

Christian Soutou

Couvre les versions
5.1 à 5.7 de MySQL

Programmer avec MySQL

4^e édition

**SQL - Transactions - PHP
Java - Optimisations**

Avec 40 exercices corrigés

EYROLLES

Christian Soutou

Maître de conférences rattaché au département Réseaux et Télécoms de l'IUT de Blagnac, Christian Soutou intervient en licence et master professionnels. Il est aussi consultant indépendant chez Orsys et auteur de nombreux ouvrages aux éditions Eyrolles.

Apprendre SQL par l'exemple

Particulièrement destiné aux débutants, cet ouvrage permet de découvrir tous les aspects de la programmation SQL (création de tables, évolution, mises à jour et extractions) par le biais du système de gestion de bases de données MySQL. Les concepts du langage procédural de MySQL y sont décrits avec précision : variables, structure de contrôle, interactions avec la base, sous-programmes, curseurs, transactions, gestion des exceptions, déclencheurs, SQL dynamique... L'auteur explique en outre comment exploiter une base MySQL (connexion et transactions) en programmant avec Java (JDBC) ou PHP. Chaque notion importante du livre est introduite à l'aide d'exemples simples et chaque chapitre se clôt par une série d'exercices, avec corrigés disponibles en ligne, qui permettront au lecteur de tester ses connaissances.

Une nouvelle édition mise à jour avec MySQL 5.7

Cette quatrième édition inclut les fonctionnalités de la version de production 5.7 de MySQL : gestion du XML et JSON, signaux et événements. L'optimisation des requêtes est également détaillée, notamment le fonctionnement de l'optimiseur, l'utilisation des statistiques et les plans d'exécution.

À qui s'adresse cet ouvrage ?

- À tous ceux qui souhaitent s'initier à MySQL
- Aux développeurs Java et PHP

Au sommaire

Partie I : Les bases de SQL. Définition et manipulation des données. Évolution d'un schéma. Interrogation et contrôle des données. **Partie II : Programmation procédurale.** Bases du langage de programmation. Programmation avancée. **Partie III : Langages et outils.** Utilisation avec Java. Utilisation avec PHP. Optimisations.



Sur le site www.editions-eyrolles.com

- Téléchargez le code source des exemples et le corrigé des exercices
- Consultez les mises à jour et les compléments
- Dialoguez avec l'auteur

Code éditeur : G14302
ISBN : 978-2-212-14302-7

Programmer
avec
MySQL

DU MÊME AUTEUR

C. SOUTOU. – **SQL pour Oracle (7^e édition).**
N°14156, 2015, 672 pages.

C. SOUTOU, F. BROUARD, N. SOUQUET et D. BARBARIN. – **SQL Server 2014.**
N°13592, 2015, 890 pages.

C. SOUTOU. – **Modélisation de bases de données (3^e édition).**
N°14206, 2015, 368 pages.

AUTRES OUVRAGES

R. BRUCHEZ. – **Les bases de données NoSQL et le Big Data (2^e édition).**
N°14155, 2015, 332 pages.

C. PIERRE DE GEYER et G. PONÇON. – **Mémento PHP 5 et SQL (3^e édition).**
N°13602, 2014, 14 pages.

R. RIMELÉ. – **Mémento MySQL 5 (4^e édition).**
N°14039, 2014, 14 pages.

P. BORGHINO, O. DASINI et A. GADAL. – **Audit et optimisation MySQL 5. Bonnes pratiques pour l'administrateur.**
N°12634, 2010, 266 pages.

D. SEGUY et P. GAMACHE. – **Sécurité PHP 5 et MySQL (3^e édition).**
N°13339, 2011, 278 pages.

Christian Soutou

Programmer avec MySQL

4^e édition

**SQL - Transactions - PHP
Java - Optimisations**

EYROLLES

The logo for EYROLLES, featuring the word "EYROLLES" in a bold, sans-serif font, centered above a horizontal line with a small circle in the middle.

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.

© Groupe Eyrolles, 2006, 2011, 2013, 2015, ISBN : 978-2-212-14302-7

Avant-propos

Nombre d'ouvrages traitent de MySQL ; certains ressemblent à des bottins téléphoniques ou proviennent de la traduction hasardeuse de la documentation officielle. Les survivants ne sont peut-être plus vraiment à jour.

Ce livre a été rédigé avec une volonté de concision et de progression dans la démarche ; il est illustré, par ailleurs, de nombreux exemples et figures. Bien que la source principale d'informations fût la documentation officielle de MySQL (<http://dev.mysql.com/doc>), l'ouvrage ne constitue pas un condensé de commandes SQL. Chaque notion importante est introduite par un exemple simple et que j'espère démonstratif. En fin de chaque chapitre, des exercices vous permettront de tester vos connaissances.

La documentation en ligne des différentes versions de MySQL (*MySQL 5.x Reference Manual*) représente une dizaine de mégaoctets au format HTML. Tous les concepts et commandes s'y trouvant ne pourraient pas être ici décemment expliqués, sauf peut-être si cet ouvrage ressemblait à un annuaire. J'ai tenté d'en extraire seulement les aspects fondamentaux sous la forme d'une synthèse. Vous n'y trouverez donc pas des considérations à propos d'aspects avancés du langage ou du serveur comme l'administration, la mise en cluster ou la réplication.

Ce livre résulte de mon expérience de l'enseignement dans le domaine des bases de données en premier, deuxième et troisième cycles universitaires dans des cursus d'informatique à vocation professionnelle (IUT, licences et masters professionnels).

Cet ouvrage s'adresse principalement aux novices désireux de découvrir SQL en programmant sous MySQL.

- Les étudiants et enseignants trouveront des exemples pédagogiques pour chaque concept abordé, ainsi que des exercices thématiques.
- Les développeurs PHP ou Java découvriront des moyens de stocker leurs données.

Guide de lecture

Ce livre s'organise autour de trois parties distinctes mais complémentaires. La première intéressera le lecteur débutant en la matière, car elle concerne les instructions SQL et les notions de base de MySQL. La deuxième partie décrit la programmation avec le langage procédural de MySQL. La troisième partie attirera l'attention des programmeurs qui envisagent d'utiliser MySQL à l'aide d'outils natifs, ou tout en programmant avec des langages évolués ou via des interfaces Web (PHP ou Java). Enfin des aspects plus avancés de SQL sont abordés comme l'optimisation des requêtes.

Première partie : SQL de base

Cette partie présente les différents aspects du langage SQL de MySQL, en étudiant en détail les instructions de base. À partir d'exemples, j'explique notamment comment déclarer, manipuler, faire évoluer et interroger des tables avec leurs différentes caractéristiques et leurs éléments associés (contraintes, index, vues, séquences). Nous étudions aussi SQL dans un contexte multi-utilisateur (droits d'accès), et au niveau du dictionnaire de données.

Deuxième partie : programmation procédurale

Cette partie décrit les caractéristiques du langage procédural de MySQL. Le chapitre 6 traite des éléments de base (structure d'un programme, variables, structures de contrôle, interactions avec la base et transactions). Le chapitre 7 traite des sous-programmes, des curseurs, de la mise en œuvre d'exceptions, de déclencheurs et l'utilisation du SQL dynamique. La gestion des documents XML et JSON est également décrite.

Troisième partie : langages et outils

Cette partie intéressera les programmeurs qui envisagent d'exploiter une base MySQL en utilisant un langage de programmation. Le chapitre 8 détaille l'API JDBC qui permet de manipuler une base MySQL par l'intermédiaire d'un programme Java. Le chapitre 9 décrit les principales fonctions de l'API *mysql* qui permet d'interfacer un programme PHP avec une base MySQL.

Le chapitre 10 est enfin consacré à l'optimisation des requêtes et des schémas. Plusieurs aspects sont étudiés : le fonctionnement de l'optimiseur, l'utilisation de statistiques et les plans d'exécution. Enfin, différents mécanismes permettant d'optimiser les traitements sont présentés : contraintes, index, tables temporaires, partitionnement et dénormalisation.

Annexes

Les annexes contiennent une bibliographie, des adresses Web et un index complet.

Convention d'écriture

La police *courrier* est utilisée pour souligner les instructions SQL, noms de types, tables, contraintes, etc. (ex : `SELECT nom FROM Pilote`).

Les majuscules sont employées pour les directives SQL, et les minuscules pour les autres éléments. Les noms des tables, index, vues, fonctions, procédures... sont précédés d'une majuscule (exemple : la table *CompagnieAerienne* contient la colonne *nomComp*).

Les termes de MySQL (bien souvent traduits littéralement de l'anglais) sont notés en italique, exemple : *trigger*, *table*, *column*, etc.

Dans une instruction SQL, les symboles { } désignent une liste d'éléments, et le symbole « | » un choix (exemple `CREATE { TABLE | VIEW }`). Les symboles « [» et «] » précisent le caractère optionnel d'une directive au sein d'une commande (exemple : `CREATE [UNIQUE | FULL-TEXT | SPATIAL] INDEX index_name [USING index_type] ON table_name (index_col_name, ...)`).



Ce sigle introduit une définition, un concept ou une remarque importante. Il apparaît soit dans une partie théorique soit dans une partie technique pour souligner des instructions importantes ou la marche à suivre avec SQL.



Ce sigle annonce soit une impossibilité de mise en œuvre d'un concept soit une mise en garde. Il est principalement utilisé dans la partie consacrée à SQL.

J'en profite pour faire passer le message suivant : si vous travaillez en version 4 de MySQL, certaines instructions décrites dans ce livre ne fonctionneront pas. Cet ouvrage n'est pas un guide de référence ! Vous trouverez sur le Web des ressources pour connaître la compatibilité de telle ou telle fonction SQL.



Ce sigle indique à partir de quelle version du serveur MySQL la fonctionnalité (commande SQL ou mécanisme de programmation) est opérationnelle. Les versions couvertes vont de 5.1 à 5.7.



Ce sigle signale une astuce ou un conseil personnel.

Contact avec l'auteur

Si vous avez des remarques à formuler sur le contenu de cet ouvrage, n'hésitez pas à m'écrire à l'adresse christian.soutou@gmail.com, mais seules les remarques relatives à l'ouvrage devraient trouver une réponse.

Par ailleurs, un site d'accompagnement de l'ouvrage (*errata*, corrigés des exercices, source des exemples et compléments) est disponible, accessible via www.editions-eyrolles.com sur la fiche de l'ouvrage.

Table des matières

Introduction	1
SQL, une norme, un succès	1
Modèle de données	2
Tables et données	2
Les clés	3
MySQL	3
Les produits	4
Licences	4
Les versions	4
Architecture	5
Moteurs de stockage	6
Oracle devient propriétaire	7
Notion de database [schéma]	8
Notion d'hôte	8
Aspects étudiés	9
Mise en œuvre de MySQL [sous Windows]	10
Installation	10
Premiers pas	11
L'interface de commande	12
Création d'un utilisateur	13
Connexion au serveur	13
Vérification de la version	14
Options de base	14
Batch	15
Votre prompt, et vite !	16
Commandes de base	17
Partie I SQL de base	21
1 Définition des données	23
Tables relationnelles	23
Création d'une table [CREATE TABLE]	23
Délimiteurs	24
Sensibilité à la casse	24
Commentaires	25
Premier exemple	26

Contraintes	27
Conventions recommandées	28
Types des colonnes	30
Structure d'une table [DESCRIBE]	34
Restrictions	34
Index	35
Arbres balancés	35
Création d'un index [CREATE INDEX]	36
Bilan	37
Destruction d'un schéma	37
Suppression d'une table [DROP TABLE]	38
Ordre des suppressions	38
Exercices	39
2 Manipulation des données	41
Insertions d'enregistrements [INSERT]	41
Syntaxe	41
Les doublons	42
Renseigner toutes les colonnes	42
Renseigner certaines colonnes	43
Renseignez vos colonnes !	43
Plusieurs enregistrements	44
Ne pas respecter des contraintes	44
Insertions multilignes	45
Données binaires	46
Énumérations	46
Dates et heures	49
Séquences	53
Utilisation en tant que clé primaire	53
Modification d'une séquence	54
Utilisation en tant que clé étrangère	55
Modifications de colonnes	56
Syntaxe [UPDATE]	56
Modification d'une colonne	57
Modification de plusieurs colonnes	57
Modification de plusieurs enregistrements	57
Ne pas respecter les contraintes	58
Restrictions	59
Dates et intervalles	59
Remplacement d'un enregistrement	63
Suppressions d'enregistrements	63
Instruction DELETE	64
Instruction TRUNCATE	64
Intégrité référentielle	65
Syntaxe	65

	Cohérences assurées	66
	Contraintes côté « père »	67
	Contraintes côté « fils »	67
	Clés composites et nulles	68
	Cohérence du fils vers le père	68
	Cohérence du père vers le fils	69
	En résumé	71
	Insertions à partir d'un fichier	71
	Exercices	74
3	Évolution d'un schéma	77
	Renommer une table [RENAME]	77
	Modifications structurelles [ALTER TABLE]	78
	Ajout de colonnes	78
	Renommer des colonnes	79
	Modifier le type des colonnes	79
	Valeurs par défaut	80
	Supprimer des colonnes	80
	Énumérations	81
	Colonnes virtuelles	82
	Modifications comportementales	84
	Ajout de contraintes	84
	Suppression de contraintes	86
	Désactivation des contraintes	88
	Réactivation des contraintes	89
	Contraintes différées	92
	Exercices	93
4	Interrogation des données	97
	Généralités	97
	Syntaxe [SELECT]	98
	Pseudotable	98
	Projection [éléments du SELECT]	99
	Extraction de toutes les colonnes	100
	Extraction de certaines colonnes	101
	Alias	101
	Duplicatas	102
	Expressions simples	102
	Ordonnement	103
	Concaténation	104
	Insertion multiligne	104
	Limitation du nombre de lignes	105
	Restriction [WHERE]	105
	Opérateurs de comparaison	106

Opérateurs logiques	107
Opérateurs intégrés.	107
Alias.	109
Fonctions	109
Caractères.	109
Numériques.	113
Fonction pour les bits	114
Dates.	115
Conversions	119
Comparaisons	120
Énumérations	121
Autres fonctions.	123
Regroupements	124
Fonctions de groupe	124
Étude du GROUP BY et HAVING	126
Opérateurs ensemblistes	129
Restrictions	129
Exemple	130
Intersection	130
Opérateurs UNION et UNION ALL.	131
Différence	132
Ordonner les résultats.	133
Produit cartésien	134
Bilan	135
Jointures	136
Classification	136
Jointure relationnelle	137
Jointures SQL2	137
Types de jointures.	138
Équijointure	138
Autojointure	140
Inéquijointure.	142
Jointures externes.	143
Jointures procédurales	145
Sous-interrogation dans la clause FROM	149
Sous-interrogations synchronisées	150
Autres directives SQL2	152
Division	155
Définition	155
Division inexacte	156
Division exacte	157
Résultats en HTML ou XML	157
Écriture dans un fichier	158
Exercices	160

5	Contrôle des données	163
	Gestion des utilisateurs	164
	Classification	164
	Création d'un utilisateur [CREATE USER]	164
	Modification d'un utilisateur	166
	Renommer un utilisateur [RENAME USER]	166
	Suppression d'un utilisateur [DROP USER]	167
	Gestion des bases de données	167
	Création d'une base [CREATE DATABASE]	167
	Sélection d'une base de données [USE]	169
	Modification d'une base [ALTER DATABASE]	170
	Suppression d'une base [DROP DATABASE]	170
	Privilèges	170
	Niveaux de privilèges	171
	Tables de la base mysql	171
	Table mysql.user	172
	Attribution de privilèges [GRANT]	175
	Table mysql.db	179
	Table mysql.host	180
	Table mysql.tables_priv	180
	Table mysql.columns_priv	180
	Table mysql.procs_priv	181
	Révocation de privilèges [REVOKE]	182
	Attributions et révocations « sauvages »	184
	Accès distants	185
	Connexion par l'interface de commande	185
	Table mysql.host	186
	Vues	187
	Création d'une vue [CREATE VIEW]	188
	Classification	189
	Vues monotables	189
	Vues complexes	194
	Autres utilisations de vues	198
	Transmission de droits	201
	Modification d'une vue [ALTER VIEW]	202
	Visualisation d'une vue [SHOW CREATE VIEW]	202
	Suppression d'une vue [DROP VIEW]	202
	Dictionnaire des données	203
	Constitution	203
	Modèle graphique du dictionnaire des données	204
	Démarche à suivre	204
	Classification des vues	207
	Moteurs du serveur	209
	Bases de données du serveur	209

Composition d'une base	210
Détail de stockage d'une base	210
Structure d'une table	211
Recherche des contraintes d'une table	213
Composition des contraintes d'une table	214
Détails des contraintes référentielles	214
Recherche du code source d'un sous-programme	216
Paramètres des sous-programmes stockés	217
Privilèges des utilisateurs d'une base de données	217
Commande SHOW	219
Exercices	221

Partie II Programmation procédurale 223

6 Bases du langage de programmation 225

Généralités	225
Environnement client-serveur	225
Avantages	226
Structure d'un bloc	226
Portée des objets	227
Casse et lisibilité	227
Identificateurs	228
Commentaires	228
Variables	228
Variables scalaires	229
Affectations	229
Restrictions	229
Résolution de noms	230
Opérateurs	230
Variables de session	231
Conventions recommandées	232
Test des exemples	232
Structures de contrôle	233
Structures conditionnelles	233
Structures répétitives	235
Interactions avec la base	238
Extraire des données	238
Manipuler des données	240
Gestion des transactions	242
Début et fin d'une transaction	243
Gestion des anomalies transactionnelles	246
Transactions en lecture seule	251
Le problème du verrou mortel [deadlock]	252

Verrouillage manuel	253
Contrôle des transactions	254
Quel mode adopter ?	255
Où placer les transactions ?	255
Modes d'exécution SQL	256
Le contexte	256
Programmation transactionnelle	256
Les dates	257
Les séquences	259
Chaînes de caractères	260
Les moteurs	261
Portabilité	262
Les combinaisons	263
Exercices	264
7 Programmation avancée	267
Sous-programmes	267
Généralités	267
Procédures cataloguées	268
Fonctions cataloguées	269
Structure d'un sous-programme	269
Exemples	270
Fonction n'interagissant pas avec la base	272
Compilation	273
Appel d'un sous-programme	273
Récursivité	275
Sous-programmes imbriqués	276
Modification d'un sous-programme	277
Destruction d'un sous-programme	277
Restrictions	278
Curseurs	278
Généralités	278
Instructions	279
Parcours d'un curseur	280
Accès concurrents [FOR UPDATE]	281
Restrictions	282
Erreurs [codes et messages]	282
Exceptions	283
Généralités	284
Exceptions avec EXIT	285
Exceptions avec CONTINUE	289
Gestion des autres erreurs [SQLEXCEPTION]	290
Même erreur sur différentes instructions	292
Exceptions nommées	295

Déroutements [SIGNAL et RESIGNAL]	297
Déclencheurs	304
Généralités	304
À quoi sert un déclencheur ?	304
Mécanisme général	305
Syntaxe	306
Déclencheurs LMD [de lignes]	307
Appel de sous-programmes	313
Dictionnaire des données	315
Programmation d'une contrainte de vérification	316
Programmation dans un déclencheur	318
Exceptions dans un déclencheur	318
Tables mutantes	321
Restrictions	322
Suppression d'un déclencheur	322
SQL dynamique	322
Syntaxe	323
Exemples	324
Restrictions	325
Paramètres de retour	328
Programmation d'événements	329
Le contexte	329
Création d'une planification	329
Exemple	331
Dictionnaire des données	333
Modification	334
Restrictions actuelles	335
Gestion de XML	335
Fonctions XML	336
Gestion des erreurs	342
Limitations	343
Chargement XML [LOAD XML]	345
Gestion de JSON	350
Exercices	352

Partie III Langages et outils 355

8 Utilisation avec Java	357
JDBC avec Connector/J	357
Classification des pilotes [drivers]	358
Le paquetage <code>java.sql</code>	359
Structure d'un programme	359
Test de votre configuration	360

Connexion à une base	361
Base Access	362
Base MySQL	363
Interface <code>Connection</code>	363
États d'une connexion	364
Interfaces disponibles	364
Méthodes génériques pour les paramètres	364
États simples [interface <code>Statement</code>]	365
Méthodes à utiliser	366
Correspondances de types	366
Manipulations avec la base	368
Suppression de données	368
Ajout d'enregistrements	369
Modification d'enregistrements	369
Extraction de données	369
Curseurs statiques	370
Curseurs navigables	371
Curseurs modifiables	375
Suppressions	377
Modifications	378
Insertions	378
Gestion des séquences	379
Méthode <code>getGeneratedKeys</code>	380
Curseur modifiable	380
Interface <code>ResultSetMetaData</code>	381
Interface <code>DatabaseMetaData</code>	382
Instructions paramétrées [<code>PreparedStatement</code>]	384
Extraction de données [<code>executeQuery</code>]	384
Mises à jour [<code>executeUpdate</code>]	385
Instruction LDD [<code>execute</code>]	385
Procédures cataloguées	386
Exemple	387
Transactions	388
Points de validation	389
Traitement des exceptions	390
Affichage des erreurs	391
Traitement des erreurs	391
Exercices	393
9 Utilisation avec PHP	395
Configuration adoptée	395
Logiciels	396
Fichiers de configuration	396

	Test d'Apache et de PHP	397
	Test d'Apache, de PHP et de MySQL	397
	API de PHP pour MySQL	398
	Connexion	398
	Interactions avec la base	399
	Extractions	401
	Instructions paramétrées	404
	Gestion des séquences	406
	Traitement des erreurs	407
	Procédures cataloguées	408
	Métadonnées	410
	Style d'écriture objet	414
	Exercices	415
10	Optimisations	419
	Cadre général	419
	Les acteurs	419
	Contexte et objectifs	420
	Causes possibles	420
	Présentation du jeu d'exemples	421
	L'optimiseur	421
	L'estimateur	423
	Les statistiques destinées à l'optimiseur	424
	Collecte	425
	Visualisation des statistiques	426
	Quand mettre à jour les statistiques ?	427
	Outils de mesure de performances	428
	MySQL Query Analyzer	428
	Visualisation des plans d'exécution	430
	Organisation des données	435
	Les contraintes	435
	Indexation	437
	Jointures	450
	Configuration de l'optimiseur [les hints]	452
	Tables temporaires	454
	Partitionnement	456
	Annexe : bibliographie et webographie	467
	Index	469

Partie I

SQL de base

Chapitre 1

Définition des données

Ce chapitre décrit les instructions SQL qui constituent l'aspect LDD (langage de définition des données). À cet effet, nous verrons notamment comment déclarer une table avec ses éventuels index et contraintes.

Tables relationnelles

Une table est créée en SQL par l'instruction `CREATE TABLE`, modifiée au niveau de sa structure par l'instruction `ALTER TABLE` et supprimée par la commande `DROP TABLE`.

Création d'une table (`CREATE TABLE`)

Pour pouvoir créer une table dans votre base, il faut que vous ayez reçu le privilège `CREATE`. Le mécanisme des privilèges est décrit au chapitre 5.

La syntaxe SQL simplifiée est la suivante :

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] [nomBase.]nomTable
  (colonne1 type1
    [NOT NULL | NULL] [DEFAULT valeur1] [COMMENT 'chaine1']
  [, colonne2 type2
    [NOT NULL | NULL] [DEFAULT valeur2] [COMMENT 'chaine2'] ]
  [CONSTRAINT nomContrainte1 typeContrainte1 ...]
  [ENGINE= InnoDB | MyISAM | ...];
```

- `TEMPORARY` : pour créer une table qui n'existera que durant la session courante (la table sera supprimée à la déconnexion). Deux connexions peuvent ainsi créer deux tables temporaires de même nom sans risquer de conflit. Il faut posséder le privilège `CREATE TEMPORARY TABLES`.
- `IF NOT EXISTS` : permet d'éviter qu'une erreur se produise si la table existe déjà (si c'est le cas, elle n'est aucunement affectée par la tentative de création).
- *nomBase* : (jusqu'à 64 caractères permis dans un nom de répertoire ou de fichier sauf « / », « \ » et « . ») s'il est omis, il sera assimilé à la base connectée. S'il est précisé, il

désigne soit la base connectée soit une autre base (dans ce cas, il faut que l'utilisateur courant ait le droit de créer une table dans l'autre base). Nous aborderons ces points dans le chapitre Contrôle des données et nous considérerons jusque-là que nous travaillons dans la base courante (ce sera votre configuration la plupart du temps).

- *nomTable* : mêmes limitations que pour le nom de la base.
- *colonnei typei* : nom d'une colonne (mêmes caractéristiques que pour les noms des tables) et son type (INTEGER, CHAR, DATE...). Nous verrons quels types sont disponibles sous MySQL. La directive DEFAULT fixe une valeur par défaut. La directive NOT NULL interdit que la valeur de la colonne soit nulle.



NULL représente une valeur qu'on peut considérer comme non disponible, non affectée, inconnue ou inapplicable. Elle est différente d'un espace pour un caractère ou d'un zéro pour un nombre.

- COMMENT : (jusqu'à 60 caractères) permet de commenter une colonne. Ce texte sera ensuite automatiquement affiché à l'aide des commandes SHOW CREATE TABLE et SHOW FULL COLUMNS (voir le chapitre 5).
- *nomContrainte typeContrainte* : nom de la contrainte et son type (clé primaire, clé étrangère, etc.). Nous allons détailler dans le paragraphe suivant les différentes contraintes possibles.
- ENGINE : définit le type de table (par défaut InnoDB, bien adapté à la programmation de transactions et adopté dans cet ouvrage). Le moteur MYISAM est très utilisé du fait de sa robustesse, mais il ne supporte ni les transactions ni l'intégrité référentielle. D'autres types existent, citons MEMORY pour les tables temporaires, ARCHIVE, etc.
- « ; » : symbole par défaut qui termine une instruction MySQL en mode ligne de commande (en l'absence d'un autre délimiteur).

Délimiteurs

En mode ligne de commande, il est possible (par la directive `delimiter`) de choisir le symbole qui terminera chaque instruction. Dans l'exemple suivant on choisit le dollar ; au cours de l'ouvrage nous resterons avec le symbole par défaut de MySQL à savoir « ; ».

```
delimiter $
CREATE TABLE Test (t CHAR(8))$
```

Sensibilité à la casse

Alors que MySQL est sensible par défaut à la casse (au niveau des noms de base et de table) dans la plupart des distributions Unix, il ne l'est pas pour Windows ! En revanche, concernant les noms de colonnes, index, alias de colonnes, déclencheurs et procédures cataloguées,

MySQL n'est pas sensible à la casse tous systèmes confondus. En fait, tous ces noms sont stockés en minuscules dans le dictionnaire de données.



La variable `lower_case_table_names` permet de forcer la sensibilité à la casse pour les noms des tables et des bases de données (si elle vaut 0, la sensibilité à la casse est active et les noms sont stockés en minuscules ; 1, pas de sensibilité à la casse et les noms sont stockés en minuscules ; 2, pas de sensibilité à la casse et les noms sont stockés en respectant la casse).

Je vous invite à positionner cette variable à 0 de manière à homogénéiser le codage et à contrôler un peu plus l'écriture de vos instructions SQL. De plus, c'est l'option par défaut sur Linux. Dans le fichier `my.ini`, sous la section serveur identifiée par `[mysqld]`, ajouter la ligne et le commentaire suivants :

```
# Rend sensible à la CASSE les noms de tables et de database
lower_case_table_names=0
```



Refusez ce type de programmation (rendue impossible d'ailleurs si la variable `lower_case_table_names` est positionnée à 0).

```
mysql> SELECT * FROM Avion WHERE AVION.capacite > 150;
```

Par ailleurs, la casse devrait toujours avoir (quel que soit le SGBD concerné) une incidence majeure dans les expressions de comparaison entre colonnes et valeurs, que ce soit dans une instruction SQL ou un test dans un programme.



Ainsi l'expression « `nomComp='Air France'` » a la même signification que l'expression « `nomComp='AIR France'` » avec MySQL ! Horreur, oui.

Donc, si vous désirez vérifier la casse au sein même des données, il faudra utiliser la fonction `BINARY()` qui convertit en bits une expression. En effet, « `BINARY('AIR France')` » est différent de « `BINARY('Air France')` » et « `BINARY(nomComp)=BINARY('Air France')` » renverra vrai si tel est le cas en respectant la casse.

Commentaires

Dans toute instruction SQL (déclaration, manipulation, interrogation et contrôle des données), il est possible d'inclure des retours chariot, des tabulations, espaces et commentaires (sur une ligne précédée de deux tirets « `--` », en fin de ligne à l'aide du dièse « `#` », au sein d'une ligne ou sur plusieurs lignes entre « `/*` » et « `*/` »). Les scripts suivants décrivent la déclaration d'une même table en utilisant différentes conventions :

Tableau 1-1 Différentes écritures SQL

Sans commentaire	Avec commentaires
<pre>CREATE TABLE Test(colonne DECIMAL(38,8));</pre>	<pre>CREATE TABLE -- nom de la table Test(#début de la description COLONNE DECIMAL(38,8)) -- fin, ne pas oublier le point-virgule. ;</pre>
<pre>CREATE TABLE Test (colonne DECIMAL(38,8));</pre>	<pre>CREATE TABLE Test (/* une plus grande description des colonnes */ COLONNE /* type : */ DECIMAL(38,8));</pre>

Cet ouvrage utilise les conventions suivantes :



- Tous les mots-clés de SQL sont notés en majuscules.
- Les noms de tables sont notés en Minuscules (excepté la première lettre, ces noms seront quand même stockés dans le système en minuscules).
- Les noms de colonnes et de contraintes en minuscules.

Premier exemple

Le tableau ci-après décrit l'instruction SQL qui permet de créer la table *Compagnie* illustrée par la figure suivante, dans la base *bduutil* (l'absence du préfixe « *bduutil.* » conduirait au même résultat si *bduutil* était la base connectée lors de l'exécution du script).

Figure 1-1 Table à créer

Compagnie

comp	nrue	rue	ville	nomComp

Tableau 1-2 Création d'une table et de ses contraintes

Instruction SQL	Commentaires
<pre>CREATE TABLE bduтил.Compagnie (comp CHAR(4), nrue INTEGER(3), rue VARCHAR(20), ville VARCHAR(15) DEFAULT 'Paris' COMMENT 'Par default : Paris', nomComp VARCHAR(15) NOT NULL);</pre>	<p>La table contient cinq colonnes (quatre chaînes de caractères et un numérique de trois chiffres). La colonne <code>ville</code> est commentée.</p> <p>La table inclut en plus deux contraintes :</p> <ul style="list-style-type: none"> • <code>DEFAULT</code> qui fixe <i>Paris</i> comme valeur par défaut de la colonne <code>ville</code> ; • <code>NOT NULL</code> qui impose une valeur non nulle dans la colonne <code>nomComp</code>.

Contraintes

Les contraintes ont pour but de programmer des règles de gestion au niveau des colonnes des tables. Elles peuvent alléger un développement côté client (si on déclare qu'une note doit être comprise entre 0 et 20, les programmes de saisie n'ont plus à tester les valeurs en entrée mais seulement le code retour après connexion à la base ; on déporte les contraintes côté serveur).

Les contraintes peuvent être déclarées de deux manières :

- En même temps que la colonne (valable pour les contraintes monocolumnes) ; ces contraintes sont dites « en ligne » (*inline constraints*). L'exemple précédent en déclare deux.
- Après que la colonne est déclarée ; ces contraintes ne sont pas limitées à une colonne et peuvent être personnalisées par un nom (*out-of-line constraints*).

Il est recommandé de déclarer les contraintes `NOT NULL` en ligne, les autres peuvent soit être déclarées en ligne, soit être nommées. Étudions à présent les types de contraintes nommées (*out-of-line*). Les quatre types de contraintes les plus utilisées sont les suivants :

```
CONSTRAINT nomContrainte
UNIQUE (colonne1 [,colonne2]...)
PRIMARY KEY (colonne1 [,colonne2]...)
FOREIGN KEY (colonne1 [,colonne2]...)
REFERENCES nomTablePere [(colonne1 [,colonne2]...)]
           [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
           [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
CHECK (condition)
```

- La contrainte `UNIQUE` impose une valeur distincte au niveau de la table (les valeurs nulles font exception à moins que `NOT NULL` soit aussi appliquée sur les colonnes).
- La contrainte `PRIMARY KEY` déclare la clé primaire de la table. Un index est généré automatiquement sur la ou les colonnes concernées. Les colonnes clés primaires ne peuvent être ni nulles ni identiques (en totalité si elles sont composées de plusieurs colonnes).

- La contrainte `FOREIGN KEY` déclare une clé étrangère entre une table enfant (*child*) et une table père (*parent*). Ces contraintes définissent l'intégrité référentielle que nous aborderons plus tard. Les directives `ON UPDATE` et `ON DELETE` disposent de quatre options que nous détaillerons avec les directives `MATCH` à la section « Intégrité référentielle » du chapitre 2).



La contrainte `CHECK` impose un domaine de valeurs à une colonne ou une condition (exemples : `CHECK (note BETWEEN 0 AND 20)`, `CHECK (grade='Copilote' OR grade='Commandant')`). Il est permis de déclarer des contraintes `CHECK` lors de la création d'une table mais le mécanisme de vérification des valeurs n'est toujours pas implémenté.

Dans le manuel de référence (<http://dev.mysql.com/doc/refman/x.y/en/alter-table.html>), il est dit que la clause `CHECK` est comprise par MySQL mais non interprétée par tous les moteurs de stockage pour des raisons de compatibilité de portage de code vers un autre SGBD.



Il est recommandé de ne pas définir de contraintes sans les nommer (bien que cela soit possible), car il sera difficile de les faire évoluer (désactivation, réactivation, suppression), et la lisibilité des programmes en sera affectée.

Nous verrons au chapitre 3 comment ajouter, supprimer, désactiver et réactiver des contraintes (options de la commande `ALTER TABLE`).

Conventions recommandées

Adoptez les conventions d'écriture suivantes pour vos contraintes :



- Préfixez par `pk_` le nom d'une contrainte clé primaire, `fk_` une clé étrangère, `ck_` une vérification, `un_` une unicité.
 - Pour une contrainte clé primaire, suffixez du nom de la table la contrainte (exemple `pk_Avion`).
 - Pour une contrainte clé étrangère, renseignez (ou abrégez) les noms de la table source, de la clé, et de la table cible (exemple `fk_Pil_compa_Comp`).
-

En respectant nos conventions, déclarons les tables de l'exemple suivant (`Compagnie` avec sa clé primaire et `Avion` avec sa clé primaire et sa clé étrangère). Du fait de l'existence de la clé étrangère, la table `Compagnie` est dite « parent » (ou « père ») de la table `Avion` « enfant »

(ou « fils »). Cela résulte de la traduction d'une association *un-à-plusieurs* entre les deux tables (*Modélisation de bases de données*, Eyrolles 2015). Nous reviendrons sur ces principes à la section « Intégrité référentielle » du prochain chapitre.

Figure 1-2 Deux tables reliées à créer



Tableau 1-3 Contraintes en ligne et nommées

Tables	Contraintes
<pre>CREATE TABLE bduutil.Compagnie (comp CHAR(4), nrue INTEGER(3), rue VARCHAR(20), ville VARCHAR(15) DEFAULT 'Paris' COMMENT 'Par defaut : Paris', nomComp VARCHAR(15) NOT NULL, CONSTRAINT pk_Compagnie PRIMARY KEY(comp));</pre>	<p>Deux contraintes en ligne et une contrainte nommée de clé primaire.</p>
<pre>CREATE TABLE bduutil.Pilote (brevet VARCHAR(6), nom VARCHAR(30) NOT NULL, pseudo VARCHAR(8), nbhVol DECIMAL(7,2), compa CHAR(4), CONSTRAINT pk_Pilote PRIMARY KEY(brevet), CONSTRAINT ck_nbhVol CHECK (nbhVol BETWEEN 0 AND 20000), CONSTRAINT un_nom UNIQUE (nom), CONSTRAINT fk_Pil_compa_Comp FOREIGN KEY (compa) REFERENCES bduutil.Compagnie(comp));</pre>	<p>Une contrainte en ligne et quatre contraintes nommées :</p> <ul style="list-style-type: none"> • Clé primaire • NOT NULL • CHECK (nombre d'heures de vol compris entre 0 et 20 000) • UNIQUE (homonymes interdits) • Clé étrangère

Remarques



- L'ordre n'est pas important dans la déclaration des contraintes nommées.
- PRIMARY KEY équivaut à : UNIQUE + NOT NULL + index.

- L'ordre de création des tables est important quand on définit les contraintes en même temps que les tables (on peut différer la création ou l'activation des contraintes, voir le chapitre 3). Il faut créer d'abord les tables « pères » puis les tables « fils ». Le script de destruction des tables suit le raisonnement inverse.
-

Types des colonnes

Pour décrire les colonnes d'une table, MySQL fournit les types prédéfinis suivants (*built-in datatypes*) :

- caractères (CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT) ;
- valeurs numériques (TINYINT, SMALLINT, MEDIUMINT, INT, INTEGER, BIGINT, FLOAT, DOUBLE, REAL, DECIMAL, NUMERIC, et BIT) ;
- date/heure (DATE, DATETIME, TIME, YEAR, TIMESTAMP) ;
- données binaires (BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB) ;
- énumérations (ENUM, SET).

Détaillons à présent ces types. Nous verrons comment utiliser les plus courants au chapitre 2 et les autres au fil de l'ouvrage.

Caractères

Le type CHAR permet de stocker des chaînes de caractères de taille fixe. Les valeurs sont stockées en ajoutant, s'il le faut, des espaces (*trailing spaces*) à concurrence de la taille définie. Ces espaces ne seront pas considérés après extraction à partir de la table.

Le type VARCHAR permet de stocker des chaînes de caractères de taille variable. Les valeurs sont stockées sans l'ajout d'espaces à concurrence de la taille définie. Depuis la version 5.0.3 de MySQL, les éventuels espaces de fin de chaîne seront stockés et extraits en conformité avec la norme SQL. Des caractères Unicode (méthode de codage universelle qui fournit une valeur de code unique pour chaque caractère quels que soient la plate-forme, le programme ou la langue) peuvent aussi être stockés.

Les types BINARY et VARBINARY sont similaires à CHAR et VARCHAR, excepté par le fait qu'ils contiennent des chaînes d'octets sans tenir compte d'un jeu de caractères en particulier.

Les quatre types permettant aussi de stocker du texte sont TINYTEXT, TEXT, MEDIUMTEXT, et LONGTEXT. Ces types sont associés à un jeu de caractères. Il n'y a pas de mécanisme de suppression d'espaces de fin ni de possibilité d'y associer une contrainte DEFAULT.

Tableau 1-4 Types de données caractères

Type	Description	Commentaire pour une colonne
CHAR(<i>n</i>) [BINARY ASCII UNICODE]	Chaîne fixe de <i>n</i> octets ou caractères.	Taille fixe (maximum de 255 caractères).
VARCHAR(<i>n</i>) [BINARY]	Chaîne variable de <i>n</i> caractères ou octets.	Taille variable (maximum de 65 535 caractères).
BINARY(<i>n</i>)	Chaîne fixe de <i>n</i> octets.	Taille fixe (maximum de 255 octets).
VARBINARY(<i>n</i>)	Chaîne variable de <i>n</i> octets.	Taille variable (maximum de 255 octets).
TINYTEXT(<i>n</i>)	Flot de <i>n</i> octets.	Taille fixe (maximum de 255 octets).
TEXT(<i>n</i>)	Flot de <i>n</i> octets.	Taille fixe (maximum de 65 535 octets).
MEDIUMTEXT(<i>n</i>)	Flot de <i>n</i> octets.	Taille fixe (maximum de 16 mégaoctets).
LONGTEXT(<i>n</i>)	Flot de <i>n</i> octets.	Taille fixe (maximum de 4,29 gigaoctets).

Valeurs numériques

De nombreux types sont proposés par MySQL pour définir des valeurs exactes (entiers ou décimaux positifs ou négatifs : `INTEGER` et `SMALLINT`), et des valeurs à virgule fixe ou flottante (`FLOAT`, `DOUBLE` et `DECIMAL`). En plus des spécifications de la norme SQL, MySQL propose les types d'entiers restreints (`TINYINT`, `MEDIUMINT` et `BIGINT`). Le tableau suivant décrit ces types :

- *n* indique le nombre de positions de la valeur à l'affichage (le maximum est de 255). Ainsi, il est possible de déclarer une colonne `TINYINT(4)` sachant que seules 3 positions sont nécessaires en fonction du domaine de valeurs permises.
- La directive `UNSIGNED` permet de considérer seulement des valeurs positives.
- La directive `ZEROFILL` complète par des zéros à gauche une valeur (par exemple : soit un `INTEGER(5)` contenant 4, si `ZEROFILL` est appliqué, la valeur extraite sera « 00004 »). En déclarant une colonne `ZEROFILL`, MySQL l'affecte automatiquement aussi à `UNSIGNED`.

Synonymes et alias



- `INT` est synonyme de `INTEGER`.
- `DOUBLE PRECISION` et `REAL` sont synonymes de `DOUBLE`.
- `DEC NUMERIC` et `FIXED` sont synonymes de `DECIMAL`.
- `SERIAL` est un alias pour `BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE`.
- Dans toute instruction SQL, écrivez la virgule avec un point (7/2 retourne 3.5).

Tableau 1-5 Types de données numériques

Type	Description
BIT[(n)]	Ensemble de n bits. Taille de 1 à 64 (par défaut 1).
TINYINT[(n)] [UNSIGNED] [ZEROFILL]	Entier (sur un octet) de -128 à 127 signé, 0 à 255 non signé.
BOOL et BOOLEAN	Synonymes de TINYINT(1), la valeur zéro est considérée comme fausse. Le non-zéro est considéré comme vrai. Dans les prochaines versions, le type <i>boolean</i> , comme le préconise la norme SQL, sera réellement pris en charge.
SMALLINT[(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 2 octets) de -32 768 à 32 767 signé, 0 à 65 535 non signé.
MEDIUMINT[(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 3 octets) de -8 388 608 à 8 388 607 signé, 0 à 16 777 215 non signé.
INTEGER[(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 4 octets) de -2 147 483 648 à 2 147 483 647 signé, 0 à 4 294 967 295 non signé.
BIGINT[(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 8 octets) de -9 223 372 036 854 775 808 à 9 223 372 036 854 775 807 signé, 0 à 18 446 744 073 709 551 615 non signé.
FLOAT[(n[,p])] [UNSIGNED] [ZEROFILL]	Flottant (de 4 à 8 octets) p désigne la précision simple (jusqu'à 7 décimales) de $-3.4 \cdot 10^{+38}$ à $-1.1 \cdot 10^{-38}$, 0, signé, et de $1.1 \cdot 10^{-38}$ à $3.4 \cdot 10^{+38}$ non signé.
DOUBLE[(n[,p])] [UNSIGNED] [ZEROFILL]	Flottant (sur 8 octets) p désigne la précision double (jusqu'à 15 décimales) de $-1.7 \cdot 10^{+308}$ à $-2.2 \cdot 10^{-308}$, 0, signé, et de $2.2 \cdot 10^{-308}$ à $1.7 \cdot 10^{+308}$ non signé.
DECIMAL[(n[,p])] [UNSIGNED] [ZEROFILL]	Décimal (sur 4 octets) à virgule fixe, p désigne la précision (nombre de chiffres après la virgule, maximum 30). Par défaut n vaut 10, p vaut 0. Nombre maximal de chiffres pour un décimal : 65.

Dates et heures

Les types suivants permettent de stocker des moments ponctuels (dates, dates et heures, années, et heures). Les fonctions NOW() et SYSDATE() retournent la date et l'heure courantes. Dans une procédure ou un déclencheur SYSDATE est réévaluée en temps réel, alors que NOW désignera toujours l'instant de début de traitement.

Données binaires

Les types BLOB (*Binary Large Object*) permettent de stocker des données non structurées comme le multimédia (images, sons, vidéo, etc.). Les quatre types de colonnes BLOB sont TINYBLOB, BLOB, MEDIUMBLOB et LONGBLOB. Ces types sont traités comme des flots d'octets sans jeu de caractère associé.

Tableau 1-6 Types de données dates et heures

Type	Description	Commentaire pour une colonne
DATE	Dates du 1 ^{er} janvier de l'an 1000 au 31 décembre 9999 après J.-C.	Sur 3 octets. L'affichage est au format 'YYYY-MM-DD'.
DATETIME	Dates et heures (de 0 h de la première date à 23 h 59 minutes 59 secondes de la dernière date).	Sur 8 octets. L'affichage est au format 'YYYY-MM-DD HH:MM:SS'.
YEAR [(2 4)]	Sur 4 positions : de 1901 à 2155 (incluant 0000). Sur 2 positions (autorisé jusqu'en version 5.5) : de 70 à 69 désignant 1970 à 2069.	Sur 1 octet ; l'année est considérée sur 2 ou 4 positions (4 par défaut). Le format d'affichage est 'YYYY'.
TIME	Heures de -838 h 59 minutes 59 secondes à 838 h 59 minutes 59 secondes.	L'heure au format 'HHH:MM:SS' sur 3 octets.
TIMESTAMP	Instants du 1 ^{er} Janvier 1970 0 h 0 minute 0 seconde à l'année 2037.	Estampille sur 4 octets (au format 'YYYY-MM-DD HH:MM:SS') ; mise à jour à chaque modification sur la table.

Tableau 1-7 Types de données binaires

Type	Description	Commentaire pour une colonne
TINYBLOB (n)	Flot de n octets.	Taille fixe (maximum de 255 octets).
BLOB (n)		Taille fixe (maximum de 65 535 octets).
MEDIUMBLOB (n)		Taille fixe (maximum de 16 mégaoctets).
LOB (n)		Taille fixe (maximum de 4,29 gigaoctets).

Énumération

Deux types de collections sont proposés par MySQL.

- Le type ENUM définit une liste de valeurs permises (chaînes de caractères).
- Le type SET permettra de comparer une liste à une combinaison de valeurs permises à partir d'un ensemble de référence (chaînes de caractères).

Tableau 1-8 Types de données énumération

Type	Description
ENUM ('valeur1', 'valeur2', ...)	Liste de 65 535 valeurs au maximum.
SET ('valeur1', 'valeur2', ...)	Ensemble de référence (maximum de 64 valeurs).

Structure d'une table (DESCRIBE)

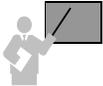
DESCRIBE (écriture autorisée DESC) est une commande qui vient de SQL*Plus d'Oracle et qui a été reprise par MySQL. Elle permet d'extraire la structure brute d'une table ou d'une vue.

■ **DESCRIBE** [*nomBase.*] *nomTableouVue* [*colonne*];

Si la base n'est pas indiquée, il s'agit de celle en cours d'utilisation. Retrouvons la structure des tables *Compagnie* et *Pilote* précédemment créées. Le type de chaque colonne apparaît :

Tableau 1-9 Structure brute des tables

Résultat						Commentaires
mysql> DESCRIBE bdbutil.Pilote;						Les clés primaires sont NOT NULL (désignées par PRI dans la colonne Key). Les unicités sont désignées par UNI dans la colonne Key. Les occurrences multiples possibles sont désignées par MUL dans la colonne Key. Les contraintes NOT NULL nommées (définies via les contraintes CHECK) n'apparaissent pas. La colonne Extra indique notamment les séquences (AUTO_INCREMENT).
Field	Type	Null	Key	Default	Extra	
brevet	varchar(6)	NO	PRI	NULL		
nom	varchar(30)	NO	UNI	NULL		
pseudo	varchar(8)	YES		NULL		
nbHVol	decimal(7,2)	YES		NULL		
compa	char(4)	YES	MUL	NULL		
mysql> DESCRIBE bdbutil.Compagnie;						
Field	Type	Null	Key	Default	Extra	
comp	char(4)	NO	PRI			
nrue	int(3)	YES		NULL		
rue	varchar(20)	YES		NULL		
ville	varchar(15)	YES		Paris		
nomComp	varchar(15)	NO				



La commande `SHOW CREATE TABLE [nom_base.]nom_table;` (voir le chapitre 5) restitue l'instruction complète qui a permis la création de la table en question. La commande `SHOW FULL COLUMNS FROM [nom_base.]nom_table;` est plus complète que l'instruction DESCRIBE.

Restrictions



Les contraintes CHECK définies ne sont pas encore opérationnelles.

La fraction de seconde du type TIME n'est pas encore pris en charge 'D HH:MM:SS.fraction'.

Les noms des colonnes doivent être uniques pour une table donnée (il est en revanche possible d'utiliser le même nom de colonne dans plusieurs tables).

Les colonnes de type SET sont évaluées par des chaînes de caractères séparés par des « , » ('Airbus, Boeing'). En conséquence aucune valeur d'un SET ne doit contenir le symbole « , ».

Les noms des objets (base, tables, colonnes, contraintes, vues, etc.) ne doivent pas emprunter des mots-clés de MySQL : TABLE, SELECT, INSERT, IF... Si vous êtes « franco-français » cela ne vous gênera pas.

Index

Comme l'index de cet ouvrage vous aide à atteindre les pages concernées par un mot recherché, un index MySQL permet d'accélérer l'accès aux données d'une table. Le but principal d'un index est d'éviter de parcourir une table séquentiellement du premier enregistrement jusqu'à celui visé (problème rencontré si c'est le Français nommé « Zidane » qu'on recherche dans une table non indexée de plus de soixante-six millions d'enregistrements...). Le principe d'un index est l'association de l'adresse de chaque enregistrement avec la valeur des colonnes indexées.

Sans index, tous les enregistrements de tous les blocs qui constituent la table seront systématiquement parcourus, et ce quelque soit la requête écrite (même celle qui doit retrouver une seule ligne de la table). Avec un index, quelques parcours de blocs peuvent suffire et ce nombre d'accès sera décorrélé de la montée en charge des enregistrements. En effet, si une table grossit d'un facteur 10000, le temps de traitement sera proportionnel sans index, alors qu'il n'augmentera éventuellement que d'un point de vue logarithmique (ici, schématiquement 4 blocs de plus parcourus au lieu de 10000 fois plus d'accès). Le parcours d'un index est dichotomique et le moyen le plus efficace de rechercher une valeur parmi un ensemble.

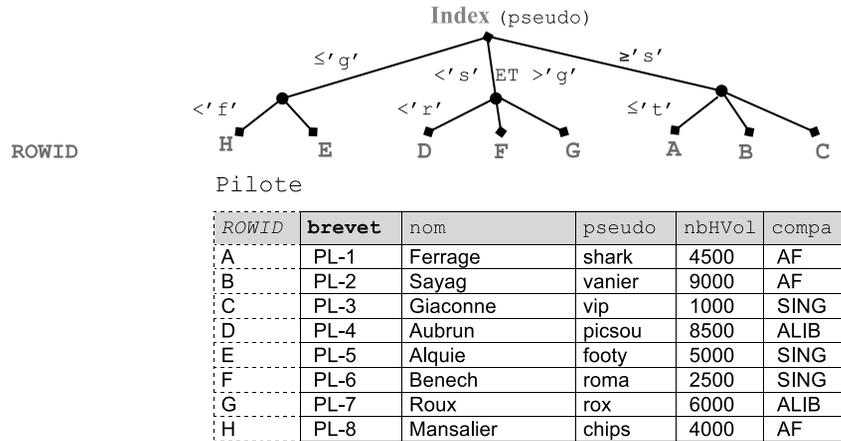
Arbres balancés

La figure suivante illustre un index unique sous la forme d'un arbre. Cet index est basé sur la colonne `pseudo` de la table `Pilote`. Dans cet exemple, trois accès à l'index seront nécessaires pour adresser directement un pilote via son `pseudo`, au lieu d'en analyser huit au plus. Plus il y aura de lignes dans la tables, plus l'index sera utile.

Un index est associé à une table et peut être défini sur une ou plusieurs colonnes (dites « indexées »). Une table peut « héberger » plusieurs index. Ils sont mis à jour automatiquement après rafraîchissement de la table (ajouts et suppressions d'enregistrements ou modification des colonnes indexées). Un index peut être déclaré unique si on sait que les valeurs des colonnes indexées seront toujours uniques.

La plupart des index de MySQL (`PRIMARY KEY`, `UNIQUE`, `INDEX`, et `FULLTEXT`) sont stockés dans des arbres équilibrés (*balanced trees* : *B-trees*). D'autres types d'index existent, citons ceux qui portent sur des colonnes `SPATIAL` (*reverse key* : *R-trees*), et ceux appliqués aux tables `MEMORY` (tables de hachage : *hash*).

Figure 1-3 Index sur la colonne nom



La particularité des index *B-tree* est qu'ils conservent en permanence une arborescence symétrique (balancée). Toutes les feuilles sont à la même profondeur. Le temps de recherche est ainsi à peu près constant quel que soit l'enregistrement cherché. Le plus bas niveau de l'index (*leaf blocks*) contient les valeurs des colonnes indexées et le *rowid*. Toutes les feuilles de l'index sont chaînées entre elles. Pour les index non uniques (par exemple si on voulait définir un index sur la colonne *compa* de la table *Pilote*) le *rowid* est inclus dans la valeur de la colonne indexée. Ces index, premiers apparus, sont désormais très fiables et performants, ils ne se dégradent pas lors de la montée en charge de la table.

Création d'un index (CREATE INDEX)

Pour pouvoir créer un index dans une base, la table à indexer doit appartenir à la base en question. Si l'utilisateur a le privilège `INDEX`, il peut créer et supprimer des index dans sa base. Un index est créé par l'instruction `CREATE INDEX` et supprimé par `DROP INDEX`.

La syntaxe de création d'un index est la suivante :

```
CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX nomIndex
  [USING BTREE | HASH]
  ON [nomBase.]nomTable (colonne1 [(taille1)] [ASC | DESC],...);
```

- `UNIQUE` permet de créer un index qui n'accepte pas les doublons.
- `FULLTEXT` permet de bénéficier de fonctions de recherche dans des textes (flot de caractères).
- `SPATIAL` permet de profiter de fonctions pour les données géographiques.

- ASC et DESC précisent l'ordre (croissant ou décroissant).

Créons deux index sur la table `Pilote`.

Tableau 1-10 Créations d'index

Instruction SQL	Commentaires
<pre>USE boutil; CREATE UNIQUE INDEX idx_Pilote_nom3 USING BTREE ON Pilote (nom(3) DESC);</pre>	Index <i>B-tree</i> , ordre décroissant sur les trois premiers caractères du nom des pilotes.
<pre>CREATE INDEX idx_Pilote_compa USING BTREE ON Pilote (compa);</pre>	Index <i>B-tree</i> , ordre croissant sur la colonne clé étrangère <code>compa</code> .

Bilan



- Un index ralentit les rafraîchissements de la base (conséquence de la mise à jour de l'arbre ou des *bitmaps*). En revanche il accélère les accès.
- Il est conseillé de créer des index sur des colonnes (majoritairement des clés étrangères) utilisées dans les clauses de jointures (voir chapitre 4).
- Il est possible de créer des index pour toutes les colonnes d'une table (jusqu'à concurrence de 16).
- Les index sont pénalisants lorsqu'ils sont définis sur des colonnes très souvent modifiées ou si la table contient peu de lignes.

Destruction d'un schéma

Il vous sera utile d'écrire un script de destruction d'un schéma (j'entends « schéma » comme ensemble de tables, contraintes et index composant une base de données et non pas en tant qu'ensemble de tous les objets d'un utilisateur) pour pouvoir recréer une base propre. Bien entendu, si des données sont déjà présentes dans les tables, et que vous souhaitez les garder, il faudra utiliser une stratégie pour les réimporter dans les nouvelles tables. À ce niveau de l'ouvrage, vous n'en êtes pas là, et le script de destruction va vous permettre de corriger vos erreurs de syntaxe du script de création des tables.

Nous avons vu qu'il fallait créer d'abord les tables « pères » puis les tables « fils » (si des contraintes sont définies en même temps que les tables). L'ordre de destruction des tables, pour des raisons de cohérence, est inverse (il faut détruire les tables « fils » puis les tables « pères »). Dans notre exemple, il serait malvenu de supprimer la table `Compagnie` avant la table `Pilote`. En effet la clé étrangère `compa` n'aurait plus de sens.

Suppression d'une table (DROP TABLE)

Pour pouvoir supprimer une table dans une base, il faut posséder le privilège DROP sur cette base. L'instruction DROP TABLE entraîne la suppression des données, de la structure, de la description dans le dictionnaire des données, des index, des déclencheurs associés (*triggers*) et la récupération de la place dans l'espace de stockage.

```
DROP [TEMPORARY] TABLE [IF EXISTS]
      [nomBase.] nomTable1 [, [nomBase2.] nomTable2, ...]
      [RESTRICT | CASCADE]
```

- **TEMPORARY** : pour supprimer des tables temporaires. Les transactions en cours ne sont pas affectées. L'utilisation de **TEMPORARY** peut être un bon moyen de s'assurer qu'on ne détruit pas accidentellement une table non temporaire...
- **IF EXISTS** : permet d'éviter qu'une erreur se produise si la table n'existe pas.
- **RESTRICT** (par défaut) permet de vérifier qu'aucun autre élément n'utilise la table (autre table via une clé étrangère, vue, déclencheur, etc.).
- **CASCADE**, option qui n'est pas encore opérationnelle, doit répercuter la destruction à toutes les autres tables référencées par des clés étrangères.

Les éléments qui utilisaient la table (vues, synonymes, fonctions et procédures) ne sont pas supprimés mais sont temporairement inopérants. Attention, une suppression ne peut pas être par la suite annulée.



Ordre des suppressions



Il suffit de relire à l'envers le script de création de vos tables pour en déduire l'ordre de suppression à écrire dans le script de destruction de votre schéma.

Le tableau suivant présente deux écritures possibles pour détruire le schéma contenant les tables *Pilote* et *Compagnie* et deux index. La première écriture ne se soucie pas de l'ordre des tables (il faut alors supprimer les clés étrangères, les index, puis les tables dans un ordre cohérent). La seconde écriture suit l'ordre des tables enfants, puis parents.

Tableau 1-11 Scripts de destruction

Sans ordre des tables	En respectant l'ordre des tables
DROP INDEX idx_Pilote_nom3 ON bduutil.Pilote;	
ALTER TABLE bduutil.Pilote	DROP TABLE bduutil.Pilote;
DROP FOREIGN KEY fk_Pil_compa_Comp;	DROP TABLE bduutil.Compagnie;
DROP INDEX idx_Pilote_compa	
ON bduutil.Pilote;	
DROP TABLE bduutil.Compagnie;	
DROP TABLE bduutil.Pilote;	

Exercices

L'objectif de ces exercices est de créer des tables, leur clé primaire et des contraintes de vérification (NOT NULL et CHECK).

Exercice 1.1 Présentation de la base de données

Une entreprise désire gérer son parc informatique à l'aide d'une base de données. Le bâtiment est composé de trois étages. Chaque étage possède son réseau (ou segment distinct) ethernet. Ces réseaux traversent des salles équipées de postes de travail. Un poste de travail est une machine sur laquelle sont installés certains logiciels. Quatre catégories de postes de travail sont recensées (stations Unix, terminaux X, PC Windows et PC NT). La base de données devra aussi décrire les installations de logiciels.

Les noms et types des colonnes sont les suivants :

Tableau 1-12 Caractéristiques des colonnes

Colonne	Commentaire	Type
indIP	trois premiers groupes IP (exemple : 130.120.80)	VARCHAR (11)
nomSegment	nom du segment	VARCHAR (20)
etage	étage du segment	TINYINT (1)
nSalle	numéro de la salle	VARCHAR (7)
nomSalle	nom de la salle	VARCHAR (20)
nbPoste	nombre de postes de travail dans la salle	TINYINT (2)
nPoste	code du poste de travail	VARCHAR (7)
nomPoste	nom du poste de travail	VARCHAR (20)
ad	dernier groupe de chiffres IP (exemple : 11)	VARCHAR (3)
typePoste	type du poste (UNIX, TX, PCWS, PCNT)	VARCHAR (9)
dateIns	date d'installation du logiciel sur le poste	dateTIME
nLog	code du logiciel	VARCHAR (5)
nomLog	nom du logiciel	VARCHAR (20)
dateAch	date d'achat du logiciel	dateTIME
version	version du logiciel	VARCHAR (7)
typeLog	type du logiciel (UNIX, TX, PCWS, PCNT)	VARCHAR (9)
prix	prix du logiciel	DECIMAL (6, 2)
numIns	numéro séquentiel des installations	INTEGER (5)
dateIns	date d'installation du logiciel	TIMESTAMP
delai	intervalle entre achat et installation	SMALLINT
typeLP	types des logiciels et des postes	VARCHAR (9)
nomType	noms des types (Terminaux X, PC Windows...)	VARCHAR (20)

Exercice 1.2 Création des tables

Écrire puis exécuter le script SQL (que vous appellerez `creParc.sql`) de création des tables avec leur clé primaire (en gras dans le schéma suivant) et les contraintes suivantes :

- Les noms des segments, des salles et des postes sont non nuls.
- Le domaine de valeurs de la colonne `ad` s'étend de 0 à 255.
- La colonne `prix` est supérieure ou égale à 0.
- La colonne `dateIns` est égale à la date du jour par défaut.

Figure 1-4 Composition des tables

Segment

indIP	nomSegment	etage
--------------	------------	-------

Salle

nSalle	nomSalle	nbPoste	indIP
---------------	----------	---------	-------

Poste

nPoste	nomPoste	indIP	ad	typePoste	nSalle
---------------	----------	-------	----	-----------	--------

Logiciel

nLog	nomLog	dateAch	version	typeLog	prix
-------------	--------	---------	---------	---------	------

Installer

nPoste	nLog	numIns	dateIns	delai
--------	------	---------------	---------	-------

Types

typeLP	nomType
---------------	---------

Exercice 1.3 Structure des tables

Écrire puis exécuter le script SQL (que vous appellerez `descParc.sql`) qui affiche la description de toutes ces tables (en utilisant des commandes `DESCRIBE`). Comparer le résultat obtenu avec le schéma ci-dessus.

Exercice 1.4 Destruction des tables

Écrire puis exécuter le script SQL de destruction des tables (que vous appellerez `dropParc.sql`). Lancer ce script puis celui de la création des tables à nouveau.