/make_binary_distribution that resulted in a remaining @HOSTNAME@} variable instead of replacing it with the correct path to the @code{hostname binary.
- Fixed a very unlikely bug that could cause SHOW PROCESSLIST to core dump in pthread_mutex_unlock() if a new thread was connecting.
- Forbid SLAVE STOP if the thread executing the query has locked tables. This removes a possible deadlock situation.

## 13.4.4 Changes in release 3.23.54 (05 Dec 2002)

- Fixed a bug, that allowed to crash mysqld with a specially crafted packet.
- Fixed a rare crash (double free 'd pointer) when altering a temporary table.
- Fixed buffer overrun in libmysqlclient library that allowed malicious MySQL server to crash the client application.
- Fixed security-related bug in mysql_change_user() handling. All users are strongly recommended to upgrade to the version 3.23.54.
- Fixed bug that prevented --chroot command-line option of mysqld from working.
- Fixed bug that made OPTIMIZE TABLE to corrupt the table under some rare circumstances.
- Fixed mysqlcheck so it can deal with table names containing dashes.
- Fixed shutdown problem on Mac OS X.
- Fixed bug with comparing an indexed NULL field with <=> NULL .
- Fixed bug that caused IGNORE INDEX and USE INDEX sometimes to be ignored.
- Fixed rare core dump problem in complicated GROUP BY queries that didn't return any result.
- Fixed a bug where MATCH ... AGAINST () >=0 was treated as if it was > .
- Fixed core dump in SHOW PROCESSLIST when running with an active slave (unlikely timing bug).
- Allow one to start multiple MySQL servers on windows (code backported from 4.0.2).
- One can create TEMPORARY MERGE tables now.
- Fixed that --core-file works on Linux (at least on kernel 2.4.18).
- Fixed a problem with BDB and ALTER TABLE .
- Fixed reference to freed memory when doing complicated GROUP BY ... ORDER BY queries. Symptom was that mysqld died in function send_fields .
- Allocate heap rows in smaller blocks to get better memory usage.
- Fixed memory allocation bug when storing BLOB values in internal temporary tables used for some (unlikely) GROUP BY queries.
- Fixed a bug in key optimizing handling where the expression WHERE column_name = key_column_name was calculated as true for NULL values.
- Fixed core dump bug when doing LEFT JOIN ... WHERE key_column=NULL .
- Fixed MyISAM crash when using dynamic-row tables with huge numbers of packed fields.
- Updated source tree to be built using automake 1.5 and libtool 1.4 .

## 13.4.5 Changes in release 3.23.53 (09 Oct 2002)

- Fixed crash when SHOW INNODB STATUS was used and skip-innodb was defined.
- Fixed possible memory corruption bug in binary log file handling when slave rotated the logs (only affected 3.23, not 4.0).
- Fixed problem in LOCK TABLES on windows when one connects to a database that contains upper case letters.
- Fixed that --skip-show-databases doesn't reset the --port option.
- Small fix in safe_mysqld for some shells.
- Fixed that FLUSH STATUS doesn't reset Delayed_insert_threads .
- Fixed core dump bug when using the BINARY cast on a NULL value.
- Fixed race condition when someone did a GRANT at the same time a new user logged in or did a USE DATABASE .
- Fixed bug in ALTER TABLE and RENAME TABLE when running with -O lower_case_table_names=1 (typically on windows) when giving the table name in uppercase.
- Fixed that -O lower_case_table_names=1 also converts database names to lower case.
- Fixed unlikely core dump with SELECT ... ORDER BY ... LIMIT .
- Changed AND/OR to report that they can return NULL. This fixes a bug in GROUP BY on AND/OR expressions that return NULL .
- Fixed a bug that OPTIMIZE of locked and modified MyISAM table, reported table corruption.
- Fixed a BDB -related ALTER TABLE bug with dropping a column and shutting down immediately thereafter.

- Fixed problem with configure ... --localstatedir=... .
- Fixed problem with UNSIGNED BIGINT on AIX (again).
- Fixed bug in pthread_mutex_trylock() on HPUX 11.0.
- Tests de stress multi-threads pour InnoDB .

## 13.4.6 Changes in release 3.23.52 (14 Aug 2002)

- Wrap BEGIN / COMMIT around transaction in the binary log. This makes replication honour transactions.
- Fixed security bug when having an empty database name in the user.db table.
- Changed initialisation of RND() to make it less predicatable.
- Fixed problem with GROUP BY on result with expression that created a BLOB field.
- Fixed problem with GROUP BY on columns that have NULL values. To solve this we now create an MyISAM temporary table when doing a GROUP BY on a possible NULL item. From MySQL 4.0.5 we can use in memory HEAP tables for this case.
- Fixed problem with privilege tables when downgrading from 4.0.2 to 3.23.
- Fixed thread bug in SLAVE START , SLAVE STOP and automatic repair of MyISAM tables that could cause table cache to be corrupted.
- Fixed possible thread related key-cache-corruption problem with OPTIMIZE TABLE and REPAIR TABLE .
- Added name of 'administrator command' logs.
- Fixed bug with creating an auto-increment value on second part of a UNIQUE() key where first part could contain NULL values.
- Don't write slave-timeout reconnects to the error log.
- Fixed bug with slave net read timeouting
- Fixed a core-dump bug with MERGE tables and MAX() function.
- Fixed bug in ALTER TABLE with BDB tables.
- Fixed bug when logging LOAD DATA INFILE to binary log with no active database.
- Fixed a bug in range optimiser (causing crashes).
- Fixed possible problem in replication when doing DROP DATABASE on a database with InnoDB tables.
- Fixed that mysql_info() returns 0 for 'Duplicates' when using INSERT DELAYED IGNORE .
- Added -DHAVE_BROKEN_REALPATH to the Mac OS X (darwin) compile options in configure.in to fix a failure under high load.

## 13.4.7 Changes in release 3.23.51 (31 May 2002)

- Fix bug with closing tags missing slash for mysqldump XML output.
- Remove end space from ENUM values. (This fixed a problem with SHOW CREATE TABLE .)
- Fixed bug in CONCAT_WS() that cut the result.
- Changed name of server variables Com_show_master_stat to Com_show_master_status and Com_show_slave_stat to Com_show_slave_status .
- Changed handling of gethostbyname() to make the client library thread-safe even if gethostbyname_r doesn't exist.
- Fixed core-dump problem when giving a wrong password string to GRANT .
- Fixed bug in DROP DATABASE with symlinked directory.
- Fixed optimisation problem with DATETIME and value outside DATETIME range.
- Removed Sleepycat's BDB doc files from the source tree, as they're not needed (MySQL covers BDB in its own documentation).
- Fixed MIT-pthreads to compile with glibc 2.2 (needed for make dist ).
- Fixed the FLOAT(X+1,X) is not converted to FLOAT(X+2,X) . (This also affected DECIMAL , DOUBLE and REAL types)
- Fixed the result from IF() is case in-sensitive if the second and third arguments are case sensitive.
- Fixed core dump problem on OSF/1 in gethostbyname_r .
- Fixed that underflowed decimal fields are not zero filled.
- If we get an overflow when inserting '+11111' for DECIMAL(5,0) UNSIGNED columns, we will just drop the sign.
- Fixed optimisation bug with ISNULL(expression_which_cannot_be_null) and ISNULL(constant_expression) .
- Fixed host lookup bug in the glibc library that we used with the 3.23.50 Linux-x86 binaries.

## 13.4.8 Changes in release 3.23.50 (21 Apr 2002)

- Fixed buffer overflow problem if someone specified a too long datadir parameter to mysqld
- Add missing <row> tags for mysqldump XML output.
- Fixed problem with crash-me and gcc 3.0.4.
- Fixed that @@unknown_variable doesn't hang server.
- Added @@VERSION as a synonym for VERSION() .
- SHOW VARIABLES LIKE 'xxx' is now case-insensitive.
- Fixed timeout for GET_LOCK() on HP-UX with DCE threads.
- Fixed memory allocation bug in the glibc library used to build Linux binaries, which caused mysqld to die in 'free()'.
- Fixed SIGINT and SIGQUIT problems in mysql .
- Fixed bug in character table converts when used with big ( > 64K) strings.
- InnoDB now retains foreign key constraints through ALTER TABLE and CREATE/DROP INDEX .
- InnoDB now allows foreign key constraints to be added through the ALTER TABLE syntax.
- InnoDB tables can now be set to automatically grow in size (autoextend).
- Our Linux RPMS and binaries are now compiled with gcc 3.0.4, which should make them a bit faster.
- Fixed some buffer overflow problems when reading startup parameters.
- Because of problems on shutdown we have now disabled named pipes on windows by default. One can enable named pipes by starting mysqld with --enable-named-pipe .
- Fixed bug when using WHERE key_column = 'J' or key_column='j' .
- Fixed core-dump bug when using --log-bin with LOAD DATA INFILE without an active database.
- Fixed bug in RENAME TABLE when used with lower_case_table_names=1 (default on Windows).
- Fixed unlikely core-dump bug when using DROP TABLE on a table that was in use by a thread that also used queries on only temporary tables.
- Fixed problem with SHOW CREATE TABLE and PRIMARY KEY when using 32 indexes.
- Fixed that one can use SET PASSWORD for the anonymous user.
- Fixed core dump bug when reading client groups from option files using mysql_options() .
- Memory leak (16 bytes per every **corrupted** table) closed.
- Fixed binary builds to use --enable-local-infile .
- Update source to work with new version of bison .
- Updated shell scripts to now agree with new POSIX standard.
- Fixed bug where DATE_FORMAT() returned empty string when used with GROUP BY .

## 13.4.9 Changes in release 3.23.49

- Don't give warning for a statement that is only a comment; this is needed for mysqldump --disable-keys to work.
- Fixed unlikely caching bug when doing a join without keys. In this case the last used field for a table always returned NULL .
- Added options to make LOAD DATA LOCAL INFILE more secure.
- MySQL binary release 3.23.48 for Linux contained a new glibc library, which has serious problems under high load and RedHat 7.2. The 3.23.49 binary release doesn't have this problem.
- Fixed shutdown problem on NT.

## 13.4.10 Changes in release 3.23.48 (07 Feb 2002)

- Added --xml option to mysqldump for producing XML output.
- Changed to use autoconf 2.52 (from autoconf 2.13)
- Fixed bug in complicated join with const tables.
- Added internal safety checks for InnoDB .
- Some InnoDB variables were always shown in SHOW VARIABLES as OFF on high-byte-first systems (like SPARC).
- Fixed problem with one thread using an InnoDB table and another thread doing an ALTER TABLE on the same table. Before that, mysqld could crash with an assertion failure in row0row.c , line 474.
- Tuned the InnoDB SQL optimiser to favor index searches more often over table scans.
- Fixed a performance problem with InnoDB tables when several large SELECT queries are run concurrently on a multiprocessor Linux computer. Large CPU-bound SELECT queries will now also generally run faster on all platforms.
- If MySQL binlogging is used, InnoDB now prints after crash recovery the latest MySQL binlog name and the offset InnoDB was able to recover to. This is useful, for example, when resynchronising a master and a slave database in replication.
- Added better error messages to help in installation problems of InnoDB tables.
- It is now possible to recover MySQL temporary tables that have become orphaned inside the InnoDB tablespace.
- InnoDB now prevents a FOREIGN KEY declaration where the signedness is not the same in the referencing and referenced integer columns.
- Calling SHOW CREATE TABLE or SHOW TABLE STATUS could cause memory corruption and make mysqld crash. Especially at risk was mysqldump , because it frequently calls SHOW CREATE TABLE .
- If inserts to several tables containing an AUTO_INCREMENT column were wrapped inside one LOCK TABLES , InnoDB asserted in lock0lock.c .
- In 3.23.47 we allowed several NULL values in a UNIQUE secondary index for an InnoDB table. But CHECK TABLE was not relaxed: it reports the table as corrupt. CHECK TABLE no longer complains in this situation.
- SHOW GRANTS now shows REFERENCES instead of REFERENCE .

## 13.4.11 Changes in release 3.23.47 (27 Dec 2001)

- Fixed bug when using the following construct: SELECT ... WHERE key=@var_name OR key=@var_name2
- Restrict InnoDB keys to 500 bytes.
- InnoDB now supports NULL in keys.
- Fixed shutdown problem on HP-UX. (Introduced in 3.23.46)
- Fixed core dump bug in replication when using SELECT RELEASE_LOCK() .
- Added new command: DO expression,[expression]
- Added slave-skip-errors option.
- Added statistics variables for all MySQL commands. ( SHOW STATUS is now much longer.)
- Fixed default values for InnoDB tables.
- Fixed that GROUP BY expr DESC works.
- Fixed bug when using t1 LEFT JOIN t2 ON t2.key=constant .
- mysql_config now also works with binary (relocated) distributions.

## 13.4.12 Changes in release 3.23.46 (29 Nov 2001)

- Fixed problem with aliased temporary table replication.
- InnoDB and BDB tables will now use index when doing an ORDER BY on the whole table.
- Fixed bug where one got an empty set instead of a DEADLOCK error when using BDB tables.
- One can now kill ANALYZE , REPAIR , and OPTIMIZE TABLE when the thread is waiting to get a lock on the table.
- Fixed race condition in ANALYZE TABLE .
- Fixed bug when joining with caching (unlikely to happen).
- Fixed race condition when using the binary log and INSERT DELAYED which could cause the binary log to have rows that were not yet written to MyISAM tables.
- Changed caching of binary log to make replication slightly faster.
- Fixed bug in replication on Mac OS X.

## 13.4.13 Changes in release 3.23.45 (22 Nov 2001)

- (UPDATE|DELETE) ...WHERE MATCH bugfix.
- shutdown should now work on Darwin (Mac OS X).
- Fixed core dump when repairing corrupted packed MyISAM files.
- --core-file now works on Solaris.
- Fix a bug which could cause InnoDB to complain if it cannot find free blocks from the buffer cache during recovery.
- Fixed bug in InnoDB insert buffer B-tree handling that could cause crashes.
- Fixed bug in InnoDB lock timeout handling.
- Fixed core dump bug in ALTER TABLE on a TEMPORARY InnoDB table.
- Fixed bug in OPTIMIZE TABLE that reset index cardinality if it was up to date.
- Fixed problem with t1 LEFT_JOIN t2 ... WHERE t2.date_column IS NULL when date_column was declared as NOT NULL .
- Fixed bug with BDB tables and keys on BLOB columns.
- Fixed bug in MERGE tables on OS with 32-bit file pointers.
- Fixed bug in TIME_TO_SEC() when using negative values.

## 13.4.14 Changes in release 3.23.44 (31 Oct 2001)

- Fixed Rows_examined count in slow query log.
- Fixed bug when using a reference to an AVG() column in HAVING .
- Fixed that date functions that require correct dates, like DAYOFYEAR(column) , will return NULL for 0000-00-00 dates.
- Fixed bug in const-propagation when comparing columns of different types. ( SELECT * FROM date_col="2001-01-01" and date_col=time_col )
- Fixed bug that caused error message Can't write, because of unique constraint with some GROUP BY queries.
- Fixed problem with sjis character strings used within quoted table names.
- Fixed core dump when using CREATE ... FULLTEXT keys with other table handlers than MyISAM .
- Don't use signal() on Windows because this appears to not be 100% reliable.
- Fixed bug when doing WHERE col_name=NULL on an indexed column that had NULL values.
- Fixed bug when doing LEFT JOIN ... ON (col_name = constant) WHERE col_name = constant .
- When using replications, aborted queries that contained % could cause a core dump.
- TCP_NODELAY was not used on some systems. (Speed problem.)
- Applied portability fixes for OS/2. (Patch by Yuri Dario.)

The following changes are for InnoDB tables:

- Add missing InnoDB variables to SHOW VARIABLES .
- Foreign keys checking is now done for InnoDB tables.
- DROP DATABASE now works also for InnoDB tables.
- InnoDB now supports datafiles and raw disk partitions bigger than 4 GB on those operating systems that have big files.
- InnoDB calculates better table cardinality estimates for the MySQL optimiser.
- Accent characters in the default character set latin1 are ordered according to the MySQL ordering.Note: if you are using latin1 and have inserted characters whose code is greater than 127 into an indexed CHAR column, you should run CHECK TABLE on your table when you upgrade to 3.23.44, and drop and reimport the table if CHECK TABLE reports an error!
- A new my.cnf parameter, innodb_thread_concurrency , helps in performance tuning in heavily concurrent environments.
- A new my.cnf parameter, innodb_fast_shutdown , speeds up server shutdown.
- A new my.cnf parameter, innodb_force_recovery , helps to save your data in case the disk image of the database becomes corrupt.
- innodb_monitor has been improved and a new innodb_table_monitor added.
- Increased maximum key length from 500 to 7000 bytes.
- Fixed a bug in replication of AUTO_INCREMENT columns with multiple-line inserts.
- Fixed a bug when the case of letters changes in an update of an indexed secondary column.
- Fixed a hang when there are > 24 datafiles.
- Fixed a crash when MAX(col) is selected from an empty table, and col is not the first column in a multi-column index.
- Fixed a bug in purge which could cause crashes.

## 13.4.15 Changes in release 3.23.43 (04 Oct 2001)

- Fixed a bug in INSERT DELAYED and FLUSH TABLES introduced in 3.23.42.
- Fixed unlikely bug, which returned non-matching rows, in SELECT with many tables and multi-column indexes and 'range' type.
- Fixed an unlikely core dump bug when doing EXPLAIN SELECT when using many tables and ORDER BY .
- Fixed bug in LOAD DATA FROM MASTER when using table with CHECKSUM=1 .
- Added unique error message when one gets a DEADLOCK during a transaction with BDB tables.
- Fixed problem with BDB tables and UNIQUE columns defined as NULL .
- Fixed problem with myisampack when using pre-space filled CHAR columns.
- Applied patch from Yuri Dario for OS/2.
- Fixed bug in --safe-user-create .

## 13.4.16 Changes in release 3.23.42 (08 Sep 2001)

- Fixed problem when using LOCK TABLES and BDB tables.
- Fixed problem with REPAIR TABLE on MyISAM tables with row lengths in the range from 65517 to 65520 bytes.
- Fixed rare hang when doing mysqladmin shutdown when there was a lot of activity in other threads.
- Fixed problem with INSERT DELAYED where delay thread could be hanging on upgrading locks with no apparent reason.
- Fixed problem with myisampack and BLOB .
- Fixed problem when one edited .MRG tables by hand. (Patch from Benjamin Pflugmann).
- Enforce that all tables in a MERGE table come from the same database.
- Fixed bug with LOAD DATA INFILE and transactional tables.
- Fix bug when using INSERT DELAYED with wrong column definition.
- Fixed core dump during REPAIR of some particularly broken tables.
- Fixed bug in InnoDB and AUTO_INCREMENT columns.
- Fixed bug in InnoDB and RENAME TABLE columns.
- Fixed critical bug in InnoDB and BLOB columns. If you have used BLOB columns larger than 8000 bytes in an InnoDB table, it is necessary to dump the table with mysqldump , drop it and restore it from the dump.
- Applied large patch for OS/2 from Yuri Dario.
- Fixed problem with InnoDB when one could get the error Can't execute the given command... even when no transaction was active.
- Applied some minor fixes that concern Gemini.
- Use real arithmetic operations even in integer context if not all arguments are integers. (Fixes uncommon bug in some integer contexts).
- Don't force everything to lowercase on Windows. (To fix problem with Windows and ALTER TABLE ). Now --lower_case_names also works on Unix.
- Fixed that automatic rollback is done when thread end doesn't lock other threads.

## 13.4.17 Changes in release 3.23.41 (11 Aug 2001)

- Added --sql-mode=option[,option[,option]] option to mysqld . Options en ligne de commande de mysqld .
- Fixed possible problem with shutdown on Solaris where the .pid file wasn't deleted.
- InnoDB now supports < 4 GB rows. The former limit was 8000 bytes.
- The doublewrite file flush method is used in InnoDB . It reduces the need for Unix fsync() calls to a fraction and improves performance on most Unix flavors.
- You can now use the InnoDB Monitor to print a lot of InnoDB state information, including locks, to the standard output. This is useful in performance tuning.

- Several bugs which could cause hangs in InnoDB have been fixed.
- Split record_buffer to record_buffer and record_rnd_buffer . To make things compatible to previous MySQL versions, if record_rnd_buffer is not set, then it takes the value of record_buffer .
- Fixed optimising bug in ORDER BY where some ORDER BY parts where wrongly removed.
- Fixed overflow bug with ALTER TABLE and MERGE tables.
- Added prototypes for my_thread_init() and my_thread_end() to mysql_com.h
- Added --safe-user-create option to mysqld .
- Fixed bug in SELECT DISTINCT ... HAVING that caused error message Can't find record in #...

## 13.4.18 Changes in release 3.23.40

- Fixed problem with --low-priority-updates and INSERT statements.
- Fixed bug in slave thread when under some rare circumstances it could get 22 bytes ahead on the offset in the master.
- Added slave_net_timeout for replication.
- Fixed problem with UPDATE and BDB tables.
- Fixed hard bug in BDB tables when using key parts.
- Fixed problem when using GRANT FILE ON database.* ... ; previously we added the DROP privilege for the database.
- Fixed DELETE FROM tbl_name ... LIMIT 0 and UPDATE FROM tbl_name ... LIMIT 0 , which acted as though the LIMIT clause was not present (they deleted or updated all selected rows).
- CHECK TABLE now checks if an AUTO_INCREMENT column contains the value 0.
- Sending a SIGHUP to mysqld will now only flush the logs, not reset the replication.
- Fixed parser to allow floats of type 1.0e1 (no sign after e ).
- Option --force to myisamchk now also updates states.
- Added option --warnings to mysqld . Now mysqld prints the error Aborted connection only if this option is used.
- Fixed problem with SHOW CREATE TABLE when you didn't have a PRIMARY KEY .
- Properly fixed the rename of innodb_unix_file_flush_method variable to innodb_flush_method .
- Fixed bug when converting BIGINT UNSIGNED to DOUBLE . This caused a problem when doing comparisons with BIGINT values outside of the signed range.
- Fixed bug in BDB tables when querying empty tables.
- Fixed a bug when using COUNT(DISTINCT) with LEFT JOIN and there weren't any matching rows.
- Removed all documentation referring to the GEMINI table type. GEMINI is not released under an Open Source license.

## 13.4.19 Changes in release 3.23.39 (12 Jun 2001)

- The AUTO_INCREMENT sequence wasn't reset when dropping and adding an AUTO_INCREMENT column.
- CREATE ... SELECT now creates non-unique indexes delayed.
- Fixed problem where LOCK TABLES tbl_name READ followed by FLUSH TABLES put an exclusive lock on the table.
- REAL @variable values were represented with only 2 digits when converted to strings.
- Fixed problem that client ``hung'' when LOAD TABLE FROM MASTER failed.
- myisamchk --fast --force will no longer repair tables that only had the open count wrong.
- Added functions to handle symbolic links to make life easier in 4.0.
- We are now using the -lcma thread library on HP-UX 10.20 so that MySQL will be more stable on HP-UX.
- Fixed problem with IF() and number of decimals in the result.
- Fixed date-part extraction functions to work with dates where day and/or month is 0.
- Extended argument length in option files from 256 to 512 chars.
- Fixed problem with shutdown when INSERT DELAYED was waiting for a LOCK TABLE .
- Fixed core dump bug in InnoDB when tablespace was full.
- Fixed problem with MERGE tables and big tables (> 4G) when using ORDER BY .

## 13.4.20 Changes in release 3.23.38 (09 May 2001)

- Fixed a bug when SELECT from MERGE table sometimes results in incorrectly ordered rows.
- Fixed a bug in REPLACE() when using the ujis character set.
- Applied Sleepycat BDB patches 3.2.9.1 and 3.2.9.2.
- Added --skip-stack-trace option to mysqld .
- CREATE TEMPORARY now works with InnoDB tables.
- InnoDB now promotes sub keys to whole keys.
- Added option CONCURRENT to LOAD DATA .
- Better error message when slave max_allowed_packet is too low to read a very long log event from the master.
- Fixed bug when too many rows where removed when using SELECT DISTINCT ... HAVING .
- SHOW CREATE TABLE now returns TEMPORARY for temporary tables.
- Added Rows_examined to slow query log.
- Fixed problems with function returning empty string when used together with a group function and a WHERE that didn't match any rows.
- New program mysqlcheck .
- Added database name to output for administrative commands like CHECK , REPAIR , OPTIMIZE .
- Lots of portability fixes for InnoDB .
- Changed optimiser so that queries like SELECT * FROM tbl_name,tbl_name2 ... ORDER BY key_part1 LIMIT # will use index on key_part1 instead of filesort .
- Fixed bug when doing LOCK TABLE to_table WRITE,...; INSERT INTO to_table... SELECT ... when to_table was empty.
- Fixed bug with LOCK TABLE and BDB tables.

## 13.4.21 Changes in release 3.23.37 (17 Apr 2001)

- Fixed a bug when using MATCH() in HAVING clause.
- Fixed a bug when using HEAP tables with LIKE .
- Added --mysql-version option to safe_mysqld
- Changed INNOBASE to InnoDB (because the INNOBASE name was already used). All configure options and mysqld start options now use innodb instead of innobase . This means that before upgrading to this version, you have to change any configuration files where you have used innobase options!
- Fixed bug when using indexes on CHAR(255) NULL columns.
- Slave thread will now be started even if master-host is not set, as long as server-id is set and valid master.info is present.
- Partial updates (terminated with kill) are now logged with a special error code to the binary log. Slave will refuse to

execute them if the error code indicates the update was terminated abnormally, and will have to be recovered with SET SQL_SLAVE_SKIP_COUNTER=1; SLAVE START after a manual sanity check/correction of data integrity.
- Fixed bug that erroneously logged a drop of internal temporary table on thread termination to the binary log -- this bug affected replication.
- Fixed a bug in REGEXP on 64-bit machines.
- UPDATE and DELETE with WHERE unique_key_part IS NULL didn't update/delete all rows.
- Disabled INSERT DELAYED for tables that support transactions.
- Fixed bug when using date functions on TEXT / BLOB column with wrong date format.
- UDFs now also work on Windows. (Patch by Ralph Mason.)
- Fixed bug in ALTER TABLE and LOAD DATA INFILE that disabled key-sorting. These commands should now be faster in most cases.
- Fixed performance bug where reopened tables (tables that had been waiting for FLUSH or REPAIR ) would not use indexes for the next query.
- Fixed problem with ALTER TABLE to InnoDB tables on FreeBSD.
- Added mysqld variables myisam_max_sort_file_size and myisam_max_extra_sort_file_size .
- Initialise signals early to avoid problem with signals in InnoDB .
- Applied patch for the tis620 character set to make comparisons case-independent and to fix a bug in LIKE for this character set. **Note** : All tables that uses the tis620 character set must be fixed with myisamchk -r or REPAIR TABLE !
- Added --skip-safemalloc option to mysqld .

## 13.4.22 Changes in release 3.23.36 (27 Mar 2001)

- Fixed a bug that allowed use of database names containing a '.' character. This fixes a serious security issue when mysqld is run as root.
- Fixed bug when thread creation failed (could happen when doing a **lot** of connections in a short time).
- Fixed some problems with FLUSH TABLES and TEMPORARY tables. (Problem with freeing the key cache and error Can't reopen table... .)
- Fixed a problem in InnoDB with other character sets than latin1 and another problem when using many columns.
- Fixed bug that caused a core dump when using a very complex query involving DISTINCT and summary functions.
- Added SET TRANSACTION ISOLATION LEVEL ...
- Added SELECT ... FOR UPDATE .
- Fixed bug where the number of affected rows was not returned when MySQL was compiled without transaction support.
- Fixed a bug in UPDATE where keys weren't always used to find the rows to be updated.
- Fixed a bug in CONCAT_WS() where it returned incorrect results.
- Changed CREATE ... SELECT and INSERT ... SELECT to not allow concurrent inserts as this could make the binary log hard to repeat. (Concurrent inserts are enabled if you are not using the binary or update log.)
- Changed some macros to be able to use fast mutex with glibc 2.2.

## 13.4.23 Changes in release 3.23.35 (15 Mar 2001)

- Fixed newly introduced bug in ORDER BY .
- Fixed wrong define CLIENT_TRANSACTIONS .
- Fixed bug in SHOW VARIABLES when using INNOBASE tables.
- Setting and using user variables in SELECT DISTINCT didn't work.
- Tuned SHOW ANALYZE for small tables.
- Fixed handling of arguments in the benchmark script run-all-tests .

## 13.4.24 Changes in release 3.23.34a

- Added extra files to the distribution to allow INNOBASE support to be compiled.

## 13.4.25 Changes in release 3.23.34 (10 Mar 2001)

- Added the INNOBASE table handler and the BDB table handler to the MySQL source distribution.
- Updated the documentation about GEMINI tables.
- Fixed a bug in INSERT DELAYED that caused threads to hang when inserting NULL into an AUTO_INCREMENT column.
- Fixed a bug in CHECK TABLE / REPAIR TABLE that could cause a thread to hang.
- REPLACE will not replace a row that conflicts with an AUTO_INCREMENT generated key.
- mysqld now only sets CLIENT_TRANSACTIONS in mysql->server_capabilities if the server supports a transaction-safe handler.
- Fixed LOAD DATA INFILE to allow numeric values to be read into ENUM and SET columns.
- Improved error diagnostic for slave thread exit.
- Fixed bug in ALTER TABLE ... ORDER BY .
- Added max_user_connections variable to mysqld .
- Limit query length for replication by max_allowed_packet , not the arbitrary limit of 4 MB.
- Allow space around = in argument to --set-variable .
- Fixed problem in automatic repair that could leave some threads in state Waiting for table .
- SHOW CREATE TABLE now displays the UNION() for MERGE tables.
- ALTER TABLE now remembers the old UNION() definition.
- Fixed bug when replicating timestamps.
- Fixed bug in bidirectional replication.
- Fixed bug in the BDB table handler that occurred when using an index on multi-part key where a key part may be NULL .
- Fixed MAX() optimisation on sub-key for BDB tables.
- Fixed problem where garbage results were returned when using BDB tables and BLOB or TEXT fields when joining many tables.
- Fixed a problem with BDB tables and TEXT columns.
- Fixed bug when using a BLOB key where a const row wasn't found.
- Fixed that mysqlbinlog writes the timestamp value for each query. This ensures that one gets same values for date functions like NOW() when using mysqlbinlog to pipe the queries to another server.
- Allow --skip-gemini , --skip-bdb , and --skip-innodb options to be specified when invoking mysqld , even if these table handlers are not compiled in to mysqld .
- One can now do GROUP BY ... DESC .
- Fixed a deadlock in the SET code, when one ran SET @foo=bar , where bar is a column reference, an error was not properly generated.

## 13.4.26 Changes in release 3.23.33 (09 Feb 2001)

- Fixed DNS lookups not to use the same mutex as the hostname cache. This will enable known hosts to be quickly resolved even if a DNS lookup takes a long time.
- Added --character-sets-dir option to myisampack .
- Removed warnings when running REPAIR TABLE ... EXTENDED .
- Fixed a bug that caused a core dump when using GROUP BY on an alias, where the alias was the same as an existing column name.
- Added SEQUENCE() as an example UDF function.
- Changed mysql_install_db to use BINARY for CHAR columns in the privilege tables.
- Changed TRUNCATE tbl_name to TRUNCATE TABLE tbl_name to use the same syntax as Oracle. Until 4.0 we will also allow TRUNCATE tbl_name to not crash old code.
- Fixed ``no found rows'' bug in MyISAM tables when a BLOB was first part of a multi-part key.
- Fixed bug where CASE didn't work with GROUP BY .
- Added --sort-recover option to myisamchk .
- myisamchk -S and OPTIMIZE TABLE now work on Windows.
- Fixed bug when using DISTINCT on results from functions that referred to a group function, like:

```
SELECT a, DISTINCT SEC_TO_TIME(SUM(a))
FROM tbl_name GROUP BY a, b;
```

- Fixed buffer overrun in libmysqlclient library. Fixed bug in handling STOP event after ROTATE event in replication.
- Fixed another buffer overrun in DROP DATABASE .
- Added Table_locks_immediate and Table_locks_waited status variables.
- Fixed bug in replication that broke slave server start with existing master.info . This fixes a bug introduced in 3.23.32.
- Added SET SQL_SLAVE_SKIP_COUNTER=n command to recover from replication glitches without a full database copy.
- Added max_binlog_size variable; the binary log will be rotated automatically when the size crosses the limit.
- Added Last_error , Last_errno , and Slave_skip_counter variables to SHOW SLAVE STATUS .
- Fixed bug in MASTER_POS_WAIT() function.
- Execute core dump handler on SIGILL , and SIGBUS in addition to SIGSEGV .
- On x86 Linux, print the current query and thread (connection) id, if available, in the core dump handler.
- Fixed several timing bugs in the test suite.
- Extended mysqltest to take care of the timing issues in the test suite.
- ALTER TABLE can now be used to change the definition for a MERGE table.
- Fixed creation of MERGE tables on Windows.
- Portability fixes for OpenBSD and OS/2.
- Added --temp-pool option to mysqld . Using this option will cause most temporary files created to use a small set of names, rather than a unique name for each new file. This is to work around a problem in the Linux kernel dealing with creating a bunch of new files with different names. With the old behaviour, Linux seems to "leak" memory, as it's being allocated to the directory entry cache instead of the disk cache.

## 13.4.27 Changes in release 3.23.32 (22 Jan 2001: Stable)

- Changed code to get around compiler bug in Compaq C++ on OSF/1, that broke BACKUP , RESTORE , CHECK , REPAIR , and ANALYZE TABLE .
- Added option FULL to SHOW COLUMNS . Now we show the privilege list for the columns only if this option is given.
- Fixed bug in SHOW LOGS when there weren't any BDB logs.
- Fixed a timing problem in replication that could delay sending an update to the client until a new update was done.
- Don't convert field names when using mysql_list_fields() . This is to keep this code compatible with SHOW FIELDS .
- MERGE tables didn't work on Windows.
- Fixed problem with SET PASSWORD=... on Windows.
- Added missing my_config.h to RPM distribution.
- TRIM("foo" from "foo") didn't return an empty string.
- Added --with-version-suffix option to configure .
- Fixed core dump when client aborted connection without mysql_close() .
- Fixed a bug in RESTORE TABLE when trying to restore from a non-existent directory.
- Fixed a bug which caused a core dump on the slave when replicating SET PASSWORD .
- Added MASTER_POS_WAIT() .

## 13.4.28 Changes in release 3.23.31 (17 Jan 2001)

- The test suite now tests all reachable BDB interface code. During testing we found and fixed many errors in the interface code.
- Using HAVING on an empty table could produce one result row when it shouldn't.
- Fixed the MySQL RPM not to depend on Perl5 anymore.
- Fixed some problems with HEAP tables on Windows.
- SHOW TABLE STATUS didn't show correct average row length for tables larger than 4G.
- CHECK TABLE ... EXTENDED didn't check row links for fixed size tables.
- Added option MEDIUM to CHECK TABLE .
- Fixed problem when using DECIMAL() keys on negative numbers.
- HOUR() (and some other TIME functions) on a CHAR column always returned NULL .
- Fixed security bug in something (please upgrade if you are using an earlier MySQL 3.23 version).
- Fixed buffer overflow bug when writing a certain error message.
- Added usage of setrlimit() on Linux to get -O --open-files-limit=# to work on Linux.
- Added bdb_version variable to mysqld .
- Fixed bug when using expression of type:

```
SELECT ... FROM t1 LEFT JOIN t2 ON (t1.a=t2.a) WHERE t1.a=t2.a
```

  In this case the test in the WHERE clause was wrongly optimised away.
- Fixed bug in MyISAM when deleting keys with possible NULL values, but the first key-column was not a prefix-compressed text column.
- Fixed mysql.server to read the [mysql.server] option file group rather than the [mysql_server] group.
- Fixed safe_mysqld and mysql.server to also read the server option section.
- Added Threads_created status variable to mysqld .

## 13.4.29 Changes in release 3.23.30 (04 Jan 2001)

- Added SHOW OPEN TABLES command.
- Fixed that myisamdump works against old mysqld servers.
- Fixed myisamchk -k# so that it works again.
- Fixed a problem with replication when the binary log file went over 2G on 32-bit systems.
- LOCK TABLES will now automatically start a new transaction.
- Changed BDB tables to not use internal subtransactions and reuse open files to get more speed.
- Added --mysqld=# option to safe_mysqld .
- Allow hex constants in the --fields-*-by and --lines-terminated-by options to mysqldump and mysqlimport . By Paul DuBois.
- Added --safe-show-database option to mysqld .
- Added have_bdb , have_gemini , have_innobase , have_raid and have_openssl to SHOW VARIABLES to make it easy to test for supported extensions.
- Added --open-files-limit option to mysqld .
- Changed --open-files option to --open-files-limit in safe_mysqld .
- Fixed a bug where some rows were not found with HEAP tables that had many keys.
- Fixed that --bdb-no-sync works.
- Changed --bdb-recover to --bdb-no-recover as recover should be on by default.
- Changed the default number of BDB locks to 10000.
- Fixed a bug from 3.23.29 when allocating the shared structure needed for BDB tables.
- Changed mysqld_multi.sh to use configure variables. Patch by Christopher McCrory.
- Added fixing of include files for Solaris 2.8.
- Fixed bug with --skip-networking on Debian Linux.
- Fixed problem that some temporary files where reported as having the name UNOPENED in error messages.
- Fixed bug when running two simultaneous SHOW LOGS queries.

## 13.4.30 Changes in release 3.23.29 (16 Dec 2000)

- Configure updates for Tru64, large file support, and better TCP wrapper support. By Albert Chin-A-Young.
- Fixed bug in <=> operator.
- Fixed bug in REPLACE with BDB tables.
- LPAD() and RPAD() will shorten the result string if it's longer than the length argument.
- Added SHOW LOGS command.
- Remove unused BDB logs on shutdown.
- When creating a table, put PRIMARY keys first, followed by UNIQUE keys.
- Fixed a bug in UPDATE involving multi-part keys where one specified all key parts both in the update and the WHERE part. In this case MySQL could try to update a record that didn't match the whole WHERE part.
- Changed drop table to first drop the tables and then the .frm file.
- Fixed a bug in the hostname cache which caused mysqld to report the hostname as '' in some error messages.
- Fixed a bug with HEAP type tables; the variable max_heap_table_size wasn't used. Now either MAX_ROWS or max_heap_table_size can be used to limit the size of a HEAP type table.
- Changed the default server-id to 1 for masters and 2 for slaves to make it easier to use the binary log.
- Renamed bdb_lock_max variable to bdb_max_lock .
- Added support for AUTO_INCREMENT on sub-fields for BDB tables.
- Added ANALYZE of BDB tables.
- In BDB tables, we now store the number of rows; this helps to optimise queries when we need an approximation of the number of rows.
- If we get an error in a multi-row statement, we now only roll back the last statement, not the entire transaction.
- If you do a ROLLBACK when you have updated a non-transactional table you will get an error as a warning.
- Added --bdb-shared-data option to mysqld .
- Added Slave_open_temp_tables status variable to mysqld
- Added binlog_cache_size and max_binlog_cache_size variables to mysqld .
- DROP TABLE , RENAME TABLE , CREATE INDEX and DROP INDEX are now transaction endpoints.
- If you do a DROP DATABASE on a symbolically linked database, both the link and the original database is deleted.
- Fixed DROP DATABASE to work on OS/2.
- Fixed bug when doing a SELECT DISTINCT ... table1 LEFT JOIN table2 ... when table2 was empty.
- Added --abort-slave-event-count and --disconnect-slave-event-count options to mysqld for debugging and testing of replication.
- Fixed replication of temporary tables. Handles everything except slave server restart.
- SHOW KEYS now shows whether key is FULLTEXT .
- New script mysqld_multi . mysqld_multi , un programme pour gérer plusieurs serveurs MySQL .
- Added new script, mysql-multi.server.sh . Thanks to Tim Bunce Tim.Bunce@ig.co.uk for modifying mysql.server to easily handle hosts running many mysqld processes.
- safe_mysqld , mysql.server , and mysql_install_db have been modified to use mysql_print_defaults instead of various hacks to read the my.cnf files. In addition, the handling of various paths has been made more consistent with how mysqld handles them by default.
- Automatically remove Berkeley DB transaction logs that no longer are in use.
- Fixed bug with several FULLTEXT indexes in one table.
- Added a warning if number of rows changes on REPAIR / OPTIMIZE .
- Applied patches for OS/2 by Yuri Dario .
- FLUSH TABLES tbl_name didn't always flush the index tree to disk properly.
- --bootstrap is now run in a separate thread. This fixes a problem that caused mysql_install_db to core dump on some Linux machines.
- Changed mi_create() to use less stack space.
- Fixed bug with optimiser trying to over-optimise MATCH() when used with UNIQUE key.
- Changed crash-me and the MySQL benchmarks to also work with FrontBase.
- Allow RESTRICT and CASCADE after DROP TABLE to make porting easier.
- Reset status variable which could cause problem if one used --slow-log .
- Added connect_timeout variable to mysql and mysqladmin .
- Added connect-timeout as an alias for timeout for option files read by mysql_options() .

## 13.4.31 Changes in release 3.23.28 (22 Nov 2000: Gamma)

- Added new options --pager[=...] , --no-pager , --tee=... and --no-tee to the mysql client. The new corresponding interactive commands are pager , nopager , tee and notee . L'outil de ligne de commande , mysql --help and the interactive help for more information.
- Fixed crash when automatic repair of MyISAM table failed.
- Fixed a major performance bug in the table locking code when one constantly had a lot of SELECT , UPDATE and INSERT statements running. The symptom was that the UPDATE and INSERT queries were locked for a long time while new SELECT statements were executed before the updates.
- When reading options_files with mysql_options() the return-found-rows option was ignored.

- One can now specify interactive-timeout in the option file that is read by mysql_options() . This makes it possible to force programs that run for a long time (like mysqlhotcopy ) to use the interactive_timeout time instead of the wait_timeout time.
- Added to the slow query log the time and the user name for each logged query. If you are using --log-long-format then also queries that do not use an index are logged, even if the query takes less than long_query_time seconds.
- Fixed a problem in LEFT JOIN which caused all columns in a reference table to be NULL .
- Fixed a problem when using NATURAL JOIN without keys.
- Fixed a bug when using a multi-part keys where the first part was of type TEXT or BLOB .
- DROP of temporary tables wasn't stored in the update/binary log.
- Fixed a bug where SELECT DISTINCT * ... LIMIT # only returned one row.
- Fixed a bug in the assembler code in strstr() for SPARC and cleaned up the global.h header file to avoid a problem with bad aliasing with the compiler submitted with RedHat 7.0. (Reported by Trond Eivind Glomsrød)
- The --skip-networking option now works properly on NT.
- Fixed a long outstanding bug in the ISAM tables when a row with a length of more than 65K was shortened by a single byte.
- Fixed a bug in MyISAM when running multiple updating processes on the same table.
- Allow one to use FLUSH TABLE tbl_name .
- Added --replicate-ignore-table , --replicate-do-table , --replicate-wild-ignore-table , and --replicate-wild-do-table options to mysqld .
- Changed all log files to use our own IO_CACHE mechanism instead of FILE to avoid OS problems when there are many files open.
- Added --open-files and --timezone options to safe_mysqld .
- Fixed a fatal bug in CREATE TEMPORARY TABLE ... SELECT ... .
- Fixed a problem with CREATE TABLE ... SELECT NULL .
- Added variables large_file_support , net_read_timeout , net_write_timeout and query_buffer_size to SHOW VARIABLES .
- Added status variables created_tmp_files and sort_merge_passes to SHOW STATUS .
- Fixed a bug where we didn't allow an index name after the FOREIGN KEY definition.
- Added TRUNCATE table_name as a synonym for DELETE FROM table_name .
- Fixed a bug in a BDB key compare function when comparing part keys.
- Added bdb_lock_max variable to mysqld .
- Added more tests to the benchmark suite.
- Fixed an overflow bug in the client code when using overly long database names.
- mysql_connect() now aborts on Linux if the server doesn't answer in timeout seconds.
- SLAVE START did not work if you started with --skip-slave-start and had not explicitly run CHANGE MASTER TO .
- Fixed the output of SHOW MASTER STATUS to be consistent with SHOW SLAVE STATUS . (It now has no directory in the log name.)
- Added PURGE MASTER LOGS TO .
- Added SHOW MASTER LOGS .
- Added --safemalloc-mem-limit option to mysqld to simulate memory shortage when compiled with the --with-debug=full option.
- Fixed several core dumps in out-of-memory conditions.
- SHOW SLAVE STATUS was using an uninitialised mutex if the slave had not been started yet.
- Fixed bug in ELT() and MAKE_SET() when the query used a temporary table.
- CHANGE MASTER TO without specifying MASTER_LOG_POS would set it to 0 instead of 4 and hit the magic number in the master binlog.
- ALTER TABLE ... ORDER BY ... syntax added. This will create the new table with the rows in a specific order.

## 13.4.32 Changes in release 3.23.27 (24 Oct 2000)

- Fixed a bug where the automatic repair of MyISAM tables sometimes failed when the datafile was corrupt.
- Fixed a bug in SHOW CREATE when using AUTO_INCREMENT columns.
- Changed BDB tables to use new compare function in Berkeley DB 3.2.3.
- You can now use Unix sockets with MIT-pthreads.
- Added the latin5 (turkish) character set.
- Small portability fixes.

## 13.4.33 Changes in release 3.23.26 (18 Oct 2000)

- Renamed FLUSH MASTER and FLUSH SLAVE to RESET MASTER and RESET SLAVE .
- Fixed <> to work properly with NULL .
- Fixed a problem with SUBSTRING_INDEX() and REPLACE() . (Patch by Alexander Igonitchev)
- Fix CREATE TEMPORARY TABLE IF NOT EXISTS not to produce an error if the table exists.
- If you don't create a PRIMARY KEY in a BDB table, a hidden PRIMARY KEY will be created.
- Added read-only-key optimisation to BDB tables.
- LEFT JOIN in some cases preferred a full table scan when there was no WHERE clause.
- When using --log-slow-queries , don't count the time waiting for a lock.
- Fixed bug in lock code on Windows which could cause the key cache to report that the key file was crashed even if it was okay.
- Automatic repair of MyISAM tables if you start mysqld with --myisam-recover .
- Removed the TYPE= keyword from CHECK and REPAIR . Allow CHECK options to be combined. (You can still use TYPE= , but this usage is deprecated.)
- Fixed mutex bug in the binary replication log -- long update queries could be read only in part by the slave if it did it at the wrong time, which was not fatal, but resulted in a performance-degrading reconnect and a scary message in the error log.
- Changed the format of the binary log -- added magic number, server version, binlog version. Added server id and query error code for each query event.
- Replication thread from the slave now will kill all the stale threads from the same server.
- Long replication user names were not being handled properly.
- Added --replicate-rewrite-db option to mysqld .
- Added --skip-slave-start option to mysqld .
- Updates that generated an error code (such as INSERT INTO foo(some_key) values (1),(1) ) erroneously terminated the slave thread.
- Added optimisation of queries where DISTINCT is only used on columns from some of the tables.
- Allow floating-point numbers where there is no sign after the exponent (like 1e1 ).
- SHOW GRANTS didn't always show all column grants.
- Added --default-extra-file=# option to all MySQL clients.
- Columns referenced in INSERT statements now are initialised properly.
- UPDATE didn't always work when used with a range on a timestamp that was part of the key that was used to find rows.
- Fixed a bug in FULLTEXT index when inserting a NULL column.
- Changed to use mkstemp() instead of tempnam() . Based on a patch from John Jones.

## 13.4.34 Changes in release 3.23.25 (29 Sep 2000)

- Fixed that databasename works as second argument to mysqlhotcopy .
- The values for the UMASK and UMASK_DIR environment variables now can be specified in octal by beginning the value with a zero.
- Added RIGHT JOIN . This makes RIGHT a reserved word.
- Added @@IDENTITY as a synonym for LAST_INSERT_ID() . (This is for MSSQL compatibility.)
- Fixed a bug in myisamchk and REPAIR when using FULLTEXT index.
- LOAD DATA INFILE now works with FIFOs. (Patch by Toni L. Harbaugh-Blackford.)
- FLUSH LOGS broke replication if you specified a log name with an explicit extension as the value of the log-bin option.
- Fixed a bug in MyISAM with packed multi-part keys.
- Fixed crash when using CHECK TABLE on Windows.
- Fixed a bug where FULLTEXT index always used the koi8_ukr character set.
- Fixed privilege checking for CHECK TABLE .
- The MyISAM repair/reindex code didn't use the --tmpdir option for its temporary files.
- Added BACKUP TABLE and RESTORE TABLE .
- Fixed core dump on CHANGE MASTER TO when the slave did not have the master to start with.
- Fixed incorrect Time in the processlist for Connect of the slave thread.
- The slave now logs when it connects to the master.
- Fixed a core dump bug when doing FLUSH MASTER if you didn't specify a filename argument to --log-bin .
- Added missing ha_berkeley.x files to the MySQL Windows distribution.
- Fixed some mutex bugs in the log code that could cause thread blocks if new log files couldn't be created.
- Added lock time and number of selected processed rows to slow query log.
- Added --memlock option to mysqld to lock mysqld in memory on systems with the mlockall() call (like in Solaris).
- HEAP tables didn't use keys properly. (Bug from 3.23.23.)
- Added better support for MERGE tables (keys, mapping, creation, documentation...). Tables MERGE .
- Fixed bug in mysqldump from 3.23 which caused some CHAR columns not to be quoted.
- Merged analyze , check , optimize and repair code.
- OPTIMIZE TABLE is now mapped to REPAIR with statistics and sorting of the index tree. This means that for the moment it only works on MyISAM tables.
- Added a pre-alloced block to root_malloc to get fewer mallocs.
- Added a lot of new statistics variables.
- Fixed ORDER BY bug with BDB tables.
- Removed warning that mysqld couldn't remove the .pid file under Windows.
- Changed --log-isam to log MyISAM tables instead of isam tables.
- Fixed CHECK TABLE to work on Windows.
- Added file mutexes to make pwrite() safe on Windows.

## 13.4.35 Changes in release 3.23.24 (08 Sep 2000)

- Added created_tmp_disk_tables variable to mysqld .
- To make it possible to reliably dump and restore tables with TIMESTAMP(X) columns, MySQL now reports columns with X other than 14 or 8 to be strings.
- Changed sort order for latin1 as it was before MySQL Version 3.23.23. Any table that was created or modified with 3.23.22 must be repaired if it has CHAR columns that may contain characters with ASCII values greater than 128!
- Fixed small memory leak introduced from 3.23.22 when creating a temporary table.
- Fixed problem with BDB tables and reading on a unique (not primary) key.
- Restored the win1251 character set (it's now only marked deprecated).

## 13.4.36 Changes in release 3.23.23 (01 Sep 2000)

- Changed sort order for 'German'; all tables created with 'German' sortorder must be repaired with REPAIR TABLE or myisamchk before use!
- Added --core-file option to mysqld to get a core file on Linux if mysqld dies on the SIGSEGV signal.
- MySQL client mysql now starts with option --no-named-commands ( -g ) by default. This option can be disabled with --enable-named-commands ( -G ). This may cause incompatibility problems in some cases, for example, in SQL scripts that use named commands without a semicolon, etc. ! Long format commands still work from the first line.
- Fixed a problem when using many pending DROP TABLE statements at the same time.
- Optimiser didn't use keys properly when using LEFT JOIN on an empty table.
- Added shorter help text when invoking mysqld with incorrect options.
- Fixed non-fatal free() bug in mysqlimport .
- Fixed bug in MyISAM index handling of DECIMAL / NUMERIC keys.
- Fixed a bug in concurrent insert in MyISAM tables. In some contexts, usage of MIN(key_part) or MAX(key_part) returned an empty set.
- Updated mysqlhotcopy to use the new FLUSH TABLES table_list syntax. Only tables which are being backed up are flushed now.
- Changed behaviour of --enable-thread-safe-client so that both non-threaded ( -lmysqlclient ) and threaded ( -lmysqlclient_r ) libraries are built. Users who linked against a threaded -lmysqlclient will need to link against -lmysqlclient_r now.
- Added atomic RENAME TABLE command.
- Don't count NULL values in COUNT(DISTINCT ...) .
- Changed ALTER TABLE , LOAD DATA INFILE on empty tables and INSERT ... SELECT ... on empty tables to create non-unique indexes in a separate batch with sorting. This will make the above calls much faster when you have many indexes.
- ALTER TABLE now logs the first used insert_id correctly.
- Fixed crash when adding a default value to a BLOB column.
- Fixed a bug with DATE_ADD/DATE_SUB where it returned a datetime instead of a date.
- Correction d'un problème avec le cache de thread qui faisait que certains threads apparaissaient comme ***DEAD*** dans SHOW PROCESSLIST .
- Fixed a lock in our thr_rwlock code, which could make selects that run at the same time as concurrent inserts crash. This only affects systems that don't have the pthread_rwlock_rdlock code.
- When deleting rows with a non-unique key in a HEAP table, all rows weren't always deleted.
- Fixed bug in range optimiser for HEAP tables for searches on a part index.
- Fixed SELECT on part keys to work with BDB tables.
- Fixed INSERT INTO bdb_table ... SELECT to work with BDB tables.
- CHECK TABLE now updates key statistics for the table.
- ANALYZE TABLE will now only update tables that have been changed since the last ANALYZE . Note that this is a new feature and tables will not be marked to be analysed until they are updated in any way with 3.23.23 or newer. For older tables, you have to do CHECK TABLE to update the key distribution.
- Fixed some minor privilege problems with CHECK , ANALYZE , REPAIR and SHOW CREATE commands.
- Added CHANGE MASTER TO statement.
- Added FAST , QUICK EXTENDED check types to CHECK TABLES .
- Changed myisamchk so that --fast and --check-only-changed are also honored with --sort-index and --analyze .
- Fixed fatal bug in LOAD TABLE FROM MASTER that did not lock the table during index re-build.

- LOAD DATA INFILE broke replication if the database was excluded from replication.
- More variables in SHOW SLAVE STATUS and SHOW MASTER STATUS .
- SLAVE STOP now will not return until the slave thread actually exits.
- Full-text search via the MATCH() function and FULLTEXT index type (for MyISAM files). This makes FULLTEXT a reserved word.

## 13.4.37 Changes in release 3.23.22 (31 Jul 2000)

- Fixed that lex_hash.h is created properly for each MySQL distribution.
- Fixed that MASTER and COLLECTION are not reserved words.
- The log generated by --slow-query-log didn't contain the whole queries.
- Fixed that open transactions in BDB tables are rolled back if the connection is closed unexpectedly.
- Added workaround for a bug in gcc 2.96 (intel) and gcc 2.9 (IA64) in gen_lex_hash.c .
- Fixed memory leak in the client library when using host= in the my.cnf file.
- Optimised functions that manipulate the hours/minutes/seconds.
- Fixed bug when comparing the result of DATE_ADD() / DATE_SUB() against a number.
- Changed the meaning of -F, --fast for myisamchk . Added -C, --check-only-changed option to myisamchk .
- Added ANALYZE tbl_name to update key statistics for tables.
- Changed binary items 0x... to be regarded as integers by default.
- Fix for SCO and SHOW PROCESSLIST .
- Added auto-rehash on reconnect for the mysql client.
- Fixed a newly introduced bug in MyISAM , where the index file couldn't get bigger than 64M.
- Added SHOW MASTER STATUS and SHOW SLAVE STATUS .

## 13.4.38 Changes in release 3.23.21

- Added mysql_character_set_name() function to the MySQL C API.
- Made the update log ASCII 0 safe.
- Added the mysql_config script.
- Fixed problem when using < or > with a char column that was only partly indexed.
- One would get a core dump if the log file was not readable by the MySQL user.
- Changed mysqladmin to use CREATE DATABASE and DROP DATABASE statements instead of the old deprecated API calls.
- Fixed chown warning in safe_mysqld .
- Fixed a bug in ORDER BY that was introduced in 3.23.19.
- Only optimise the DELETE FROM tbl_name to do a drop+create of the table if we are in AUTOCOMMIT mode (needed for BDB tables).
- Added extra checks to avoid index corruption when the ISAM / MyISAM index files get full during an INSERT / UPDATE .
- myisamchk didn't correctly update row checksum when used with -ro (this only gave a warning in subsequent runs).
- Fixed bug in REPAIR TABLE so that it works with tables without indexes.
- Fixed buffer overrun in DROP DATABASE .
- LOAD TABLE FROM MASTER is sufficiently bug-free to announce it as a feature.
- MATCH and AGAINST are now reserved words.

## 13.4.39 Changes in release 3.23.20

- Fixed bug in 3.23.19; DELETE FROM tbl_name removed the .frm file.
- Added SHOW CREATE TABLE .

## 13.4.40 Changes in release 3.23.19

- Changed copyright for all files to GPL for the server code and utilities and to LGPL for the client libraries.
- Fixed bug where all rows matching weren't updated on a MyISAM table when doing update based on key on a table with many keys and some key changed values.
- The Linux MySQL RPM's and binaries are now statically linked with a linuxthread version that has faster mutex handling when used with MySQL.
- ORDER BY can now use REF keys to find subsets of the rows that need to be sorted.
- Changed name of print_defaults program to my_print_defaults to avoid name confusion.
- Fixed NULLIF() to work according to ANSI SQL99.
- Added net_read_timeout and net_write_timeout as startup parameters to mysqld .
- Fixed bug that destroyed index when doing myisamchk --sort-records on a table with prefix compressed index.
- Added pack_isam and myisampack to the standard MySQL distribution.
- Added the syntax BEGIN WORK (the same as BEGIN ).
- Fixed core dump bug when using ORDER BY on a CONV() expression.
- Added LOAD TABLE FROM MASTER .
- Added FLUSH MASTER and FLUSH SLAVE .
- Fixed big/little endian problem in the replication.

## 13.4.41 Changes in release 3.23.18

- Fixed a problem from 3.23.17 when choosing character set on the client side.
- Added FLUSH TABLES WITH READ LOCK to make a global lock suitable for making a copy of MySQL datafiles.
- CREATE TABLE ... SELECT ... PROCEDURE now works.
- Internal temporary tables will now use compressed index when using GROUP BY on VARCHAR/CHAR columns.
- Fixed a problem when locking the same table with both a READ and a WRITE lock.
- Fixed problem with myisamchk and RAID tables.

## 13.4.42 Changes in release 3.23.17

- Fixed a bug in FIND_IN_SET() when the first argument was NULL .
- Added table locks to Berkeley DB.
- Fixed a bug with LEFT JOIN and ORDER BY where the first table had only one matching row.
- Added 4 sample my.cnf example files in the support-files directory.
- Fixed duplicated key problem when doing big GROUP BY operations. (This bug was probably introduced in 3.23.15.)
- Changed syntax for INNER JOIN to match ANSI SQL.
- Added NATURAL JOIN syntax.
- A lot of fixes in the BDB interface.
- Added handling of --no-defaults and --defaults-file to safe_mysqld.sh and mysql_install_db.sh .
- Fixed bug in reading compressed tables with many threads.
- Fixed that USE INDEX works with PRIMARY keys.
- Added BEGIN statement to start a transaction in AUTOCOMMIT mode.
- Added support for symbolic links for Windows.
- Changed protocol to let client know if the server is in AUTOCOMMIT mode and if there is a pending transaction. If there is a pending transaction, the client library will give an error before reconnecting to the server to let the client know that the server did a rollback. The protocol is still backward-compatible with old clients.
- KILL now works on a thread that is locked on a 'write' to a dead client.
- Fixed memory leak in the replication slave thread.
- Added new log-slave-updates option to mysqld , to allow daisy-chaining the slaves.
- Fixed compile error on FreeBSD and other systems where pthread_t is not the same as int .
- Fixed master shutdown aborting the slave thread.
- Fixed a race condition in INSERT DELAYED code when doing ALTER TABLE .
- Added deadlock detection sanity checks to INSERT DELAYED .

## 13.4.43 Changes in release 3.23.16

- Added SLAVE START and SLAVE STOP statements.
- Added TYPE=QUICK option to CHECK and to REPAIR .
- Fixed bug in REPAIR TABLE when the table was in use by other threads.
- Added a thread cache to make it possible to debug MySQL with gdb when one does a lot of reconnects. This will also improve systems where you can't use persistent connections.
- Lots of fixes in the Berkeley DB interface.
- UPDATE IGNORE will not abort if an update results in a DUPLICATE_KEY error.
- Put CREATE TEMPORARY TABLE commands in the update log.
- Fixed bug in handling of masked IP numbers in the privilege tables.
- Fixed bug with delay_key_write tables and CHECK TABLE .
- Added replicate-do-db and replicate-ignore-db options to mysqld , to restrict which databases get replicated.
- Added SQL_LOG_BIN option.

## 13.4.44 Changes in release 3.23.15 (May 2000: Beta)

- To start mysqld as root , you must now use the --user=root option.
- Added interface to Berkeley DB. (This is not yet functional; play with it at your own risk!)
- Réplication entre maître et esclaves.
- Fixed bug that other threads could steal a lock when a thread had a lock on a table and did a FLUSH TABLES command.
- Added the slow_launch_time variable and the Slow_launch_threads status variable to mysqld . These can be examined with mysqladmin variables and mysqladmin extended-status .
- Added functions INET_NTOA() and INET_ATON() .
- The default type of IF() now depends on the second and third arguments and not only on the second argument.
- Fixed case when myisamchk could go into a loop when trying to repair a crashed table.
- Don't write INSERT DELAYED to update log if SQL_LOG_UPDATE=0 .
- Fixed problem with REPLACE on HEAP tables.
- Added possible character sets and time zone to SHOW VARIABLES output.
- Fixed bug in locking code that could result in locking problems with concurrent inserts under high load.
- Fixed a problem with DELETE of many rows on a table with compressed keys where MySQL scanned the index to find the rows.
- Fixed problem with CHECK on table with deleted keyblocks.
- Fixed a bug in reconnect (at the client side) where it didn't free memory properly in some contexts.
- Fixed problems in update log when using LAST_INSERT_ID() to update a table with an AUTO_INCREMENT key.
- Added NULLIF() function.
- Fixed bug when using LOAD DATA INFILE on a table with BLOB/TEXT columns.
- Optimised MyISAM to be faster when inserting keys in sorted order.
- EXPLAIN SELECT ... now also prints out whether MySQL needs to create a temporary table or use file sorting when resolving the SELECT .
- Added optimisation to skip ORDER BY parts where the part is a constant expression in the WHERE part. Indexes can now be used even if the ORDER BY doesn't match the index exactly, as long as all the unused index parts and all the extra ORDER BY columns are constants in the WHERE clause. Comment MySQL utilise les index .
- UPDATE and DELETE on a whole unique key in the WHERE part are now faster than before.
- Changed RAID_CHUNKSIZE to be in 1024-byte increments.
- Fixed core dump in LOAD_FILE(NULL) .

## 13.4.45 Changes in release 3.23.14

- Added mysql_real_escape_string() function to the MySQL C API.
- Fixed a bug in CONCAT() where one of the arguments was a function that returned a modified argument.
- Fixed a critical bug in myisamchk , where it updated the header in the index file when one only checked the table. This confused the mysqld daemon if it updated the same table at the same time. Now the status in the index file is only updated if one uses --update-state . With older myisamchk versions you should use --read-only when only checking tables, if there is the slightest chance that the mysqld server is working on the table at the same time!
- Fixed that DROP TABLE is logged in the update log.
- Fixed problem when searching on DECIMAL() key field where the column data contained leading zeros.
- Fix bug in myisamchk when the AUTO_INCREMENT column isn't the first key.
- Allow DATETIME in ISO8601 format: 2000-03-12T12:00:00
- Dynamic character sets. A mysqld binary can now handle many different character sets (you can choose which when starting mysqld ).
- Added command REPAIR TABLE .

- Added mysql_thread_safe() function to the MySQL C API.
- Added the UMASK_DIR environment variable.
- Added CONNECTION_ID() function to return the client connection thread ID.
- When using = on BLOB or VARCHAR BINARY keys, where only a part of the column was indexed, the whole column of the result row wasn't compared.
- Fix for sjis character set and ORDER BY .
- When running in ANSI mode, don't allow columns to be used that aren't in the GROUP BY part.

## 13.4.46 Changes in release 3.23.13

- Fixed problem when doing locks on the same table more than 2 times in the same LOCK TABLE command; this fixed the problem one got when running the test-ATIS test with --fast or --check-only-changed .
- Added SQL_BUFFER_RESULT option to SELECT .
- Removed end space from double/float numbers in results from temporary tables.
- Added CHECK TABLE command.
- Added changes for MyISAM in 3.23.12 that didn't get into the source distribution because of CVS problems.
- Fixed bug so that mysqladmin shutdown will wait for the local server to close down.
- Fixed a possible endless loop when calculating timestamp.
- Added print_defaults program to the .rpm files. Removed mysqlbug from the client .rpm file.

## 13.4.47 Changes in release 3.23.12 (07 Mar 2000)

- Fixed bug in MyISAM involving REPLACE ... SELECT ... which could give a corrupted table.
- Fixed bug in myisamchk where it incorrectly reset the AUTO_INCREMENT value.
- LOTS of patches for Linux Alpha. MySQL now appears to be relatively stable on Alpha.
- Changed DISTINCT on HEAP temporary tables to use hashed keys to quickly find duplicated rows. This mostly concerns queries of type SELECT DISTINCT ... GROUP BY ... . This fixes a problem where not all duplicates were removed in queries of the above type. In addition, the new code is MUCH faster.
- Added patches to make MySQL compile on Mac OS X.
- Added IF NOT EXISTS clause to CREATE DATABASE .
- Added --all-databases and --databases options to mysqldump to allow dumping of many databases at the same time.
- Fixed bug in compressed DECIMAL() index in MyISAM tables.
- Fixed bug when storing 0 into a timestamp.
- When doing mysqladmin shutdown on a local connection, mysqladmin now waits until the PID file is gone before terminating.
- Fixed core dump with some COUNT(DISTINCT ...) queries.
- Fixed that myisamchk works properly with RAID tables.
- Fixed problem with LEFT JOIN and key_field IS NULL .
- Fixed bug in net_clear() which could give the error Aborted connection in the MySQL clients.
- Added options USE INDEX (key_list) and IGNORE INDEX (key_list) as parameters in SELECT .
- DELETE and RENAME should now work on RAID tables.

## 13.4.48 Changes in release 3.23.11

- Allow the ALTER TABLE tbl_name ADD (field_list) syntax.
- Fixed problem with optimiser that could sometimes use incorrect keys.
- Fixed that GRANT/REVOKE ALL PRIVILEGES doesn't affect GRANT OPTION .
- Removed extra ')' from the output of SHOW GRANTS .
- Fixed problem when storing numbers in timestamps.
- Fix problem with timezones that have half hour offsets.
- Allow the syntax UNIQUE INDEX in CREATE statements.
- mysqlhotcopy - fast online hot-backup utility for local MySQL databases. By Tim Bunce.
- New more secure mysqlaccess . Thanks to Steve Harvey for this.
- Added --i-am-a-dummy and --safe-updates options to mysql .
- Added select_limit and max_join_size variables to mysql .
- Added SQL_MAX_JOIN_SIZE and SQL_SAFE_UPDATES options.
- Added READ LOCAL lock that doesn't lock the table for concurrent inserts. (This is used by mysqldump .)
- Changed that LOCK TABLES ... READ doesn't anymore allow concurrent inserts.
- Added --skip-delay-key-write option to mysqld .
- Fixed security problem in the protocol regarding password checking.
- _rowid can now be used as an alias for an integer type unique indexed column.
- Added back blocking of SIGPIPE when compiling with --thread-safe-clients to make things safe for old clients.

## 13.4.49 Changes in release 3.23.10

- Fixed bug in 3.23.9 where memory wasn't properly freed when using LOCK TABLES .

## 13.4.50 Changes in release 3.23.9

- Fixed problem that affected queries that did arithmetic on group functions.
- Fixed problem with timestamps and INSERT DELAYED .
- Fixed that date_col BETWEEN const_date AND const_date works.
- Fixed problem when only changing a 0 to NULL in a table with BLOB/TEXT columns.
- Fixed bug in range optimiser when using many key parts and or on the middle key parts: WHERE K1=1 and K3=2 and (K2=2 and K4=4 or K2=3 and K4=5)
- Added source command to mysql to allow reading of batch files inside the mysql client. Original patch by Matthew Vanecek.
- Fixed critical problem with the WITH GRANT OPTION option.
- Don't give an unnecessary GRANT error when using tables from many databases in the same query.
- Added VIO wrapper (needed for SSL support; by Andrei Errapart and Tõnu Samuel).
- Fixed optimiser problem on SELECT when using many overlapping indexes. MySQL should now be able to choose keys even better when there are many keys to choose from.

- Changed optimiser to prefer a range key instead of a ref key when the range key can uses more columns than the ref key (which only can use columns with = ). For example, the following type of queries should now be faster: SELECT * from key_part_1=const and key_part_2 > const2
- Fixed bug that a change of all VARCHAR columns to CHAR columns didn't change row type from dynamic to fixed.
- Disabled floating-point exceptions for FreeBSD to fix core dump when doing SELECT FLOOR(POW(2,63)) .
- Renamed mysqld startup option from --delay-key-write to --delay-key-write-for-all-tables .
- Added read-next-on-key to HEAP tables. This should fix all problems with HEAP tables when using non- UNIQUE keys.
- Added option to print default arguments to all clients.
- Added --log-slow-queries option to mysqld to log all queries that take a long time to a separate log file with a time indicating how long the query took.
- Fixed core dump when doing WHERE key_col=RAND(...) .
- Fixed optimisation bug in SELECT ... LEFT JOIN ... key_col IS NULL , when key_col could contain NULL values.
- Fixed problem with 8-bit characters as separators in LOAD DATA INFILE .

## 13.4.51 Changes in release 3.23.8 (02 Jan 2000)

- Fixed problem when handling indexfiles larger than 8G.
- Added latest patches to MIT-pthreads for NetBSD.
- Fixed problem with timezones that are < GMT - 11.
- Fixed a bug when deleting packed keys in NISAM .
- Fixed problem with ISAM when doing some ORDER BY ... DESC queries.
- Fixed bug when doing a join on a text key which didn't cover the whole key.
- Option --delay-key-write didn't enable delayed key writing.
- Fixed update of TEXT column which involved only case changes.
- Fixed that INSERT DELAYED doesn't update timestamps that are given.
- Added function YEARWEEK() and options x , X , v and V to DATE_FORMAT() .
- Fixed problem with MAX(indexed_column) and HEAP tables.
- Fixed problem with BLOB NULL keys and LIKE "prefix%" .
- Fixed problem with MyISAM and fixed-length rows < 5 bytes.
- Fixed problem that could cause MySQL to touch freed memory when doing very complicated GROUP BY queries.
- Fixed core dump if you got a crashed table where an ENUM field value was too big.

## 13.4.52 Changes in release 3.23.7 (10 Dec 1999)

- Fixed workaround under Linux to avoid problems with pthread_mutex_timedwait , which is used with INSERT DELAYED . Notes relatives à Linux (toutes les versions de Linux) .
- Fixed that one will get a 'disk full' error message if one gets disk full when doing sorting (instead of waiting until we got more disk space).
- Fixed a bug in MyISAM with keys > 250 characters.
- In MyISAM one can now do an INSERT at the same time as other threads are reading from the table.
- Added max_write_lock_count variable to mysqld to force a READ lock after a certain number of WRITE locks.
- Inverted flag delay_key_write on show variables .
- Renamed concurrency variable to thread_concurrency .
- The following functions are now multi-byte-safe: LOCATE(substr,str) , POSITION(substr IN str) , LOCATE(substr,str,pos) , INSTR(str,substr) , LEFT(str,len) , RIGHT(str,len) , SUBSTRING(str,pos,len) , SUBSTRING(str FROM pos FOR len) , MID(str,pos,len) , SUBSTRING(str,pos) , SUBSTRING(str FROM pos) , SUBSTRING_INDEX(str,delim,count) , RTRIM(str) , TRIM([[BOTH | TRAILING] [remstr] FROM] str) , REPLACE(str,from_str,to_str) , REVERSE(str) , INSERT(str,pos,len,newstr) , LCASE(str) , LOWER(str) , UCASE(str) and UPPER(str) ; patch by Wei He.
- Fix core dump when releasing a lock from a non-existent table.
- Remove locks on tables before starting to remove duplicates.
- Added option FULL to SHOW PROCESSLIST .
- Added option --verbose to mysqladmin .
- Fixed problem when automatically converting HEAP to MyISAM .
- Fixed bug in HEAP tables when doing insert + delete + insert + scan the table.
- Fixed bugs on Alpha with REPLACE() and LOAD DATA INFILE .
- Added interactive_timeout variable to mysqld .
- Changed the argument to mysql_data_seek() from ulong to ulonglong .

## 13.4.53 Changes in release 3.23.6

- Added -O lower_case_table_names={0|1} option to mysqld to allow users to force table names to lowercase.
- Added SELECT ... INTO DUMPFILE .
- Added --ansi option to mysqld to make some functions ANSI SQL compatible.
- Temporary table names now start with #sql .
- Added quoting of identifiers with ` ( " in --ansi mode).
- Changed to use snprintf() when printing floats to avoid some buffer overflows on FreeBSD.
- Made FLOOR() overflow safe on FreeBSD.
- Added --quote-names option to mysqldump .
- Fixed bug that one could make a part of a PRIMARY KEY NOT NULL .
- Fixed encrypt() to be thread-safe and not reuse buffer.
- Added mysql_odbc_escape_string() function to support big5 characters in MyODBC.
- Rewrote the table handler to use classes. This introduces a lot of new code, but will make table handling faster and better.
- Added patch by Sasha for user-defined variables.
- Changed that FLOAT and DOUBLE (without any length modifiers) no longer are fixed decimal point numbers.
- Changed the meaning of FLOAT(X) : Now this is the same as FLOAT if X <= 24 and a DOUBLE if 24 < X <= 53.
- DECIMAL(X) is now an alias for DECIMAL(X,0) and DECIMAL is now an alias for DECIMAL(10,0) . The same goes for NUMERIC .
- Added option ROW_FORMAT={default | dynamic | fixed | compressed} to CREATE_TABLE .
- DELETE FROM table_name didn't work on temporary tables.
- Changed function CHAR_LENGTH() to be multi-byte character safe.
- Added function ORD(string) .

## 13.4.54 Changes in release 3.23.5 (20 Oct 1999)

- Fixed some Y2K problems in the new date handling in 3.23.
- Fixed problem with SELECT DISTINCT ... ORDER BY RAND() .
- Added patches by Sergei A. Golubchik for text searching on the MyISAM level.
- Fixed cache overflow problem when using full joins without keys.
- Fixed some configure issues.
- Some small changes to make parsing faster.
- Adding a column after the last field with ALTER TABLE didn't work.
- Fixed problem when using an AUTO_INCREMENT column in two keys
- With MyISAM , you now can have an AUTO_INCREMENT column as a key sub part: CREATE TABLE foo (a INT NOT NULL AUTO_INCREMENT, b CHAR(5), PRIMARY KEY (b,a))
- Fixed bug in MyISAM with packed char keys that could be NULL .
- AS on field name with CREATE TABLE table_name SELECT ... didn't work.
- Allow use of NATIONAL and NCHAR when defining character columns. This is the same as not using BINARY .
- Don't allow NULL columns in a PRIMARY KEY (only in UNIQUE keys).
- Clear LAST_INSERT_ID() if one uses this in ODBC: WHERE auto_increment_column IS NULL . This seems to fix some problems with Access.
- SET SQL_AUTO_IS_NULL=0|1 now turns on/off the handling of searching after the last inserted row with WHERE auto_increment_column IS NULL .
- Added new variable concurrency to mysqld for Solaris.
- Added --relative option to mysqladmin to make extended-status more useful to monitor changes.
- Fixed bug when using COUNT(DISTINCT ...) on an empty table.
- Added support for the Chinese character set GBK.
- Fixed problem with LOAD DATA INFILE and BLOB columns.
- Added bit operator ~ (negation).
- Fixed problem with UDF functions.

## 13.4.55 Changes in release 3.23.4 (28 Sep 1999)

- Inserting a DATETIME into a TIME column no longer will try to store 'days' in it.
- Fixed problem with storage of float/double on little endian machines. (This affected SUM() .)
- Added connect timeout on TCP/IP connections.
- Fixed problem with LIKE "%" on an index that may have NULL values.
- REVOKE ALL PRIVILEGES didn't revoke all privileges.
- Allow creation of temporary tables with same name as the original table.
- When granting a user a GRANT option for a database, he couldn't grant privileges to other users.
- New command: SHOW GRANTS FOR user (by Sinisa).
- New date_add syntax: date/datetime + INTERVAL # interval_type . By Joshua Chamas.
- Fixed privilege check for LOAD DATA REPLACE .
- Automatic fixing of broken include files on Solaris 2.7
- Some configure issues to fix problems with big filesystem detection.
- REGEXP is now case-insensitive if you use non-binary strings.

## 13.4.56 Changes in release 3.23.3

- Added patches for MIT-pthreads on NetBSD.
- Fixed range bug in MyISAM .
- ASC is now the default again for ORDER BY .
- Added LIMIT to UPDATE .
- Added mysql_change_user() function to the MySQL C API.
- Added character set to SHOW VARIABLES .
- Added support of --[whitespace] comments.
- Allow INSERT into tbl_name VALUES () , that is, you may now specify an empty value list to insert a row in which each column is set to its default value.
- Changed SUBSTRING(text FROM pos) to conform to ANSI SQL. (Before this construct returned the rightmost pos characters.)
- SUM() with GROUP BY returned 0 on some systems.
- Changed output for SHOW TABLE STATUS .
- Added DELAY_KEY_WRITE option to CREATE TABLE .
- Allow AUTO_INCREMENT on any key part.
- Fixed problem with YEAR(NOW()) and YEAR(CURDATE()) .
- Added CASE construct.
- New function COALESCE() .

## 13.4.57 Changes in release 3.23.2 (09 Aug 1999)

- Fixed range optimiser bug: SELECT * FROM table_name WHERE key_part1 >= const AND (key_part2 = const OR key_part2 = const) . The bug was that some rows could be duplicated in the result.
- Running myisamchk without -a updated the index distribution incorrectly.
- SET SQL_LOW_PRIORITY_UPDATES=1 was causing a parse error.
- You can now update index columns that are used in the WHERE clause. UPDATE tbl_name SET KEY=KEY+1 WHERE KEY > 100
- Date handling should now be a bit faster.
- Added handling of fuzzy dates (dates where day or month is 0), such as '1999-01-00' .
- Fixed optimisation of SELECT ... WHERE key_part1=const1 AND key_part_2=const2 AND key_part1=const4 AND key_part2=const4 ; indextype should be range instead of ref .
- Fixed egcs 1.1.2 optimiser bug (when using BLOB s) on Linux Alpha.
- Fixed problem with LOCK TABLES combined with DELETE FROM table .
- MyISAM tables now allow keys on NULL and BLOB/TEXT columns.
- The following join is now much faster: SELECT ... FROM t1 LEFT JOIN t2 ON ... WHERE t2.not_null_column IS NULL .
- ORDER BY and GROUP BY can be done on functions.
- Changed handling of 'const_item' to allow handling of ORDER BY RAND() .
- Indexes are now used for WHERE key_column = function .
- Indexes are now used for WHERE key_column = col_name even if the columns are not identically packed.
- Indexes are now used for WHERE col_name IS NULL .
- Changed heap tables to be stored in low_byte_first order (to make it easy to convert to MyISAM tables)
- Automatic change of HEAP temporary tables to MyISAM tables in case of 'table is full' errors.
- Added --init-file=file_name option to mysqld .

- Added COUNT(DISTINCT value, [value, ...]) .
- CREATE TEMPORARY TABLE now creates a temporary table, in its own namespace, that is automatically deleted if connection is dropped.
- New reserved words (required for CASE ): CASE, THEN, WHEN, ELSE and END .
- New functions EXPORT_SET() and MD5() .
- Support for the GB2312 Chinese character set.

## 13.4.58 Changes in release 3.23.1

- Fixed some compilation problems.

## 13.4.59 Changes in release 3.23.0 (05 Aug 1999: Alpha)

- A new table handler library ( MyISAM ) with a lot of new features. Tables MyISAM .
- You can create in-memory HEAP tables which are extremely fast for lookups.
- Support for big files (63-bit) on OSes that support big files.
- New function LOAD_FILE(filename) to get the contents of a file as a string value.
- New operator <=> which will act as = but will return TRUE if both arguments are NULL . This is useful for comparing changes between tables.
- Added the ODBC 3.0 EXTRACT(interval FROM datetime) function.
- Columns defined as FLOAT(X) are not rounded on storage and may be in scientific notation (1.0 E+10) when retrieved.
- REPLACE is now faster than before.
- Changed LIKE character comparison to behave as = ; This means that 'e' LIKE ''e' is now true. (If the line doesn't display correctly, the latter 'e' is a French 'e' with a dot above.)
- SHOW TABLE STATUS returns a lot of information about the tables.
- Added LIKE to the SHOW STATUS command.
- Added Privileges column to SHOW COLUMNS .
- Added Packed and Comment columns to SHOW INDEX .
- Added comments to tables (with CREATE TABLE ... COMMENT "xxx" ).
- Added UNIQUE , as in CREATE TABLE table_name (col int not null UNIQUE)
- New create syntax: CREATE TABLE table_name SELECT ...
- New create syntax: CREATE TABLE IF NOT EXISTS ...
- Allow creation of CHAR(0) columns.
- DATE_FORMAT() now requires '%' before any format character.
- DELAYED is now a reserved word (sorry about that :( ).
- An example procedure is added: analyse , file: sql_analyse.c . This will describe the data in your query. Try the following:

```
 SELECT ... FROM ...
WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
```

  This procedure is extremely useful when you want to check the data in your table!
- BINARY cast to force a string to be compared in case-sensitive fashion.
- Added --skip-show-database option to mysqld .
- Check whether a row has changed in an UPDATE now also works with BLOB / TEXT columns.
- Added the INNER join syntax. **NOTE** : This made INNER a reserved word!
- Added support for netmasks to the hostname in the MySQL grant tables. You can specify a netmask using the IP/NETMASK syntax.
- If you compare a NOT NULL DATE/DATETIME column with IS NULL , this is changed to a compare against 0 to satisfy some ODBC applications. (By shreeve@uci.edu .)
- NULL IN (...) now returns NULL instead of 0 . This will ensure that null_column NOT IN (...) doesn't match NULL values.
- Fix storage of floating-point values in TIME columns.
- Changed parsing of TIME strings to be more strict. Now the fractional second part is detected (and currently skipped). The following formats are supported:
  - [[DAYS] [H]H:]MM:]SS[.fraction]
  - [[[[[H]H]H]MM]SS[.fraction]
- Detect (and ignore) fractional second part from DATETIME .
- Added the LOW_PRIORITY attribute to LOAD DATA INFILE .
- The default index name now uses the same case as the column name on which the index name is based.
- Changed default number of connections to 100.
- Use bigger buffers when using LOAD DATA INFILE .
- DECIMAL(x,y) now works according to ANSI SQL.
- Added aggregate UDF functions. Thanks to Andreas F. Bobak ( bobak@relog.ch ) for this!
- LAST_INSERT_ID() is now updated for INSERT INTO ... SELECT .
- Some small changes to the join table optimiser to make some joins faster.
- SELECT DISTINCT is much faster; it uses the new UNIQUE functionality in MyISAM . One difference compared to MySQL Version 3.22 is that the output of DISTINCT is not sorted anymore.
- All C client API macros are now functions to make shared libraries more reliable. Because of this, you can no longer call mysql_num_fields() on a MYSQL object, you must use mysql_field_count() instead.
- Added use of LIBWRAP ; patch by Henning P. Schmiedehausen.
- Don't allow AUTO_INCREMENT for other than numerical columns.
- Using AUTO_INCREMENT will now automatically make the column NOT NULL .
- Show NULL as the default value for AUTO_INCREMENT columns.
- Added SQL_BIG_RESULT ; SQL_SMALL_RESULT is now default.
- Added a shared library RPM. This enhancement was contributed by David Fox ( dsfox@cogsci.ucsd.edu ).
- Added --enable-large-files and --disable-large-files switches to configure . See configure.in for some systems where this is automatically turned off because of broken implementations.
- Upgraded readline to 4.0.
- New CREATE TABLE options: PACK_KEYS and CHECKSUM .
- Added --default-table-type option to mysqld .

# 13.5 Changes in release 3.22.x (Older; still supported)

The 3.22 version has faster and safer connect code than version 3.21, as well as a lot of new nice enhancements. As there aren't really any major changes, upgrading from 3.21 to 3.22 should be very easy and painless. Upgrading from Version 3.21 to 3.22 .

## 13.5.1 Changes in release 3.22.35

- Fixed problem with STD() .
- Merged changes from the newest ISAM library from 3.23.
- Fixed problem with INSERT DELAYED .
- Fixed a bug core dump when using a LEFT JOIN / STRAIGHT_JOIN on a table with only one row.

## 13.5.2 Changes in release 3.22.34

- Fixed problem with GROUP BY on TINYBLOB columns; this caused bugzilla to not show rows in some queries.
- Had to do total recompile of the Windows binary version as VC++ didn't compile all relevant files for 3.22.33 :(

## 13.5.3 Changes in release 3.22.33

- Fixed problems in Windows when locking tables with LOCK TABLE .
- Quicker kill of SELECT DISTINCT queries.

## 13.5.4 Changes in release 3.22.32 (14 Feb 2000)

- Fixed problem when storing numbers in timestamps.
- Fix problem with timezones that have half hour offsets.
- Added mysqlhotcopy , a fast online hot-backup utility for local MySQL databases. By Tim Bunce.
- New more secure mysqlaccess . Thanks to Steve Harvey for this.
- Fixed security problem in the protocol regarding password checking.
- Fixed problem that affected queries that did arithmetic on GROUP functions.
- Fixed a bug in the ISAM code when deleting rows on tables with packed indexes.

## 13.5.5 Changes in release 3.22.31

- A few small fixes for the Windows version.

## 13.5.6 Changes in release 3.22.30

- Fixed optimiser problem on SELECT when using many overlapping indexes.
- Disabled floating-point exceptions for FreeBSD to fix core dump when doing SELECT FLOOR(POW(2,63)) .
- Added print of default arguments options to all clients.
- Fixed critical problem with the WITH GRANT OPTION option.
- Fixed non-critical Y2K problem when writing short date to log files.

## 13.5.7 Changes in release 3.22.29 (02 Jan 2000)

- Upgraded the configure and include files to match the latest 3.23 version. This should increase portability and make it easier to build shared libraries.
- Added latest patches to MIT-pthreads for NetBSD.
- Fixed problem with timezones that are < GMT -11.
- Fixed a bug when deleting packed keys in NISAM.
- Fixed problem that could cause MySQL to touch freed memory when doing very complicated GROUP BY queries.
- Fixed core dump if you got a crashed table where an ENUM field value was too big.
- Added mysqlshutdown.exe and mysqlwatch.exe to the Windows distribution.
- Fixed problem when doing ORDER BY on a reference key.
- Fixed that INSERT DELAYED doesn't update timestamps that are given.

## 13.5.8 Changes in release 3.22.28 (20 Oct 1999)

- Fixed problem with LEFT JOIN and COUNT() on a column which was declared NULL + and it had a DEFAULT value.
- Fixed core dump problem when using CONCAT() in a WHERE clause.
- Fixed problem with AVG() and STD() with NULL values.

## 13.5.9 Changes in release 3.22.27

- Fixed prototype in my_ctype.h when using other character sets.
- Some configure issues to fix problems with big filesystem detection.
- Fixed problem when sorting on big BLOB columns.
- ROUND() will now work on Windows.

## 13.5.10 Changes in release 3.22.26 (16 Sep 1999)

- Fixed core dump with empty BLOB/TEXT column argument to REVERSE() .
- Extended /*! */ with version numbers.
- Changed SUBSTRING(text FROM pos) to conform to ANSI SQL. (Before this construct returned the rightmost 'pos' characters.)
- Fixed problem with LOCK TABLES combined with DELETE FROM table
- Fixed problem that INSERT ... SELECT didn't use BIG_TABLES .
- SET SQL_LOW_PRIORITY_UPDATES=# didn't work.
- Password wasn't updated correctly if privileges didn't change on: GRANT ... IDENTIFIED BY
- Fixed range optimiser bug in SELECT * FROM table_name WHERE key_part1 >= const AND (key_part2 = const OR key_part2 = const) .
- Fixed bug in compression key handling in ISAM .

## 13.5.11 Changes in release 3.22.25

- Fixed some small problems with the installation.

## 13.5.12 Changes in release 3.22.24 (05 Jul 1999)

- DATA is not a reserved word anymore.
- Fixed optimiser bug with tables with only one row.
- Fixed bug when using LOCK TABLES table_name READ; FLUSH TABLES;
- Applied some patches for HP-UX.
- isamchk should now work on Windows.
- Changed configure to not use big file handling on Linux as this crashes some RedHat 6.0 systems

## 13.5.13 Changes in release 3.22.23 (08 Jun 1999)

- Upgraded to use Autoconf 2.13, Automake 1.4 and libtool 1.3.2.
- Better support for SCO in configure .
- Added option --defaults-file=### to option file handling to force use of only one specific option file.
- Extended CREATE syntax to ignore MySQL Version 3.23 keywords.
- Fixed deadlock problem when using INSERT DELAYED on a table locked with LOCK TABLES .
- Fixed deadlock problem when using DROP TABLE on a table that was locked by another thread.
- Add logging of GRANT/REVOKE commands in the update log.
- Fixed isamchk to detect a new error condition.
- Fixed bug in NATURAL LEFT JOIN .

## 13.5.14 Changes in release 3.22.22 (30 Apr 1999)

- Fixed problem in the C API when you called mysql_close() directly after mysql_init() .
- Better client error message when you can't open socket.
- Fixed delayed_insert_thread counting when you couldn't create a new delayed_insert thread.
- Fixed bug in CONCAT() with many arguments.
- Added patches for DEC 3.2 and SCO.
- Fixed path-bug when installing MySQL as a service on NT.
- The MySQL-Windows version is now compiled with VC++ 6.0 instead of with VC++ 5.0.
- New installation setup for MySQL-Windows.

## 13.5.15 Changes in release 3.22.21

- Fixed problem with DELETE FROM TABLE when table was locked by another thread.
- Fixed bug in LEFT JOIN involving empty tables.
- Changed the mysql.db column from CHAR(32) to CHAR(60) .
- MODIFY and DELAYED are not reserved words anymore.
- Fixed a bug when storing days in a TIME column.
- Fixed a problem with Host '...' is not allowed to connect to this MySQL server after one had inserted a new MySQL user with a GRANT command.
- Changed to use TCP_NODELAY also on Linux (should give faster TCP/IP connections).

## 13.5.16 Changes in release 3.22.20 (18 Mar 1999)

- Fixed STD() for big tables when result should be 0.
- The update log didn't have newlines on some operating systems.
- INSERT DELAYED had some garbage at end in the update log.

## 13.5.17 Changes in release 3.22.19 (Mar 1999: Stable)

- Fixed bug in mysql_install_db (from 3.22.17).
- Changed default key cache size to 8M.
- Fixed problem with queries that needed temporary tables with BLOB columns.

## 13.5.18 Changes in release 3.22.18

- Fixes a fatal problem in 3.22.17 on Linux; after shutdown not all threads died properly.
- Added option -O flush_time=# to mysqld . This is mostly useful on Windows and tells how often MySQL should close all unused tables and flush all updated tables to disk.
- Fixed problem that a VARCHAR column compared with CHAR column didn't use keys efficiently.

## 13.5.19 Changes in release 3.22.17

- Fixed a core dump problem when using --log-update and connecting without a default database.
- Fixed some configure and portability problems.
- Using LEFT JOIN on tables that had circular dependencies caused mysqld to hang forever.

## 13.5.20 Changes in release 3.22.16 (Feb 1999: Gamma)

- mysqladmin processlist could kill the server if a new user logged in.
- DELETE FROM tbl_name WHERE key_column=col_name didn't find any matching rows. Fixed.
- DATE_ADD(column, ...) didn't work.
- INSERT DELAYED could deadlock with status 'upgrading lock'
- Extended ENCRYPT() to take longer salt strings than 2 characters.
- longlong2str is now much faster than before. For Intel x86 platforms, this function is written in optimised assembler.
- Added the MODIFY keyword to ALTER TABLE .

## 13.5.21 Changes in release 3.22.15

- GRANT used with IDENTIFIED BY didn't take effect until privileges were flushed.
- Name change of some variables in SHOW STATUS .
- Fixed problem with ORDER BY with 'only index' optimisation when there were multiple key definitions for a used column.
- DATE and DATETIME columns are now up to 5 times faster than before.
- INSERT DELAYED can be used to let the client do other things while the server inserts rows into a table.
- LEFT JOIN USING (col1,col2) didn't work if one used it with tables from 2 different databases.
- LOAD DATA LOCAL INFILE didn't work in the Unix version because of a missing file.
- Fixed problems with VARCHAR / BLOB on very short rows (< 4 bytes); error 127 could occur when deleting rows.
- Updating BLOB/TEXT through formulas didn't work for short (< 256 char) strings.
- When you did a GRANT on a new host, mysqld could die on the first connect from this host.
- Fixed bug when one used ORDER BY on column name that was the same name as an alias.
- Added BENCHMARK(loop_count,expression) function to time expressions.

## 13.5.22 Changes in release 3.22.14

- Allow empty arguments to mysqld to make it easier to start from shell scripts.
- Setting a TIMESTAMP column to NULL didn't record the timestamp value in the update log.
- Fixed lock handler bug when one did INSERT INTO TABLE ... SELECT ... GROUP BY .
- Added a patch for localtime_r() on Windows so that it will not crash anymore if your date is > 2039, but instead will return a time of all zero.
- Names for user-defined functions are no longer case-sensitive.
- Added escape of ^Z (ASCII 26) to \Z as ^Z doesn't work with pipes on Windows.
- mysql_fix_privileges adds a new column to the mysql.func to support aggregate UDF functions in future MySQL releases.

## 13.5.23 Changes in release 3.22.13

- Saving NOW() , CURDATE() or CURTIME() directly in a column didn't work.
- SELECT COUNT(*) ... LEFT JOIN ... didn't work with no WHERE part.
- Updated config.guess to allow MySQL to configure on UnixWare 7.0.x.
- Changed the implementation of pthread_cond() on the Windows version. get_lock() now correctly times out on Windows!

## 13.5.24 Changes in release 3.22.12

- Fixed problem when using DATE_ADD() and DATE_SUB() in a WHERE clause.
- You can now set the password for a user with the GRANT ... TO user IDENTIFIED BY 'password' syntax.
- Fixed bug in GRANT checking with SELECT on many tables.
- Added missing file mysql_fix_privilege_tables to the RPM distribution. This is not run by default because it relies on the client package.
- Added option SQL_SMALL_RESULT to SELECT to force use of fast temporary tables when you know that the result set will be small.
- Allow use of negative real numbers without a decimal point.
- Day number is now adjusted to maximum days in month if the resulting month after DATE_ADD / DATE_SUB() doesn't have enough days.
- Fix that GRANT compares columns in case-insensitive fashion.
- Fixed a bug in sql_list.h that made ALTER TABLE dump core in some contexts.
- The hostname in user@hostname can now include '.' and '-' without quotes in the context of the GRANT , REVOKE and SET PASSWORD FOR ... statements.
- Fix for isamchk for tables which need big temporary files.

## 13.5.25 Changes in release 3.22.11

- **Important** : You must run the mysql_fix_privilege_tables script when you upgrade to this version! This is needed because of the new GRANT system. If you don't do this, you will get Access denied when you try to use ALTER TABLE , CREATE INDEX , or DROP INDEX .
- GRANT to allow/deny users table and column access.
- Changed USER() to return a value in user@host format. Formerly it returned only user .
- Changed the syntax for how to set PASSWORD for another user.
- New command FLUSH STATUS that resets most status variables to zero.
- New status variables: aborted_threads , aborted_connects .
- New option variable: connection_timeout .
- Added support for Thai sorting (by Pruet Boonma pruet@ds90.intanon.nectec.or.th ).
- Slovak and japanese error messages.
- Configuration and portability fixes.
- Added option SET SQL_WARNINGS=1 to get a warning count also for simple inserts.
- MySQL now uses SIGTERM instead of SIGQUIT with shutdown to work better on FreeBSD.
- Added option \G (print vertically) to mysql .
- SELECT HIGH_PRIORITY ... killed mysqld .
- IS NULL on a AUTO_INCREMENT column in a LEFT JOIN didn't work as expected.
- New function MAKE_SET() .

## 13.5.26 Changes in release 3.22.10

- mysql_install_db no longer starts the MySQL server! You should start mysqld with safe_mysqld after installing it! The MySQL RPM will, however, start the server as before.
- Added --bootstrap option to mysqld and recoded mysql_install_db to use it. This will make it easier to install MySQL with RPMs.
- Changed + , - (sign and minus), * , / , % , ABS() and MOD() to be BIGINT aware (64-bit safe).
- Fixed a bug in ALTER TABLE that caused mysqld to crash.
- MySQL now always reports the conflicting key values when a duplicate key entry occurs. (Before this was only reported for INSERT .)
- New syntax: INSERT INTO tbl_name SET col_name=value, col_name=value, ...
- Most errors in the .err log are now prefixed with a time stamp.
- Added option MYSQL_INIT_COMMAND to mysql_options() to make a query on connect or reconnect.
- Added option MYSQL_READ_DEFAULT_FILE and MYSQL_READ_DEFAULT_GROUP to mysql_options() to read the following parameters from the MySQL option files: port , socket , compress , password , pipe , timeout , user , init-command , host and database .
- Added maybe_null to the UDF structure.
- Added option IGNORE to INSERT statements with many rows.
- Fixed some problems with sorting of the koi8 character sets; users of koi8 **must** run isamchk -rq on each table that has an index on a CHAR or VARCHAR column.
- New script mysql_setpermission , by Luuk de Boer. It allows easy creation of new users with permissions for specific databases.
- Allow use of hexadecimal strings (0x...) when specifying a constant string (like in the column separators with LOAD DATA INFILE ).
- Ported to OS/2 (thanks to Antony T. Curtis antony.curtis@olcs.net ).
- Added more variables to SHOW STATUS and changed format of output to be like SHOW VARIABLES .
- Added extended-status command to mysqladmin which will show the new status variables.

## 13.5.27 Changes in release 3.22.9

- SET SQL_LOG_UPDATE=0 caused a lockup of the server.
- New SQL command: FLUSH [ TABLES | HOSTS | LOGS | PRIVILEGES ] [, ...]
- New SQL command: KILL thread_id .
- Added casts and changed include files to make MySQL easier to compile on AIX and DEC OSF/1 4.x
- Fixed conversion problem when using ALTER TABLE from a INT to a short CHAR() column.
- Added SELECT HIGH_PRIORITY ; this will get a lock for the SELECT even if there is a thread waiting for another SELECT to get a WRITE LOCK .
- Moved wild_compare() to string class to be able to use LIKE on BLOB / TEXT columns with \0 .
- Added ESCAPE option to LIKE .
- Added a lot more output to mysqladmin debug .
- You can now start mysqld on Windows with the --flush option. This will flush all tables to disk after each update. This makes things much safer on the Windows platforms but also **much** slower.

## 13.5.28 Changes in release 3.22.8

- Czech character sets should now work much better. You must also install http://www.mysql.com/Downloads/Patches/czech-3.22.8-patch . This patch should also be installed if you are using a character set which uses my_strcoll() ! The patch should always be safe to install (for any system), but as this patch changes ISAM internals it's not yet in the default distribution.
- DATE_ADD() and DATE_SUB() didn't work with group functions.
- mysql will now also try to reconnect on USE DATABASE commands.
- Fix problem with ORDER BY and LEFT JOIN and const tables.
- Fixed problem with ORDER BY if the first ORDER BY column was a key and the rest of the ORDER BY columns wasn't part of the key.
- Fixed a big problem with OPTIMIZE TABLE .
- MySQL clients on NT will now by default first try to connect with named pipes and after this with TCP/IP.
- Fixed a problem with DROP TABLE and mysqladmin shutdown on Windows (a fatal bug from 3.22.6).
- Fixed problems with TIME columns and negative strings.
- Added an extra thread signal loop on shutdown to avoid some error messages from the client.
- MySQL now uses the next available number as extension for the update log file.
- Added patches for UNIXWARE 7.

## 13.5.29 Changes in release 3.22.7 (Sep 1998: Beta)

- Added LIMIT clause for the DELETE statement.
- You can now use the /*! ... */ syntax to hide MySQL-specific keywords when you write portable code. MySQL will parse the code inside the comments as if the surrounding /*! and */ comment characters didn't exist.
- OPTIMIZE TABLE tbl_name can now be used to reclaim disk space after many deletes. Currently, this uses ALTER TABLE to regenerate the table, but in the future it will use an integrated isamchk for more speed.
- Upgraded libtool to get the configure more portable.
- Fixed slow UPDATE and DELETE operations when using DATETIME or DATE keys.
- Changed optimiser to make it better at deciding when to do a full join and when using keys.
- You can now use mysqladmin proc to display information about your own threads. Only users with the PROCESS privilege can get information about all threads. (In 4.0.2 one needs the SUPER privilege for this.)
- Added handling of formats YYMMDD , YYYYMMDD , YYMMDDHHMMSS for numbers when using DATETIME and TIMESTAMP types. (Formerly these formats only worked with strings.)
- Added connect option CLIENT_IGNORE_SPACE to allow use of spaces after function names and before '(' (Powerbuilder requires this). This will make all function names reserved words.
- Added the --log-long-format option to mysqld to enable timestamps and INSERT_ID's in the update log.
- Added --where option to mysqldump (patch by Jim Faucette).
- The lexical analyser now uses ``perfect hashing'' for faster parsing of SQL statements.

## 13.5.30 Changes in release 3.22.6

- Faster mysqldump .
- For the LOAD DATA INFILE statement, you can now use the new LOCAL keyword to read the file from the client. mysqlimport will automatically use LOCAL when importing with the TCP/IP protocol.
- Fixed small optimise problem when updating keys.
- Changed makefiles to support shared libraries.
- MySQL-NT can now use named pipes, which means that you can now use MySQL-NT without having to install TCP/IP.

## 13.5.31 Changes in release 3.22.5

- All table lock handing is changed to avoid some very subtle deadlocks when using DROP TABLE , ALTER TABLE , DELETE FROM TABLE and mysqladmin flush-tables under heavy usage. Changed locking code to get better handling of locks of different types.
- Updated DBI to 1.00 and DBD to 1.2.0.
- Added a check that the error message file contains error messages suitable for the current version of mysqld . (To avoid errors if you accidentally try to use an old error message file.)
- All count structures in the client ( affected_rows() , insert_id() , ...) are now of type BIGINT to allow 64-bit values to be used. This required a minor change in the MySQL protocol which should affect only old clients when using tables with AUTO_INCREMENT values > 16M.
- The return type of mysql_fetch_lengths() has changed from uint * to ulong * . This may give a warning for old clients but should work on most machines.
- Change mysys and dbug libraries to allocate all thread variables in one struct. This makes it easier to make a threaded libmysql.dll library.
- Use the result from gethostname() (instead of uname() ) when constructing .pid file names.
- New better compressed server/client protocol.
- COUNT() , STD() and AVG() are extended to handle more than 4G rows.
- You can now store values in the range -838:59:59 <= x <= 838:59:59 in a TIME column.
- **Warning: incompatible change!!** If you set a TIME column to too short a value, MySQL now assumes the value is given as: [[[D ]HH:]MM:]SS instead of HH[:MM[:SS]] .
- TIME_TO_SEC() and SEC_TO_TIME() can now handle negative times and hours up to 32767.
- Added new option SET SQL_LOG_UPDATE={0|1} to allow users with the PROCESS privilege to bypass the update log. (Modified patch from Sergey A Mukhin violet@rosnet.net .)
- Fixed fatal bug in LPAD() .
- Initialise line buffer in mysql.cc to make BLOB reading from pipes safer.
- Added -O max_connect_errors=# option to mysqld . Connect errors are now reset for each correct connection.
- Increased the default value of max_allowed_packet to 1M in mysqld .
- Added --low-priority-updates option to mysqld , to give table-modifying operations ( INSERT , REPLACE , UPDATE , DELETE ) lower priority than retrievals. You can now use {INSERT | REPLACE | UPDATE | DELETE} LOW_PRIORITY ... You can also use SET SQL_LOW_PRIORITY_UPDATES={0|1} to change the priority for one thread. One side effect is that LOW_PRIORITY is now a reserved word. :(
- Add support for INSERT INTO table ... VALUES(...),(...),(...) , to allow inserting multiple rows with a single statement.
- INSERT INTO tbl_name is now also cached when used with LOCK TABLES . (Previously only INSERT ... SELECT and LOAD DATA INFILE were cached.)
- Allow GROUP BY functions with HAVING :

```
mysql> SELECT col FROM table GROUP BY col HAVING COUNT(*)>0;
```

- mysqld will now ignore trailing ';' characters in queries. This is to make it easier to migrate from some other SQL servers that require the trailing ';' .
- Fix for corrupted fixed-format output generated by SELECT INTO OUTFILE .
- **Warning: incompatible change!** Added Oracle GREATEST() and LEAST() functions. You must now use these instead of the MAX() and MIN() functions to get the largest/smallest value from a list of values. These can now handle REAL , BIGINT and string ( CHAR or VARCHAR ) values.
- **Warning: incompatible change!** DAYOFWEEK() had offset 0 for Sunday. Changed the offset to 1.
- Give an error for queries that mix GROUP BY columns and fields when there is no GROUP BY specification.
- Added --vertical option to mysql , for printing results in vertical mode.
- Index-only optimisation; some queries are now resolved using only indexes. Until MySQL 4.0, this works only for numeric columns. Comment MySQL utilise les index .
- Lots of new benchmarks.
- A new C API chapter and lots of other improvements in the manual.

## 13.5.32 Changes in release 3.22.4

- Added --tmpdir option to mysqld , for specifying the location of the temporary file directory.
- MySQL now automatically changes a query from an ODBC client:

```
SELECT ... FROM table WHERE auto_increment_column IS NULL
```

to:

```
SELECT ... FROM table WHERE auto_increment_column == LAST_INSERT_ID()
```

This allows some ODBC programs (Delphi, Access) to retrieve the newly inserted row to fetch the AUTO_INCREMENT id.

- DROP TABLE now waits for all users to free a table before deleting it.
- Fixed small memory leak in the new connect protocol.
- New functions BIN() , OCT() , HEX() and CONV() for converting between different number bases.
- Added function SUBSTRING() with 2 arguments.
- If you created a table with a record length smaller than 5, you couldn't delete rows from the table.
- Added optimisation to remove const reference tables from ORDER BY and GROUP BY .
- mysqld now automatically disables system locking on Linux and Windows, and for systems that use MIT-pthreads. You can force the use of locking with the --enable-external-locking option.
- Added --console option to mysqld , to force a console window (for error messages) when using Windows.
- Fixed table locks for Windows.
- Allow '$' in identifiers.
- Changed name of user-specific configuration file from my.cnf to .my.cnf (Unix only).
- Added DATE_ADD() and DATE_SUB() functions.

## 13.5.33 Changes in release 3.22.3

- Fixed a lock problem (bug in MySQL Version 3.22.1) when closing temporary tables.
- Added missing mysql_ping() to the client library.
- Added --compress option to all MySQL clients.
- Changed byte to char in mysql.h and mysql_com.h .

## 13.5.34 Changes in release 3.22.2

- Searching on multiple constant keys that matched more than 30% of the rows didn't always use the best possible key.
- New functions << , >> , RPAD() and LPAD() .
- You can now save default options (like passwords) in a configuration file ( my.cnf ).
- Lots of small changes to get ORDER BY to work when no records are found when using fields that are not in GROUP BY (MySQL extension).
- Added --chroot option to mysqld , to start mysqld in a chroot environment (by Nikki Chumakov nikkic@cityline.ru ).
- Trailing spaces are now ignored when comparing case-sensitive strings; this should fix some problems with ODBC and flag 512!
- Fixed a core dump bug in the range optimiser.
- Added --one-thread option to mysqld , for debugging with LinuxThreads (or glibc ). (This replaces the -T32 flag)
- Added DROP TABLE IF EXISTS to prevent an error from occurring if the table doesn't exist.
- IF and EXISTS are now reserved words (they would have to be sooner or later).
- Added lots of new options to mysqldump .
- Server error messages are now in mysqld_error.h .
- The server/client protocol now supports compression.
- All bug fixes from MySQL Version 3.21.32.

## 13.5.35 Changes in release 3.22.1 (Jun 1998: Alpha)

- Added new C API function mysql_ping() .
- Added new API functions mysql_init() and mysql_options() . You now MUST call mysql_init() before you call mysql_real_connect() . You don't have to call mysql_init() if you only use mysql_connect() .
- Added mysql_options(...,MYSQL_OPT_CONNECT_TIMEOUT,...) so you can set a timeout for connecting to a server.
- Added --timeout option to mysqladmin , as a test of mysql_options() .
- Added AFTER column and FIRST options to ALTER TABLE ... ADD columns . This makes it possible to add a new column at some specific location within a row in an existing table.
- WEEK() now takes an optional argument to allow handling of weeks when the week starts on Monday (some European countries). By default, WEEK() assumes the week starts on Sunday.
- TIME columns weren't stored properly (bug in MySQL Version 3.22.0).
- UPDATE now returns information about how many rows were matched and updated, and how many ``warnings'' occurred when doing the update.
- Fixed incorrect result from FORMAT(-100,2) .
- ENUM and SET columns were compared in binary (case-sensitive) fashion; changed to be case-insensitive.

## 13.5.36 Changes in release 3.22.0

- New (backward-compatible) connect protocol that allows you to specify the database to use when connecting, to get much faster connections to a specific database.

  The `mysql_real_connect()` call is changed to:

```
mysql_real_connect(MYSQL *mysql, const char *host, const char *user,
                   const char *passwd, const char *db, uint port,
                   const char *unix_socket, uint client_flag)
```

- Each connection is handled by its own thread, rather than by the master accept() thread. This fixes permanently the telnet bug that was a topic on the mail list some time ago.
- All TCP/IP connections are now checked with backward-resolution of the hostname to get better security. mysqld now has a local hostname resolver cache so connections should actually be faster than before, even with this feature.
- A site automatically will be blocked from future connections if someone repeatedly connects with an ``improper header'' (like when one uses telnet).
- You can now refer to tables in different databases with references of the form tbl_name@db_name or db_name.tbl_name . This makes it possible to give a user read access to some tables and write access to others simply by keeping them in different databases!
- Added --user option to mysqld , to allow it to run as another Unix user (if it is started as the Unix root user).
- Added caching of users and access rights (for faster access rights checking)
- Normal users (not anonymous ones) can change their password with mysqladmin password 'new_password' . This uses encrypted passwords that are not logged in the normal MySQL log!

- All important string functions are now coded in assembler for x86 Linux machines. This gives a speedup of 10% in many cases.
- For tables that have many columns, the column names are now hashed for much faster column name lookup (this will speed up some benchmark tests a lot!)
- Some benchmarks are changed to get better individual timing. (Some loops were so short that a specific test took < 2 seconds. The loops have been changed to take about 20 seconds to make it easier to compare different databases. A test that took 1–2 seconds before now takes 11–24 seconds, which is much better)
- Re–arranged SELECT code to handle some very specific queries involving group functions (like COUNT(*) ) without a GROUP BY but with HAVING . The following now works:

```
mysql> SELECT COUNT(*) as C FROM table HAVING C > 1;
```

- Changed the protocol for field functions to be faster and avoid some calls to malloc() .
- Added –T32 option to mysqld , for running all queries under the main thread. This makes it possible to debug mysqld under Linux with gdb !
- Added optimisation of not_null_column IS NULL (needed for some Access queries).
- Allow STRAIGHT_JOIN to be used between two tables to force the optimiser to join them in a specific order.
- String functions now return VARCHAR rather than CHAR and the column type is now VARCHAR for fields saved as VARCHAR . This should make the MyODBC driver better, but may break some old MySQL clients that don't handle FIELD_TYPE_VARCHAR the same way as FIELD_TYPE_CHAR .
- CREATE INDEX and DROP INDEX are now implemented through ALTER TABLE . CREATE TABLE is still the recommended (fast) way to create indexes.
- Added ––set–variable option wait_timeout to mysqld .
- Added time column to mysqladmin processlist to show how long a query has taken or how long a thread has slept.
- Added lots of new variables to show variables and some new to show status .
- Added new type YEAR . YEAR is stored in 1 byte with allowable values of 0, and 1901 to 2155.
- Added new DATE type that is stored in 3 bytes rather than 4 bytes. All new tables are created with the new date type if you don't use the ––old–protocol option to mysqld .
- Fixed bug in record caches; for some queries, you could get Error from table handler: # on some operating systems.
- Added ––enable–assembler option to configure , for x86 machines (tested on Linux + gcc ). This will enable assembler functions for the most important string functions for more speed!

# 13.6 Changes in release 3.21.x

Version 3.21 is quite old now, and should be avoided if possible. This information is kept here for historical purposes only.

## 13.6.1 Changes in release 3.21.33

- Fixed problem when sending SIGHUP to mysqld ; mysqld core dumped when starting from boot on some systems.
- Fixed problem with losing a little memory for some connections.
- DELETE FROM tbl_name without a WHERE condition is now done the long way when you use LOCK TABLES or if the table is in use, to avoid race conditions.
- INSERT INTO TABLE (timestamp_column) VALUES (NULL); didn't set timestamp.

## 13.6.2 Changes in release 3.21.32

- Fixed some possible race conditions when doing many reopen/close on the same tables under heavy load! This can happen if you execute mysqladmin refresh often. This could in some very rare cases corrupt the header of the index file and cause error 126 or 138.
- Fixed fatal bug in refresh() when running with the ––skip–external–locking option. There was a ``very small'' time gap after a mysqladmin refresh when a table could be corrupted if one thread updated a table while another thread did mysqladmin refresh and another thread started a new update ont the same table before the first thread had finished. A refresh (or ––flush–tables ) will now not return until all used tables are closed!
- SELECT DISTINCT with a WHERE clause that didn't match any rows returned a row in some contexts (bug only in 3.21.31).
- GROUP BY + ORDER BY returned one empty row when no rows where found.
- Fixed a bug in the range optimiser that wrote Use_count: Wrong count for ... in the error log file.

## 13.6.3 Changes in release 3.21.31

- Fixed a sign extension problem for the TINYINT type on Irix.
- Fixed problem with LEFT("constant_string",function) .
- Fixed problem with FIND_IN_SET() .
- LEFT JOIN core dumped if the second table is used with a constant WHERE/ON expression that uniquely identifies one record.
- Fixed problems with DATE_FORMAT() and incorrect dates. DATE_FORMAT() now ignores '%' to make it possible to extend it more easily in the future.

## 13.6.4 Changes in release 3.21.30

- mysql now returns an exit code > 0 if the query returned an error.
- Saving of command–line history to file in mysql client. By Tommy Larsen tommy@mix.hive.no .
- Fixed problem with empty lines that were ignored in mysql.cc .
- Save the pid of the signal handler thread in the pid file instead of the pid of the main thread.
- Added patch by tommy@valley.ne.jp to support Japanese characters SJIS and UJIS.
- Changed safe_mysqld to redirect startup messages to 'hostname'.err instead of 'hostname'.log to reclaim file space on mysqladmin refresh .
- ENUM always had the first entry as default value.
- ALTER TABLE wrote two entries to the update log.
- sql_acc() now closes the mysql grant tables after a reload to save table space and memory.
- Changed LOAD DATA to use less memory with tables and BLOB columns.

- Sorting on a function which made a division / 0 produced a wrong set in some cases.
- Fixed SELECT problem with LEFT() when using the czech character set.
- Fixed problem in isamchk ; it couldn't repair a packed table in a very unusual case.
- SELECT statements with  or | (bit functions) failed on columns with NULL values.
- When comparing a field = field, where one of the fields was a part key, only the length of the part key was compared.

## 13.6.5 Changes in release 3.21.29

- LOCK TABLES + DELETE from tbl_name never removed locks properly.
- Fixed problem when grouping on an OR function.
- Fixed permission problem with umask() and creating new databases.
- Fixed permission problem on result file with SELECT ... INTO OUTFILE ...
- Fixed problem in range optimiser (core dump) for a very complex query.
- Fixed problem when using MIN(integer) or MAX(integer) in GROUP BY .
- Fixed bug on Alpha when using integer keys. (Other keys worked on Alpha.)
- Fixed bug in WEEK("XXXX-xx-01") .

## 13.6.6 Changes in release 3.21.28

- Fixed socket permission (clients couldn't connect to Unix socket on Linux).
- Fixed bug in record caches; for some queries, you could get Error from table handler: # on some operating systems.

## 13.6.7 Changes in release 3.21.27

- Added user level lock functions GET_LOCK(string,timeout) , RELEASE_LOCK(string) .
- Added Opened_tables to show status .
- Changed connect timeout to 3 seconds to make it somewhat harder for crackers to kill mysqld through telnet + TCP/IP.
- Fixed bug in range optimiser when using WHERE key_part_1 >= something AND key_part_2 <= something_else .
- Changed configure for detection of FreeBSD 3.0 9803xx and above
- WHERE with string_col_key = constant_string didn't always find all rows if the column had many values differing only with characters of the same sort value (like e and 'e).
- Strings keys looked up with 'ref' were not compared in case-sensitive fashion.
- Added umask() to make log files non-readable for normal users.
- Ignore users with old (8-byte) password on startup if not using --old-protocol option to mysqld .
- SELECT which matched all key fields returned the values in the case of the matched values, not of the found values. (Minor problem.)

## 13.6.8 Changes in release 3.21.26

- FROM_DAYS(0) now returns "0000-00-00".
- In DATE_FORMAT() , PM and AM were swapped for hours 00 and 12.
- Extended the default maximum key size to 256.
- Fixed bug when using BLOB / TEXT in GROUP BY with many tables.
- An ENUM field that is not declared NOT NULL has NULL as the default value. (Previously, the default value was the first enumeration value.)
- Fixed bug in the join optimiser code when using many part keys on the same key: INDEX (Organisation,Surname(35),Initials(35)) .
- Added some tests to the table order optimiser to get some cases with SELECT ... FROM many_tables much faster.
- Added a retry loop around accept() to possibly fix some problems on some Linux machines.

## 13.6.9 Changes in release 3.21.25

- Changed typedef 'string' to typedef 'my_string' for better portability.
- You can now kill threads that are waiting on a disk-full condition.
- Fixed some problems with UDF functions.
- Added long options to isamchk . Try isamchk --help .
- Fixed a bug when using 8 bytes long (alpha); filesort() didn't work. Affects DISTINCT , ORDER BY and GROUP BY on 64-bit processors.

## 13.6.10 Changes in release 3.21.24

- Dynamic loadable functions. Based on source from Alexis Mikhailov.
- You couldn't delete from a table if no one had done a SELECT on the table.
- Fixed problem with range optimiser with many OR operators on key parts inside each other.
- Recoded MIN() and MAX() to work properly with strings and HAVING .
- Changed default umask value for new files from 0664 to 0660 .
- Fixed problem with LEFT JOIN and constant expressions in the ON part.
- Added Italian error messages from brenno@dewinter.com .
- configure now works better on OSF/1 (tested on 4.0D).
- Added hooks to allow LIKE optimisation with international character support.
- Upgraded DBI to 0.93.

## 13.6.11 Changes in release 3.21.23

- The following symbols are now reserved words: TIME , DATE , TIMESTAMP , TEXT , BIT , ENUM , NO , ACTION , CHECK , YEAR , MONTH , DAY , HOUR , MINUTE , SECOND , STATUS , VARIABLES .
- Setting a TIMESTAMP to NULL in LOAD DATA INFILE ... didn't set the current time for the TIMESTAMP .
- Fix BETWEEN to recognise binary strings. Now BETWEEN is case-sensitive.
- Added --skip-thread-priority option to mysqld , for systems where mysqld 's thread scheduling doesn't work properly (BSDI 3.1).
- Added ODBC functions DAYNAME() and MONTHNAME() .
- Added function TIME_FORMAT() . This works like DATE_FORMAT() , but takes a time string ( 'HH:MM:DD' ) as argument.
- Fixed unlikely(?) key optimiser bug when using OR s of key parts inside AND s.
- Added variables command to mysqladmin .
- A lot of small changes to the binary releases.
- Fixed a bug in the new protocol from MySQL Version 3.21.20.
- Changed ALTER TABLE to work with Windows (Windows can't rename open files). Also fixed a couple of small bugs in the Windows version.
- All standard MySQL clients are now ported to MySQL-Windows.
- MySQL can now be started as a service on NT.

## 13.6.12 Changes in release 3.21.22

- Starting with this version, all MySQL distributions will be configured, compiled and tested with crash-me and the benchmarks on the following platforms: SunOS 5.6 sun4u, SunOS 5.5.1 sun4u, SunOS 4.14 sun4c, SunOS 5.6 i86pc, Irix 6.3 mips5k, HP-UX 10.20 hppa, AIX 4.2.1 ppc, OSF/1 V4.0 alpha, FreeBSD 2.2.2 i86pc and BSDI 3.1 i386.
- Fix COUNT(*) problems when the WHERE clause didn't match any records. (Bug from 3.21.17.)
- Removed that NULL = NULL is true. Now you must use IS NULL or IS NOT NULL to test whether a value is NULL . (This is according to ANSI SQL but may break old applications that are ported from mSQL .) You can get the old behaviour by compiling with -DmSQL_COMPLIANT .
- Fixed bug that core dumped when using many LEFT OUTER JOIN clauses.
- Fixed bug in ORDER BY on string formula with possible NULL values.
- Fixed problem in range optimiser when using <= on sub index.
- Added functions DAYOFYEAR() , DAYOFMONTH() , MONTH() , YEAR() , WEEK() , QUARTER() , HOUR() , MINUTE() , SECOND() and FIND_IN_SET() .
- Added SHOW VARIABLES command.
- Added support of ``long constant strings'' from ANSI SQL:

```
mysql> SELECT 'first ' 'second';       -> 'first second'
```

- Upgraded Msql-Mysql-modules to 1.1825.
- Upgraded mysqlaccess to 2.02.
- Fixed problem with Russian character set and LIKE .
- Ported to OpenBSD 2.1.
- New Dutch error messages.

## 13.6.13 Changes in release 3.21.21a

- Configure changes for some operating systems.

## 13.6.14 Changes in release 3.21.21

- Fixed optimiser bug when using WHERE data_field = date_field2 AND date_field2 = constant .
- Added SHOW STATUS command.
- Removed manual.ps from the source distribution to make it smaller.

## 13.6.15 Changes in release 3.21.20

- Changed the maximum table name and column name lengths from 32 to 64.
- Aliases can now be of ``any'' length.
- Fixed mysqladmin stat to return the right number of queries.
- Changed protocol (downward compatible) to mark if a column has the AUTO_INCREMENT attribute or is a TIMESTAMP . This is needed for the new Java driver.
- Added Hebrew sorting order by Zeev Suraski.
- Solaris 2.6: Fixed configure bugs and increased maximum table size from 2G to 4G.

## 13.6.16 Changes in release 3.21.19

- Upgraded DBD to 1.1823. This version implements mysql_use_result in DBD-Mysql .
- Benchmarks updated for empress (by Luuk).
- Fixed a case of slow range searching.
- Configure fixes ( Docs directory).
- Added function REVERSE() (by Zeev Suraski).

## 13.6.17 Changes in release 3.21.18

- Issue error message if client C functions are called in wrong order.
- Added automatic reconnect to the libmysql.c library. If a write command fails, an automatic reconnect is done.
- Small sort sets no longer use temporary files.
- Upgraded DBI to 0.91.
- Fixed a couple of problems with LEFT OUTER JOIN .
- Added CROSS JOIN syntax. CROSS is now a reserved word.
- Recoded yacc / bison stack allocation to be even safer and to allow MySQL to handle even bigger expressions.
- Fixed a couple of problems with the update log.
- ORDER BY was slow when used with key ranges.

## 13.6.18 Changes in release 3.21.17

- Changed documentation string of --with-unix-socket-path to avoid confusion.
- Added ODBC and ANSI SQL style LEFT OUTER JOIN .
- The following are new reserved words: LEFT , NATURAL , USING .
- The client library now uses the value of the environment variable MYSQL_HOST as the default host if it's defined.
- SELECT col_name, SUM(expr) now returns NULL for col_name when there are matching rows.
- Fixed problem with comparing binary strings and BLOB s with ASCII characters over 127.
- Fixed lock problem: when freeing a read lock on a table with multiple read locks, a thread waiting for a write lock would have been given the lock. This shouldn't affect data integrity, but could possibly make mysqld restart if one thread was reading data that another thread modified.
- LIMIT offset,count didn't work in INSERT ... SELECT .
- Optimised key block caching. This will be quicker than the old algorithm when using bigger key caches.

## 13.6.19 Changes in release 3.21.16

- Added ODBC 2.0 &3.0 functions POWER() , SPACE() , COT() , DEGREES() , RADIANS() , ROUND(2 arg) and TRUNCATE() .
- **Warning: Incompatible change!** LOCATE() parameters were swapped according to ODBC standard. Fixed.
- Added function TIME_TO_SEC() .
- In some cases, default values were not used for NOT NULL fields.
- Timestamp wasn't always updated properly in UPDATE SET ... statements.
- Allow empty strings as default values for BLOB and TEXT , to be compatible with mysqldump .

## 13.6.20 Changes in release 3.21.15

- **Warning: Incompatible change!** mysqlperl is now from Msql-Mysql-modules. This means that connect() now takes host , database , user , password arguments! The old version took host , database , password , user .
- Allow DATE '1997-01-01' , TIME '12:10:10' and TIMESTAMP '1997-01-01 12:10:10' formats required by ANSI SQL. **Warning: Incompatible change!** This has the unfortunate side-effect that you no longer can have columns named DATE , TIME or TIMESTAMP . :( Old columns can still be accessed through tablename.columnname !)
- Changed Makefiles to hopefully work better with BSD systems. Also, manual.dvi is now included in the distribution to avoid having stupid make programs trying to rebuild it.
- readline library upgraded to version 2.1.
- A new sortorder german-1. That is a normal ISO-Latin1 with a german sort order.
- Perl DBI / DBD is now included in the distribution. DBI is now the recommended way to connect to MySQL from Perl.
- New portable benchmark suite with DBD , with test results from mSQL 2.0.3, MySQL, PostgreSQL 6.2.1 and Solid server 2.2.
- crash-me is now included with the benchmarks; this is a Perl program designed to find as many limits as possible in a SQL server. Tested with mSQL , PostgreSQL, Solid and MySQL.
- Fixed bug in range-optimiser that crashed MySQL on some queries.
- Table and column name completion for mysql command-line tool, by Zeev Suraski and Andi Gutmans.
- Added new command REPLACE that works like INSERT but replaces conflicting records with the new record. REPLACE INTO TABLE ... SELECT ... works also.
- Added new commands CREATE DATABASE db_name and DROP DATABASE db_name .
- Added RENAME option to ALTER TABLE : ALTER TABLE name RENAME TO new_name .
- make_binary_distribution now includes libgcc.a in libmysqlclient.a . This should make linking work for people who don't have gcc .
- Changed net_write() to my_net_write() because of a name conflict with Sybase.
- New function DAYOFWEEK() compatible with ODBC.
- Stack checking and bison memory overrun checking to make MySQL safer with weird queries.

## 13.6.21 Changes in release 3.21.14b

- Fixed a couple of small configure problems on some platforms.

## 13.6.22 Changes in release 3.21.14a

- Ported to SCO Openserver 5.0.4 with FSU Pthreads.
- HP-UX 10.20 should work.
- Added new function DATE_FORMAT() .
- Added NOT IN .
- Added automatic removal of 'ODBC function conversions': {fn now() }
- Handle ODBC 2.50.3 option flags.
- Fixed comparison of DATE and TIME values with NULL .
- Changed language name from germany to german to be consistent with the other language names.
- Fixed sorting problem on functions returning a FLOAT . Previously, the values were converted to INT s before sorting.
- Fixed slow sorting when sorting on key field when using key_column=constant .
- Sorting on calculated DOUBLE values sorted on integer results instead.

- mysql no longer requires a database argument.
- Changed the place where HAVING should be. According to ANSI, it should be after GROUP BY but before ORDER BY . MySQL Version 3.20 incorrectly had it last.
- Added Sybase command USE DATABASE to start using another database.
- Added automatic adjusting of number of connections and table cache size if the maximum number of files that can be opened is less than needed. This should fix that mysqld doesn't crash even if you haven't done a ulimit -n 256 before starting mysqld .
- Added lots of limit checks to make it safer when running with too little memory or when doing weird queries.

## 13.6.23 Changes in release 3.21.13

- Added retry of interrupted reads and clearing of errno . This makes Linux systems much safer!
- Fixed locking bug when using many aliases on the same table in the same SELECT .
- Fixed bug with LIKE on number key.
- New error message so you can check whether the connection was lost while the command was running or whether the connection was down from the start.
- Added --table option to mysql to print in table format. Moved time and row information after query result. Added automatic reconnect of lost connections.
- Added != as a synonym for <> .
- Added function VERSION() to make easier logs.
- New multi-user test tests/fork_test.pl to put some strain on the thread library.

## 13.6.24 Changes in release 3.21.12

- Fixed ftruncate() call in MIT-pthreads. This made isamchk destroy the .ISM files on (Free)BSD 2.x systems.
- Fixed broken __P_ patch in MIT-pthreads.
- Many memory overrun checks. All string functions now return NULL if the returned string should be longer than max_allowed_packet bytes.
- Changed the name of the INTERVAL type to ENUM , because INTERVAL is used in ANSI SQL.
- In some cases, doing a JOIN + GROUP + INTO OUTFILE , the result wasn't grouped.
- LIKE with '_' as last character didn't work. Fixed.
- Added extended ANSI SQL TRIM() function.
- Added CURTIME() .
- Added ENCRYPT() function by Zeev Suraski.
- Fixed better FOREIGN KEY syntax skipping. New reserved words: MATCH , FULL , PARTIAL .
- mysqld now allows IP number and hostname for the --bind-address option.
- Added SET CHARACTER SET cp1251_koi8 to enable conversions of data to and from the cp1251_koi8 character set.
- Lots of changes for Windows 95 port. In theory, this version should now be easily portable to Windows 95.
- Changed the CREATE COLUMN syntax of NOT NULL columns to be after the DEFAULT value, as specified in the ANSI SQL standard. This will make mysqldump with NOT NULL and default values incompatible with MySQL Version 3.20.
- Added many function name aliases so the functions can be used with ODBC or ANSI SQL92 syntax.
- Fixed syntax of ALTER TABLE tbl_name ALTER COLUMN col_name SET DEFAULT NULL .
- Added CHAR and BIT as synonyms for CHAR(1) .
- Fixed core dump when updating as a user who has only SELECT privilege.
- INSERT ... SELECT ... GROUP BY didn't work in some cases. An Invalid use of group function error occurred.
- When using LIMIT , SELECT now always uses keys instead of record scan. This will give better performance on SELECT and a WHERE that matches many rows.
- Added Russian error messages.

## 13.6.25 Changes in release 3.21.11

- Configure changes.
- MySQL now works with the new thread library on BSD/OS 3.0.
- Added new group functions BIT_OR() and BIT_AND() .
- Added compatibility functions CHECK and REFERENCES . CHECK is now a reserved word.
- Added ALL option to GRANT for better compatibility. ( GRANT is still a dummy function.)
- Added partly-translated Dutch error messages.
- Fixed bug in ORDER BY and GROUP BY with NULL columns.
- Added function LAST_INSERT_ID() SQL function to retrieve last AUTO_INCREMENT value. This is intended for clients to ODBC that can't use the mysql_insert_id() API function, but can be used by any client.
- Added --flush-logs option to mysqladmin .
- Added command STATUS to mysql .
- Fixed problem with ORDER BY / GROUP BY because of bug in gcc .
- Fixed problem with INSERT ... SELECT ... GROUP BY .

## 13.6.26 Changes in release 3.21.10

- New mysqlaccess .
- CREATE now supports all ODBC types and the mSQL TEXT type. All ODBC 2.5 functions are also supported (added REPEAT ). This provides better portability.
- Added text types TINYTEXT , TEXT , MEDIUMTEXT and LONGTEXT . These are actually BLOB types, but all searching is done in case-insensitive fashion.
- All old BLOB fields are now TEXT fields. This only changes that all searching on strings is done in case-sensitive fashion. You must do an ALTER TABLE and change the field type to BLOB if you want to have tests done in case-sensitive fashion.
- Fixed some configure issues.
- Made the locking code a bit safer. Fixed very unlikely deadlock situation.
- Fixed a couple of bugs in the range optimiser. Now the new range benchmark test-select works.

## 13.6.27 Changes in release 3.21.9

- Added --enable-unix-socket=pathname option to configure .
- Fixed a couple of portability problems with include files.
- Fixed bug in range calculation that could return empty set when searching on multiple key with only one entry (very rare).
- Most things ported to FSU Pthreads, which should allow MySQL to run on Caldera (SCO). Notes sur Caldera .

## 13.6.28 Changes in release 3.21.8

- Works now in Solaris 2.6.
- Added handling of calculation of SUM() functions. For example, you can now use SUM(column)/COUNT(column) .
- Added handling of trigometric functions: PI() , ACOS() , ASIN() , ATAN() , COS() , SIN() and TAN() .
- New languages: norwegian, norwegian-ny and portuguese.
- Fixed parameter bug in net_print() in procedure.cc .
- Fixed a couple of memory leaks.
- Now allow also the old SELECT ... INTO OUTFILE syntax.
- Fixed bug with GROUP BY and SELECT on key with many values.
- mysql_fetch_lengths() sometimes returned incorrect lengths when you used mysql_use_result() . This affected at least some cases of mysqldump --quick .
- Fixed bug in optimisation of WHERE const op field .
- Fixed problem when sorting on NULL fields.
- Fixed a couple of 64-bit (Alpha) problems.
- Added --pid-file=# option to mysqld .
- Added date formatting to FROM_UNIXTIME() , originally by Zeev Suraski.
- Fixed bug in BETWEEN in range optimiser (did only test = of the first argument).
- Added machine-dependent files for MIT-pthreads i386-SCO. There is probably more to do to get this to work on SCO 3.5.

## 13.6.29 Changes in release 3.21.7

- Changed Makefile.am to take advantage of Automake 1.2.
- Added the beginnings of a benchmark suite.
- Added more secure password handling.
- Added new client function mysql_errno() , to get the error number of the error message. This makes error checking in the client much easier. This makes the new server incompatible with the 3.20.x server when running without --old-protocol . The client code is backward-compatible. More information can be found in the README file!
- Fixed some problems when using very long, illegal names.

## 13.6.30 Changes in release 3.21.6

- Fixed more portability issues (incorrect sigwait and sigset defines).
- configure should now be able to detect the last argument to accept() .

## 13.6.31 Changes in release 3.21.5

- Should now work with FreeBSD 3.0 if used with FreeBSD-3.0-libc_r-1.0.diff , which can be found at http://www.mysql.com/Downloads/Patches/ .
- Added new -O tmp_table_size=# option to mysqld .
- New function FROM_UNIXTIME(timestamp) which returns a date string in 'YYYY-MM-DD HH:MM:DD' format.
- New function SEC_TO_TIME(seconds) which returns a string in 'HH:MM:SS' format.
- New function SUBSTRING_INDEX() , originally by Zeev Suraski.

## 13.6.32 Changes in release 3.21.4

- Should now configure and compile on OSF/1 4.0 with the DEC compiler.
- Configuration and compilation on BSD/OS 3.0 works, but due to some bugs in BSD/OS 3.0, mysqld doesn't work on it yet.
- Configuration and compilation on FreeBSD 3.0 works, but I couldn't get pthread_create to work.

## 13.6.33 Changes in release 3.21.3

- Added reverse check lookup of hostnames to get better security.
- Fixed some possible buffer overflows if filenames that are too long are used.
- mysqld doesn't accept hostnames that start with digits followed by a '.' , because the hostname may look like an IP number.
- Added --skip-networking option to mysqld , to allow only socket connections. (This will not work with MIT-pthreads!)
- Added check of too long table names for alias.
- Added check if database name is okay.
- Added check if too long table names.
- Removed incorrect free() that killed the server on CREATE DATABASE or DROP DATABASE .
- Changed some mysqld -O options to better names.
- Added -O join_cache_size=# option to mysqld .
- Added -O max_join_size=# option to mysqld , to be able to set a limit how big queries (in this case big = slow) one should be able to handle without specifying SET SQL_BIG_SELECTS=1 . A # = is about 10 examined records. The default is ``unlimited''.
- When comparing a TIME , DATE , DATETIME or TIMESTAMP column to a constant, the constant is converted to a time value before performing the comparison. This will make it easier to get ODBC (particularly Access97) to work with the above types. It should also make dates easier to use and the comparisons should be quicker than before.
- Applied patch from Jochen Wiedmann that allows query() in mysqlperl to take a query with \0 in it.

- Storing a timestamp with a 2-digit year ( YYMMDD ) didn't work.
- Fix that timestamp wasn't automatically updated if set in an UPDATE clause.
- Now the automatic timestamp field is the FIRST timestamp field.
- SELECT * INTO OUTFILE , which didn't correctly if the outfile already existed.
- mysql now shows the thread ID when starting or doing a reconnect.
- Changed the default sort buffer size from 2M to 1M.

## 13.6.34 Changes in release 3.21.2

- The range optimiser is coded, but only 85% tested. It can be enabled with --new , but it crashes core a lot yet...
- More portable. Should compile on AIX and alpha-digital. At least the isam library should be relatively 64-bit clean.
- New isamchk which can detect and fix more problems.
- New options for isamlog .
- Using new version of Automake.
- Many small portability changes (from the AIX and alpha-digital port) Better checking of pthread(s) library.
- czech error messages by snajdr@pvt.net .
- Decreased size of some buffers to get fewer problems on systems with little memory. Also added more checks to handle ``out of memory'' problems.
- mysqladmin : you can now do mysqladmin kill 5,6,7,8 to kill multiple threads.
- When the maximum connection limit is reached, one extra connection by a user with the **process_acl** privilege is granted.
- Added -O backlog=# option to mysqld .
- Increased maximum packet size from 512K to 1024K for client.
- Almost all of the function code is now tested in the internal test suite.
- ALTER TABLE now returns warnings from field conversions.
- Port changed to 3306 (got it reserved from ISI).
- Added a fix for Visual FoxBase so that any schema name from a table specification is automatically removed.
- New function ASCII() .
- Removed function BETWEEN(a,b,c) . Use the standard ANSI syntax instead: expr BETWEEN expr AND expr .
- MySQL no longer has to use an extra temporary table when sorting on functions or SUM() functions.
- Fixed bug that you couldn't use tbl_name.field_name in UPDATE .
- Fixed SELECT DISTINCT when using 'hidden group'. For example:

```
mysql> SELECT DISTINCT MOD(some_field,10) FROM test
   ->        GROUP BY some_field;
```

Note: some_field is normally in the SELECT part. ANSI SQL should require it.

## 13.6.35 Changes in release 3.21.0

- New reserved words used: INTERVAL , EXPLAIN , READ , WRITE , BINARY .
- Added ODBC function CHAR(num,...) .
- New operator IN . This uses a binary search to find a match.
- New command LOCK TABLES tbl_name [AS alias] {READ|WRITE} ...
- Added --log-update option to mysqld , to get a log suitable for incremental updates.
- New command EXPLAIN SELECT ... to get information about how the optimiser will do the join.
- For easier client code, the client should no longer use FIELD_TYPE_TINY_BLOB , FIELD_TYPE_MEDIUM_BLOB , FIELD_TYPE_LONG_BLOB or FIELD_TYPE_VAR_STRING (as previously returned by mysql_list_fields ). You should instead only use FIELD_TYPE_BLOB or FIELD_TYPE_STRING . If you want exact types, you should use the command SHOW FIELDS .
- Added varbinary syntax: 0x@# which can be used as a string (default) or a number.
- FIELD_TYPE_CHAR is renamed to FIELD_TYPE_TINY .
- Changed all fields to C++ classes.
- Removed FORM struct.
- Fields with DEFAULT values no longer need to be NOT NULL .
- New field types:
- Now all function calculation is done with double or long long . This will provide the full 64-bit range with bit functions and fix some conversions that previously could result in precision losses. One should avoid using unsigned long long columns with full 64-bit range (numbers bigger than 9223372036854775807) because calculations are done with signed long long .
- ORDER BY will now put NULL field values first. GROUP BY will also work with NULL values.
- Full WHERE with expressions.
- New range optimiser that can resolve ranges when some keypart prefix is constant. Example:

```
mysql> SELECT * FROM tbl_name
   ->          WHERE key_part_1="customer"
   ->          AND key_part_2>=10 AND key_part_2<=10;
```

# 13.7 Changes in release 3.20.x

Version 3.20 is quite old now, and should be avoided if possible. This information is kept here for historical purposes only.

Changes from 3.20.18 to 3.20.32b are not documented here because the 3.21 release branched here. And the relevant changes are also documented as changes to the 3.21 version.

## 13.7.1 Changes in release 3.20.18

- Added -p# (remove # directories from path) to isamlog . All files are written with a relative path from the database directory Now mysqld shouldn't crash on shutdown when using the --log-isam option.
- New mysqlperl version. It is now compatible with msqlperl-0.63 .
- New DBD module available at http://www.mysql.com/Downloads/Contrib/ site.
- Added group function STD() (standard deviation).
- The mysqld server is now compiled by default without debugging information. This will make the daemon smaller and faster.
- Now one usually only has to specify the --basedir option to mysqld . All other paths are relative in a normal installation.
- BLOB columns sometimes contained garbage when used with a SELECT on more than one table and ORDER BY .

- Fixed that calculations that are not in GROUP BY work as expected (ANSI SQL extension). Example:

```
mysql> SELECT id,id+1 FROM table GROUP BY id;
```

- The test of using MYSQL_PWD was reversed. Now MYSQL_PWD is enabled as default in the default release.
- Fixed conversion bug which caused mysqld to core dump with Arithmetic error on SPARC-386.
- Added --unbuffered option to mysql , for new mysqlaccess .
- When using overlapping (unnecessary) keys and join over many tables, the optimiser could get confused and return 0 records.

## 13.7.2 Changes in release 3.20.17

- You can now use BLOB columns and the functions IS NULL and IS NOT NULL in the WHERE clause.
- All communication packets and row buffers are now allocated dynamically on demand. The default value of max_allowed_packet is now 64K for the server and 512K for the client. This is mainly used to catch incorrect packets that could trash all memory. The server limit may be changed when it is started.
- Changed stack usage to use less memory.
- Changed safe_mysqld to check for running daemon.
- The ELT() function is renamed to FIELD() . The new ELT() function returns a value based on an index: FIELD() is the inverse of ELT() Example: ELT(2,"A","B","C") returns "B" . FIELD("B","A","B","C") returns 2 .
- COUNT(field) , where field could have a NULL value, now works.
- A couple of bugs fixed in SELECT ... GROUP BY .
- Fixed memory overrun bug in WHERE with many unoptimisable brace levels.
- Fixed some small bugs in the grant code.
- If hostname isn't found by get_hostname , only the IP is checked. Previously, you got Access denied .
- Inserts of timestamps with values didn't always work.
- INSERT INTO ... SELECT ... WHERE could give the error Duplicated field .
- Added some tests to safe_mysqld to make it ``safer''.
- LIKE was case-sensitive in some places and case-insensitive in others. Now LIKE is always case-insensitive.
- mysql.cc : Allow '#' anywhere on the line.
- New command SET SQL_SELECT_LIMIT=# . See the FAQ for more details.
- New version of the mysqlaccess script.
- Change FROM_DAYS() and WEEKDAY() to also take a full TIMESTAMP or DATETIME as argument. Before they only took a number of type YYYYMMDD or YYMMDD .
- Added new function UNIX_TIMESTAMP(timestamp_column) .

## 13.7.3 Changes in release 3.20.16

- More changes in MIT-pthreads to get them safer. Fixed also some link bugs at least in SunOS.
- Changed mysqld to work around a bug in MIT-pthreads. This makes multiple small SELECT operations 20 times faster. Now lock_test.pl should work.
- Added mysql_FetchHash(handle) to mysqlperl .
- The mysqlbug script is now distributed built to allow for reporting bugs that appear during the build with it.
- Changed libmysql.c to prefer getpwuid() instead of cuserid() .
- Fixed bug in SELECT optimiser when using many tables with the same column used as key to different tables.
- Added new latin2 and Russian KOI8 character tables.
- Added support for a dummy GRANT command to satisfy Powerbuilder.

## 13.7.4 Changes in release 3.20.15

- Fixed fatal bug packets out of order when using MIT-pthreads.
- Removed possible loop when a thread waits for command from client and fcntl() fails. Thanks to Mike Bretz for finding this bug.
- Changed alarm loop in mysqld.cc because shutdown didn't always succeed in Linux.
- Removed use of termbits from mysql.cc . This conflicted with glibc 2.0.
- Fixed some syntax errors for at least BSD and Linux.
- Fixed bug when doing a SELECT as superuser without a database.
- Fixed bug when doing SELECT with group calculation to outfile.

## 13.7.5 Changes in release 3.20.14

- If one gives -p or --password option to mysql without an argument, the user is solicited for the password from the tty.
- Added default password from MYSQL_PWD (by Elmar Haneke).
- Added command kill to mysqladmin to kill a specific MySQL thread.
- Sometimes when doing a reconnect on a down connection this succeeded first on second try.
- Fixed adding an AUTO_INCREMENT key with ALTER_TABLE .
- AVG() gave too small value on some SELECT s with GROUP BY and ORDER BY .
- Added new DATETIME type (by Giovanni Maruzzelli maruzz@matrice.it ).
- Fixed that defining DONT_USE_DEFAULT_FIELDS works.
- Changed to use a thread to handle alarms instead of signals on Solaris to avoid race conditions.
- Fixed default length of signed numbers. (George Harvey georgeh@pinacl.co.uk .)
- Allow anything for CREATE INDEX .
- Add prezeros when packing numbers to DATE , TIME and TIMESTAMP .
- Fixed a bug in OR of multiple tables (gave empty set).
- Added many patches to MIT-pthreads. This fixes at least one lookup bug.

## 13.7.6 Changes in release 3.20.13

- Added ANSI SQL94 DATE and TIME types.
- Fixed bug in SELECT with AND - OR levels.
- Added support for Slovenian characters. The Contrib directory contains source and instructions for adding other character sets.
- Fixed bug with LIMIT and ORDER BY .

- Allow ORDER BY and GROUP BY on items that aren't in the SELECT list. (Thanks to Wim Bonis bonis@kiss.de , for pointing this out.)
- Allow setting of timestamp values in INSERT .
- Fixed bug with SELECT ... WHERE ... = NULL .
- Added changes for glibc 2.0. To get glibc to work, you should add the gibc−2.0−sigwait−patch before compiling glibc .
- Fixed bug in ALTER TABLE when changing a NOT NULL field to allow NULL values.
- Added some ANSI92 synonyms as field types to CREATE TABLE . CREATE TABLE now allows FLOAT(4) and FLOAT(8) to mean FLOAT and DOUBLE .
- New utility program mysqlaccess by Yves.Carlier@rug.ac.be . This program shows the access rights for a specific user and the grant rows that determine this grant.
- Added WHERE const op field (by bonis@kiss.de ).

## 13.7.7 Changes in release 3.20.11

- When using SELECT ... INTO OUTFILE , all temporary tables are ISAM instead of HEAP to allow big dumps.
- Changed date functions to be string functions. This fixed some ``funny'' side effects when sorting on dates.
- Extended ALTER TABLE according to SQL92.
- Some minor compatibility changes.
- Added −−port and −−socket options to all utility programs and mysqld .
- Fixed MIT−pthreads readdir_r() . Now mysqladmin create database and mysqladmin drop database should work.
- Changed MIT−pthreads to use our tempnam() . This should fix the ``sort aborted'' bug.
- Added sync of records count in sql_update . This fixed slow updates on first connection. (Thanks to Vaclav Bittner for the test.)

## 13.7.8 Changes in release 3.20.10

- New insert type: INSERT INTO ... SELECT ...
- MEDIUMBLOB fixed.
- Fixed bug in ALTER TABLE and BLOB s.
- SELECT ... INTO OUTFILE now creates the file in the current database directory.
- DROP TABLE now can take a list of tables.
- Oracle synonym DESCRIBE ( DESC ).
- Changes to make_binary_distribution .
- Added some comments to installation instructions about configure 's C++ link test.
- Added −−without−perl option to configure .
- Lots of small portability changes.

## 13.7.9 Changes in release 3.20.9

- ALTER TABLE didn't copy null bit. As a result, fields that were allowed to have NULL values were always NULL .
- CREATE didn't take numbers as DEFAULT .
- Some compatibility changes for SunOS.
- Removed config.cache from old distribution.

## 13.7.10 Changes in release 3.20.8

- Fixed bug with ALTER TABLE and multi−part keys.

## 13.7.11 Changes in release 3.20.7

- New commands: ALTER TABLE , SELECT ... INTO OUTFILE and LOAD DATA INFILE .
- New function: NOW() .
- Added new field File_priv to mysql/user table.
- New script add_file_priv which adds the new field File_priv to the user table. This script must be executed if you want to use the new SELECT ... INTO and LOAD DATA INFILE ... commands with a version of MySQL earlier than 3.20.7.
- Fixed bug in locking code, which made lock_test.pl test fail.
- New files NEW and BUGS .
- Changed select_test.c and insert_test.c to include config.h .
- Added status command to mysqladmin for short logging.
- Increased maximum number of keys to 16 and maximum number of key parts to 15.
- Use of sub keys. A key may now be a prefix of a string field.
- Added −k option to mysqlshow , to get key information for a table.
- Added long options to mysqldump .

## 13.7.12 Changes in release 3.20.6

- Portable to more systems because of MIT−pthreads, which will be used automatically if configure cannot find a −lpthreads library.
- Added GNU−style long options to almost all programs. Test with program −−help .
- Some shared library support for Linux.
- The FAQ is now in .texi format and is available in .html , .txt and .ps formats.
- Added new SQL function RAND([init]) .
- Changed sql_lex to handle \0 unquoted, but the client can't send the query through the C API, because it takes a str pointer. You must use mysql_real_query() to send the query.
- Added API function mysql_get_client_info() .
- mysqld now uses the N_MAX_KEY_LENGTH from nisam.h as the maximum allowable key length.

- The following now works:

```
mysql> SELECT filter_nr,filter_nr FROM filter ORDER BY filter_nr;
```

  Previously, this resulted in the error: Column: 'filter_nr' in order clause is ambiguous .
- mysql now outputs '\0' , '\t' , '\n' and '\\' when encountering ASCII 0, tab, newline or '\' while writing tab-separated output. This is to allow printing of binary data in a portable format. To get the old behaviour, use -r (or --raw ).
- Added german error messages (60 of 80 error messages translated).
- Added new API function mysql_fetch_lengths(MYSQL_RES *) , which returns an array of column lengths (of type uint ).
- Fixed bug with IS NULL in WHERE clause.
- Changed the optimiser a little to get better results when searching on a key part.
- Added SELECT option STRAIGHT_JOIN to tell the optimiser that it should join tables in the given order.
- Added support for comments starting with '--' in mysql.cc (Postgres syntax).
- You can have SELECT expressions and table columns in a SELECT which are not used in the group part. This makes it efficient to implement lookups. The column that is used should be a constant for each group because the value is calculated only once for the first row that is found for a group.

```
mysql> SELECT id,lookup.text,SUM(*) FROM test,lookup
   ->        WHERE test.id=lookup.id GROUP BY id;
```

- Fixed bug in SUM(function) (could cause a core dump).
- Changed AUTO_INCREMENT placement in the SQL query:

```
INSERT INTO table (auto_field) VALUES (0);
```

  inserted 0, but it should insert an AUTO_INCREMENT value.
- mysqlshow.c : Added number of records in table. Had to change the client code a little to fix this.
- mysql now allows doubled '' or "" within strings for embedded ' or " .
- New math functions: EXP() , LOG() , SQRT() , ROUND() , CEILING() .

## 13.7.13 Changes in release 3.20.3

- The configure source now compiles a thread-free client library -lmysqlclient . This is the only library that needs to be linked with client applications. When using the binary releases, you must link with -lmysql -lmysys -ldbug -lmystrings as before.
- New readline library from bash-2.0 .
- LOTS of small changes to configure and makefiles (and related source).
- It should now be possible to compile in another directory using VPATH . Tested with GNU Make 3.75.
- safe_mysqld and mysql.server changed to be more compatible between the source and the binary releases.
- LIMIT now takes one or two numeric arguments. If one argument is given, it indicates the maximum number of rows in a result. If two arguments are given, the first argument indicates the offset of the first row to return, the second is the maximum number of rows. With this it's easy to do a poor man's next page/previous page WWW application.
- Changed name of SQL function FIELDS() to ELT() . Changed SQL function INTERVALL() to INTERVAL() .
- Made SHOW COLUMNS a synonym for SHOW FIELDS . Added compatibility syntax FRIEND KEY to CREATE TABLE . In MySQL, this creates a non-unique key on the given columns.
- Added CREATE INDEX and DROP INDEX as compatibility functions. In MySQL, CREATE INDEX only checks if the index exists and issues an error if it doesn't exist. DROP INDEX always succeeds.
- mysqladmin.c : added client version to version information.
- Fixed core dump bug in sql_acl (core on new connection).
- Removed host , user and db tables from database test in the distribution.
- FIELD_TYPE_CHAR can now be signed (-128 to 127) or unsigned (0 to 255) Previously, it was always unsigned.
- Bug fixes in CONCAT() and WEEKDAY() .
- Changed a lot of source to get mysqld to be compiled with SunPro compiler.
- SQL functions must now have a '(' immediately after the function name (no intervening space). For example, 'USER(' is regarded as beginning a function call, and 'USER (' is regarded as an identifier USER followed by a '(' , not as a function call.

## 13.7.14 Changes in release 3.20.0

- The source distribution is done with configure and Automake. It will make porting much easier. The readline library is included in the distribution.
- Separate client compilation: the client code should be very easy to compile on systems which don't have threads.
- The old Perl interface code is automatically compiled and installed. Automatic compiling of DBD will follow when the new DBD code is ported.
- Dynamic language support: mysqld can now be started with Swedish or English (default) error messages.
- New functions: INSERT() , RTRIM() , LTRIM() and FORMAT() .
- mysqldump now works correctly for all field types (even AUTO_INCREMENT ). The format for SHOW FIELDS FROM tbl_name is changed so the Type column contains information suitable for CREATE TABLE . In previous releases, some CREATE TABLE information had to be patched when re-creating tables.
- Some parser bugs from 3.19.5 ( BLOB and TIMESTAMP ) are corrected. TIMESTAMP now returns different date information depending on its create length.
- Changed parser to allow a database, table or field name to start with a number or '_' .
- All old C code from Unireg changed to C++ and cleaned up. This makes the daemon a little smaller and easier to understand.
- A lot of small bug fixes done.
- New INSTALL files (not final version) and some information regarding porting.

# 13.8 Changes in release 3.19.x

Version 3.19 is quite old now, and should be avoided if possible. This information is kept here for historical purposes only.

## 13.8.1 Changes in release 3.19.5

- Some new functions, some more optimisation on joins.
- Should now compile clean on Linux (2.0.x).
- Added functions DATABASE() , USER() , POW() , LOG10() (needed for ODBC).
- In a WHERE with an ORDER BY on fields from only one table, the table is now preferred as first table in a multi-join.
- HAVING and IS NULL or IS NOT NULL now works.
- A group on one column and a sort on a group function ( SUM() , AVG() ...) didn't work together. Fixed.
- mysqldump : Didn't send password to server.

## 13.8.2 Changes in release 3.19.4

- Fixed horrible locking bug when inserting in one thread and reading in another thread.
- Fixed one-off decimal bug. 1.00 was output as 1.0.
- Added attribute 'Locked' to process list as info if a query is locked by another query.
- Fixed full magic timestamp. Timestamp length may now be 14, 12, 10, 8, 6, 4 or 2 bytes.
- Sort on some numeric functions could sort incorrectly on last number.
- IF(arg,syntax_error,syntax_error) crashed.
- Added functions CEILING() , ROUND() , EXP() , LOG() and SQRT() .
- Enhanced BETWEEN to handle strings.

## 13.8.3 Changes in release 3.19.3

- Fixed SELECT with grouping on BLOB columns not to return incorrect BLOB info. Grouping, sorting and distinct on BLOB columns will not yet work as expected (probably it will group/sort by the first 7 characters in the BLOB ). Grouping on formulas with a fixed string size (use MID() on a BLOB ) should work.
- When doing a full join (no direct keys) on multiple tables with BLOB fields, the BLOB was garbage on output.
- Fixed DISTINCT with calculated columns.

# 14 Port vers d'autres systèmes

Cet appendice vous aidera à porter MySQL vers un autre système d'exploitation. Vérifiez d'abord la liste des systèmes supportés avant toute chose. Operating Systems Supported by MySQL . Si vous avez crée un nouveau port de MySQL, merci de nous en avertir pour que nous puissions le lister ici et sur notre site web ( http://www.mysql.com/ ), pour le recommander aux autres utilisateurs.

Note : Si vous créez un nouveau port de MySQL, vous êtes libre de le copier et le distribuer sous la licence GPL, mais cela ne signifie pas que vous êtes détenteur de droits sur MySQL.

Une librairie de threads Posix qui fonctionne est requise pour le serveur. Pour Solaris 2.5 nous utilisons Sun PThreads (le support natif des threads de la version 2.4 et plus ancienne n'est pas assez bonne) et sur Linux nous utilisons LinuxThreads de Xavier Leroy, Xavier.Leroy@inria.fr .

La partie la plus difficile du port vers une nouvelle variante Unix ne bénéficiant pas d'un bon support natif des threads est probablement le port de MIT−pthreads. Voyez `mit−pthreads/README` et Programming POSIX Threads ( http://www.humanfactor.com/pthreads/ ).

La distribution MySQL inclut une version patchée des Pthreads de Provenzano de MIT (voyez la page web des Pthreads MIT http://www.mit.edu:8001/people/proven/pthreads.html ). Cela peut être utilisé pour certains systèmes d'exploitation qui n'ont pas les threads POSIX.

Il est aussi possible d'utiliser un autre package de threads au niveau utilisateur nommé FSU Pthreads (Voir la page de FSU Pthreads ). Cette implémentation est utilisée pour le port vers SCO.

Consultez les programmes thr_lock.c et thr_alarm.c dans le dossier mysys pour quelques tests/exemples de ces problèmes.

Le serveur et le client on besoin d'un compilateur C++ fonctionnel (nous utilisons `gcc` et avons essayé SPARCworks). Un autre compilateur connu maintenant pour fonctionner est Irix `cc` .

Pour ne compiler que le client, utilisez ./configure −−without−server .

Il n'y a actuellement aucun support pour ne compiler que le serveur, et il n'est pas prévu d'en ajouter un à moins que quelqu'un n'ait une bonne raison de le faire.

Si vous voulez ou avez besoin de changer un fichier Makefile ou le script de configuration vous aurez besoin d'avoir Automake et Autoconf. Nous avons utilisé les distributions automake−1.2 et autoconf−2.12 .

Toutes les étapes dont vous avez besoin pour reconstruire le tout à partir des fichiers de base.

```
 /bin/rm */.deps/*.P
/bin/rm −f config.cache
aclocal
autoheader
aclocal
automake
autoconf
./configure −−with−debug=full −−prefix='votre dossier installation'

 # les fichiers make générés plus haut ont besoin de GNU make 3.75 ou plus récent.
# (appelé gmake ci−dessous)
gmake clean all install init−db
```

Si vous rencontrez des problèmes avec un nouveau port, vois devrez faire du débogage de MySQL ! Déboguer un serveur MySQL .

**Note** : avant de commencer à déboguer `mysqld` , faites d'abord fonctionner les programmes de tests `mysys/thr_alarm` et `mysys/thr_lock` . Cela assurera que votre installation des threads a une chance de fonctionner !

## 14.1 Déboguer un serveur MySQL

Si vous utilisez des fonctionnalités qui ont été ajoutées il y a peu de temps à MySQL, vous pouvez essayer de démarrer `mysqld` avec `−−skip−new` (qui désactivera toutes les fonctionnalités nouvelles, qui sont potentiellement non−stables) ou avec `−−safe−mode` qui désactive un tas d'optimisations qui

pourraient poser problèmes. Que faire si MySQL crashe constamment .

Si mysqld ne veut pas démarrer, vous devez vérifier que vous n'avez pas de fichiers my.cnf qui interfèrent avec votre configuration ! Vous pouvez vérifier les arguments de votre my.cnf avec mysqld --print-defaults et éviter de les utiliser en démarrant avec mysqld --no-defaults ... .

Si `mysqld` se met à trop consommer de mémoire ou de CPU ou si il se bloque, vous pouvez utiliser `mysqladmin processlist status` pour trouver si quelqu'un utilise une requête qui prend trop de temps à s'exécuter. C'est un bonne idée d'exécuter `mysqladmin -i10 processlist status` dans un terminal si vous avez des problèmes de performances ou des problèmes à la connexion de nouveaux clients.

La commande mysqladmin debug écrira des informations à propos des verrous en cours d'utilisation, de la mémoire utilisée et des requêtes dans le fichier de log de MySQL. Cela peut vous aider à résoudre certains problèmes. Cette commande fournit aussi des informations utiles même si vous n'avez pas compilé MySQL pour le débogage !

Si le problème vient du fait que certaines tables sont de plus en plus lentes vous devez essayer de les optimiser en utilisant `OPTIMIZE TABLE` ou `myisamchk` . Administration de la base . Vous devez aussi vérifier les requêtes qui prennent trop de temps avec la commande `EXPLAIN` .

Vous devriez aussi consulter les sections spécifiques aux systèmes d'exploitations dans ce manuel pour les problèmes pouvant être uniques à votre environnement. Remarques spécifiques aux systèmes d'exploitation .

## 14.1.1 Compiler MYSQL pour le débogage

Si vous avez un problème spécifique, vous pouvez toujours essayer de déboguer MySQL. Pour ce faire, vous devez configurer MySQL avec l'option `--with-debug` ou `--with-debug=full` . Vous pouvez vérifier si MySQL est déjà compilé avec le support du débogage en faisant ceci : `mysqld --help` . Si l'attribut `--debug` est listé avec les options, cela veut dire que le support du débogage est activé. Dans ce cas, `mysqladmin ver` liste aussi la version de `mysqld` en tant que `mysql ... --debug` .

Si vous utilisez gcc ou egcs, la ligne de configuration recommandée est :

```
CC=gcc CFLAGS="-O2" CXX=gcc CXXFLAGS="-O2 -felide-constructors \
  -fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
  --with-debug --with-extra-charsets=complex
```

Cela évitera les problèmes avec la librairie libstdc++ et les exceptions C++ (plusieurs compilateurs ont des problèmes avec les exceptions C++ dans le code threadé) et compilera une version MySQL avec le support de tous les jeux de caractères.

Si vous suspectez un dépassement de la mémoire, vous pouvez configurer MySQL avec `--with-debug=full` , qui installera un vérificateur d'allocation mémoire ( `SAFEMALLOC` ). Fonctionner avec `SAFEMALLOC` est cependant un peu ralentissant, et donc, si vous rencontrez des problèmes de performances, vous devez démarrer `mysqld` avec l'option `--skip-safemalloc` . Cela désactivera les vérifications de dépassements de mémoire pour chaque appel à `malloc` ou `free` .

Si mysqld ne crashe plus lorsque vous le compilez avec --with-debug , vous avez probablement trouvé un bogue du compilateur ou un bogue de temporisation dans MySQL. Dans ce cas, vous pouvez essayer d'ajouter -g aux variables CFLAGS et CXXFLAGS vues plus haut et ne pas utiliser --with-debug . Si mysqld crashe maintenant, vous pouvez vous y attacher avec gdb ou utiliser gdb sur le fichier noyau pour trouver ce qui est arrivé.

Lorsque vous configurez MySQL pour le support du débogage, vous activez automatiquement un tas de fonctions de tests supplémentaires qui se chargent de surveiller le bon fonctionnement de `mysqld` . Si elles trouvent quelque chose d'inattendu (``unexpected''), une entrée sera écrite dans `stderr` , que `safe_mysqld` redirige vers le log d'erreurs ! Cela signifie aussi que si vous avez quelques problèmes inattendus avec MySQL et que vous utilisez une distribution des sources, la première chose à faire est de configurer MySQL avec le support du débogage ! (la seconde, bien sûr, étant d'envoyer un mail à mysql@lists.mysql.com et demander de l'aide. Merci d'utiliser le script `mysqlbug` pour tous les rapport de bogues ou questions concernant la version de MySQL que vous utilisez !

Dans la distribution Windows de MySQL, mysqld.exe est par défaut compilé avec le support des fichiers de traçage.

## 14.1.2 Créer un fichier de traçage

Si le serveur `mysqld` ne démarre pas ou que vous pouvez le crasher facilement, vous pouvez essayer de créer un fichier de traçage pour trouver le problème.

Pour ce faire, vous devez avoir un `mysqld` qui est compilé pour le débogage. Vous pouvez le vérifier en exécutant `mysqld -V` . Si le numéro de version se termine par `-debug` , il est compilé avec le support des fichiers de traçage.

Démarrez le serveur mysqld avec un journal de suivi dans /tmp/mysqld.trace (ou C:\mysqld.trace sous Windows) :

```
mysqld --debug
```

Sous Windows vous devez aussi utiliser l'option --standalone pour ne pas démarrer mysqld en tant que service :

Dans une console DOS entrez :

```
mysqld --debug --standalone
```

Après cela, vous pouvez utiliser l'outil en ligne de commande mysql.exe dans une seconde fenêtre pour reproduire le problème. Vous pouvez couper le serveur avec la commande mysqladmin shutdown .

Notez que le fichier de traçage deviendra **très gros** ! si vous voulez obtenir un fichier plus petit, utilisez ce qui suit par exemple :

```
mysqld --debug=d,info,error,query,general,where:O,/tmp/mysqld.trace
```

qui n'écrit que les informations les plus intéressantes dans `/tmp/mysqld.trace` .

Si vous créez un rapport de bogue, merci de n'envoyer que les lignes du fichier de traçage où le problème se concrétise à la liste de diffusion appropriée ! Si vous n'arrivez pas à trouver le bon endroit dans le fichier, vous pouvez envoyer la totalité du fichier ainsi que le rapport de bogue via FTP à ftp://support.mysql.com/pub/mysql/secret/ pour qu'un développeur MySQL y jette un coup d'oeil.

Le fichier de traçage est généré avec le package **DBUG** de Fred Fish. Le package DBUG .

## 14.1.3 Déboguer mysqld sous gdb

Sur la plupart des systèmes, vous pouvez démarrer mysqld à partir de gdb pour obtenir plus d'informations si mysqld crashe.

Avec quelques anciennes versions de `gdb` sous Linux vous devez exécuter `run --one-thread` si vous voulez être capables de déboguer les threads de `mysqld` threads. Dans ce cas, vous ne pouvez n'avoir qu'un thread actif à la fois. Nous vous recommandons de mettre à jour gdb à la version 5.1 dès que possible vu que le débogage des threads fonctionne mieux avec cette version !

Lors de l'utilisation de mysqld sous gdb, vous devez désactiver le traçage de la pile avec --skip-stack-trace pour pouvoir trouver les erreurs de segmentations avec gdb.

Il est très difficile de déboguer MySQL sous `gdb` si vous effectuez plusieurs nouvelles connexions tout le temps vu que `gdb` ne libère pas la mémoire occupée par les anciens threads. Vous pouvez contourner ce problème en démarrant `mysqld` avec `-O thread_cache_size= 'max_connections +1'` . Dans la plupart des cas, le simple fait d'utiliser `-O thread_cache_size=5'` vous aidera beaucoup !

Si vous voulez obtenir un core dump sur Linux si mysqld se termine avec un signal SIGSEGV, vous pouvez démarrer mysqld avec l'option --core-file . Ce fichier noyau peut être utilisé pour effectuer des traçages qui peuvent vous aider à trouver pourquoi mysqld s'est terminée :

```
shell> gdb mysqld core
gdb>   backtrace full
gdb>   exit
```

Que faire si MySQL crashe constamment .Si vous utilisez gdb 4.17.x ou plus récent sous Linux, vous devez installer un fichier .gdb , avec les informations suivantes, dans votre répertoire courant :

```
set print sevenbit off
handle SIGUSR1 nostop noprint
```

```
handle SIGUSR2 nostop noprint
handle SIGWAITING nostop noprint
handle SIGLWP nostop noprint
handle SIGPIPE nostop
handle SIGALRM nostop
handle SIGHUP nostop
handle SIGTERM nostop noprint
```

Si vous rencontrez des problèmes lors du débogage des threads avec gdb, vous devez obtenir la version 5.x de gdb et essayer cela à la place. La nouvelle version de gdb a une meilleur gestion des threads !Voilà un exemple de comment déboguer mysqld :

```
 shell> gdb /usr/local/libexec/mysqld
gdb> run
...
backtrace full # A faire lorsque mysqld crashe
```

Incluez la sortie suivante dans un mail généré avec mysqlbug et envoyez le à mysql@lists.mysql.com .Si mysqld ne répond plus, vous pouvez utiliser des outils système tel que strace ou /usr/proc/bin/pstack pour savoir où mysqld s'est bloqué.

```
 strace /tmp/log libexec/mysqld
```

Si vous utilisez l'interface DBI de Perl, vous pouvez activer le débogage en utilisant la méthode trace ou en définissant la variable d'environnement DBI_TRACE . L'interface DBI .

## 14.1.4 Utilisation d'un traçage de pile mémoire

Sur quelques systèmes d'exploitation, le log d'erreurs contiendra un fichier de pile mémoire si `mysqld` se termine soudainement. Vous pouvez utiliser ceci pour trouver où (et peut−être pourquoi) `mysqld` s'est terminé. Le log d'erreurs . Pour obtenir un traçage de la pile, vous ne devez pas compiler `mysqld` avec l'option `−fomit−frame−pointer` de gcc. Compiler MYSQL pour le débogage .

Si le fichier d'erreurs contient quelque chose qui ressemble à ce qui suit :

```
 mysqld got signal 11;
The manual section 'Debugging a MySQL server' tells you how to use a
stack trace and/or the core file to produce a readable backtrace that may
help in finding out why mysqld died
Attemping backtrace. You can use the following information to find out
where mysqld died.  If you see no messages after this, something went
terribly wrong
stack range sanity check, ok, backtrace follows
0x40077552
0x81281a0
0x8128f47
0x8127be0
0x8127995
0x8104947
0x80ff28f
0x810131b
0x80ee4bc
0x80c3c91
0x80c6b43
0x80c1fd9
0x80c1686
```

vous pouvez trouver où s'est terminé mysqld en exécutant ce qui suit :

- Copiez les nombres précédents dans un fichier, mysqld.stack par exemple.
- créez un fichier symbolique pour le serveur mysqld :

  ```
  nm −n libexec/mysqld > /tmp/mysqld.sym
  ```

  Notez que beaucoup de distributions binaires MySQL fournissent le fichier précédent, nommé mysqld.sym.gz . Dans ce cas, décompressez le en faisant :

  ```
  gunzip < bin/mysqld.sym.gz > /tmp/mysqld.sym
  ```

- Exécutez resolve_stack_dump −s /tmp/mysqld.sym −n mysqld.stack .Cela affichera l'endroit où mysqld a planté. Si cela ne vous aide pas à trouver pourquoi mysqld a crashé, vous devez créer un rapport de bogue et y inclure le résultat de la commande précédente.Notez toutefois que dans la plupart de cas le fait de n'avoir que le traçage de la pile ne nous aidera pas à trouver d'où vient le problème. Pour être capable de trouver le bogue, ou fournir une parade, nous aurons besoin dans la plupart des cas, nous aurons besoin de connaître la requête qui a fait planter mysqld et une batterie de tests pour que nous puissions reproduire le problème ! Comment rapporter des bogues ou problèmes .

## 14.1.5 Utilisation des fichiers de log pour trouver d'où viennent les erreurs de mysqld

Notez qu'avant de démarrer `mysqld` avec `−−log` vous devez vérifier toutes vos tables avec `myisamchk` . Administration de la base .

Si `mysqld` se termine ou se bloque, vous devez démarrer `mysqld` avec `−−log` . Lorsque `mysqld` se termine à nouveau, vous pouvez examiner la fin de votre fichier de log pour trouver les requêtes qui ont terminé `mysqld` .

Si vous utilisez --log sans spécifier un nom de fichier, le log est enregistré dans le dossier des bases de données en tant que 'hostname'.log. Dans la plupart des cas, c'est la dernière requête dans le fichier de log qui a terminé mysqld , mais si possible, vérifiez le en redémarrant mysqld et exécutant à nouveau la requête en question à partir du client en ligne de commande mysql . Si elle fonctionne, vous devez aussi tester les autres requêtes complexes qui n'ont pas abouties.

Vous pouvez aussi utiliser la commande `EXPLAIN` sur toutes vos requêtes `SELECT` qui prennent beaucoup de temps à s'exécuter pour être sûrs que `mysqld` utilise les index convenablement. Syntaxe de `EXPLAIN` (Obtenir des informations `SELECT` ) .

Vous pouvez trouver les requêtes qui prennent trop de temps à s'exécuter en démarrant mysqld avec --log-slow-queries . Le log des requêtes lentes .

Si vous trouvez le texte `mysqld restarted` dans le log d'erreurs (normalement nommé `hostname.err` ) vous avez probablement trouvé une requête qui fait planter `mysqld` . Si tel est le cas, vous devez vérifier toutes vos tables avec `myisamchk` ( Administration de la base ), et tester les requêtes dans les fichiers de log MySQL pour voir si elles ne fonctionnent toujours pas. si vous trouvez une requête de ce genre, essayez d'abord de mettre à jour votre version de MySQL en prenant la version la plus récente. Si cela ne vous aide pas et que vous ne pouvez trouver d'aide dans les archives des mails de `mysql` , vous devez reporter ce bogue à mysql@lists.mysql.com . Des liens vers les archives de mails sont disponibles en ligne à l'adresse suivante : http://lists.mysql.com/ .

Si vous avez démarré mysqld avec myisam-recover , MySQL vérifiera et essayera automatiquement de réparer les tables MyISAM si elles sont marquées comme 'not closed properly' ou 'crashed'. Si cela arrive, MySQL ajoutera une entrée dans le fichier hostname.err 'Warning: Checking table ...' qui sera suivie de Warning: Repairing table si la table devait être réparée. si vous obtenez beaucoup de ces erreurs, sans que mysqld n'ai crashé juste avant, alors quelque chose ne va pas, et une enquête plus approfondie est nécessaire. Options en ligne de commande de mysqld .

Ce n'est bien sûr pas de bon augure si `mysqld` a crashé, mais dans ce cas, il ne faut pas s'attarder sur les messages `Checking table...` mais plutôt essayer de savoir pourquoi `mysqld` a crashé.

### 14.1.6 Faire une batterie de tests lorsque vous faites face à un problème de table corrompue

Si vos tables sont corrompues ou que `mysqld` échoue toujours avec quelques commandes de mises à jour, vous pouvez tester si le bogue est reproduisable en effectuant ce qui suit :

- Coupez le démon MySQL (avec mysqladmin shutdown ).
- Créez une copie de vos tables (pour prévoir le cas très improbable ou la réparation tournerait mal).
- Vérifiez toutes les tables avec myisamchk -s base/*.MYI . Réparez toute table corrompue avec myisamchk -r base/table.MYI .
- Créez une seconde copie des tables.
- Effacez (ou déplacez) tout les vieux fichiers de log du répertoire de données de MySQL si vous avez besoin de plus d'espace.
- Démarrez mysqld avec --log-bin . Le log binaire des mises à jour . Si vous voulez trouver une requête qui fait crasher mysqld , vous devez utiliser --log --log-bin .
- Lorsque vous obtenez une table crashée, stoppez le serveur mysqld .
- Restaurez les sauvegardes.
- Redémarrez le serveur mysqld **sans** --log-bin
- Re-exécutez les commandes avec mysqlbinlog update-log-file | mysql . Le log des mises à jour est est sauvegardé dans le dossier des données de MySQL avec le nom hostname-bin.# .
- Si les tables sont à nouveau corrompues ou que vous pouvez faire échouer mysqld avec la commande précédente, vous avez trouvé un bogue reproductible qui devrait être facile à corriger ! Envoyez les tables et le log binaire via FTP à ftp://support.mysql.com/pub/mysql/secret/ et envoyez un mail à bugs@lists.mysql.com ou (si vous êtes client du support) à support@mysql.com à propos du problème et l'équipe MySQL le corrigera le plus vite possible.

Vous pouvez aussi utiliser le script mysql_find_rows pour n'exécuter que quelques requêtes de mises à jour si vous voulez mieux cerner le problème.

## 14.2 Débogage un client MySQL

Pour pouvoir déboguer un client MySQL avec le package de débogage intégré, vous devez configurer MySQL avec `--with-debug` ou `--with-debug=full` . Les options habituelles de `configure` .

Avant de mettre en marche un client, vous devez définir la variable d'environnement MYSQL_DEBUG :

```
 shell> MYSQL_DEBUG=d:t:O,/tmp/client.trace
shell> export MYSQL_DEBUG
```

Cela fait générer au client un fichier de traçage dans /tmp/client.trace .Si vous avez un problème avec votre propre code client, vous devez essayer de vous connecter au serveur et exécuter vos requêtes en utilisant un client qui fonctionne. Faites le en utilisant mysql

en mode débogage (en supposant que vous ayez compilé MySQL avec le support du débogage) :

```
shell> mysql --debug=d:t:O,/tmp/client.trace
```

Il vous fournira des informations utiles si vous voulez envoyer un rapport de bogue. Comment rapporter des bogues ou problèmes .Si votre client crashe au niveau d'un code qui vous parait 'légal', vous devez vérifier que votre fichier mysql.h inclu correspond à votre librairie mysql. Une erreur très courante est d'utiliser un vieux fichier mysql.h d'une anciene installation avec la nouvelle librairie MySQL.

# 14.3 Le package DBUG

Le serveur MySQL et la plupart des clients MySQL sont compilés avec le package DBUG écrit, à l'origine, par Fred Fish. Lorsque MySQL est compilé avec le support du débogage, ce package permet d'obtenir des fichiers de traçage de ce que le programme débogue. Créer des fichiers de traçage .

Le package de débogage est utilisé en invoquant le programme avec l'option --debug="..." ou -#... .

La plupart des programmes MySQL ont une chaîne de débogage par défaut qui sera utilisée si vous ne spécifiez aucune option à `--debug` . Le fichier de traçage par défaut est usuellement `/tmp/nomprogramme.trace` sur Unix et `\nomprogramme.trace` sur Windows.

La chaîne de caractères de controle du débogage est une séquence de champs séparés par des deux-points (:) comme celle qui suit :

```
<champ_1>:<champ_2>:...:<champ_N>
```

Chaque champ consiste d'un caractère attribut suivi d'une liste de modificateurs, commençant optionnellement par une virgule, séparés par des virgules :

```
flag[,modificateur,modificateur,...,modificateur]
```

Les caractères attributs actuellement reconnus sont :

| Attribut | Description |
|---|---|
| d | Active les sorties des macros DBUG_<N> pour l'état courant. Peut être suivi d'une liste de mots clefs, ce qui sélectionne la sortie seuelemnt pour les macros DBUG contenant ces mots. Une liste de mots clefs vides implique les sorties de toutes les macros. |
| D | Attendre après chaque ligne résultante du débogueur. L'arguement est le nombre de dixièmme de secondes à attendre, sujet aux capabilités de la machine. Et donc, -#D,20 est une attente de deux secondes. |
| f | Limiter le débogage et/ou le traçage aux fonctions citées. Notez qu'une liste nulle désactivera toutes les fonctions. Les attributs appropriés "d" ou "t" doivent quand même être donnés, cet attribut ne limite que leurs actions si ils sont activés. |
| F | Identifie le nom du fichier source pour chaque ligne de débogage ou de traçage affichée. |
| i | Identifie le processus avec son identifiant pour chaque ligne de débogage ou de traçage affichée. |
| g | Active le profiling. Crée un fichier nommé 'dbugmon.out' contenant des informations qui peuvent être utilisées pour profiler le programme. Peut être suivi d'une liste de mots clefs qui sélectionnent le profiling uniquement pour les fonctions présentes dans ctte liste. Une liste nulle implique que toutes les fonctions sont considérées. |
| L | Identifie le numéro de ligne du fichier source pour chaque ligne renvoyée par le traçage ou le débogage. |
| n | Imprime le niveau de profondeur de la fonction en cours d'exécution pour la sortie du traçage ou du débogage. |
| N | Numérote chaque ligne de la sortie du débogage. |
| o | Redirige le flux de sortie du débogueur vers le fichier spécifié. Par défaut, c'est stderr. |
| O | Comme o mais le fichier est vraiement écrit entre chaque ajout de contenu. Lorsque le besoin en est, le fichier est fermé puis réouvert entre chaque écriture. |
| p | Limite les actions du débogueur aux processus spécifiés. Un processus peut être identifié avec la macro DBUG_PROCESS et correspondre à un processus dans la liste pour que le débogage ait lieu. |
| P | Affiche le nom du processus courant pour chaque ligne de sortie de débogage ou de traçage. |
| r | Lors du passage à un nouvel état, ne pas hériter le niveau de profondeur de l'état de la fonction précédente. Utile lorsque l'affichage commence à la marge gauche. |
| S | Exécute la fonction _sanity(_file_,_line_) sur chaque fonction déboguée jusqu'à ce que la valeur de retour de _sanity() diffère de 0. (La plupart du temps utilisée avec safemalloc opur trouver les pertes de mémoire) |
| t | Active le traçage des appels/sorties des fonctions. Peut être suivi d'une liste (ne contenant qu'un seul modificateur) donnant un maximum numérique du traçage, au delà duquel aucune sortie ne sera affichée pour les macros de débogage ou de traçage. Par défaut, c'est une option de temps de compilation. |

Quelques exemples de chaînes de controle de débogage pouvant être utilisée en ligne de commande dans le shell ("-#" est typiquement utilisé pour introduire une chaîne de controle à un programme d'application) sont :

```
-#d:t
-#d:f,main,subr1:F:L:t,20
-#d,input,output,files:n
-#d:t:i:O,\\mysqld.trace
```

En MySQL, les balises communes affichées (avec l'option d ) sont : enter , exit , error , warning , info et loop .

# 14.4 Méthodes de verrouillage

Actuellement, MySQL ne supporte que le verrouillage de table pour les tables ISAM / MyISAM et HEAP , le verrouillage de page pour les tables BDB et le verrouillage de ligne pour InnoDB . Comment MySQL verrouille les tables . Avec les tables MyISAM , vous pouvez mélanger librement les INSERT et SELECT sans poser de verrous ( Versioning ).

Depuis la version 3.23.33, vous pouvez analyser l'utilisation des verrous sur vos tables avec les variables d'environnement `Table_locks_waited` et `Table_locks_immediate` .

Pour décider si vous voulez utiliser un type de table avec verrouillage de ligne, vous devez commencer par étudier ce que votre application fait, et quel est le schéma d'utilisation des sélections et modifications.

Avantages du verrouillage de ligne :

- Moins de conflits de lignes, lorsque les mêmes lignes sont utilisées par différents threads.
- Moins de modifications pour les annulations ( ROLLBACK )
- Rend possible le verrouillage d'une ligne pour une longue durée.

Inconvénients du verrouillage de ligne :

- Prend plus de mémoire que les verrous de page ou de table.
- Est plus lent que les verrous de page ou de table, lorsqu'il est utilisé sur une grand partie de la table, car il faut alors poser plusieurs verrous.
- Est vraiment bien pire que les autres verrous si vous utilisez souvent la requête GROUP BY sur la majeure partie des données, ou si vous avez à scanner toute la table.
- Avec des verrous de plus haut niveau, vous pouvez aussi supporter des verrous d'autres types, pour optimiser l'application, car le coût de l'administration est moindre que pour le verrouillage de ligne.

Les verrous de tables sont supérieurs aux verrous de page ou de ligne dans les cas suivants :

- Les lectures.
- Les lectures et les modifications sur des clés strictes : c'est le cas si une modification ou un effacement de ligne peut être lu en une seule opération dans l'index.

```
 UPDATE table_name SET column=value WHERE unique_key#
DELETE FROM table_name WHERE unique_key=#
```

- SELECT combiné avec INSERT (et quelques UPDATE et DELETE rares).
- De nombreux scans / GROUP BY sur toute la table, sans aucune écriture.

Autres possibiltés alternatives au verrouillage de ligne ou de page :

Le versionnage (comme celui que nous utilisons pour les insertions simultanées avec MySQL), où vous pouvez avoir un thread qui écrit et de nombreux autres qui lisent. Cela signifie que les bases ou tables supportent différentes vues des données, suivants le moment d'accès aux données. D'autres noms pour cette techniques sont `time travel` , `copy on write` ou `copy on demand` .

La copy on demand (copie sur demande) est dans de nombreuses situations bien meilleure que le verrouillage de page ou de ligne. Le pire reste l'utilisation de mémoire, qui est bien plus forte qu'avec les verrous normaux.

Au lieu d'utiliser le verrouillage de ligne, vous pouvez utiliser des verrous au niveau de l'application (comme les get_lock/release_lock de MySQL). Cela ne fonctionne qu'avec les applications bien élevées.

Dans de nombreux cas, vous pouvez faire une estimation éclairée du type de verrouillage qui est le meilleur pour votre application, mais la plus part du temps, il est très difficile de dire si un type de verrouillage est plus adapté que l'autre. Tout dépend de votre application, et des différentes parties qui réclament des verrous.

Voici quelques conseils sur le verrouillage de MySQL :

La plupart des sites web fond de nombreuses lecture, très peu d'effacements, des modifications avec des clés, et des insertions dans des tables spécifiques. La base MySQL est très bien optimisée pour cela.

Les utilisateurs simultanés ne sont pas un problème si ils ne mélangent pas les insertions et les modifications, qui déclenchent l'analyse de nombreuses lignes dans la même table.

Si vous mélangez les insertions et les effacements sur les mêmes tables, alors utilisez INSERT DELAYED pour améliorer la situation.

Vous pouvez aussi utiliser la commande `LOCK TABLES` pour accélérer les opérations (de nombreuses modifications dans le même verrou et bien plus rapide que de le faire sans le verrou). Répartir les opérations sur plusieurs tables peut aussi aider.

Si vous avez des problèmes avec les verrous de tables de MySQL, vous pourriez les résoudre en convertissant les tables au format InnoDB ou BDB . Tables InnoDB . Tables BDB ou BerkeleyDB .

Le chapitre d'optimisation du manuel couvre de nombreux aspects du paramétrage de l'application. Autres conseils d'optimisation .

# 14.5 Commaires à propos des threads RTS

J'ai essayé d'utiliser le package de threads RTS avec MySQL mais je suis resté bloqué au niveau des problèmes suivants :

Ils utilisent une veille version avec beaucoup d'appels POSIX et il est vraiment difficile de créer une couche d'abstraction pour toutes les fonctions. Je pense qu'il serait plus facile de changer la librairie des threads pour qu'elle suive les nouvelles spécifications POSIX.

Quelques couches d'abstractions sont déjà écrites. Voyez mysys/my_pthread.c pour plus d'informations.

Au minimum, ce qui suit devra être changé :

pthread_get_specific doit utiliser un seul argument. sigwait doit prendre deux arguments. Beaucoup de fonctions (du moins pthread_cond_wait , pthread_cond_timedwait ) doivent retourner le code erreur lorsqu'elles en rencontrent. Elle retournent à présent −1 et définissent errno .

Un autre problème est que les threads au niveau utilisateurs utilisent les signaux ALRM et que ceux−ci fait échouer beaucoup de fonctions ( read , write , open ...). MySQL devrait faire une autre tentative à chaque interruption mais cela n'est pas facile à vérifier.

Le plus gros problème non−résolu est le suivant :

Pour avoir des alertes au niveau des threads, j'ai changé mysys/thr_alarm.c pour avoir une attente entre les alarmes avec pthread_cond_timedwait() , mais cela échoue avec une erreur EINTR . J'ai essayé de déboguer la librairie des threads pour voir d'où cela venait, mais je n'ai pu trouver aucune solution simple.

Si quelqu'un veut utiliser MySQL avec les threads RTS je suggère ce qui suit :

- Changez les fonctions que MySQL utilise à partir de la librairie des threads en POSIX. Cela ne devrait pas vous prendre beaucoup de temps.
- Compilez toutes les librairies avec −DHAVE_rts_threads .
- Compilez thr_alarm .
- S'il y a de petites différences dans l'implémentation, elles peuvent être corrigées en changeant les fichiers my_pthread.h et my_pthread.c .
- Exécutez thr_alarm . S'il tourne sans aucun message du type ``warning'', ``error'' ou ``aborted'', vous êtes sur le bon chemin. Voici une bonne exécution se déroulant sur Solaris :

```
 Main thread: 1
Thread 0 (5) started
Thread: 5  Waiting
process_alarm
Thread 1 (6) started
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 1 (1) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 2 (2) sec
Thread: 6  Simulation of no alarm needed
Thread: 6  Slept for 0 (3) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 4 (4) sec
Thread: 6  Waiting
process_alarm
thread_alarm
Thread: 5  Slept for 10 (10) sec
Thread: 5  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 5 (5) sec
Thread: 6  Waiting
process_alarm
process_alarm

 ...
thread_alarm
Thread: 5  Slept for 0 (1) sec
end
```

# 14.6 Différences entre les différents packages de threads

MySQL est très dépendant du package de threads utilisé. Ce qui fait que lors du choix d'une bonne plate-forme pour MySQL, le package des threads est très important.

## Il y a au moins trois types de packages de threads :

- Threads utilisateurs dans un processus unique. Le changement de threads est géré avec des alarmes et la librairie des threads gère les fonctions non-utilisables avec les threads par des verrouillages. Les opérations de lectures, d'écritures et de sélections sont usuellement gérées avec une sélection spécifique aux threads qui passe à un autre thread si celui en cours d'exécution attend des données. Si les packages des threads utilisateurs sont intégrés dans les librairies standards (threads FreeBSD et BSDI) le package de thread nécessite moins de ressources que les packages de thread qui doivent mapper toutes les appels non-sûrs (MIT-pthreads, FSU Pthreads et RTS threads). Avec quelques environnements, (SCO par exemple), tous les appels système sont sûrs pour les threads, ce qui fait que la liaison peut se faire très facilement (Pthreads FSU sur SCO). Mauvais côté : Tous les appels mappés prennent un peu de temps, et il est assez difficile de pouvoir gérer toutes les situations. Il y a aussi souvent des appels système qui ne sont pas gérés par le package de threads (comme MIT-pthreads et les sockets). La gestion des threads n'est pas toujours optimale.
- Threads utilisateurs dans des processus séparés. Les changements de threads sont effectués par le noyau et toutes les données sont partagées entre les threads. Le package de thread gère les appels threads standards pour permettre le partage entre les threads. LinuxThreads utilise cette méthode. Mauvais côté : Beaucoup de processus. La création des threads est lente. Si un thread s'interrompt, les autres restent en suspens et vous devez tous les terminer avant de redémarrer. Le changement de thread est d'une certaine façon consommateur de ressources.
- Threads noyau. Le changement de threads est géré par la librairie de threads ou le noyau est très rapide. Tout est fait en un seul processus, mais sur certains systèmes, ps peut montrer plusieurs threads. Si un thread échoue, tout le processus échoue. La plupart des appels système sont bons pour les threads et ne devraient avoir besoin que d'une petite perte de performances. Solaris, HP-UX, AIX et OSF/1 ont des threads noyau.

Avec quelques systèmes, les threads du noyau sont gérés en intégrant les threads niveau utilisateur dans les librairies du système. Dans ces cas, le changement de thread ne peut être fait qu'avec la librairie des threads et le noyau n'est pas vraiment ``attentif aux threads''.

# 15 Variables d'environnement

Voici une liste de toutes les variables d'environnement qui sont directement ou indirectement utilisées par MySQL. La plupart peuvent être retrouvées dans d'autres parties du manuel.

Notez que toute option en ligne de commande prendra la précédence par rapport aux valeurs spécifiées dans les fichiers de configurations et les variables d'environnement, et que les valeurs dans les fichiers de configurations prennent la précédence sur les valeurs des variables d'environnement.

Dans beaucoup de cas, il est préférable d'utiliser un fichier de configuration plutôt que les variables d'environnement pour modifier le comportement de MySQL. Fichier d'options my.cnf .

| Variable | Description |
|---|---|
| CCX | A passer au compilateur C++ lors de l'exécution de configure. |
| CC | A passer au compilateur C lors de l'exécution de configure. |
| CFLAGS | A passer au compilateur C lors de l'exécution de configure. |
| CXXFLAGS | A passer au compilateur C++ lors de l'exécution de configure. |
| DBI_USER | L'utilisateur par défaut pour Perl DBI. |
| DBI_TRACE | Utilisée lors du traçage de Perl DBI. |
| HOME | Le chemin par défaut vers les fichiers d'hostorique de mysql est $HOME/.mysql_history . |
| LD_RUN_PATH | Utilisé pour spécifier où se trouve libmysqlclient.so . |
| MYSQL_DEBUG | Options de débogage/traçage lors de la phase de débogage. |
| MYSQL_HISTFILE | Le chemin vers le fichier d'historique de mysql . |
| MYSQL_HOST | Nom d'hôte par défaut utilisé par le client en ligne de commande mysql . |
| MYSQL_PS1 | Invite de commande à utiliser dans le client en ligne de commande mysql . L'outil de ligne de commande. |
| MYSQL_PWD | Le mot de passe par défaut lors de la connexion à mysqld . Notez que cette utilisation n'est pas sécurisée. |
| MYSQL_TCP_PORT | Le port TCP/IP par défaut. |
| MYSQL_UNIX_PORT | La socket par défaut; utilisée pour les connexion à localhost . |
| PATH | Utilisé par le shell pour trouver les programmes MySQL. |
| TMPDIR | Le dossier ou les tables et fichiers tomporaires sont crées. |
| TZ | Cela devrait être configuré à votre fuseau horaire local. Les problèmes de fuseau horaire. |
| UMASK_DIR | Le masque (mask) de création des dossiers. Notez que cette valeur est soumise à un ET logique par rapport UMASK ! |
| UMASK | Le masque (mask) de création des fichiers utilisateurs lors de la création de fichiers. |
| USER | L'utilisateur par défaut de Windows à utiliser lors de la connexion à mysqld . |

# 16 Expressions régulières MySQL

Une expression régulière (regex) est la meilleure méthode pour spécifier une recherche complexe.

MySQL utilise l'implémentation de Henry Spencer des expressions régulières qui tend à être conforme à POSIX 1003.2. MySQL en utilise la version étendue.

Ceci est une référence simplifiée qui n'aborde pas les détails. Pour avoir plus d'ionformations détaillées, regardez la page de manuel regex(7) de Henry Spencer qui est inclue dans la distribution des sources. Contributeurs .

Une expression régulière décrit un jeu de chaînes de caractères. La plus simple est celle qui ne comporte pas de caractères spéciaux. Par exemple, la regexp bonjour trouvera bonjour et rien d'autre.

Les expression régulières non−trivialles utilisent des constructions spéciales pour pouvoir trouver plus d'une chaîne. Par exemple, la regexp bonjour|monde trouve la chaîne bonjour ou la chaîne monde .

Voici un exemple encore plus complexe : la regexp B[an]*s trouve l'une des chaînes suivantes Bananas , Baaaaas , Bs , et n'importe quelle autre chaîne commençant par un B , se terminant par un s , et contenant n'importe quel nombre de a et de n au milieu. Une expression régulière peut utiliser l'un des caractères spéciaux ou constructions suivants :

*^*

Correspond au début de la chaîne.

```
 mysql> SELECT "fo\nfo" REGEXP "^fo$";           -> 0
mysql> SELECT "fofo" REGEXP "^fo";              -> 1
```

*$*

Correspond à la fin de la chaîne.

```
 mysql> SELECT "fo\no" REGEXP "^fo\no$";         -> 1
mysql> SELECT "fo\no" REGEXP "^fo$";            -> 0
```

*.*

N'importe quel caractère (nouvelle ligne inclus).

```
 mysql> SELECT "fofo" REGEXP "^f.*";             -> 1
mysql> SELECT "fo\nfo" REGEXP "^f.*";           -> 1
```

*a**

Correspond à toute séquence de zéro ou plus caractères a .

```
 mysql> SELECT "Ban" REGEXP "^Ba*n";             -> 1
mysql> SELECT "Baaan" REGEXP "^Ba*n";           -> 1
mysql> SELECT "Bn" REGEXP "^Ba*n";              -> 1
```

*a+*

Correspond à toute séquence de un ou plus caractères a .

```
 mysql> SELECT "Ban" REGEXP "^Ba+n";             -> 1
mysql> SELECT "Bn" REGEXP "^Ba+n";              -> 0
```

*a?*

Correspond à zéro ou un caractère a .

```
 mysql> SELECT "Bn" REGEXP "^Ba?n";              -> 1
mysql> SELECT "Ban" REGEXP "^Ba?n";             -> 1
mysql> SELECT "Baan" REGEXP "^Ba?n";            -> 0
```

*de|abc*

Correspond aux séquences de de ou de abc .

```
 mysql> SELECT "pi" REGEXP "pi|apa";             -> 1
mysql> SELECT "axe" REGEXP "pi|apa";            -> 0
mysql> SELECT "apa" REGEXP "pi|apa";            -> 1
mysql> SELECT "apa" REGEXP "^(pi|apa)$";        -> 1
mysql> SELECT "pi" REGEXP "^(pi|apa)$";         -> 1
mysql> SELECT "pix" REGEXP "^(pi|apa)$";        -> 0
```

*(abc)**

Correspond à zéro ou plus séquences de abc .

```
 mysql> SELECT "pi" REGEXP "^(pi)*$";            -> 1
mysql> SELECT "pip" REGEXP "^(pi)*$";           -> 0
mysql> SELECT "pipi" REGEXP "^(pi)*$";          -> 1
```

*{1}*
*{2,3}*

Voici une façon plus générale d'écrire les regexps qui correspondent à plusieurs occurences du dernier atome.

*a**

Peut être écrit a{0,} .

*a+*

Peut être écrit a{1,} .

*a?*

Peut être écrit a{0,1} .

Pour être plus précis, un atome suivi d'une accolade contenant un entier i et pas de virgule trouve une séquence de exactement i atomes.

Un atome suivi d'une accolade contenant un entier i et une virgule trouve une séquence de i ou plus atomes.

Un atome suivi d'une accolade contenant deux entiers i et j séparés d'une virgule trouve les séquences de i à j (inclusif) atomes.

Les deux arguements doivent être compris entre 0 et RE_DUP_MAX (par défaut 255), inclusif. S'il y a deux arguements, le second doit être supérieur ou égal au premier.

*[a-dX]*
*[^a-dX]*

Trouve n'importe quel caractère qui est (ou n'est pas, si ^ est utilisé) a , b , c , d ou X . Pour inclure le caractère litéral ] , il doit suivre immédiatement le crochet ouvrant [ . Pour inclure le caractère litéral - , il doit être écrit en premier ou en dernier. Ce qui fait que [0-9] correspond à n'importe quel chiffre. Chaque caractère qui n'a pas de signification spéciale à l'intérieur une paire de [] ne joue pas de rôle spécial et ne correspond qu'à lui même.

```
 mysql> SELECT "aXbc" REGEXP "[a-dXYZ]";         -> 1
mysql> SELECT "aXbc" REGEXP "^[a-dXYZ]$";       -> 0
mysql> SELECT "aXbc" REGEXP "^[a-dXYZ]+$";      -> 1
mysql> SELECT "aXbc" REGEXP "^[^a-dXYZ]+$";     -> 0
mysql> SELECT "gheis" REGEXP "^[^a-dXYZ]+$";    -> 1
mysql> SELECT "gheisa" REGEXP "^[^a-dXYZ]+$";   -> 0
```

*[[.charactères.]]*

La séquence de caractères de cet élément d'assemblage. La séquence est un élément de la liste contenue entre les crochets. Une telle expression contenant un élément d'assemblage multi-caractères peut ainsi trouver plus d'un caractère. Par exemple, si la séquence d'assemblage inclut un élément ch , alors l'expression regulière [[.ch.]]*c trouve les cinq premiers caractères de chchcc .

*[=character_class=]*

Une classe d'équivalence, remplaçant les séquences de caractères de tous les éléments de l'assemblage équivalents à celui-ci, lui même inclut.Par exemple, si o et (+) sont membres d'une classe d'équivalence, alors [[=o=]] , [[=(+)=]] , et [o(+)] sont tous des synonymes. Une classe d'équivalence ne doit pas être un point final d'intervalle.

*[:classe_de_caractères:]*

Dans une expression entre crochets, le nom d'une classe de caractères entourée de [: et :] remplace la liste de tous les caractères appartenant à cette classe. Les noms des classes de caractères sont :

| Nom | Nom | Nom |
|---|---|---|
| alnum | digit | punct |
| alpha | graph | space |
| blank | lower | upper |
| cntrl | print | xdigit |

Voilà les classes de caractères définies dans la page de manuel ctype(3) . Une locale peut en fournir d'autres. Une classe de caractère ne doit pas être utilisée en tant que point final d'intervalle.

```
 mysql> SELECT "justalnums" REGEXP "[[:alnum:]]+";       -> 1
mysql> SELECT "!!" REGEXP "[[:alnum:]]+";               -> 0
```

*[[:<:]]*
*[[:>:]]*

Ceux là trouvent la chaîne nulle qui précède et suit chaque mot. Un mot est défini comme étant une séquence de caractères qui n'est ni suivi ni précédée d'un caractère de mot. Un caractère de mot est un caractère alnum (défini par ctype(3) ) ou un tiret bas ( _ ).

```
 mysql> SELECT "a word a" REGEXP "[[:<:]]word[[:>:]]";      -> 1
mysql> SELECT "a xword a" REGEXP "[[:<:]]word[[:>:]]";     -> 0
```

```
mysql> SELECT "weeknights" REGEXP "^(wee|week)(knights|nights)$"; -> 1
```

# 17 Licence Publique Générale GNU

### Notice d'accompagnement de la traduction non officielle à conserver dans toute reproduction de cette traduction

This is an unofficial translation of the GNU General Public License into french. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL−−only the original English text of the GNU GPL does that. However, we hope that this translation will help french speakers understand the GNU GPL better.

Ceci est une traduction non officielle de la GNU General Public License en français. Elle n'a pas été publiée par la Free Software Foundation, et ne détermine pas les termes de distribution pour les logiciels qui utilisent la GNU GPL−−seul le texte anglais original de la GNU GPL en a le droit. Cependant, nous espérons que cette traduction aidera les francophones à mieux comprendre la GPL.

Cette traduction est sous Copyright 2001 APRIL ( http://www.april.org ). La version la plus à jour de ce document est disponible sur http://www.april.org/gnu/gpl_french.html

Il est permis à tout le monde de reproduire et distribuer des copies conformes de cette traduction, mais aucune modification ne doit y être apportée, et la présente notice doit être préservée.

Nous autorisons la FSF à apporter toute modification qu'elle jugera nécessaire pour rendre la traduction plus claire.

Version 2, June 1991


Copyright © 1989, 1991 Free Software Foundation, Inc.
59 Temple Place − Suite 330, Boston, MA 02111−1307, USA


Il est permis à tout le monde de reproduire et distribuer des copies conformes de ce document de licence, mais aucune modification ne doit y être apportée.

**Préambule** Les licences relatives à la plupart des logiciels sont destinées à supprimer votre liberté de les partager et de les modifier. Par contraste, la licence publique générale GNU General Public License veut garantir votre liberté de partager et de modifier les logiciels libres, pour qu'ils soient vraiment libres pour tous leurs utilisateurs. La présente licence publique générale s'applique à la plupart des logiciels de la Free Software Foundation, ainsi qu'à tout autre programme dont les auteurs s'engagent à l'utiliser. (Certains autres logiciels sont couverts par la Licence Publique Générale pour Bibliothèques GNU à la place). Vous pouvez aussi l'appliquer à vos programmes.

Quand nous parlons de logiciels libres, nous parlons de liberté, non de gratuité. Nos licences publiques générales veulent vous garantir que vous avez toute liberté de distribuer des copies des logiciels libres (et de facturer ce service, si vous le souhaitez), que vous recevez les codes sources ou pouvez les obtenir si vous le souhaitez, que vous pouvez modifier les logiciels ou en utiliser des éléments dans de nouveaux programmes libres, et que vous savez que vous pouvez le faire. Pour protéger vos droits, nous devons apporter des restrictions, qui vont interdire à quiconque de vous dénier ces droits, ou de vous demander de vous en désister. Ces restrictions se traduisent par certaines responsabilités pour ce qui vous concerne, si vous distribuez des copies de logiciels, ou si vous les modifiez. Par exemple, si vous distribuez des copies d'un tel programme, gratuitement ou contre une rémunération, vous devez transférer aux destinataires tous les droits dont vous disposez. Vous devez vous garantir qu'eux−mêmes, par ailleurs, reçoivent ou peuvent recevoir le code source. Et vous devez leur montrer les présentes dispositions, de façon qu'ils connaissent leurs droits. Nous protégeons vos droits en deux étapes : (1) Nous assurons le droit d'auteur (copyright) du logiciel, et (2) Nous vous proposons cette licence, qui vous donne l'autorisation légale de dupliquer, distribuer et/ou modifier le logiciel.

De même, pour la protection de chacun des auteurs, et pour notre propre protection, nous souhaitons nous assurer que tout le monde comprenne qu'il n'y a aucune garantie portant sur ce logiciel libre. Si le logiciel est modifié par quelqu'un d'autre puis transmis à des tiers, nous souhaitons que les destinataires sachent que ce qu'ils possèdent n'est pas l'original, de façon que tous problèmes introduits par d'autres ne se traduisent pas par une répercussion négative sur la réputation de l'auteur original.

Enfin, tout programme libre est en permanence menacé par des brevets de logiciels. Nous souhaitons éviter le danger que des sous−distributeurs d'un programme libre obtiennent à titre individuel des licences de brevets, avec comme conséquence qu'ils ont un droit de propriété sur le programme. Pour éviter cette situation, nous avons fait tout ce qui est nécessaire pour que tous brevets doivent faire l'objet d'une concession de licence qui en permette l'utilisation libre par quiconque, ou bien qu'il ne soit pas concédé du tout.

Nous présentons ci−dessous les clauses et dispositions concernant la duplication, la distribution et

la modification.

- Le présent contrat de licence s'applique à tout programme ou autre ouvrage contenant un avis, apposé par le détenteur du droit de propriété, disant qu'il peut être distribué au titre des dispositions de la présente Licence Publique Générale. Ci−après, le "Programme" désigne l'un quelconque de ces programmes ou ouvrages, et un "ouvrage fondé sur le programme" désigne soit le programme, soit un ouvrage qui en dérive au titre de la loi sur le droit d'auteur ; plus précisément, il s'agira d'un ouvrage contenant le programme ou une version de ce dernier, soit mot à mot, soit avec des modifications et/ou traduit en une autre langue (ci−après, le terme "modification" englobe, sans aucune limitation, les traductions qui en sont faites). Chaque titulaire de licence sera appelé "concessionnaire". Les activités autres que la duplication, la distribution et la modification ne sont pas couvertes par la présente licence ; elles n'entrent pas dans le cadre de cette dernière. L'exécution du programme n'est soumise à aucune restriction, et les résultats du programme ne sont couverts que si son contenu constitue un ouvrage fondé sur le programme (indépendamment du fait qu'il a été réalisé par exécution du programme). La véracité de ce qui précède dépend de ce que fait le programme.
- Le concessionnaire peut dupliquer et distribuer des copies mot à mot du code source du programme tel qu'il les reçoit, et ce sur un support quelconque, du moment qu'il appose, d'une manière parfaitement visible et appropriée, sur chaque exemplaire, un avis approprié de droits d'auteur (Copyright) et de renonciation à garantie ; qu'il maintient intacts tous les avis qui se rapportent à la présente licence et à l'absence de toute garantie ; et qu'il transmet à tout destinataire du programme un exemplaire de la présente licence en même temps que le programme.Le concessionnaire peut facturer l'acte physique de transfert d'un exemplaire, et il peut, à sa discrétion, proposer en échange d'une rémunération une protection en garantie.
- Le concessionnaire peut modifier son ou ses exemplaires du programme ou de toute portion de ce dernier, en formant ainsi un ouvrage fondé sur le programme, et dupliquer et distribuer ces modifications ou cet ouvrage selon les dispositions de la section 1 ci−dessus, du moment que le concessionnaire satisfait aussi à toutes ces conditions :
  - Le concessionnaire doit faire en sorte que les fichiers modifiés portent un avis, parfaitement visible, disant que le concessionnaire a modifié les fichiers, avec la date de tout changement.
  - Le concessionnaire doit faire en sorte que tout ouvrage qu'il distribue ou publie, et qui, en totalité ou en partie, contient le programme ou une partie quelconque de ce dernier ou en dérive, soit concédé en bloc, à titre gracieux, à tous tiers au titre des dispositions de la présente licence.
  - Si le programme modifié lit normalement des instructions interactives lors de son exécution, le concessionnaire doit, quand il commence l'exécution du programme pour une telle utilisation interactive de la manière la plus usuelle, faire en sorte que ce programme imprime ou affiche une annonce, comprenant un avis approprié de droits d'auteur, et un avis selon lequel il n'y a aucune garantie (ou autrement, que le concessionnaire fournit une garantie), et que les utilisateurs peuvent redistribuer le programme au titre de ces dispositions, et disant à l'utilisateur comment visualiser une copie de cette licence (exception : si le programme par lui−même est interactif mais n'imprime normalement pas une telle annonce, l'ouvrage du concessionnaire se fondant sur le programme n'a pas besoin d'imprimer une annonce).

Les exigences ci−dessus s'appliquent à l'ouvrage modifié pris en bloc. Si des sections identifiables de cet ouvrage ne dérivent pas du programme et peuvent être considérées raisonnablement comme représentant des ouvrages indépendants et distincts par eux−mêmes, alors la présente licence, et ses dispositions, ne s'appliquent pas à ces sections quand le concessionnaire les distribue sous forme d'ouvrages distincts. Mais quand le concessionnaire distribue ces mêmes sections en tant qu'élément d'un tout qui représente un ouvrage se fondant sur le programme, la distribution de ce tout doit se faire conformément aux dispositions de la présente licence, dont les autorisations, portant sur d'autres concessionnaires, s'étendent à la totalité dont il est question, et ainsi à chacune de ces parties, indépendamment de celui qu'il a écrite.

Ainsi, cette section n'a pas pour but de revendiquer des droits ou de contester vos droits sur un ouvrage entièrement écrit par le concessionnaire ; bien plus, l'intention est d'exercer le droit de surveiller la distribution d'ouvrages dérivée ou collective se fondant sur le programme.

De plus, un simple assemblage d'un autre ouvrage ne se fondant pas sur le programme, avec le programme (ou avec un ouvrage se fondant sur le programme) sur un volume d'un support de stockage ou distribution, ne fait pas entrer l'autre ouvrage dans le cadre de la présente licence.

- Le concessionnaire peut dupliquer et distribuer le programme (ou un ouvrage se fondant sur ce dernier, au titre de la Section 2), en code objet ou sous une forme exécutable, au titre des dispositions des Sections 1 et 2 ci−dessus, du moment que le concessionnaire effectue aussi l'une des opérations suivantes :
  - Lui joindre le code source complet correspondant, exploitable par une machine, code qui doit être distribué au titre des Sections 1 et 2 ci−dessus sur un support couramment utilisé pour l'échange de logiciels ; ou bien
  - Lui joindre une offre écrite, dont la validité se prolonge pendant au moins 3 ans, de transmettre à un tiers quelconque, pour un montant non supérieur au coût pour le concessionnaire, de réalisation physique de la distribution de la source, un exemplaire complet, exploitable par une machine, du code source correspondant, qui devra être distribué au titre des dispositions des Sections 1 et 2 ci−dessus sur un support couramment utilisé pour l'échange des logiciels ; ou bien
  - Lui joindre les informations que le concessionnaire a reçues, pour proposer une distribution du code source correspondant (cette variante n'est autorisée que pour la distribution non commerciale, et seulement si le concessionnaire a reçu le programme sous forme exécutable ou sous forme d'un code objet, avec une telle offre, conformément à l'alinéa b) ci−dessus).

Le code source d'un ouvrage représente la forme préférée de l'ouvrage pour y effectuer des modifications. Pour un ouvrage exécutable, le code source complet représente la totalité du code source pour tous les modules qu'il contient, plus tous fichiers de définitions d'interface associés, plus les informations en code machine pour commander la compilation et l'installation du programme exécutable. Cependant, à titre d'exceptions spéciales, le code source distribué n'a pas besoin de comprendre quoi que ce soit qui est normalement distribué (sous forme source ou sous forme binaire) avec les composants principaux (compilateur, noyau de système d'exploitation, etc.) du système d'exploitation sur lequel est exécuté le programme exécutable, à moins que le composant, par lui−même, soit joint au programme exécutable.

Si la distribution de l'exécutable ou du code objet est réalisée de telle sorte qu'elle offre d'accéder à une copie à partir d'un lieu désigné, alors le fait d'offrir un accès équivalent à la duplication du code source à partir de ce même lieu s'entend comme distribution du code source, même si des tiers ne sont pas contraints de dupliquer la source en même temps que le code objet.

- Le concessionnaire ne peut dupliquer, modifier, concéder en sous-licence ou distribuer le programme, sauf si cela est expressément prévu par les dispositions de la présente licence. Toute tentative pour autrement dupliquer, modifier, concéder en sous-licence ou distribuer le programme est répétée nulle, et met automatiquement fin aux droits du concessionnaire au titre de la présente licence. Cependant, les parties qui ont reçu des copies, ou des droits, de la part du concessionnaire au titre de la présente licence, ne verront pas expirer leur contrat de licence, tant que ces parties agissent d'une manière parfaitement conforme.
- Il n'est pas exigé du concessionnaire qu'il accepte la présente licence, car il ne l'a pas signée. Cependant, rien d'autre n'octroie au concessionnaire l'autorisation de modifier ou de distribuer le programme ou ses ouvrages dérivés. Ces actions sont interdites par la loi si le concessionnaire n'accepte pas la présente licence. En conséquence, par le fait de modifier ou de distribuer le programme (ou un ouvrage quelconque se fondant sur le programme), le concessionnaire indique qu'il accepte la présente licence, et qu'il a la volonté de se conformer à toutes les clauses et dispositions concernant la duplication, la distribution ou la modification du programme ou d'ouvrages se fondant sur ce dernier.
- Chaque fois que le concessionnaire redistribue le programme (ou tout ouvrage se fondant sur le programme), le destinataire reçoit automatiquement une licence de l'émetteur initial de la licence, pour dupliquer, distribuer ou modifier le programme, sous réserve des présentes clauses et dispositions. Le concessionnaire ne peut imposer aucune restriction plus poussée sur l'exercice, par le destinataire, des droits octroyés au titre des présentes. Le concessionnaire n'a pas pour responsabilité d'exiger que des tiers se conforment à la présente licence.
- Si, en conséquence une décision de justice ou une allégation d'infraction au droit des brevets, ou pour toute autre raison (qui n'est pas limitée à des problèmes de propriétés industrielles), des conditions sont imposées au concessionnaire (par autorité de justice, par convention ou autrement), qui entrent en contradiction avec les dispositions de la présente licence, elles n'exemptent pas le concessionnaire de respecter les dispositions de la présente licence. Si le concessionnaire ne peut procéder à la distribution de façon à satisfaire simultanément à ces obligations au titre de la présente licence et à toutes autres obligations pertinentes, alors, en conséquence de ce qui précède, le concessionnaire peut ne pas procéder du tout à la distribution du programme. Par exemple, si une licence de brevet ne permettait pas une redistribution du programme, sans redevances, par tous ceux qui reçoivent des copies directement ou indirectement par l'intermédiaire du concessionnaire, alors le seul moyen par lequel le concessionnaire pourrait satisfaire tant à cette licence de brevet qu'à la présente licence, consisterait à s'abstenir complètement de distribuer le programme.Si une partie quelconque de cette section est considérée comme nulle ou non exécutoire dans certaines circonstances particulières, le reste de cette section est réputé s'appliquer, et la section dans son ensemble est considérée comme s'appliquant dans les autres circonstances.

La présente section n'a pas pour objet de pousser le concessionnaire à enfreindre tous brevets ou autres revendications à droit de propriété, ou encore à contester la validité de une ou plusieurs quelconques de ces revendications ; la présente section a pour objet unique de protéger l'intégrité du système de distribution des logiciels libres, système qui est mis en oeuvre par les pratiques liées aux licences publiques. De nombreuses personnes ont apporté une forte contribution à la gamme étendue des logiciels distribués par ce système, en comptant sur l'application systématique de ce système ; c'est à l'auteur/donateur de décider s'il a la volonté de distribuer le logiciel par un quelconque autre système, et un concessionnaire ne peut imposer ce choix.

La présente section veut rendre parfaitement claire ce que l'on pense être une conséquence du reste de la présente licence.

- Si la distribution et/ou l'utilisation du Programme est restreinte dans certains pays, sous l'effet de brevets ou d'interfaces présentant un droit d'auteur, le détenteur du droit d'auteur original, qui soumet le Programme aux dispositions de la présente licence, pourra ajouter une limitation expresse de distribution géographique excluant ces pays, de façon que la distribution ne soit autorisée que dans les pays ou parmi les pays qui ne sont pas ainsi exclus. Dans ce cas, la limitation fait partie intégrante de la présente licence, comme si elle était écrite dans le corps de la présente licence.
- La Free Software Foundation peut, de temps à autre, publier des versions révisées et/ou nouvelles du General Public License. Ces nouvelles versions seront analogues, du point de vue de leur esprit, à la présente version, mais pourront en différer dans le détail, pour résoudre de nouveaux problèmes ou de nouvelles situations.

Chaque version reçoit un numéro de version qui lui est propre. Si le programme spécifie un numéro de version de la présente licence, qui s'applique à cette dernier et "à toute autre version ultérieure", le concessionnaire a le choix de respecter les clauses et dispositions de cette version, ou une quelconque version ultérieure publiée par la Free Software Foundation. Si le programme ne spécifie pas de numéro de version de la présente licence, le concessionnaire pourra choisir une version quelconque publiée à tout moment par la Free Software Foundation.

- Si le concessionnaire souhaite incorporer des parties du programme dans d'autres programmes libres dont les conditions de distribution sont différentes, il devrait écrire à l'auteur pour demander son autorisation. Pour un logiciel soumis à droit d'auteur par la Free Software Foundation, il devra écrire à la Free Software Foundation ; nous faisons quelquefois des exceptions à cette règle. Notre décision va être guidée par le double objectif de protéger le statut libre de tous les dérivés de nos logiciels libres, et de favoriser le partage et la réutilisation des logiciels en général.
- COMME LA LICENCE DU PROGRAMME EST CONCEDEE A TITRE GRATUIT, IL N'Y AUCUNE GARANTIE S'APPLIQUANT AU PROGRAMME, DANS LA MESURE AUTORISEE PAR LA LOI EN VIGUEUR. SAUF MENTION CONTRAIRE ECRITE, LES DETENTEURS DU DROIT D'AUTEUR ET/OU LES AUTRES PARTIES METTENT LE PROGRAMME A DISPOSITON "EN L'ETAT", SANS AUCUNE GARANTIE DE QUELQUE NATURE QUE CE SOIT, EXPRESSE OU IMPLICITE, Y COMPRIS, MAIS SANS LIMITATION, LES GARANTIES IMPLICITES DE COMMERCIALISATION ET DE L'APTITUDE A UN OBJET PARTICULIER. C'EST LE CONCESSIONNAIRE QUI PREND LA TOTALITE DU RISQUE QUANT A LA QUALITE ET AUX PERFORMANCES DU PROGRAMME. SI LE PROGRAMME SE REVELAIT DEFECTUEUX, C'EST LE CONCESSIONNAIRE QUI PRENDRAIT A SA CHARGE LE COUT DE L'ENSEMBLE DES OPERATIONS NECESSAIRES D'ENTRETIEN, REPARATION OU CORRECTION.
- EN AUCUN CAS, SAUF SI LA LOI EN VIGUEUR L'EXIGE OU SI UNE CONVENTION ECRITE EXISTE A CE SUJET, AUCUN DETENTEUR DE DROITS D'AUTEUR, OU AUCUNE PARTIE AYANT LE POUVOIR DE MODIFIER ET/OU DE REDISTRIBUER LE PROGRAMME CONFORMEMENT AUX AUTORISATIONS CI-DESSUS, N'EST RESPONSABLE VIS-A-VIS DU CONCESSIONNAIRE POUR CE QUI EST DES DOMMAGES, Y COMPRIS TOUS DOMMAGES GENERAUX, SPECIAUX, ACCIDENTELS OU INDIRECTS, RESULTANT DE L'UTILISATION OU DU PROGRAMME OU DE L'IMPOSSIBILITE D'UTILISER LE PROGRAMME (Y COMPRIS, MAIS SANS LIMITATION, LA PERTE DE DONNEES, OU LE FAIT QUE DES DONNEES SONT RENDUES IMPRECISES, OU ENCORE LES PERTES EPROUVEES PAR LE CONCESSIONNAIRE OU PAR DES TIERS, OU ENCORE UN MANQUEMENT DU PROGRAMME A FONCTIONNER AVEC TOUS AUTRES PROGRAMMES), MEME SI CE DETENTEUR OU CETTE AUTRE PARTIE A ETE AVISE DE LA POSSIBILITE DE TELS DOMMAGES.

## Comment appliquer ces dispositions a vos nouveaux programmes?

Si le concessionnaire développe un nouveau programme, et s'il en souhaite l'utilisation la plus large possible dans le public, le meilleur moyen d'y arriver est d'en faire un logiciel libre, que tout le monde pourra redistribuer et modifier au titre des présentes dispositions. Dans ce but, il convient de rattacher au programme les avis suivants. Le moyen le plus sûr consiste à les rattacher au

début de chaque fichier source, pour avertir le plus efficacement possible de l'exclusion de garantie ; et chaque fichier doit comporter au moins la ligne ``copyright'', et un pointeur indiquant où est localisée la totalité de l'avis.

```
 Une ligne pour donner le nom du programme et une idée de ce qu'il fait.
Copyright (C) yyyy  nom de l'auteur

 Ce programme est un logiciel libre ; vous pouvez le redistribuer et/ou le
modifier conformément aux dispositions de la Licence Publique Générale GNU,
telle que publiée par la Free Software Foundation ; version 2 de la licence, ou
encore (à votre choix) toute version ultérieure.

 Ce programme est distribué dans l'espoir qu'il sera utile, mais SANS AUCUNE
GARANTIE ; sans même la garantie implicite de COMMERCIALISATION ou D'ADAPTATION
A UN OBJET PARTICULIER. Pour plus de détail, voir la Licence Publique Générale
GNU .

 Vous devez avoir reçu un exemplaire de la Licence Publique Générale GNU en même
temps que ce programme ; si ce n'est pas le cas, écrivez à la Free Software
Foundation Inc., 675 Mass Ave, Cambridge, MA 02139, Etats-Unis.
```

Ajoutez aussi des informations sur le moyen permettant d'entrer en contact avec vous par courrier électronique (e-mail) et courrier normal.Si le programme est interactif, prévoyez en sortie un court avis, tel que celui qui est présenté ci-dessous, lors du démarrage en mode interactif.

```
 Gnomovision version 69, Copyright (C) yyyy nom de l'auteur

 Gnomovision est livré absolument SANS AUCUNE GARANTIE ; pour plus de détail,
tapez 'show w'. Il s'agit d'un logiciel libre, et vous avez le droit de le
redistribuer dans certaines conditions ; pour plus de détail, tapez 'show c'.
```

Les instructions hypothétiques 'show w' et 'show c' doivent présenter les parties appropriées de la Licence Publique Générale. Bien évidemment, les instructions que vous utilisez peuvent porter d'autres noms que 'show w' et 'show c' ; elles peuvent même correspondre à des clics de souris ou à des éléments d'un menu, selon ce qui convient à votre programme.Si nécessaire, vous devrez aussi demander à votre employeur (si vous travaillez en tant que programmeur) ou à votre éventuelle école ou université, de signer une ``renonciation à droit d'auteur'' concernant le programme. En voici un échantillon (il suffit de modifier les noms) :

```
 Yoyodyne, Inc., par la présente, renonce à tout intérêt de droits d'auteur dans
le programme `Gnomovision' (qui fait des passages au niveau des compilateurs)
écrit par James Hacker.

 signature de Ty Coon, 1er avril 1989

 Ty Coon, President of Vice
```

La présente Licence Publique Générale n'autorise pas le concessionnaire à incorporer son programme dans des programmes propriétaires. Si votre programme est une bibliothèque de sous-programmes, vous pouvez considérer comme plus intéressant d'autoriser une édition de liens des applications propriétaires avec la bibliothèque. Si c'est ce que vous souhaitez, vous devrez utiliser non pas la présente licence, mais la Licence Publique Générale pour Bibliothèques GNU.Sauf mention contraire indiquée plus haut, le présent document est soumis aux conditions d'exploitation suivantes :

## Copyright 2001 APRIL

Ce document peut être reproduit par n'importe quel moyen que ce soit, pourvu qu'aucune modification ne soit effectuée et que cette notice soit préservée.

# 18 Licence Publique Générale GNU Limitée

Version 2.1, Février 1999

This is an unofficial translation of the GNU Lesser General Public License into French. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU LGPL−−only the original English text of the GNU LGPL does that. However, we hope that this translation will help French speakers understand the GNU LGPL better. Voici (http://www.linux−france.org/article/these/lgpl.html) une adaptation non officielle de la Licence Publique Générale Limitée du projet GNU. Elle n'a pas été publiée par la Free Software Foundation et son contenu n'a aucune portée légale car seule la version anglaise de ce document détaille le mode de distribution des logiciels sous GNU LGPL. Nous espérons cependant qu'elle permettra aux francophones de mieux comprendre la LGPL.


Copyright © Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111−1307 Etats−Unis, 1991, 1999.

La copie et la distribution de copies exactes de ce document sont autorisées, mais aucune modification n'est permise.


[Ceci est la première version publiée de la GPL Limitée. Elle joue aussi le rôle de successeur de la Licence Publique Générale GNU pour les bibliothèques, version 2, d'où le numéro de version 2.1.]

**Préambule**

Les licences d'utilisation de la plupart des programmes sont définies pour limiter ou supprimer toute liberté à l'utilisateur. A l'inverse, les Licences Publiques Générales GNU (GNU General Public Licenses) sont destinées à vous garantir la liberté de partager et de modifier les logiciels libres, et de s'assurer que ces logiciels sont effectivement accessibles à tout utilisateur.

Cette licence, la Licence Publique Générale Limitée, s'applique à certains programmes de la Free Software Foundation, typiquement les bibliothèques, comme à tout autre programme dont l'auteur l'aura décidé. Vous pouvez vous aussi l'utiliser, mais nous vous suggérons de réfléchir attentivement, en vous fondant sur les explications données ci−dessous, à la meilleure stratégie à employer dans chaque cas particulier, de la présente licence ou de la Licence Publique Générale ordinaire.

Liberté des logiciels ne signifie pas nécessairement gratuité. Nos Licences Publiques Générales sont conçues pour vous assurer la liberté de distribuer des copies des programmes, gratuitement ou non, de recevoir le code source ou de pouvoir l'obtenir, de modifier les programmes ou d'en utiliser des éléments dans de nouveaux programmes libres, en sachant que vous y êtes autorisé.

Afin de garantir ces droits, nous avons dû introduire des restrictions interdisant à quiconque de vous les refuser ou de vous demander d'y renoncer. Ces restrictions vous imposent en retour certaines obligations si vous distribuez ou modifiez des copies de bibliothèques protégées par la présente Licence. En d'autres termes, il vous incombera en ce cas de :

- transmettre aux destinataires tous les droits que vous possédez,
- expédier aux destinataires le code source ou bien tenir celui−ci à leur disposition,
- si vous liez du code à la bibliothèque, leur fournir des fichiers objets complets, de telle sorte qu'ils puissent les lier de nouveau à la bibliothèque après l'avoir modifiée et recompilée,
- leur remettre cette Licence afin qu'ils prennent connaissance de leurs droits.

Nous protégeons vos droits de deux façons: d'abord par le copyright du logiciel, ensuite par la remise de cette Licence qui vous autorise légalement à copier, distribuer et/ou modifier la bibliothèque.

Pour protéger chaque auteur, nous stipulons bien que la bibliothèque concernée ne fait l'objet d'aucune garantie. En outre, si un tiers la modifie puis la redistribue, tous ceux qui en recevront une copie doivent savoir qu'il ne s'agit pas de la version originale afin qu'une copie défectueuse n'entache pas la réputation de l'auteur de la bibliothèque.

Enfin, tout programme libre est sans cesse menacé par des dépôts de brevets. Nous souhaitons à tout prix éviter que des sociétés puissent déposer des brevets sur les Logiciels Libres pour leur propre compte, en en restreignant de ce fait les utilisateurs. Par conséquent, nous exigeons que tout dépôt de brevet accordé à une version de la bibliothèque soit compatible avec la totale liberté d'utilisation exposée dans la présente licence.

La plupart des logiciels du projet GNU, y compris certaines bibliothèques, sont couverts par la Licence Publique Générale ordinaire. La présente licence, la Licence Publique Générale GNU Limitée, concerne un certain nombre de bibliothèques, et diffère beaucoup de la Licence Publique

Générale ordinaire. Nous couvrons par la présente licence certaines bibliothèques afin de permettre à des programmes non libres d'être liés avec ces dernières.

Quand un programme est lié à une bibliothèque, que ce soit de manière statique ou par l'utilisation d'une bibliothèque partagée, l'ensemble forme légalement parlant un travail combiné, dérivé de la bibliothèque originale. C'est pourquoi la Licence Publique Générale ordinaire n'autorise une telle édition de liens que si l'ensemble qui en résulte satisfait ses critères de liberté. La Licence Publique Générale Limitée est permissive quant aux critères que doit remplir un code lié avec la bibliothèque en question.

Nous qualifions cette licence de Limitée car les garanties de liberté qu'elle apporte à l'utilisateur sont **limitées** par rapport à celles de la Licence Publique Générale ordinaire. Elle limite également les avantages que peuvent acquérir d'autres développeurs de logiciels libres dans la concurrence avec les programmes non libres. C'est à cause de ces limitations que nous utilisons la Licence Publique Générale ordinaire pour de nombreuses bibliothèques. Cependant, la Licence Limitée est avantageuse dans certaines circonstances particulières.

Par exemple, on observe (rarement) un besoin particulier d'encourager autant que possible l'utilisation d'une certaine bibliothèque, de telle sorte qu'elle devienne un standard de fait. Pour atteindre un tel but, il faut autoriser des programmes non libres à utiliser cette bibliothèque. Un cas plus fréquemment rencontré est celui où une bibliothèque libre remplit la même fonction que des bibliothèques non libres et très répandues. Il y a alors peu à gagner à limiter la bibliothèque libre aux logiciels libres, et on utilisera la Licence Publique Générale Limitée.

Dans d'autres cas, autoriser des programmes non libres à utiliser une bibliothèque particulière autorise plus de monde à utiliser une grande quantité de logiciels libres. Par exemple, la permission d'utiliser la bibliothèque GNU pour le langage C dans des programmes non libres permet à beaucoup plus de gens d'utiliser l'ensemble du système d'exploitation GNU, ainsi que sa variante GNU/Linux.

Bien que la Licence Publique Générale Limitée Limite la liberté des utilisateurs, elle assure que l'utilisateur d'un programme lié avec la Bibliothèque a la liberté et la possibilité d'exécuter ce programme en utilisant une version modifiée de la Bibliothèque.

Les termes et conditions précis selon lesquels on peut copier, distribuer, et modifier une telle bibliothèque suivent. Accordez une attention toute particulière à la différence entre un ``travail fondé sur la bibliothèque'' et un ``travail qui utilise la bibliothèque''. Celui−là renferme du code dérivé de la bibliothèque, alors que celui−ci doit être combiné à la bibliothèque pour être exécuté.

- Le présent Accord de Licence s'applique à toute bibliothèque logicielle ou tout autre programme où figure une note, placée par le détenteur des droits ou un tiers autorisé à ce faire, stipulant que ladite bibliothèque ou programme peut être distribuée selon les termes de la présente Licence Publique Générale Limitée (également appelée ``cette licence''). Chaque personne concernée par la Licence Publique Générale Limitée sera désignée par le terme ``Vous''.

  Une ``bibliothèque'' signifie une collection de fonctions et/ou de données logicielles préparées de manière à être convenablement liées avec des programmes d'application (qui utilisent certaines des fonctions et des données) dans le but de former des exécutables.

  Ci−dessous, le terme ``Bibliothèque'' se rapporte à toute bibliothèque ou oeuvre logicielle distribuée selon les présents termes. Un ``travail fondé sur la Bibliothèque'' signifie aussi bien la Bibliothèque elle−même que tout travail qui en est dérivé selon la loi, c'est−à−dire tout ouvrage reproduisant la Bibliothèque ou une partie de cette dernière, à l'identique ou bien modifiée, et/ou traduit dans une autre langue (la traduction est considérée comme une modification).

  Le ``code source'' d'un travail désigne la forme de cet ouvrage sous laquelle les modifications sont les plus aisées. Sont ainsi désignés la totalité du code source de tous les modules composant une bibliothèque logicielle, de même que tout fichier de définition associé, ainsi que les scripts utilisés pour effectuer la compilation et l'installation de la bibliothèque.

  Les activités autres que copie, distribution et modification ne sont pas couvertes par la présente Licence et sortent de son cadre. Rien ne restreint l'utilisation de la Bibliothèque, et les données issues de celle−ci ne sont couvertes que si leur contenu constitue un travail fondé sur la Bibliothèque (indépendemment du fait d'avoir été réalisé en utilisant la Bibliothèque). Tout dépend de ce que la Bibliothèque et le programme qui y fait appel sont censés produire.
- Vous pouvez copier et distribuer des copies conformes et complètes du code source de la Bibliothèque, tel que Vous l'avez reçue, sur n'importe quel support, à condition de placer sur chaque copie un copyright approprié et une restriction de garantie, de ne pas modifier ou omettre toutes les stipulations se référant à la présente Licence et à la limitation de garantie, et de fournir avec toute copie de la Bibliothèque un exemplaire de la Licence.Vous pouvez demander une rétribution financière pour la réalisation de la copie et demeurez libre de proposer une garantie assurée par vos soins, moyennant finances.
- Vous pouvez modifier votre copie ou vos copies de la Bibliothèque ou partie de celle−ci, ou d'un travail fondé sur cette Bibliothèque, et copier et distribuer ces modifications selon les termes de l'article 1, à condition de Vous conformer également aux conditions suivantes :
  - Le travail dérivé doit être lui−même une bibliothèque logicielle.
  - Ajouter aux fichiers modifiés l'indication très claire des modifications effectuées, ainsi que la date de chaque changement.

- Distribuer sous les termes de la présente Licence l'ensemble de la réalisation, à tous, et sans frais.
- Si une fonctionnalité de la Bibliothèque modifiée se réfère à une fonction ou à des données fournies par le programme d'application qui utilise la fonctionnalité en question sans pour cela utiliser d'argument lors de l'appel de cette dernière, vous devez agir au mieux pour assurer que la fonctionnalité se comporte correctement et remplit ceux de ses buts qui ont encore un sens lorsqu'une application ne fournit pas la fonction ou les données auxquelles la Bibliothèque se réfère. (Par exemple, si une fonction d'une bibliothèque calcule des racines carrées, elle a un but absolument bien défini, indépendamment de l'application. Par conséquent, la clause 2d exige que toute fonction ou toutes données fournies par l'application et utilisation cette fonction soient optionnelles : si l'application ne les propose pas, la fonction de calcul de racines carrées doit encore calculer des racines carrées.)

Toutes ces conditions s'appliquent à l'ensemble des modifications. Si des éléments identifiables de ce travail ne sont pas dérivés de la Bibliothèque et peuvent être raisonnablement considérés comme indépendants, la présente Licence ne s'applique pas à ces éléments lorsque Vous les distribuez seuls. Mais, si Vous distribuez ces mêmes éléments comme partie d'un ensemble cohérent dont le reste est fondé sur une Bibliothèque soumise à la Licence, ils lui sont également soumis, et la Licence s'étend ainsi à l'ensemble du produit, quel qu'en soit l'auteur.

Cet article n'a pas pour but de s'approprier ou de contester vos droits sur un travail entièrement réalisé par Vous, mais plutôt d'ouvrir droit à un contrôle de la libre distribution de tout travail dérivé ou collectif fondé sur la Bibliothèque.

En outre, toute fusion d'un autre travail, non fondé sur la Bibliothèque, avec la Bibliothèque (ou avec un travail dérivé de cette dernière), effectuée sur un support de stockage ou de distribution, ne fait pas tomber cet autre travail sous le contrôle de la Licence.

- Vous pouvez opter pour la Licence Publique Générale GNU ordinaire pour protéger une copie donnée de la Bibliothèque. Pour cela, il vous faudra modifier toutes les notes se référant à la présente Licence, pour qu'elles se réfèrent plutôt à la Licence Publique Générale GNU ordinaire, version 2 (si une version plus récente de la Licence Publique Générale GNU a vu le jour, vous pouvez alors spécifier cet autre numéro de version si tel est votre désir). Ne vous livrez à aucune autre modification dans ces notes.

Une fois que ce changement a été effectué dans une copie donnée, il est irréversible pour cette copie, aussi la Licence Publique Générale ordinaire s'appliquera à toutes les copies et tous les travaux dérivés qui en seront extraits.

Cette option vous servira lorsque vous souhaiterez copier une portion du code de la Bibliothèque dans un programme qui n'est pas lui−même une bibliothèque.

- Vous pouvez copier et distribuer la Bibliothèque (ou tout travail dérivé selon les conditions énoncées dans l'article 2) sous forme de code objet ou exécutable, selon les termes des articles 1 et 2, à condition de fournir le code source complet de la Bibliothèque, sous une forme lisible par un ordinateur et selon les termes des articles 1 et 2, sur un support habituellement utilisé pour l'échange de données.Si la distribution du code objet consiste à offrir un accès permettant de copier la Bibliothèque depuis un endroit particulier, l'offre d'un accès équivalent pour se procurer le code source au même endroit satisfait l'obligation de distribution de ce code source, même si l'utilisateur choisit de ne pas profiter de cette offre.
- On appelle ``travail qui utilise la Bibliothèque'' tout programme qui n'est dérivé d'aucune portion de la Bibliothèque, mais qui est conçu dans le but de fonctionner avec cette dernière en l'incluant à la compilation ou à l'édition de liens. Isolée, une telle oeuvre n'est pas un travail dérivé de la Bibliothèque, et sort donc du cadre de cette License. Cependant, lier un ``travail qui utilise la Bibliothèque'' à cette dernière produit un exécutable qui est dérivé de la Bibliothèque (en ce qu'il en contient des portions), et non plus un ``travail qui utilise la Bibliothèque''. Cet exécutable est donc couvert par la présente License. C'est l'article 6 qui énonce les conditions de distribution de tels exécutables.

Dans le cas où un ``travail qui utilise la Bibliothèque'' utilise des portions d'un fichier d'en−têtes inclus dans cette dernière, le code objet qui en résulte peut fort bien être un travail dérivé de la Bibliothèque, quand bien même ce n'est pas le cas du code source. Cette précision prend toute son importance si on peut lier ce travail sans la Bibliothèque, ou si le travail est lui−même une bibliothèque. Le seuil à partir duquel cela prend effet n'est pas exactement défini par la loi.

Si un tel fichier objet n'utilise que des paramètres numériques, les représentations des structures de données et ce par quoi elles sont lues ou modifiées, ainsi que de petites instructions macros ou fonctions embarquées (de moins de dix lignes de longueur), alors on pourra utiliser le fichier objet de la manière que l'on souhaite, qu'il soit ou non, légalement parlant, un travail dérivé (mais les exécutables renfermant ce code objet et des portions de la Bibliothèque continuent à être soumis à l'article 6).

Sinon, si le travail est dérivé de la Bibliothèque, vous pouvez distribuer le code objet de ce travail selon les conditions de l'article 6. Tout exécutable renfermant ce travail est lui aussi soumis à l'article 6, qu'il soit ou non directement lié avec la Bibliothèque à proprement parler.

- Le précédent article fait exception aux précédents. Vous pouvez également combiner ou lier un ``travail qui utilise la Bibliothèque'' avec cette dernière pour produire un travail contenant des portions de la Bibliothèque, et distribuer ce dernier sous les conditions qui vous siéront, pourvu que ces conditions autorisent la modification de ce travail pour utilisation personnelle, ainsi que l'ingénierie à revers afin de déboguer ces modifications.

Vous devez fournir avec chaque copie du travail une note très claire expliquant que la Bibliothèque fut utilisée dans sa conception, et que la Bibliothèque et son utilisation sont

couvertes par la présente Licence. Vous devez également fournir une copie de la présente Licence. Si le travail, lors de son exécution, affiche des copyrights, vous devez inclure parmi ces derniers le copyright de la Bibliothèque, ainsi qu'une référence expliquant à l'utilisateur où il pourra trouver une copie de la présente Licence. Vous devez aussi vous conformer à l'un des points suivants :

- Accompagner le travail avec l'intégralité du code source pour la Bibliothèque, sous une forme lisible par un ordinateur, ainsi que les éventuelles modifications que vous lui avez apportées pour réaliser ce travail (lequel doit être distribué selon les termes des articles 1 et 2). Si le travail est un exécutable lié avec la Bibliothèque, il vous faut proposer également, sous une forme lisible par un ordinateur, l'ensemble du ``travail qui utilise la Bibliothèque'', sous forme de code source ou objet, de telle sorte que l'utilisateur puisse modifier la Bibliothèque et effectuer de nouveau l'édition de liens, afin de produire un exécutable modifié, renfermant une version modifiée de la Bibliothèque (il est entendu que l'utilisateur qui modifie le contenu des fichiers de définitions de la Bibliothèque ne sera pas forcément capable de recompiler l'application afin d'utiliser la version modifiée de ces définitions).
- Utiliser un mécanisme de partage de bibliothèques convenable pour l'édition de liens avec la Bibliothèque. Un mécanisme convenable est un mécanisme qui : (1) utilise une copie de la bibliothèque déjà présente sur le système de l'utilisateur, plutôt que de copier des fonctions de la bibliothèque au sein de l'exécutable, et (2) fonctionnera correctement avec une version modifiée de la bibliothèque, si l'utilisateur en installe une, tant que la version modifiée sera compatible avec la version qui a servi à la réalisation du travail.
- Faire une offre écrite, valable pendant au moins trois ans, proposant de distribuer à cet utilisateur les éléments spécifiés dans l'article 6a, ci−dessus, pour un tarif n'excédant pas le coût de la copie.
- Si le travail est distribué en proposant un accès à une copie située à un endroit désigné, proposer de manière équivalente, depuis ce même endroit, un accès aux objets spécifiés ci−dessus.
- Vérifier que l'utilisateur a déjà reçu une copie de ces objets, ou que vous la lui avez déjà envoyée.

Pour un programme exécutable, la forme requise du ``travail qui utilise la Bibliothèque'' doit comprendre toute donnée et tout utilitaire nécessaires pour pouvoir reconstruire l'exécutable. Toutefois, l'environnement standard de développement du système d'exploitation mis en oeuvre (source ou binaire) −− compilateurs, bibliothèques, noyau, etc. −− constitue une exception, sauf si ces éléments sont diffusés en même temps que le programme exécutable.

Il est possible que cette clause soit en contradiction avec les restrictions apportées par les licences d'autres bibliothèques propriétaires qui habituellement n'accompagnent pas le système d'exploitation. Une telle contradiction signifie qu'il Vous est impossible d'utiliser ces dernières en conjonction avec la Bibliothèque au sein d'un exécutable distribué par Vous.

- Vous pouvez incorporer au sein d'une même bibliothèque des fonctionnalités fondées sur la Bibliothèque, qui forment un travail fondé sur cette dernière, avec des fonctionnalités issues d'autres bibliothèques, non couvertes par la présente Licence, et distribuer la bibliothèque résultante, si tant est qu'il est autorisé par ailleurs de distribuer séparément le travail fondé sur la Bibliothèque et les autres fonctionnalités, et pourvu que vous vous acquittiez des deux obligations suivantes:
  - Accompagner la bibliothèque résultante d'une copie du travail fondé sur la Bibliothèque, sans le combiner aux autres fontionnalités de bibliothèques. Cet ensemble doit être distribué selon les conditions des articles ci−dessus.
  - Ajouter à la bibliothèque mixte l'indication très claire du fait qu'une portion de la bibliothèque est un travail fondé sur la Bibliothèque, et en expliquant où trouver la version non mélangée du même travail.
- Vous ne pouvez pas copier, modifier, céder, déposer ou distribuer la Bibliothèque d'une autre manière que l'autorise la présente Licence. Toute tentative de ce type annule immédiatement vos droits d'utilisation de la Bibliothèque sous cette Licence. Toutefois, les tiers ayant reçu de Vous des copies de la Bibliothèque ou le droit d'utiliser ces copies continueront à bénéficier de leur droit d'utilisation tant qu'ils respecteront pleinement les conditions de la présente Licence.
- Ne l'ayant pas signée, Vous n'êtes pas obligé d'accepter la présente Licence. Cependant, rien d'autre ne Vous autorise à modifier ou distribuer la Bibliothèque ou quelque travaux dérivés: la loi l'interdit tant que Vous n'acceptez pas les termes de la présente Licence. En conséquence, en modifiant ou en distribuant la Bibliothèque (ou tout travail fondé sur elle), Vous acceptez implicitement tous les termes et conditions de la présente Licence.
- La diffusion d'une Bibliothèque (ou de tout travail dérivé) suppose l'envoi simultané d'une licence autorisant la copie, la distribution, l'édition de liens avec, ou la modification de la Bibliothèque, aux termes et conditions de la Licence. Vous n'avez pas le droit d'imposer de restrictions supplémentaires aux droits transmis au destinataire. Vous n'êtes pas responsable du respect de la Licence par un tiers.
- Si, à la suite d'une décision de Justice, d'une plainte en contrefaçon ou pour toute autre raison (liée ou non à la contrefaçon), des conditions Vous sont imposées (que ce soit par ordonnance, accord amiable ou autre) qui se révèlent incompatibles avec les termes de la présente Licence, Vous n'êtes pas pour autant dégagé des obligations liées à celle−ci: si Vous ne pouvez concilier vos obligations légales ou autres avec les conditions de cette Licence, Vous ne devez pas distribuer la Bibliothèque. Si une partie quelconque de cet article est invalidée ou inapplicable pour quelque raison que ce soit, le reste de l'article continue de s'appliquer et l'intégralité de l'article s'appliquera en toute autre circonstance.

Le présent article n'a pas pour but de Vous pousser à enfreindre des droits ou des dispositions légales ni en contester la validité; son seul objectif est de protéger l'intégrité du système de distribution du Logiciel Libre. De nombreuses personnes ont généreusement contribué à la large gamme de logiciels distribuée de cette façon en toute confiance; il appartient à chaque auteur/donateur de décider de diffuser ses logiciels selon les critères de son choix.

- Si la distribution et/ou l'utilisation de la Bibliothèque est limitée dans certains pays par des brevets ou des droits sur des interfaces, le détenteur original des droits qui place la Bibliothèque sous la Licence Publique Générale peut ajouter explicitement une clause de limitation géographique excluant ces pays. Dans ce cas, cette clause devient une partie intégrante de la Licence.
- La Free Software Foundation se réserve le droit de publier périodiquement des mises à jour ou de nouvelles versions de la Licence. Rédigées dans le même esprit que la présente version, elles seront cependant susceptibles d'en modifier certains détails à mesure que de nouveaux problèmes se font jour.Chaque version possède un numéro distinct. Si la Bibliothèque précise un numéro de version de cette Licence et ``toute version ultérieure'', Vous avez le choix de suivre les termes et conditions de cette version ou de toute autre version plus récente publiée par la Free Software Foundation. Si la Bibliothèque ne spécifie aucun numéro de version, Vous pouvez alors choisir l'une quelconque des versions publiées par la Free Software Foundation.
- Si Vous désirez incorporer des éléments de la Bibliothèque dans d'autres programmes libres dont les conditions de distribution diffèrent, Vous devez écrire à l'auteur pour lui en demander la permission. Pour ce qui est des programmes

directement déposés par la Free Software Foundation, écrivez−nous: une exception est toujours envisageable. Notre décision sera basée sur notre volonté de préserver la liberté de notre Programme ou de ses dérivés et celle de promouvoir le partage et la réutilisation du logiciel en général.

- PARCE QUE L'UTILISATION DE LA BIBLIOTHEQUE EST LIBRE ET GRATUITE, AUCUNE GARANTIE N'EST FOURNIE, COMME LE PERMET LA LOI. SAUF MENTION ECRITE, LES DETENTEURS DU COPYRIGHT ET/OU LES TIERS FOURNISSENT LA BIBLIOTHEQUE EN L'ETAT, SANS AUCUNE SORTE DE GARANTIE EXPLICITE OU IMPLICITE, Y COMPRIS LES GARANTIES DE COMMERCIALISATION OU D'ADAPTATION DANS UN BUT PARTICULIER. VOUS ASSUMEZ TOUS LES RISQUES QUANT A LA QUALITE ET AUX EFFETS DE LA bIBLIOTHEQUE. SI LA BIBLIOTHEQUE EST DEFECTUEUSE, VOUS ASSUMEZ LE COUT DE TOUS LES SERVICES, CORRECTIONS OU REPARATIONS NECESSAIRES.
- SAUF LORSQU'EXPLICITEMENT PREVU PAR LA lOI OU ACCEPTE PAR ECRIT, NI LE DETENTEUR DES DROITS, NI QUICONQUE AUTORISE A MODIFIER ET/OU REDISTRIBUER LA BIBLIOTHEQUE COMME IL EST PERMIS CI−DESSUS NE POURRA ETRE TENU POUR RESPONSABLE DE TOUT DOMMAGE DIRECT, INDIRECT, SECONDAIRE OU ACCESSOIRE (PERTES FINANCIERES DUES AU MANQUE A GAGNER, A L'INTERRUPTION D'ACTIVITES OU A LA PERTE DE DONNEES, ETC., DECOULANT DE L'UTILISATION DE LA BIBLIOTHEQUE OU DE L'IMPOSSIBILITE D'UTILISER CELLE−CI).

## Comment appliquer ces directives à vos nouvelles bibliothèques

Si vous développez une nouvelle bibliothèque et désirez en faire bénéficier tout un chacun, la meilleure méthode est d'en faire un Logiciel Libre que tout le monde pourra redistribuer et modifier. Vous pouvez atteindre ce but en autorisant la redistribution selon les présentes clauses (ou, c'est une autres solution, selon les termes de la Licence Publique Générale ordinaire).

Pour cela, insérez les indications suivantes dans votre bibliothèque (il est préférable et plus sûr de les faire figurer au début de chaque fichier source; dans tous les cas, chaque module source devra comporter au minimum la ligne de ``copyright'' et indiquer où résident toutes les autres indications) :

```
 une ligne pour donner le nom de la bibliothèque et donner une idée de sa
finalité.
Copyright (C) année  nom de l'auteur

 Cette bibliothèque est libre, vous pouvez la redistribuer et/ou la modifier
selon les termes de la Licence Publique Générale GNU Limitée publiée par la Free
Software Foundation (version 2 ou bien toute autre version ultérieure choisie
par vous).

 Cette bibliothèque est distribuée car potentiellement utile, mais SANS AUCUNE
GARANTIE, ni explicite ni implicite, y compris les garanties de
commercialisation ou d'adaptation dans un but spécifique. Reportez−vous à la
Licence Publique Générale GNU Limitée pour plus de détails.

 Vous devez avoir reçu une copie de la Licence Publique Générale GNU Limitée en
même temps que cette bibliothèque; si ce n'est pas le cas, écrivez à la Free
Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111−1307,
Etats−Unis.
```

Ajoutez également votre adresse électronique, le cas échéant, ainsi que votre adresse postale.

Si vous officiez en tant que programmeur, n'omettez pas de demander à votre employeur, votre établissement scolaire ou autres de signer une décharge stipulant leur renoncement aux droits qu'ils pourraient avoir sur la bibliothèque :

```
 Yoyodyne, Inc., déclare par la présente ne pas revendiquer de droits sur la
bibliothèque `Frob' réalisée par James Random Hacker.

 signature de Ty Coon, 1er Avril 1990
Ty Coon, President of Vice
```

# 19 Index des commandes, types et fonctions SQL


- A
- >
- <<

- –
-
  - –password option
- <
-
  - my_init()
  - mysql_affected_rows()
  - mysql_autocommit() .
  - mysql_bind_param()
  - mysql_bind_result()
  - mysql_change_user()
  - mysql_character_set_name()
  - mysql_close()
  - mysql_commit() .
  - mysql_connect()
  - mysql_create_db()
  - mysql_data_seek()
  - mysql_debug()
  - mysql_drop_db()
  - mysql_dump_debug_info()
  - mysql_eof()
  - mysql_errno()
  - mysql_error()
  - mysql_escape_string()
  - mysql_execute()
  - mysql_fetch()
  - mysql_fetch_field()
  - mysql_fetch_field_direct()
  - mysql_fetch_fields()
  - mysql_fetch_lengths()
  - mysql_fetch_row()
  - mysql_field_count()
  - mysql_field_count()
  - mysql_field_seek()
  - mysql_field_tell()
  - mysql_free_result()
  - mysql_get_client_info()
  - mysql_get_host_info()
  - mysql_get_proto_info()
  - mysql_get_server_info()
  - mysql_get_server_version()
  - mysql_info()
  - mysql_init()
  - mysql_insert_id()
  - mysql_kill()
  - mysql_list_dbs()
  - mysql_list_fields()
  - mysql_list_processes()
  - mysql_list_tables()
  - mysql_more_results() .
  - mysql_next_result() .
  - mysql_num_rows()
  - mysql_options()
  - mysql_param_count()
  - mysql_ping()
  - mysql_prepare()
  - mysql_prepare_result .
  - mysql_query()
  - mysql_real_connect()
  - mysql_real_escape_string()
  - mysql_real_query()
  - mysql_reload()
  - mysql_rollback() .
  - mysql_row_seek()
  - mysql_row_tell()
  - mysql_select_db()
  - mysql_send_long_data() .
  - mysql_server_end()
  - mysql_server_init()
  - mysql_shutdown()
  - mysql_stat()
  - mysql_stmt_affected_rows()
  - mysql_stmt_close()
  - mysql_stmt_errno()
  - mysql_stmt_error() .
  - mysql_stmt_store_result()
  - mysql_store_result()
  - mysql_thread_end()
  - mysql_thread_id()
  - mysql_thread_init()
  - mysql_thread_safe()
  - mysql_use_result()
- A
-
  - analyze table
- B
-
  - backup table
  - bit_and()
  - bit_count()
- C

- 
  - case
  - casts
  - check table
  - command-line options
  - comment syntax
  - configure option, --with-extra-charsets
  - convert
  - create database
  - create index
  - create table
- **D**
- 
  - date
  - dbi->quote
  - delayed
  - delete
  - describe
  - describe
  - distinct
  - distinct
  - division (/)
  - do
  - drop database
  - drop index
  - drop table
  - dumpfile
- **F**
- 
  - flush
  - functions, user-defined
- **G**
- 
  - grant statemenet
- **H**
- 
  - handler
- **I**
- 
  - ignore key
  - insert
  - insert ... select
  - interval()
  - is null, and indexes
  - isolation level
- **K**
- 
  - kill
- **L**
- 
  - last_insert_id()
  - left join
  - limit
  - limit
  - load data infile
- **M**
- 
  - méthode dbi type
  - match ... against()
  - mysql_info()
  - mysql_info()
  - mysql_info()
- **N**
- 
  - null
- **O**
- 
  - optimize table
- **P**
- 
  - parentheses ( and )
  - password()
  - password()
  - processlist
- **Q**
- 
  - quote()
- **R**
- 
  - rename table
  - repair table
  - replace
  - restore table
  - revoke
  - rollback
- **S**
- 
  - select into table
  - select speed
  - select, optimising
  - set option
  - set password statement
  - show create table
  - show privileges
  - show table types
  - show table types
  - show warnings
  - sql_no_cache
  - statements, insert
- **T**

- table_cache
- time_to_sec()
- trace dbi method
- truncate
- truncate()

- **U**
- 
  - union
  - unlock tables
  - use
- **W**
- 
  - where
- **X**
- 
  - xor, logical

# 20 Index conceptuel


- **A**
- **<**
- 
  - Blob , inserting binary data
  - Dbi/dbd
  - Dbi interface
  - Myisamchk
  - Myisamchk , example output
  - Myisamchk , options
  - Myisampack
  - Myisampack
  - Mysql_install_db
  - Mysqladmin
  - Mysqladmin
  - Mysqladmin
  - Mysqlbinlog
  - Mysqlbinlog
  - Mysqlbug script, location
  - Mysqld-max
  - Mysqld , starting
  - Mysqldump
  - Mysqlimport
  - Mysqlimport
  - Mysqlimport
  - Null values, and auto_increment columns
  - Null , testing for null
  - Prompt command
  - Replace
  - Safe_mysqld
  - Select , query cache
  - Text columns, default values
  - User table, sorting
- **A**
- 
  - Access denied errors
  - Acid
  - Acls
  - Activestate perl
  - Adding, character sets
  - Age, calculating
  - Alias names, case-sensitivity
  - Aliases, for expressions
  - Ansi mode, running
  - Ansi sql, differences from
  - Ansi sql92, extensions to
  - Apache
  - Applying, patches
  - Arithmetic expressions
  - Auto-increment, odbc
  - Auto_increment
- **B**
- 
  - Backing up, databases
  - Backslash, escape character
  - Backups, database
  - Big5 chinese character encoding
  - Binary distributions
  - Binary distributions, on hp-ux
  - Bit_functions, example
  - Borland c++ compiler
- **C**
- 
  - C api, linking problems
  - C++ apis
  - Caches, clearing
  - Calling sequences for aggregate functions, udf
  - Calling sequences for simple functions, udf
  - Can't create/write to file
  - Case-sensitivity, in access checking
  - Changes, version 3.19
  - Changes, version 3.20
  - Changes, version 3.21
  - Changes, version 3.22
  - Changes, version 3.23
  - Changes, version 4.0
  - Changes, version 4.1
  - Changes, version 5.0
  - Changing socket location
  - Changing socket location
  - Checksum errors
  - Choosing types
  - Clefs
  - Clefs, multi-colonnes
  - Clients, de mysql
  - Clients, debugging
  - Columns, other types
  - Command syntax
  - Commands out of sync
  - Comments, adding
  - Comparisons, mysql vs. others
  - Compatibility, with msql
  - Compatibility, with odbc
  - Compatibility, with odbc
  - Compatibility, with odbc
  - Compatibility, with oracle
  - Compatibility, with postgresql
  - Compatibility, with sybase
  - Connecting, remotely with ssh
  - Connection, aborted
  - Connector/odbc

- Contributing companies, list of
- Contributors, list of
- Counting, table rows
- Crash
- Crash, repeated
- Crash-me
- Creating, databases
- Creating, tables
- Cvs tree

**• D**

- Data, loading into tables
- Databases, replicating
- Databases, selecting
- Databases, symbolic links
- Datatypes, c api
- Date and time types
- Date functions, y2k compliance
- Date types, y2k issues
- Date values, problems
- Dbug package
- Default installation location
- Default values
- Defaults, embedded
- Developers, list of
- Disk full
- Disks, splitting data across
- Displaying, information, show

**• E**

- E-mail lists
- Embedded mysql server library
- Environment variables
- Environment variables
- Environment variables, list of
- Errcode
- Errors, checking tables for
- Errors, common
- Errors, list of
- Examples, compressed tables
- Examples, queries
- Expressions, extended

**• F**

- Files, my.cnf
- Files, tmp
- Files, not found message
- Files, query log
- Files, repairing
- Files, script
- Files, size limits
- Files, slow query log
- Files, update log
- Fulltext
- Function, deleting
- Functions for select and where clauses
- Functions, c api
- Functions, c prepared statements api
- Functions, native, adding
- Functions, user-definable, adding
- Functions, user-defined

**• G**

- Gdb, using
- Goals of mysql
- Gpl, gnu general public license
- Grant tables
- Grouping, expressions

**• H**

- Handling, errors
- Hints
- Hints

**• I**

- Increasing, performance
- Index, deleting
- Indexes, and is null
- Indexes, and text columns
- Indexes, block size
- Inserting, speed of
- Installation overview
- Installing, mac os x pkg packages
- Installing, overview
- Installing, perl
- Installing, user-defined functions
- Internet relay chat

**• J**

- Jdbc
- Jobs at mysql

**• K**

- Key space, myisam
- Keys, foreign
- Keys, foreign
- Keys, searching on two
- Known errors

**• L**

- Language support
- Lgpl, gnu lesser general public license
- Library, mysqlclient
- Licenses
- Links, symbolic
- Linux, binary distribution
- Locking, row-level
- Log files
- Log files, maintaining

- Log, changes

**M**

- Mailing address, for customer support
- Mailing lists, guidelines
- Master-slave setup
- Max memory used
- Memory usage, myisamchk
- Memory use
- Merge tables, defined
- Methods, locking
- Mise en cache des noms d'hôte
- Modules, list of
- Msql vs. mysql, protocol
- Msql, mysql vs msql, overview
- Multi-byte character sets
- Multi-byte characters
- Multi-part index
- Myisam, size
- Myodbc, reporting problems
- Mysql ab, defined
- Mysql certification
- Mysql consulting
- Mysql mailing lists
- Mysqld options

**N**

- Named pipes
- Names, variables
- Negative values
- New procedures, adding
- Non-transactional tables
- Null value

**O**

- Odbc, administrator
- Open tables
- Operators, cast
- Optimising, distinct
- Optimising, left join
- Optimising, limit
- Optimising, tables
- Options, provided by mysql

**P**

- Partnering with mysql ab
- Passwords, for users
- Passwords, security
- Performance, benchmarks
- Performance, disk issues
- Performance, estimating
- Performance, improving
- Perl api
- Perl dbi/dbd, installation problems
- Php api
- Porting, to other systems
- Postgresql vs. mysql, benchmarks
- Postgresql vs. mysql, features
- Postgresql vs. mysql, overview
- Postgresql vs. mysql, strategies
- Prices, support
- Privilege information, location
- Privileges, display
- Problems, date columns
- Problems, installing on ibm-aix
- Problems, linking
- Problems, starting the server
- Problems, table locking
- Problems, timezone
- Processes, display
- Processing, arguments
- Products, selling
- Programs, client
- Programs, crash-me
- Prompts, meanings
- Protocol mismatch
- Python apis

**Q**

- Queries, twin studeis project

**R**

- Re-creating, grant tables
- Recovery, from crash
- Redhat package manager
- References
- Repairing, tables
- Replication, commands
- Reserved words, exceptions
- Restarting, the server
- Root user, password resetting
- Rounding errors
- Rows, matching problems
- Running a web server
- Running, multiple servers

**S**

- Sakila
- Searching, mysql web pages
- Security, against crackers
- Server, connecting
- Server, disconnecting
- Setting, passwords
- Showing, database information
- Sorting, character sets
- Speed, linking
- Speed, of queries

- Speed, of queries
- Ssl and x509 basics
- Ssl related options
- Starting many servers
- Starting, comments
- Status, tables
- Storage of data
- String collating
- String types
- Strings, quoting
- Sub select s
- Support terms
- Support, for operating systems
- Syntax, regular expression
- System variables

**T**

- Table cache
- Table is full
- Table is full
- Table, changing
- Table, deleting
- Tables, berkeley db
- Tables, heap
- Tables, changing column order
- Tables, checking
- Tables, constant
- Tables, constant
- Tables, defragment
- Tables, defragment
- Tables, deleting rows
- Tables, fragmentation
- Tables, information about
- Tables, isam
- Tables, locking
- Tables, multiple
- Tables, selecting columns
- Tables, selecting rows
- Tables, sorting rows
- Tables, too many
- Tables, unique id for last row
- Tcl apis
- Temporary tables, problems
- Testing mysqld, mysqltest
- Testing, of mysql releases
- Texinfo
- Thread packages, differences between
- Thread support, non-native
- Threaded clients
- Threads
- Threads, rts
- Timeout
- Timeout
- Tips, optimisation
- Todo list for mysql
- Todo, embedded server
- Todo, symlinks
- Tools, converting
- Trademarks
- Triggers, stored
- Troubleshooting, freebsd
- Types, of tables

**U**

- Unloading, tables
- Updating, releases of mysql
- Upgrading, 3.21 to 3.22
- Upgrading, 3.22 to 3.23
- Upgrading, 3.23 to 4.0
- Upgrading, 3.23 to 4.0
- Upgrading, 4.0 to 4.1
- Upgrading, 4.0 to 4.1
- Urls for downloading mysql
- User privileges, adding
- Users, adding
- Utilities

**V**

- Vérrouillage
- Valeurs par défaut
- Valeurs par défaut
- Variables, mysqld
- Variables, status
- Version option
- Views
- Visual basic

**W**

- What is x509/certificate?
- Windows, compiling on
- Windows, open issues
- Wrappers, eiffel


unnumbered