Capítulo 4
Construir una base de datos en MySQL

1. Crear y eliminar una base de datos

La sentencia SQL `CREATE DATABASE` permite crear una nueva base de datos.

Sintaxis simplificada

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

`nombre_base` es el nombre de la nueva base de datos. Este nombre debe respetar las reglas para los nombres de objetos MySQL.

`CREATE SCHEMA` es equivalente a `CREATE DATABASE`.

Se produce un error si ya existe una base de datos con el mismo nombre y la cláusula `IF NOT EXISTS` no está presente.

Para crear una base de datos, es necesario el privilegio global `CREATE`.

Físicamente, una base de datos MySQL se materializa en un directorio que contiene los archivos correspondientes a las diferentes tablas de la base de datos.

La sentencia SQL `CREATE DATABASE` ofrece varias opciones que permiten especificar el juego de caracteres y la intercalación por defecto de la base de datos, o cifrar la base de datos (desde la versión 8.0.16).

Ejemplo

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

La sentencia SQL `DROP DATABASE` permite eliminar una base de datos.

Sintaxis

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

`DROP SCHEMA` es equivalente a `DROP DATABASE`.

Se produce un error si no existe la base de datos y la cláusula `IF EXISTS` no está presente.

Para eliminar una base de datos, es necesario el privilegio global `DROP`.

Observación

La sentencia `DROP DATABASE` elimina todo, sin solicitar confirmación. ¡Hay que pensarlo dos veces antes de ejecutar este comando!

2. Gestionar los usuarios y los privilegios

2.1 Visión de conjunto

En la instalación de MySQL, se crea automáticamente una cuenta de superusuario con el nombre `root`.

La cuenta `root` se reserva normalmente a la administración del servidor MySQL.

Como complemento a la cuenta `root`, es recomendable crear como mínimo una cuenta por aplicación y, si es necesario, una cuenta por usuario final de la aplicación. De esta manera, será posible gestionar con mayor precisión los privilegios otorgados a cada usuario/aplicación y limitar los riesgos ligados a la utilización de la cuenta `root`.

En MySQL, un usuario está identificado de manera única por la combinación de dos datos:

- un nombre de usuario;
- un nombre de host (o dirección IP) a partir del cual el usuario puede conectarse.

Cada pareja de usuario/host es considerada por MySQL como un usuario único que tiene una contraseña para conectarse (en algún caso ninguna) y privilegios. Un mismo usuario (en el sentido de un nombre de usuario dado) puede tener privilegios diferentes según el host a partir del cual se conecte.

Por lo tanto, la sintaxis utilizada para designar a un usuario es la siguiente:

```
nombre_usuario[@nombre_host]
```

Para el nombre de host, el valor `'%'` significa «cualquier host»; es el valor predeterminado cuando no se especifica el nombre del host. El signo `%` también puede utilizarse como carácter comodín en el nombre del host o la dirección IP para especificar una lista de equipo (`oheurtel@'%.olivier-heurtel.fr'`) o un intervalo de direcciones IP (`oheurtel@'192.168.1.%'`). El nombre de usuario puede estar vacío (usuario anónimo).

Es posible tener un usuario `nombre_usuario@'%'` que puede conectarse a partir de cualquier equipo con determinados privilegios y otro usuario `nombre_usuario@nombre_host` con el mismo nombre, pero que puede conectarse a partir de un equipo con privilegios diferentes (por ejemplo, más restrictivos si el equipo es considerado como poco seguro, o menos restrictivos si el equipo es considerado como muy seguro.

Los datos sobre los usuarios y sus derechos se almacenan en la base de datos `mysql`:

Tabla	Contenido
`user`	Lista de los usuarios con sus privilegios globales (privilegios que se aplican al servidor MySQL y a todas las bases de datos del servidor).
`db`	Lista de los privilegios de nivel base de datos otorgados a los usuarios.
`tables_priv`, `columns_priv` y `procs_priv`	Lista de los privilegios de nivel objeto otorgados a los usuarios.

Observación

Para gestionar a los usuarios y los privilegios, son necesarios privilegios globales precisos (`CREATE USER`, `GRANT OPTION`, etc.). De manera predeterminada, estos privilegios se otorgan a la cuenta `root`, puesto que esta última tiene todos los privilegios. En adelante, daremos por supuesto que la gestión de los usuarios y de los privilegios se efectúa mediante la cuenta `root` y no precisaremos qué privilegio se requiere para realizar cada acción. Para saber más, consulte la documentación de MySQL.

2.2 Gestionar los usuarios

2.2.1 Crear usuarios

Antes de la versión 5.0.2, se creaba un nuevo usuario implícitamente por concesión de un primer privilegio mediante la sentencia `GRANT` (véase más adelante). Desde la versión 5.7.6, esta funcionalidad ha quedado obsoleta y se ha eliminado en la versión 8.0.11. Para administrar usuarios, hay que utilizar las sentencias SQL `CREATE USER` y `ALTER USER`.

La sentencia `CREATE USER` permite crear explícitamente un usuario.

Sintaxis simplificada

```
CREATE [IF NOT EXISTS] USER especificación_usuario [,...]
especificación_usuario =
nombre_usuario[@nombre_host] [IDENTIFIED [WITH Plugin] BY 'contraseña']
```

`nombre_usuario` y `contraseña` son respectivamente el nombre y la contraseña de la nueva cuenta. Si la cláusula `IDENTIFIED BY` está ausente, la cuenta se crea sin contraseña.

La cláusula `WITH` permite especificar un plugin de autenticación para la comprobación de la contraseña. Los valores posibles son: `mysql_native_password`, `sha256_password y caching_sha2_password` (valor por defecto en la versión 8, salvo que el servidor se haya configurado de manera distinta). El plugin de autenticación por defecto `caching_sha2_password` puede causar problemas de compatibilidad con algunos clientes que no soporten este método, como ocurre con la extensión `mysqli` de PHP que se utiliza para acceder a MySQL desde PHP. Para resolver este problema, hay que utilizar el antiguo plugin `mysql_native_password` cuando se crean los usuarios en cuestión (o modificar la configuración del servidor para utilizar este método por defecto para todos los usuarios, con ayuda de la variable de sistema `default_authentication_plugin` o de la variable `authentication_policy` a partir de la versión 8.0.27).

`nombre_host` permite especificar el nombre del equipo desde el cual se puede conectar el nuevo usuario. Si se omite esta cláusula, se utiliza el valor predeterminado `'%'`; en ese caso, el usuario puede conectarse desde cualquier host.

Si la cuenta ya existe, se produce un error, excepto si existe la cláusula `IF NOT EXISTS`, caso en el que solo se genera una simple alerta; está cláusula apareció en la versión 5.7.8.

La contraseña debe introducirse en claro; se cifrará automáticamente (hash) por MySQL antes de almacenarse en la base `mysql`.

En las versiones anteriores, la palabra clave `PASSWORD` se podía utilizar para especificar una contraseña ya cifrada (número hexadecimal de 41 cifras). Desde la versión 5.7.6, el uso de esta palabra clave quedó obsoleta y se eliminó en la versión 8.0.11; si se usa esta opción, se genera un error.

Se pueden utilizar otros métodos de autenticación en lugar del método predeterminado (véase la documentación de MySQL a este respecto).

Ejemplo

```
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> -- "Segundo usuario: "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)
```

Este «segundo» usuario podría haber sido creado con una contraseña diferente (incluso sin contraseña).

En el script `crear-base-eni.sql` utilizado para generar la base de datos de este libro, se crea un usuario `eniweb` con la sentencia SQL siguiente:

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

Para este usuario, se especifica de forma explícita el plugin de autenticación `mysql_native_password` para que pueda conectarse sin problemas desde 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)
```

Desde la versión 5.7.8, la sentencia SQL `CREATE USER` ofrece casos adicionales opcionales que permiten:

- limitar los recursos utilizados por la cuenta (número de consultas, número de conexiones, etc.);
- gestionar la expiración de la contraseña;
- crear la cuenta bloqueada (se puede bloquear/desbloquear una cuenta por medio de la sentencia SQL `ALTER USER`);
- cifrar la conexión entre el cliente y el servidor (SSL/TLS).

Desde la versión 8, hay cláusulas adicionales que permiten:

- definir los roles del usuario que se activarán por defecto durante su conexión (versión 8.0.0);
- especificar las reglas relativas a la reutilización de las antiguas contraseñas (versión 8.0.3);
- indicar que el usuario deberá introducir su contraseña actual durante un cambio de contraseña (versión 8.0.13);
- generar una contraseña aleatoria para el usuario (versión 8.0.18);
- definir una regla de bloqueo de la cuenta al cabo de cierto número de conexiones consecutivas (versión 8.0.19);
- asociar un comentario y unos atributos a un usuario (versión 8.0.21);
- utilizar hasta tres métodos de autenticación para un usuario (versión 8.0.27).

Puede consultar la documentación para saber más acerca de estas funcionalidades.

2.2.2 Eliminar usuarios

La sentencia `DROP USER` permite eliminar explícitamente un usuario.

Sintaxis

```
DROP USER [IF EXISTS] nombre_usuario[@nombre_host[,...]
```

Si se omite el nombre del host, se toma el valor predeterminado `'%'`.

Si la cuenta no existe, se produce un error, salvo si se usa la cláusula `IF NOT EXISTS`, caso en el que solo se genera una simple alerta; esta cláusula apareció en la versión 5.7.8.

En la eliminación de un usuario, los privilegios del usuario también se eliminan; en cambio, los objetos creados por el usuario, si los hay, no son eliminados.

2.2.3 Modificar la contraseña de los usuarios

La sentencia SQL `SET PASSWORD` permite modificar la contraseña de un usuario.

Sintaxis

```
SET PASSWORD [FOR nombre_usuario[@nombre_host]] =
        'nueva_contraseña'
        [REPLACE 'antigua_contraseña']
        [RETAIN CURRENT PASSWORD]
```

Si se omite el nombre del host, se toma el valor predeterminado `'%'`.

Si se omite la cláusula `FOR`, este comando permite modificar la contraseña del usuario actual.

Desde la versión 5.7.6, la nueva contraseña se introduce directamente sin cifrar.

En las versiones anteriores, se podía utilizar la función `PASSWORD` para cifrar (*hash*, de manera más precisa) la nueva contraseña introducida sin cifrar, antes de almacenarla en la base de datos `mysql`. Esta sintaxis está obsoleta desde la versión 5.7.6 y se ha eliminado en la versión 8.0.11.

La cláusula `REPLACE` permite indicar la contraseña actual. Se usa solo durante la modificación de la contraseña del usuario actual y debe estar presente si la cuenta del usuario se ha creado con la obligación de introducir la contraseña actual durante de la modificación de la contraseña. Esta funcionalidad apareció en la versión 8.0.13.

La cláusula `RETAIN CURRENT PASSWORD` permite conservar la contraseña antigua, que todavía se puede usar como contraseña secundaria. Esta posibilidad es interesante si lleva tiempo modificar las aplicaciones o para propagar la nueva contraseña al resto de bases de datos (replicación). Posteriormente, esta contraseña secundaria se puede eliminar usando la cláusula `DISCARD OLD PASSWORD` de la sentencia SQL `ALTER USER`. Esta funcionalidad apareció en la versión 8.0.13.

La sentencia SQL `SET PASSWORD` no está obsoleta, pero se desaconseja; en su lugar, se recomienda usar la sentencia SQL `ALTER USER`.

Sintaxis

```
ALTER USER nombre_usuario [@nombre_host]
IDENTIFIED BY 'nueva_contraseña'
[REPLACE ‘antigua_contraseña’]
[RETAIN CURRENT PASSWORD]
```

Las cláusulas son las mismas que las de la sentencia SQL `SET PASSWORD`.

Ejemplo

```
mysql> SET PASSWORD FOR eniadm = 'secret';
Query OK, 0 rows affected (0.00 sec)

mysql> SET PASSWORD FOR eniadm@localhost = 'secret';
Query OK, 0 rows affected (0.00 sec)

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

Observación

La sentencia SQL ALTER USER permite modificar otras propiedades de la cuenta.

2.3 Administrar los privilegios de los usuarios

2.3.1 Atribuir privilegios a los usuarios

La sentencia SQL `GRANT` permite conceder privilegios a los usuarios.

Sintaxis simplificada

```
GRANT privilegio,[,...]
ON especificación_meta
TO especificación_usuario [,...]
especificación_meta =
    *.*
  | [nombre_base.]*
  | [TABLE | FUNCTION | PROCEDURE] [nombre_base.]nombre_objeto
especificación_usuario =
nombre_usuario[@nombre_host] [IDENTIFIED BY [PASSWORD] 'contraseña']
```

La cláusula *`especificación_usuario`* es la misma que para la sentencia `CREATE USER`.

Desde la versión 8.0.11, ya no es posible crear un usuario con la sentencia SQL `GRANT` (posibilidad quedó obsoleta desde la versión 5.7.7).

Los privilegios pueden otorgarse en tres niveles: global, base de datos y objeto.

Global

Los privilegios otorgados globalmente se aplican en todas las bases de datos del servidor. Para otorgar un privilegio a nivel global, es necesario utilizar el valor `*.*` para la cláusula *`especificación_meta`*.