Chapitre 4
Construire une base de données dans MySQL

1. Créer et supprimer une base de données

L'ordre SQL `CREATE DATABASE` permet de créer une nouvelle base de données.

Syntaxe simplifiée

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] nom_base
```

`nom_base` est le nom de la nouvelle base de données. Ce nom doit respecter les règles de nommage des objets MySQL.

`CREATE SCHEMA` est un synonyme de `CREATE DATABASE`.

Une erreur se produit si une base de données de même nom existe déjà et que la clause `IF NOT EXISTS` n'est pas présente.

Pour créer une base de données, il faut le privilège global `CREATE`.

Physiquement, une base de données MySQL se matérialise par un répertoire qui contiendra les fichiers correspondant aux différentes tables de la base de données.

L'ordre SQL `CREATE DATABASE` propose plusieurs options qui permettent de spécifier le jeu de caractères et la collation par défaut de la base de données, ou de chiffrer la base de données (depuis la version 8.0.16).

Exemple

```
mysql> CREATE DATABASE biblio;
Query OK, 1 row affected (0.00 sec)
```

L'ordre SQL `DROP DATABASE` permet de supprimer une base de données.

Syntaxe

```
DROP {DATABASE | SCHEMA} [IF EXISTS] nom_base
```

`DROP SCHEMA` est un synonyme de `DROP DATABASE`.

Une erreur se produit si la base de données n'existe pas et que la clause `IF EXISTS` n'est pas présente.

Pour supprimer une base de données, il faut le privilège global `DROP`.

Remarque

*L'ordre **DROP DATABASE** supprime tout, sans demander de confirmation. Il faut y réfléchir à deux fois avant d'exécuter cette commande !*

2. Gérer les utilisateurs et les droits

2.1 Vue d'ensemble

Lors de l'installation de MySQL, un compte super-utilisateur nommé `root` est automatiquement créé.

Le compte `root` est normalement réservé à l'administration du serveur MySQL.

En complément du compte `root`, il est donc conseillé de créer au minimum un compte par application et éventuellement un compte par utilisateur final de l'application. De cette manière, il sera possible de gérer très finement les droits attribués à chaque utilisateur/application et de limiter les risques liés à l'utilisation du compte `root`.

Dans MySQL, un utilisateur est identifié de manière unique par la combinaison de deux informations :

– un nom d'utilisateur ;

– un nom d'hôte (ou adresse IP) à partir duquel l'utilisateur peut se connecter.

Chaque couple utilisateur/hôte est considéré par MySQL comme un utilisateur unique qui a un mot de passe pour se connecter (éventuellement aucun) et des droits. Un même utilisateur (au sens d'un nom d'utilisateur donné) peut donc avoir des droits différents selon l'hôte à partir duquel il se connecte.

La syntaxe utilisée pour désigner un utilisateur est donc la suivante :

```
nom_utilisateur[@nom_hôte]
```

© Editions ENI - All rights reserved

Pour le nom d'hôte, la valeur `'%'` signifie "n'importe quel hôte" ; c'est la valeur par défaut utilisée lorsque le nom d'hôte n'est pas spécifié. Le caractère `%` peut aussi être utilisé comme caractère joker dans le nom d'hôte ou l'adresse IP pour spécifier une liste de machine (`oheurtel@'%.olivier-heurtel.fr'`) ou une plage d'adresses IP (`oheurtel@'192.168.1.%'`). Le nom d'utilisateur peut être vide (utilisateur anonyme).

Il est possible d'avoir un utilisateur `nom_utilisateur@'%'` qui peut se connecter à partir de n'importe quelle machine avec certains droits et un "autre" utilisateur `nom_utilisateur@nom_hote` ayant le même nom, mais pouvant se connecter à partir d'une machine avec des droits différents (par exemple plus restrictifs si la machine est considérée comme peu sûre, ou moins restrictifs si la machine est considérée comme très sûre).

Les informations sur les utilisateurs et leurs droits sont stockées dans la base de données `mysql` :

Table	Contenu
`user`	Liste des utilisateurs avec leurs privilèges globaux (privilèges qui s'appliquent au serveur MySQL et à toutes les bases de données du serveur).
`db`	Liste des privilèges de niveau base de données attribués aux utilisateurs.
`tables_priv`, `columns_priv` et `procs_priv`	Liste des privilèges de niveau objet attribués aux utilisateurs.

Remarque

*Pour gérer les utilisateurs et les droits, il faut des privilèges globaux précis (**`CREATE USER`**, **`GRANT OPTION`**, etc.). Par défaut, ces privilèges sont attribués au compte **`root`**, puisque ce dernier a tous les droits. Dans la suite, nous supposerons que la gestion des utilisateurs et des droits est effectuée à l'aide du compte **`root`** et nous ne préciserons pas quel droit est requis pour effectuer telle action. Pour en savoir plus sur ce sujet, reportez-vous à la documentation MySQL.*

2.2 Gérer les utilisateurs

2.2.1 Créer des utilisateurs

Avant la version 5.0.2, un nouvel utilisateur était créé implicitement par attribution d'un premier droit à l'aide de l'ordre SQL `GRANT` (voir ci-après). Depuis la version 5.7.6, cette fonctionnalité était dépréciée et a été supprimée en version 8.0.11. Pour gérer les utilisateurs, il faut utiliser les ordres SQL `CREATE USER` et `ALTER USER`.

L'ordre SQL `CREATE USER` permet de créer explicitement un utilisateur.

Syntaxe simplifiée

```
CREATE [IF NOT EXISTS] USER spécification_utilisateur [,...]
spécification_utilisateur =
nom_utilisateur[@nom_hôte] [IDENTIFIED [WITH plugin] BY 'mot_de_passe']
```

`nom_utilisateur` et `mot_de_passe` sont respectivement le nom et le mot de passe du nouveau compte. Si la clause `IDENTIFIED BY` est absente, le compte est créé sans mot de passe.

La clause `WITH` permet de spécifier un plugin d'authentification pour la vérification du mot de passe. Les valeurs possibles sont : `mysql_native_password`, `sha256_password` et `caching_sha2_password` (valeur par défaut en version 8, sauf configuration différente du serveur). Le plugin d'authentification `caching_sha2_password` utilisé par défaut peut poser des problèmes de compatibilité avec certains clients qui ne supportent pas cette méthode, ce qui est le cas de l'extension `mysqli` de PHP utilisée pour accéder à MySQL à partir de PHP. Pour résoudre ce problème, il faut utiliser l'ancien plugin `mysql_native_password` lors de la création des utilisateurs concernés (ou modifier la configuration du serveur pour utiliser cette méthode par défaut pour tous les utilisateurs, à l'aide de la variable système `default_authentication_plugin` ou de la variable `authentication_policy` à partir de la version 8.0.27).

`nom_hôte` permet de spécifier le nom de la machine à partir de laquelle le nouvel utilisateur peut se connecter. Si cette clause est omise, la valeur par défaut `'%'` est utilisée ; dans ce cas, l'utilisateur peut se connecter à partir de n'importe quel hôte.

Si le compte existe déjà, une erreur se produit, sauf si la clause `IF NOT EXISTS` est présente, auquel cas une simple alerte est générée ; cette clause est apparue en version 5.7.8.

Le mot de passe doit être saisi en clair ; il sera automatiquement chiffré (haché) par MySQL avant d'être stocké dans la base `mysql`.

© Editions ENI - All rights reserved

Dans les versions précédentes, le mot-clé `PASSWORD` pouvait être utilisé pour spécifier un mot de passe déjà chiffré (nombre hexadécimal de 41 chiffres). Depuis la version 5.7.6, l'utilisation de ce mot-clé était dépréciée et a été supprimée en version 8.0.11 ; une erreur est générée si cette option est utilisée.

D'autres méthodes d'authentification peuvent être utilisées à la place de la méthode par défaut (voir la documentation de MySQL à ce sujet).

Exemple

```
mysql> CREATE USER eniadm IDENTIFIED BY 'eni';
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE USER oheurtel@localhost IDENTIFIED BY 'oh';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT host,user,plugin
    -> FROM mysql.user WHERE user IN ('eniadm','oheurtel');
+-----------+----------+-----------------------+
| host      | user     | plugin                |
+-----------+----------+-----------------------+
| %         | eniadm   | caching_sha2_password |
| localhost | oheurtel | caching_sha2_password |
+-----------+----------+-----------------------+
2 rows in set (0.00 sec)

mysql> -- "Deuxième" utilisateur "eniadm"
mysql> CREATE USER eniadm@localhost IDENTIFIED BY 'eni';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT host,user,plugin
    -> FROM mysql.user WHERE user = 'eniadm';
+------------+--------+-----------------------+
| host       | user   | authentication_string |
+------------+--------+-----------------------+
| %          | eniadm | caching_sha2_password |
| localhost  | eniadm | caching_sha2_password |
+------------+--------+-----------------------+
2 rows in set (0.00 sec)
```

Ce "deuxième" utilisateur aurait pu être créé avec un mot de passe différent (éventuellement sans mot de passe).

Dans le script `creer-base-eni.sql` utilisé pour générer la base de données de cet ouvrage, un utilisateur `eniweb` est créé à l'aide de l'ordre SQL suivant :

```
CREATE USER eniweb@localhost IDENTIFIED WITH mysql_native_password BY 'web';
```

Le plugin d'authentification `mysql_native_password` est explicitement spécifié pour cet utilisateur afin qu'il puisse se connecter sans problème à partir de PHP :

```
mysql> SELECT host,user,plugin FROM mysql.user WHERE user = 'eniweb';
+-----------+--------+-----------------------+
| host      | user   | plugin                |
+-----------+--------+-----------------------+
| localhost | eniweb | mysql_native_password |
+-----------+--------+-----------------------+
1 row in set (0.00 sec)
```

Depuis la version 5.7.8, l'ordre SQL `CREATE USER` propose des clauses supplémentaires optionnelles qui permettent :

– de limiter les ressources utilisées par le compte (nombre de requêtes, nombre de connexions, etc.) ;
– de gérer l'expiration du mot de passe ;
– de créer le compte verrouillé (un compte peut être verrouillé/déverrouillé par l'intermédiaire de l'ordre SQL `ALTER USER`) ;
– de chiffrer la connexion entre le client et le serveur (SSL/TLS).

Depuis la version 8, des clauses supplémentaires permettent :

– de définir les rôles de l'utilisateur qui seront activés par défaut lors de sa connexion (version 8.0.0) ;
– de spécifier des règles relatives à la réutilisation des anciens mots de passe (version 8.0.3) ;
– d'indiquer que l'utilisateur devra saisir son mot de passe actuel lors d'un changement de mot de passe (version 8.0.13) ;
– de générer un mot de passe aléatoire pour l'utilisateur (version 8.0.18) ;
– de définir une règle de verrouillage du compte au bout d'un certain nombre d'échecs de connexion consécutifs (version 8.0.19) ;
– d'associer un commentaire et des attributs à un utilisateur (version 8.0.21) ;
– d'utiliser jusqu'à trois méthodes d'authentification pour un utilisateur (version 8.0.27).

Vous pouvez consulter la documentation pour en savoir plus sur ces différentes fonctionnalités.

© Editions ENI - All rights reserved

2.2.2 Supprimer des utilisateurs

L'ordre SQL `DROP USER` permet de supprimer explicitement un utilisateur.

Syntaxe

```
DROP USER [IF EXISTS] nom_utilisateur[@nom_hôte[,...]
```

Si le nom d'hôte est omis, la valeur par défaut `'%'` est utilisée.

Si le compte n'existe pas, une erreur se produit, sauf si la clause `IF NOT EXISTS` est présente, auquel cas une simple alerte est générée ; cette clause est apparue en version 5.7.8.

Lors de la suppression d'un utilisateur, les droits de l'utilisateur sont aussi supprimés ; par contre, les objets éventuels créés par l'utilisateur ne sont pas supprimés.

2.2.3 Modifier le mot de passe des utilisateurs

L'ordre SQL `SET PASSWORD` permet de modifier le mot de passe d'un utilisateur.

Syntaxe

```
SET PASSWORD [FOR nom_utilisateur[@nom_hôte]] =
      'nouveau_mot_de_passe'
      [REPLACE 'ancien_mot_de_passe']
      [RETAIN CURRENT PASSWORD]
```

Si le nom d'hôte est omis, la valeur par défaut `'%'` est utilisée.

Si la clause `FOR` est omise, cette commande permet de modifier le mot de passe de l'utilisateur courant.

Depuis la version 5.7.6, le nouveau mot de passe est saisi directement en clair.

Dans les versions antérieures, la fonction `PASSWORD` pouvait être utilisée pour chiffrer (hacher, plus précisément) le nouveau mot de passe saisi en clair avant son stockage dans la base de données `mysql`. Cette syntaxe était dépréciée depuis la version 5.7.6 et a été supprimée en version 8.0.11.

La clause `REPLACE` permet d'indiquer le mot de passe actuel. Elle est utilisable uniquement lors de la modification du mot de passe de l'utilisateur courant et doit être présente si le compte utilisateur a été créé avec l'obligation de saisir le mot de passe actuel lors de la modification du mot de passe. Cette fonctionnalité est apparue en version 8.0.13.

La clause `RETAIN CURRENT PASSWORD` permet de conserver l'ancien mot de passe, qui peut encore être utilisé en mot de passe secondaire. Cette possibilité est intéressante s'il faut du temps pour modifier les applications ou pour propager le nouveau mot de passe dans d'autres bases de données (réplication). Ultérieurement, ce mot de passe secondaire peut être supprimé en utilisant la clause `DISCARD OLD PASSWORD` de l'ordre SQL `ALTER USER`. Cette fonctionnalité est apparue en version 8.0.13.