UNIDAD DIDÁCTICA 15. ADMINISTRACIÓN GENERAL DE MYSQL. CÓMO CREAR UN NUEVO USUARIO Y OTORGAR PERMISOS EN MYSQL

Introducción

MySQL es un software de administración de bases de datos de código abierto que ayuda a los usuarios a almacenar, organizar y recuperar datos. Tiene diversas opciones para otorgar permisos matizados a usuarios específicos en las tablas y bases de datos. Este tutorial dará una breve descripción general de algunas de esas opciones.

Cómo crear un nuevo usuario

Cuando empezamos a utilizar MySQL, recibimos un nombre de usuario y una contraseña. Estas credenciales iniciales nos otorgan acceso **root** o control total de todas las bases de datos y tablas.

Sin embargo, hay ocasiones en las que deberemos otorgar acceso a la base de datos a otras personas sin otorgarle el control total. En estos casos, debemos darles las credenciales de un usuario no **root**. De esta manera, podemos realizar un seguimiento de lo que los usuarios pueden y no pueden hacer con los datos.

Para crear una nueva cuenta de usuario en MySQL, sigue estos pasos debemos utilizar la sentencia ***CREATE USER***:

```
CREATE USER [IF NOT EXISTS]
    user [auth_option] [, user [auth_option]] ...
    DEFAULT ROLE role [, role ] ...
    [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
    [WITH resource_option [resource_option] ...]
    [password_option | lock_option] ...
    [COMMENT 'comment_string' | ATTRIBUTE 'json_object']

user:
    (see Section 6.2.4, “Specifying Account Names”)

auth_option: {
    IDENTIFIED BY 'auth_string'
  | IDENTIFIED BY RANDOM PASSWORD
  | IDENTIFIED WITH auth_plugin
  | IDENTIFIED WITH auth_plugin BY 'auth_string'
  | IDENTIFIED WITH auth_plugin BY RANDOM PASSWORD
  | IDENTIFIED WITH auth_plugin AS 'auth_string'
}

tls_option: {
   SSL
 | X509
 | CIPHER 'cipher'
 | ISSUER 'issuer'
 | SUBJECT 'subject'
}

resource_option: {
    MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
```

```
  | MAX_USER_CONNECTIONS count
}

password_option: {
    PASSWORD EXPIRE [DEFAULT | NEVER | INTERVAL N DAY]
  | PASSWORD HISTORY {DEFAULT | N}
  | PASSWORD REUSE INTERVAL {DEFAULT | N DAY}
  | PASSWORD REQUIRE CURRENT [DEFAULT | OPTIONAL]
  | FAILED_LOGIN_ATTEMPTS N
  | PASSWORD_LOCK_TIME {N | UNBOUNDED}
}

lock_option: {
    ACCOUNT LOCK
  | ACCOUNT UNLOCK
}
```

https://dev.mysql.com/doc/refman/8.0/en/create-user.html

```
CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';
```

En este momento, newuser no tiene permisos para hacer nada con las bases de datos. De hecho, incluso si newuser intenta iniciar sesión (con la contraseña, password), no podrá acceder al shell de MySQL.

Ejemplo:

```
CREATE USER 'bea'@'localhost' IDENTIFIED BY 'bea';
```

```
SELECT * FROM mysql.user;
```

Si creamos una conexión para este usuario, al abrirla comprobamos quew no tiene acceso a ninguna BBDD.

Por lo tanto, lo primero que se debe hacer es proporcionar al usuario acceso a la información que necesite.

ASIGNAR PERMISOS

Niveles de privilegios

En **MySQL** existen cinco niveles distintos de privilegios:

1. **Globales:** se aplican al conjunto de todas las bases de datos en un servidor. Es el nivel más alto de privilegio, en el sentido de que su ámbito es el más general. Los privilegios globales son los más potentes, ya que se aplican a cualquier base de datos.
2. **De base de datos:** se refieren a bases de datos individuales, y por extensión, a todos los objetos que contiene cada base de datos.
3. **De tabla:** se aplican a tablas individuales, y por lo tanto, a todas las columnas de esas tabla.
4. **De columna:** se aplican a una columna en una tabla concreta.
5. **De rutina:** se aplican a los procedimientos almacenados, consistentes en varias consultas SQL.

Configuración Inicial de la tabla de autorizaciones

La tabla de autorizaciones se configura durante el proceso de instalación de MyQL, con dos tipos de cuentas:

1. Cuentas con nombre de usuario **root**, son cuentas de superusuario pensadas para labores administrativas. Tienen todos los privilegios y pueden hacer cualquier cosa.
2. Cuentas no asociadas a ningún nombre de usuario, son cuentas "**anónimas**". Permiten que la gente se conecte con el servidor sin disponer de una cuenta explícita configuradas por ellos en el servidor. Los usuarios anónimos generalmente tienen pocos privilegios.

Todas las cuentas disponibles en un servidor MySQL se listan en la tabla **user** de la base de datos **mysql**, de forma, que ahí se pueden encontrar las cuentas iniciales root y anónima.

```
USE mysql;
SELECT * FROM user;
```

Cada entrada (registro) en la tabla **user** contiene un valor **Host** que indica donde puede conectar un usuario y un valor de **user** y **Password** que indica el nombre y la contraseña que debe proporcionar el usuario cuando se conecte a dicho **Host**. La tabla **user** también tiene varias columnas que indican los privilegios que tiene cada cuenta.

```
SELECT host, user, password FROM user;
```

Un administrador MySQL debe saber cómo configurar cuentas de usuario especificando que usuarios pueden conectar con el servidor, desde donde pueden conectarse y qué es lo que pueden hacer una vez que se hayan conectado. Esta información se almacena en las tablas de autorizaciones en la base de datos **mysql** y se controla mediante las siguientes sentencias:

GRANT: crear cuentas MySQL y especificar sus privilegios.

REVOKE: eliminar los privilegios de una cuenta MySQL existente.

La sintaxis de GRANT es:

```
GRANT [lista_privilegios] ON [nombre_BBDD].[nombre_tabla]
    TO [nombre_usuario]
```

```
GRANT ALL PRIVILEGES ON * . * TO 'newuser'@'localhost';
```

Ejemplo: Otorgar todos los privilegios al usuario bea

```
GRANT ALL PRIVILEGES ON * * TO 'bea'@'localhost';
```

Los asteriscos en este comando se refieren a la base de datos y la tabla (respectivamente) a los que pueden acceder. Este comando específico permite al usuario leer, editar, ejecutar y realizar todas las tareas en todas las bases de datos y tablas.

En este ejemplo estamos otorgando a **bea** acceso de **root** completo a todas nuestras base de datos.

Una vez que hayamos finalizado la otorgación de los permisos que deseamos configurar para los nuevos usuarios, tenemos siempre que volver a cargar los privilegios.

```
FLUSH PRIVILEGES;
```

Cómo otorgar diferentes permisos de usuario

Algunas de estas cláusulas son opcionales y en general los que más se utilizan son:

lista_privilegios: son los privilegios a asignar a la cuenta. Los más usuales son:

- **ALL PRIVILEGES**: asigna todos los privilegios y permite acceder a todas las Bases de datos.
- **CREATE**: permite crear nuevas tablas o bases de datos
- **DROP**: permite eliminar nuevas tablas o bases de datos
- **DELETE**: permite eliminar registros de las tablas
- **INSERT**: permite insertar registros en las tablas
- **SELECT**: permite leer registros de las tablas
- **UPDATE**: permite actualizar los registros de las tablas
- **GRANT OPTION**: permite otorgar o eliminar privilegios de otros usuarios

[nombre_BBDD].[nombre_tabla]: indica el nivel al que se aplican los privilegios. El nivel más potente es el nivel global (*.*), para él los privilegios se aplican a todas las bases de datos y tablas. Los privilegios globales se pueden considerar privilegios de superusuario, los privilegios también pueden ser específicos de una base de datos, una tabla o una columna.

TO [nombre_usuario]: indica la cuenta para la que se establecen los privilegios.

El valor de **user** es un nombre de usuario y un nombre de **host** con el formato: 'nombre_usuario'@'nombre_host', ya que hay que indicar quien se puede conectar y desde donde.

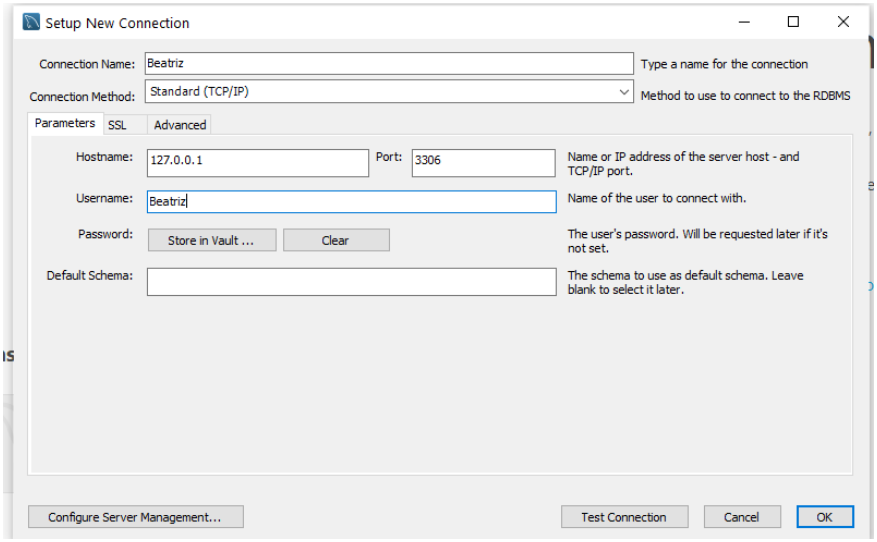
Esto permite configurar cuentas independientes para dos usuarios con el mismo nombre pero que se conectan desde ubicaciones diferentes. MySQL nos permite distinguirlos y otorgarles privilegios diferentes.

Cada vez que actualizamos o cambiamos un permiso, tenemos que utilizar el comando **Flush Privileges**.

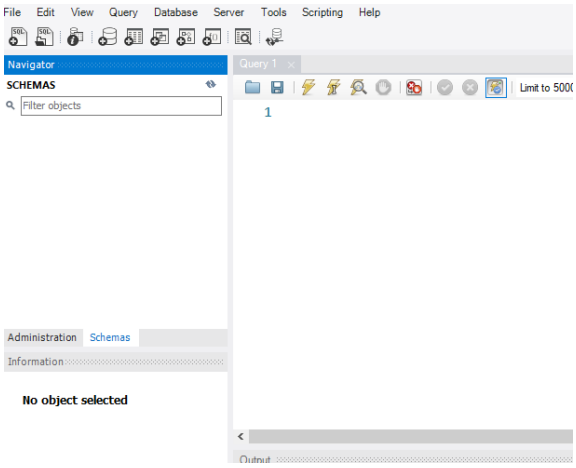
Ejemplo: Crear un usuario 'Beatriz' con todos los privilegios para la base de datos Ejemplo desde localhost y con contraseña 'Olalde'

```
CREATE USER'Beatriz'@'localhost' IDENTIFIED BY 'Olalde';

GRANT ALL ON Ejemplo.* TO 'Beatriz'@'localhost';

##Para comprobar los privilegios de Beatriz

SHOW GRANTS FOR 'Beatriz'@'localhost';
```

Para comprobar si el Nuevo usuario se ha creado correctamente, creamos una nueva conexión para el nuevo usuario y contraseña.



Como la BBDD Ejemplo no existe se muestra la pantalla vacía.




```
CREATE DATABASE Ejemplo;
(Crea la BBDD correctamente)
CREATE DATABASE Ejemplo1;
(Produce un error no tenemos permiso)
```

Ejemplo: Crear un usuario 'Manolo' con todos los privilegios para la base de datos Ejemplo que se va a conectar desde ares.mars.net y con contraseña 'Benito'

```
CREATE USER 'Manolo'@'ares.mars.net' IDENTIFIED BY 'Benito';
GRANT ALL ON Ejemplo.* TO 'Manolo'@'ares.mars.net' ;
SHOW GRANTS FOR 'Manolo'@'ares.mars.net';
FLUSH PRIVILEGES;
```

El nombre del host es desde donde se conecta el cliente, no es el servidor al que se conecta el cliente, a no ser que sea el mismo. El nombre de usuario y el del host se entrecomillan por separado.

Si queremos que un usuario se conecte desde cualquier host podemos utilizar el carácter comodín (%). Es una forma fácil de configurar un usuario pero es poco segura. El carácter comodín subrayado (_) se utiliza en valores de host para hacer coincidir cualquier carácter individual.

Ejemplo: Crear un usuario Miguel con todos los privilegios para la base de datos aaEmpresasAbc01 que se va a conectar desde cualquier cliente y con contraseña ‘Gonzalez

```
CREATE USER 'Miguel'@'%' IDENTIFIED BY 'Gonzalez';

GRANT ALL ON Ejemplo.* TO 'Miguel'@'%';

SHOW GRANTS FOR 'Miguel'@'%';

FLUSH PRIVILEGES;
```

Para comprobar si el Nuevo usuario se ha creado correctamente podemos crear una conexión .

```
GRANT ALL ON aaempresasabc01.Empleados TO 'Miguel'@'localhost' ;
```

Existe un planteamiento intermedio que nos permitirá que un usuario se conecte desde cualquier conjunto limitado de hosts.

Ejemplo: Permitir que Maria se conecte desde cualquier host en el dominio snake.net.

```
CREATE USER 'Maria'@'%.snake.net' IDENTIFIED BY 'Arrieta';

GRANT ALL ON eBanca.* TO 'Maria'@'%.snake.net';

SHOW GRANTS FOR 'Maria'@'%.snake.net';

FLUSH PRIVILEGES;
```

Se pueden utilizar bases de datos que no existan.

La parte del host del valor de cuenta puede ser también una dirección IP en lugar de un nombre de host. Se puede especificar:

- Una dirección IP.
- Una dirección que contenga caracteres patrón.
- Números IP mediante una máscara de red indicando que bits se deben utilizar en el número de red.

```
CREATE USER 'Juan'@'192.168.128.3' IDENTIFIED BY 'Pastoriza';

GRANT ALL ON Ejemplo.* TO 'Juan'@'192.168.128.3';

CREATE USER 'Marta'@'192.168.128.%' IDENTIFIED BY 'Pastoriza';

GRANT ALL ON Ejemplo.* TO 'Marta'@'192.168.128.%';

CREATE USER 'Ramon'@'192.168.128.0/255.255.128.0' IDENTIFIED BY 'Pastoriza';

GRANT ALL ON Ejemplo.* TO 'Ramon'@'192.168.128.0/255.255.128.0';
```

Ejemplos privilegios de Tabla y Columna.

Para los especificadores a nivel de tabla, se pueden especificar una cláusula (lista_columnas) después de la lista de privilegios para asignar privilegios a nivel de columna.

Los privilegios de columna son útiles cuando hay partes de una tabla que queremos ocultar a un usuario o cuando queremos que un usuario pueda modificar sólo determinadas columnas.

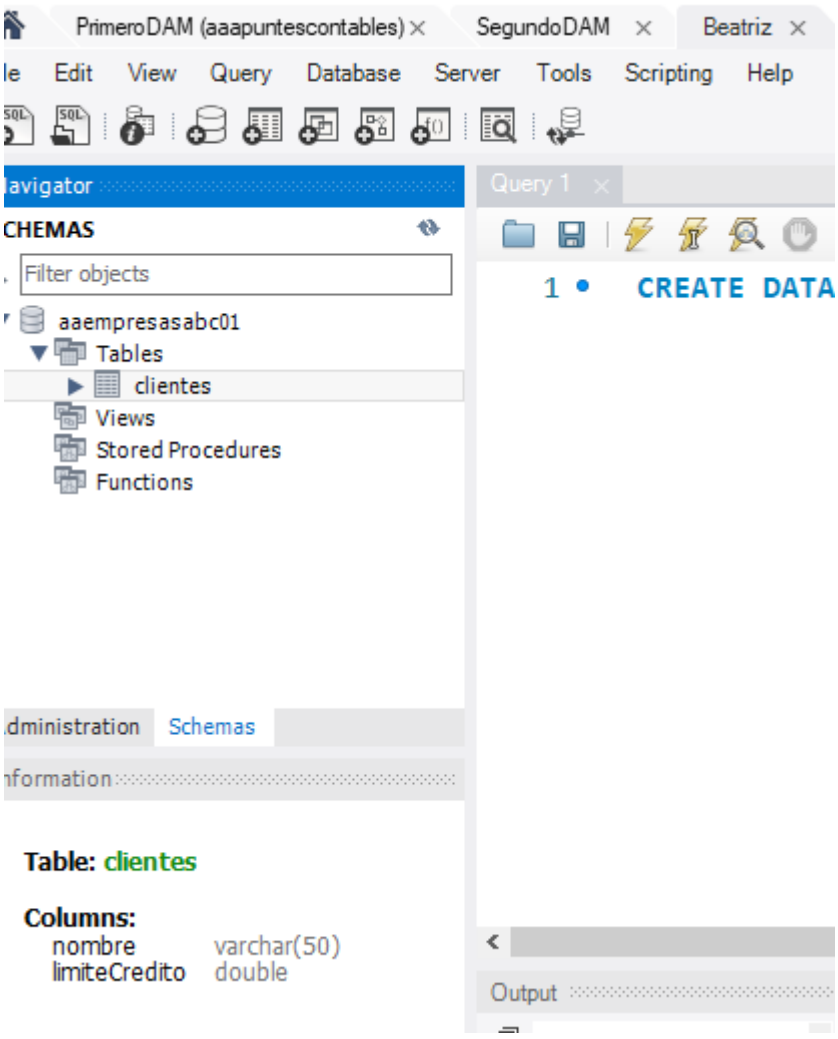
Ejemplo: Otorgar el privilegio SELECT en la tabla Clientes de la base de datos EmpresasABC en las columnas nombre y límite de crédito al usuario Beatriz contraseña Olalde.

```
GRANT SELECT (Nombre, LimiteCredito) ON aaEmpresasABC01.Clientes TO
'Beatriz'@'localhost';
SHOW GRANTS FOR 'Beatriz'@'localhost';
```

```
FLUSH PRIVILEGES;
```

Para comprobarlo entramos en MySQL con el usuario
SELECT * FROM Clientes;
Se produce un error solo tenemos permiso para dos columnas.

SELECT Nombre, LimiteCredito FROM Clientes;
No se produce el error.



Ejemplo insertar una nueva fila en la tabla Empleados.

INSERT INTO Clientes (CodCliente, Nombre, CodRepCliente, LimiteCredito) VALUES ('789', 'Rosario Aguirre', '102', 5236);
Se produce un error solo tenemos permiso de selección (SELECT), no para inserción (INSERT).

OTORGAR PRIVILEGIOS

Para poder asignar privilegios hay que disponer del privilegio GRANT OPTION.

Ejemplo Crear al usuario Esther con contraseña café, que sea un superusuario, es decir, que tenga todos los privilegios incluido el de otorgar privilegios.

CREATE USER 'Esther'@'localhost' IDENTIFIED BY 'cafe';

GRANT ALL ON *.* TO 'Esther'@'localhost' WITH GRANT OPTION;

FLUSH PRIVILEGES;

SHOW GRANTS FOR 'Esther'@'localhost';

SELECT * FROM mysql.user;

Esta sentencia la tenemos que ejecutar desde el root, si la ejecutamos desde el usuario Beatriz se produce un error, porque no puede otorgar privilegios.

La cláusula ON *.* significa todas las bases de datos y todas las tablas. Como medida de precaución Esther solo se puede conectar desde localhost (un único host).

Creamos una conexión para Esther y desde dicha conexión creamos un usuario y le otorgamos privilegios.

Se pueden especificar una lista de privilegios separándolos por comas (,).

Ejemplo Crear al usuario Juan con contraseña varios, que tenga la capacidad de leer y modificar los contenidos de las tablas de la base de datos EmpresasABC, pero no debe poder crear, o eliminar tablas.

DROP USER 'juan'@'localhost';

CREATE USER 'juan'@'localhost' IDENTIFIED BY 'varios';

GRANT SELECT, INSERT, DELETE, UPDATE ON aaempresasabc01.* TO 'juan'@'localhost';

FLUSH PRIVILEGES;

Creamos una conexión para Juan y desde dicha conexión intentamos crear una tabla.

Ejemplo Suponemos que en un momento dado necesitamos ayuda para actualizar la fecha de nacimiento de la tabla Empleados de la base de datos EmpresasAbc, creamos un usuario llamado Asistente con contraseña ayudante, que deberá tener permiso para leer todas las columnas de la

tabla, pero solo deberá tener permiso de actualización para la columna fecha de nacimiento.

```
CREATE USER 'asistente'@'localhost' IDENTIFIED BY 'ayudante';

GRANT SELECT, UPDATE (FecNacimiento) ON aaempresasabc01.Empleados TO 'asistente'@'localhost';

FLUSH PRIVILEGES;
```

Creamos una conexión para asistente y desde ella ejecutamos las siguientes sentencias

```
UPDATE empleados set sueldo = 3000 WHERE codEmpleado = 101;

UPDATE empleados SET FecNacimiento = '1986-08-16' WHERE codEmpleado = 101;
```

Para asignar privilegios específicos de columnas para más de una columna, se escriben los nombres_columnas separados por comas (,).

Ejemplo: añadir un privilegio UPDATE para las columnas Sueldo y Objetivo para el usuario asistente.

```
GRANT UPDATE (Sueldo, Objetivo) ON aaempresasabc01.Empleados TO 'asistente'@'localhost';

FLUSH PRIVILEGES;
```

Para comprobar desde el usuario asistente podemos ejecutar la sentencia

```
UPDATE empleados set sueldo = 3000 WHERE codEmpleado = 101;
```

Los nuevos privilegios se añaden a los ya existentes.

Los registros en las tablas de autorizaciones no siguen las operaciones de renombrado de las bases de datos, tablas o columnas, es decir, que cualquier privilegio asociado a una tabla no se aplica al cambio.

FLUSH

```
FLUSH [LOCAL | NO_WRITE_TO_BINLOG] flush_option [,flush_option] ...
```

Se puede usar el comando **FLUSH** si se desea limpiar algo del caché interno usado por MySQL. Para poder ejecutar **FLUSH**, se debe poseer el privilegio *RELOAD*.

```
FLUSH PRIVILEGES;
```

Mostrar los Privilegios

Para ver los privilegios que tiene una cuenta se utiliza la sentencia **SHOW GRANTS**

Ejemplo ver los privilegios para el usuario Beatriz.

```
SHOW GRANSTS FOR ‘Beatriz’@’localhost’;
```

Para ver nuestros propios privilegios, los de la cuenta por defecto

```
SHOW GRANTS;
SHOW GRANTS FOR CURRENT_USER();
USE mysql
SELECT * FROM User WHERE user = ‘Beatriz’;
```

Eliminar Privilegios y Usuarios

Para eliminar los privilegios de una cuenta utilizamos **REVOKE**. La sintaxis es similar a la sentencia GRANT excepto que la cláusula TO se sustituye por FROM y no hay cláusulas IDENTIFIED BY, REQUIRE, ni WITH.

```
REVOKE priv_type [(column_list)] [, priv_type [(column_list)]]
...
    ON {tbl_name | * | *.* | db_name.*}
    FROM user [, user] ...
REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

La sentencia **REVOKE** permite a los administradores del sistema revocar derechos de esas cuentas.

Para hacer más sencillo revocar todos los privilegios, MySQL 4.1.2 ha añadido la siguiente sintaxis, que elimina todos los privilegios de los niveles de global, de base de datos, tabla y columna para los usuarios nombrados:

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...

Para la sentencia **REVOKE**, se puede usar cualquiera de los siguientes valores para *priv_type igual que en la sentencia GRANT*:

El usuario debe coincidir con la definición de la sentencia GRANT original para la que queremos quitar los privilegios.

Los privilegios no tienen que coincidir con los de GRANT, ya que se pueden asignar unos privilegios con GRANT y luego eliminar solo parte con REVOKE.

Ejemplo: Eliminar al usuario juan, contraseña varios el privilegio SELECT.

```
REVOKE SELECT ON aaempresasabc01.* FROM 'juan'@'localhost';
FLUSH PRIVILEGES;
```

El privilegio GRANT OPTION, no se incluye en ALL, si la cuenta tiene este privilegio hay que eliminarlo explícitamente.

```
REVOKE GRANT OPTION ON *.* FROM 'Esther'@'localhost';
FLUSH PRIVILEGES;
```

Para eliminar un privilegio tenemos que disponer de ese privilegio y además tener el privilegio GRANT OPTION.

Para eliminar todos los privilegios incluyendo GRANT OPTION a nivel global, de base de datos o de tabla, podemos combinar ALL con GRANT OPTION.

```
REVOKE ALL, GRANT OPTION FROM 'Esther'@'localhost';
FLUSH PRIVILEGES;
```

Para eliminar todos los privilegios disponibles en una cuenta a cualquier nivel:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'asistente'@'localhost';
```

https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html

https://dev.mysql.com/doc/refman/5.7/en/grant.html

ELIMINAR USUARIOS

```
DROP USER [IF EXISTS]
    user
```

```
CREATE USER IF NOT EXISTS 'bea'@'localhost' IDENTIFIED BY 'Hola';
```

DROP USER 'bea'@'localhost';

```
CREATE USER IF NOT EXISTS 'teresa' IDENTIFIED BY 'Hola';
```

DROP USER IF EXISTS 'teresa';

CAMBIAR LA CONTRASEÑA DE UN USUARIO

```
ALTER USER 'nombreUsuario'@'localhost' IDENTIFIED BY 'NuevaContraseña';
```

Ejemplo: Si queremos cambiar la contraseña de la cuenta 'bea'@'localhost' por Adios.

```
CREATE USER IF NOT EXISTS 'bea'@'localhost' IDENTIFIED BY 'Hola';
```

```
ALTER USER 'bea'@'localhost' IDENTIFIED BY 'Adios';
```

Se puede comprobar que el usuario bea no puede entrar con la contraseña antigua Hola, sino que a partir de ejecutar la sentencia anterior, deberá entrar con la contraseña Adios

Hacer que una cuenta utilice conexiones seguras

El protocolo TLS (Transport Layer Security), sucesor de SSL (Segurity Sockets Layer) permite incorporar los cuatro elementos de seguridad completa en una conexión, es decir garantiza:

- la **confidencialidad** (mediante encriptación de los datos que evita que puedan ser entendidos por alguien),
- **integridad** (mediante la firma digital que evita que los datos puedan ser modificados sin que el usuario se entere),
- **autenticación** (el emisor tiene que demostrar sus credenciales) y
- **no repudio** de forma que el que recibe un mensaje sabe con seguridad quién es el autor.

En MySQL se pueden utilizar conexiones seguras utilizando el protocolo SSL (Segure Sockets Layer) que encripta el flujo de datos entre el cliente y el servidor para que la información no se trasmita de una forma comprensible. Además, se puede utilizar el protocolo X509 como medio para que el cliente proporcione información de identificación en conexiones SSL.

Las conexiones seguras proporcionan una medida adicional de protección, lo que supone un mayor consumo de recursos de la UCP para las operaciones de encriptado y desencriptado.

MySQL permite que el encriptado sea activado para conexiones individuales. Se puede escoger entre una conexión normal sin cifrar, o una segura cifrada mediante SSL dependiendo de las necesidades de las aplicaciones individualmente.

Conceptos básicos de SSL

Por defecto MySQL utiliza conexiones sin cifrar entre el cliente y el servidor. Esto significa que cualquiera con acceso a la red podría ver el tráfico y mirar los datos que están siendo enviados o recibidos. Incluso podría cambiar los datos mientras están aún en tránsito entre el cliente y el servidor. Para mejorar la seguridad un poco, se puede comprimir el tráfico entre el cliente y el servidor utilizando la opción **--compress** cuando ejecutamos programas cliente.

Cuando necesitemos mover información en una red de forma segura, no se puede utilizar una conexión sin encriptar. Hoy en día se necesitan muchos elementos adicionales de seguridad en los algoritmos de encriptado debido a los numerosos ataques que se producen.

El más utilizado es el PKI o infraestructura de clave pública. En este sistema la comunicación se basa en el uso de algoritmos de encriptado asimétricos que tienen dos claves de encriptado para cada parte de la comunicación (una pública y otra privada). Cualquier dato encriptado con la clave privada puede ser solo descifrado utilizando la clave privada correspondiente, solo en poder del propietario del certificado. También incorpora sistemas de verificación de identidad, utilizando el estándar X509.

X509 hace posible identificar a alguien en Internet. Para ello algunas empresas llamadas Autoridades de Certificación o CA asignan certificados electrónicos a cualquiera que los necesita. Un certificado consiste en la clave pública de su propietario junto con otros datos personales todo ello firmado con la clave privada de la autoridad correspondiente.

El propietario de un certificado puede enseñárselo a otra entidad como prueba de su identidad. Si la entidad confía en dicha autoridad y dispone de su clave pública podrá ver la información del certificado y así autentificar al emisor del mensaje además de conocer su clave pública.

Opciones SSL de GRANT

Para especificar el requisito de una conexión segura se utiliza la cláusula **REQUIRE** de la sentencia GRANT.

Hay diferentes maneras de limitar los tipos de conexión para una cuenta:

- Si una cuenta no tiene requerimientos de SSL o X509, las conexiones sin cifrar se permiten siempre que el nombre de usuario y la clave sean válidos. De cualquier manera, se pueden también utilizar conexiones cifradas, si el cliente tiene los certificados y archivos de claves apropiados.
- La opción REQUIRE SSL limita al servidor para que acepte únicamente conexiones cifradas SSL para la cuenta.

Para obligar a un usuario a conectarse mediante SSL sin indicarle nada más del tipo de conexión seguras utilizamos la cláusula **REQUIRE SSL**.

```
CREATE USER IF NOT EXISTS 'isabel'@'localhost' IDENTIFIED BY 'Hola' REQUIRE SSL;
```

Para ser más estrictos podemos obligar al cliente a presentar un certificado X509 válido.

REQUIRE X509 significa que el cliente debe tener un certificado pero que el certificado concreto, la entidad certificadora no importan. El único requerimiento es que debería ser posible verificar su firma con uso de los certificados CA.

```
CREATE USER IF NOT EXISTS 'carmen'@'localhost' IDENTIFIED BY 'Hola'
REQUIRE X509;
```

REQUIRE ISSUER 'issuer' coloca una restricción en la conexión mediante la cual el cliente debe presentar un certificado X509 válido, emitido por las autoridades CA cuyo nombre es 'issuer'. Si el cliente presenta un certificado que es válido pero tiene un emisor.

```
CREATE USER IF NOT EXISTS 'ignacio'@'localhost' IDENTIFIED BY 'Hola'
REQUIRE ISSUER '/c=ES/ST=Galicia/L=vigo/0=MySQL españa
AB/CN=jperez/Email=jperez@gmail.com';
```

El 'issuer' hay que introducirlo como una única cadena de caracteres.

REQUIRE SUBJECT 'subject' establece la restricción a los intentos de conexión de que el cliente debe presentar un certificado X%=) válido con sujeto 'subject'. Si el cliente presenta un certificado que, aunque válido, tiene un sujeto diferente, el servidor rechaza la conexión.

```
CREATE USER IF NOT EXISTS 'elena'@'localhost' IDENTIFIED BY 'Hola'
REQUIRE SUBJECT '/c=ES/ST=Galicia/L=vigo/0=MySQL españa
AB/CN=jperez/Email=jperez@gmail.com';
```

El 'subject' hay que introducirlo como una única cadena de caracteres.

REQUIRE CIPHER 'cipher' es necesario para asegurar que se utilizan longitudes de cifra y claves suficientemente fuertes. El protocolo SSL por sí mismo puede ser débil si se utilizan viejos algoritmos con claves de cifrado cortas. Utilizando esta opción podemos pedir un método de cifrado para permitir una conexión.

```
CREATE USER IF NOT EXISTS 'MLuisa'@'localhost' IDENTIFIED BY 'Hola'
REQUIRE CIPHER 'EDH-RSA-DES-CBC3-SHA';
```

Las opciones SUBJECT, ISSUER y CIPHER pueden combinarse en la sentencia REQUIRE de la siguiente manera:

```
CREATE USER IF NOT EXISTS 'Antxon'@'localhost' IDENTIFIED BY 'Hola'
REQUIRE SUBJECT '/c=ES/ST=Galicia/L=vigo/0=MySQL demo client
certificate/CN=jperez/Email=jperez@gmail.com'
AND ISSUER '/c=ES/ST=Galicia/L=vigo/0=MySQL españa AB/CN=L
/Email=asjperez@gmail.com'
AND CIPHER 'EDH-RSA-DES-CBC3-SHA';
```

El orden de las opciones no importa pero no se pueden repetir.

Limitar el Consumo de Recursos de una Cuenta

El sistema de autorizaciones de MySQL nos permite definir límites al número de veces por hora que un usuario puede conectarse con el servidor, y al número de sentencias o actualizaciones por hora que puede ejecutar el usuario. Para especificar estos límites se utiliza la cláusula **WITH**.

Ejemplo: Definir un usuario que tenga un acceso total a la base de datos ejemplo y que pueda conectarse solo diez veces por hora y ejecutar 200 sentencias a la hora, de las cuales como mínimo 50 pueden ser actualizaciones.

```
CREATE USER IF NOT EXISTS 'Iciar'@'localhost' IDENTIFIED BY 'Hola'
```

```
WITH MAX_CONNECTIONS_PER_HOUR 10 MAX_QUERIES_PER_HOUR 200
MAX_UPDATES_PER_HOUR 50;
```

El valor por defecto es cero, que significa “ningún límite”. Por lo tanto para eliminar las limitaciones cambiamos los valores a cero.

```
CREATE USER IF NOT EXISTS 'Santiago'@'localhost' IDENTIFIED BY 'Hola'
WITH MAX_CONNECTIONS_PER_HOUR 0;
```

No se pueden violar estas limitaciones por un usuario, utilizando varias conexiones al servidor, ya que todas las conexiones de una cuenta se contabilizan de forma global.

Actualmente, hay una nueva limitación que es el número de conexiones simultáneas, USER_CONNECTIONS si el límite es cero, el límite se controla mediante la variable del sistema **max_user_conections**. Un valor distinto de cero limita el número de conexiones simultáneas.

El orden en el que se definan las opciones no tiene importancia.