4. Cómo Crear y Gestionar Usuarios en MySQL

MySQL es uno de los sistemas de gestión de bases de datos más populares, y la administración de usuarios es una tarea esencial para cualquier administrador de bases de datos. Aprenderemos a crear y gestionar usuarios en MySQL. Veremos desde la creación de usuarios básicos hasta la asignación de permisos específicos y la eliminación de cuentas.

1. Introducción a la Gestión de Usuarios en MySQL

Gestionar usuarios en MySQL es una parte crucial de la administración de bases de datos, especialmente en entornos de producción. Cada usuario puede tener diferentes niveles de acceso a las bases de datos, lo que permite un control detallado, preciso, a un nivel específico sobre quién puede hacer qué dentro del sistema. En este artículo, abordaremos cómo crear, modificar y eliminar usuarios, así como cómo asignar permisos de forma segura.

Necesitamos una cuenta de usuario para acceder a los datos de una base de datos. Las claves de acceso se establecen cuando se crea el usuario y pueden ser modificados por el Administrador o por el propietario de dicha clave. La Base de Datos almacena encriptadas las claves en la base de datos **'mysql' y tabla 'user'.**

¿Por qué Crear Usuarios en el Servidor MySQL?

Cuando los usuarios o administradores de bases de datos instalan MySQL, el primer usuario que se crea es el usuario root, es decir, el administrador de MySQL. El usuario root tendrá permisos para hacer todo en la base de datos MySQL.

Aunque tener todos los permisos puede parecer beneficioso, hacerlo tiene sus propias vulnerabilidades de seguridad, y compartir el usuario root entre varias personas es peligroso. Los hackers a menudo intentan iniciar sesión como el usuario root y robar la información alojada o incluso destruir todo el servidor MySQL junto con sus datos.

Por lo tanto, los administradores del sistema crean usuarios con permisos específicos en algunas bases de datos. En otras palabras, si las credenciales de una cuenta se ven comprometidas, el impacto será mínimo y manejable.

Recuerda que para crear usuarios debes tener una cuenta con privilegios de Administrador.

2. Conectarse al Servidor MySQL

Antes de comenzar a crear y gestionar usuarios, debes conectarte a tu servidor MySQL.

3. Cómo listar los usuarios creados

NOTA: MySQL no dispone de un comando SHOW USERS. La información reside en las tablas del sistema.

EJEMPLO: Utiliza la siguiente consulta para mostrar los usuarios de MySQL creados en el servidor de la base de datos.

SELECT user FROM mysql.user; - BBDD mysql, tabla use.

EJEMPLO: Para ver todos los usuarios y desde qué host pueden conectarse.

SELECT user, host FROM mysql.user; - Selecciona las columnas user y host de la tabla user de la bbdd mysql

EJEMPLO: Para ver si las cuentas están bloqueadas o si la contraseña ha expirado:

SELECT user, host, account_locked, password_expired

FROM mysql.user;

NOTA: Diferencia entre Comandos A TENER EN CUENTA.

- **SHOW DATABASES;**: Válido. Muestra las bases de datos.
- **SHOW TABLES;**: Válido. Muestra las tablas de la base de datos actual.
- **SHOW USERS;**: **NO VÁLIDO**. Produce un error de sintaxis en MySQL.

---

4. Crear un Nuevo Usuario en MySQL

Para crear un nuevo usuario en MySQL, usa el comando CREATE USER. La sintaxis básica es:

```
CREATE USER [IF NOT EXISTS]
    user [auth_option] [, user [auth_option]] …
DEFAULT ROLE role [, role ] ...
```

CREATE USER IF NOT EXISTS 'nombre_usuario'@'host' IDENTIFIED BY 'contraseña';

- user → 'nombre_usuario'@'host'

- 'nombre_usuario': El nombre del usuario que deseas crear.
- 'host' Especifica desde qué host puede conectarse el usuario. Puedes usar 'localhost' para limitar el acceso al mismo servidor o '%' para permitir el acceso desde cualquier host.

  Ejemplo:

  CREATE USER 'maria'@'localhost' IDENTIFIED BY 'maria_password';

  Este comando crea un usuario llamado maria que solo puede conectarse desde localhost y debe usar la contraseña maria_password.

- 'IP': Crear Usuarios con Acceso Remoto. Si necesitas que un usuario se conecte desde una máquina remota, puedes especificar la dirección IP del host remoto en lugar de localhost.

  Ejemplo:
  CREATE USER 'remoto'@'192.168.1.100' IDENTIFIED BY 'remoto_password';
  GRANT ALL PRIVILEGES ON ejemplo_db.* TO 'remoto'@'192.168.1.100';

  Este comando crea un usuario remoto que puede conectarse desde la dirección IP 192.168.1.100 y tiene todos los privilegios en la base de datos ejemplo_db.

Ejemplo: -- Crear usuario y asignarle el rol en un solo comando

CREATE USER 'nuevo_empleado'@'localhost'

IDENTIFIED BY 'Password123'

DEFAULT ROLE 'rol_lectura';

Ejemplo C: Crear múltiples usuarios con roles distintos

Ejemplo: Creación múltiples usuarios

Puedes crear varios usuarios de una sola vez utilizando CREATE USER. Esta es la forma recomendada en versiones modernas de MySQL para separar la creación de la cuenta de la asignación de permisos.

Sintaxis básica:

```
CREATE USER 'user1'@'host1' IDENTIFIED BY 'pass1',
            'user2'@'host2' IDENTIFIED BY 'pass2',
            'user3'@'host3' IDENTIFIED BY 'pass3';
```

Mezclando diferentes hosts y contraseñas: Ideal para configurar un equipo con diferentes orígenes de conexión.

Ejemplo:

```
CREATE USER 'dev_junior'@'localhost' IDENTIFIED BY 'alfa123',
            'dev_senior'@'192.168.1.%' IDENTIFIED BY 'beta456',
            'analista'@'%' IDENTIFIED BY 'gamma789';
```

Ejemplo:

CREATE ROLE 'rol_lectura', 'rol_escritura';

- Asignar Privilegios al Rol

GRANT SELECT ON escuela.* TO 'rol_lectura';

GRANT INSERT, UPDATE, DELETE ON escuela.* TO 'rol_escritura';

CREATE USER 'becario_1'@'%' IDENTIFIED BY 'Pass1' DEFAULT ROLE 'rol_lectura',

'jefe_it'@'%' IDENTIFIED BY 'Pass2' DEFAULT ROLE 'rol_lectura', 'rol_escritura';

NOTA: Los usuarios creados con CREATE USER, a los que no se le asignen privilegios (siguiente punto) **existen**, pero **no puede hacer nada sobre las bases de datos**.

Ejemplo práctico

```
CREATE USER 'alumno'@'localhost' IDENTIFIED BY '1234';
```

El alumno intenta usar la base de datos instituto:

```
USE instituto;
```

El SGBB devuelve **Error**: Access denied for user 'alumno'

Ejemplo: Crear múltiples usuarios con roles distintos

```
CREATE USER 'becario_1'@'%' IDENTIFIED BY 'Pass1' DEFAULT ROLE 'rol_lectura',
                  'jefe_it'@'%' IDENTIFIED BY 'Pass2' DEFAULT ROLE 'rol_lectura', 'rol_escritura';
```

---

5. Privilegios usuarios

5.1. Comando GRANT y REVOKE

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON {tbl_name | * | *.* | db_name.*}
    TO user [IDENTIFIED BY [PASSWORD] 'password']
        [, user [IDENTIFIED BY [PASSWORD] 'password']] …
```

La sentencia GRANT permite a los administradores de sistemas otorgar privilegios y permisos a las cuentas de usuario para que puedan interactuar con las bases de datos.

NOTA: El comportamiento de GRANT respecto a la creación de usuarios depende de una variable de configuración llamada NO_AUTO_CREATE_USER.

- **Comportamiento:** Por defecto, si ejecutas GRANT sobre un usuario que no existe, MySQL lo crea automáticamente con la contraseña que indiques en IDENTIFIED BY.
- **Comportamiento Recomendado:** En versiones recientes (MySQL 8.0+), esta opción suele estar desactivada o eliminada. La recomendación oficial es **crear primero el usuario con CREATE USER** y luego asignar permisos con GRANT.

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

La sentencia REVOKE permite revocar los privilegios del usuario otorgados con el comando GRANT. La revocación se hará al mismo nivel de privilegios otorgados con el comando GRANT

Desglose de Elementos

priv_type: El tipo de privilegio (ej. SELECT, INSERT, UPDATE, ALL PRIVILEGES).

Para las sentencias GRANT y REVOKE, se puede usar cualquiera de los siguientes valores para priv_type que se muestran en la siguiente tabla, teniendo en cuenta lo siguiente

- Los privilegios EXECUTION, FILE, PROCESS, RELOAD, REPLICATION CLIENT, REPLICATION SLAVE, SHOW DATABASES, SHUTDOWN y SUPER son privilegios administrativos que sólo pueden ser concedidos de forma global (usando la sintaxis ON *.*).
- Otros privilegios pueden ser concedidos globalmente o en niveles más específicos.
- Los únicos valores priv_type que se pueden especificar para una tabla son SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT OPTION, INDEX y ALTER.
- Los únicos valores priv_type que se pueden especificar para una columna (esto es, cuando se usa una cláusula column_list) son SELECT, INSERT y UPDATE.
- Los únicos valores priv_type que se pueden especificar en el nivel de rutina son ALTER ROUTINE, EXECUTE y GRANT. CREATE ROUTINE no es un privilegio de nivel de rutina porque se debe

tener este privilegio para que sea posible crear una rutina en primer lugar.

Privilegio	Significado
ALL [PRIVILEGES]	Conceden todos los privilegios a este usuario. Los posibles privilegios: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE_TMP_TABLE, LOCK_TABLES, CREATE_VIEW,SHOW_VIEW, CREATE_ROUTINE, ALTER_ROUTINE, EXECUTE y GRANT.
ALTER	Permite el uso de ALTER TABLE.
CREATE	Permite el uso de CREATE TABLE.
CREATE ROUTINE	Crear rutinas almacenadas.
CREATE TEMPORARY TABLES	Permite el uso de CREATE TEMPORARY TABLE.
CREATE VIEW	Permite el uso de CREATE VIEW.
DELETE	Permite el uso de DELETE.
DROP	Permite el uso de DROP TABLE.
EXECUTE	Permite al usuario ejecutar procedimientos almacenados.
FILE	Permite el uso de SELECT ... INTO OUTFILE y LOAD DATA INFILE.

INDEX	Permite el uso de CREATE INDEX y DROP INDEX.
INSERT	Permite el uso de INSERT.
LOCK TABLES	Permite el uso de LOCK TABLES en tablas sobre las que ya se posea el privilegio *SELECT*. Permite a un usuario de base de datos bloquear explícitamente una o varias tablas para controlar el acceso concurrente, impidiendo que otros procesos las modifiquen o lean, asegurando así la integridad de los datos durante operaciones críticas como actualizaciones masivas, aunque requiere el privilegio SELECT. **Ejemplo:** **-- Bloquear la tabla 'clientes' para lectura y escritura** **LOCK TABLES clientes WRITE;** **-- Insertar un registro** **INSERT INTO clientes (nombre, ciudad) VALUES ('Juan', 'Madrid');** **-- Liberar bloqueo** **UNLOCK TABLES;**

PROCESS	Permite el uso de SHOW FULL PROCESSLIST.
	• Permite usar:
	`SHOW FULL PROCESSLIST;`
	• Muestra todas las conexiones activas al servidor, incluyendo:
	  ○ Id de conexión
	  ○ Usuario

	○ Host ○ Base de datos ○ Comando que ejecuta ○ Tiempo ○ Estado Muy útil para **monitorizar conexiones**, detectar procesos largos o bloqueos.
REFERENCES	No implementado.
RELOAD	Permite el uso de FLUSH. Permite usar **comandos de recarga del servidor**, como: `FLUSH PRIVILEGES;` -- Recarga los privilegios de usuarios sin reiniciar MySQL `FLUSH TABLES;` -- Libera las tablas abiertas
REPLICATION CLIENT	Permite al usuario **consultar información sobre servidores maestro o esclavo** para replicación.
Un servidor Maestro (Master/Origen) es el servidor principal que maneja todas las escrituras (INSERT, UPDATE, DELETE) y lecturas, mientras que uno o más servidores Esclavos (Slave/Réplica) son copias secundarias que reciben automáticamente los cambios del maestro para manejar solo las lecturas, distribuyendo la carga y proporcionando alta disponibilidad y respaldo. Esta configuración permite escalar el rendimiento al desviar consultas de lectura a los esclavos, sin afectar al maestro, y tener réplicas para copias de seguridad o recuperación ante desastres.	
REPLICATION SLAVE	Privilegio necesario para la **replicación esclava**: leer eventos del **binary log del maestro**. Solo se concede a usuarios que serán usados como **replica**.

SELECT	Permite el uso de SELECT.
SHOW DATABASES	La sentencia SHOW DATABASES muestra todas las bases de datos.
SHOW VIEW	Permite el uso de SHOW CREATE VIEW.
SHUTDOWN	Permite apagar el servidor MySQL usando el comando mysqladmin shutdown. Solo lo puede usar un **usuario con este privilegio**. No otorga acceso a bases de datos ni a tablas; es **solo para administración del servidor**.
SUPER	Es **un privilegio potente para la administración global**. Permite: 1. Ejecutar sentencias administrativas avanzadas: • CHANGE MASTER: configuración de replicación • KILL: terminar conexiones o procesos específicos • PURGE MASTER LOGS: limpiar logs de replicación • SET GLOBAL: cambiar variables globales en tiempo real 2. Usar comandos de depuración: • mysqladmin debug 3. Conectarse **aunque se haya alcanzado el límite máximo de conexiones (max_connections)** **Ejemplo práctico** **a) Usando KILL para terminar una conexión** SHOW PROCESSLIST; -- Muestra todas las conexiones KILL 12;

column_list: (Opcional): Permite restringir el privilegio a columnas específicas de una tabla. Ejemplo: UPDATE (email, telefono).

ON {tbl_name | * | *.* | db_name.*} : Define dónde se aplican los privilegios (el nivel de alcance):

Niveles Privilegios

- ***.*** 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.
- **db_name.*** De base de datos: se refieren a bases de datos individuales, y por extensión, a todos los objetos que contiene cada base de datos.
- **tbl_name** De tabla: se aplican a tablas individuales, y por lo tanto, a todas las columnas de esas tablas.
- De columna: se aplican a una columna en una tabla concreta.

- De rutina: se aplican a los procedimientos almacenados[1].

TO user

Define a quién se le otorgan los permisos. El formato suele ser 'usuario'@'host'.

- **'admin'@'localhost': Solo puede conectar desde la misma máquina.**
- **'admin'@'ip': Esta es la parte más importante para la seguridad. Define desde qué máquina o red tiene permiso este usuario para entrar.**
- **IP específica ('192.168.1.50'): El usuario admin solo podrá entrar si lo hace desde esa dirección IP exacta. Si intenta entrar desde otra PC, MySQL rechazará la conexión aunque la contraseña sea correcta.**

  GRANT SELECT ON mi_base.* TO 'admin'@'192.168.1.50';

- **'admin'@'%': Puede conectar desde cualquier dirección IP.**

IDENTIFIED BY 'password': Define la contraseña del usuario. Si el usuario no existe, MySQL lo creará automáticamente con esta contraseña (dependiendo de la versión y configuración de SQL Mode).

2. Ejemplos Prácticos con Diferentes Opciones

Ejemplo A: Privilegios Globales (Superusuario): Otorga todos los permisos sobre todos los objetos del servidor.

```
GRANT ALL PRIVILEGES ON *.* TO 'super_profe'@'%' IDENTIFIED BY 'PasswordSeguro123';
```

[1] Un procedimiento almacenado es un conjunto de instrucciones SQL que se almacena asociado a una base de datos. Es un objeto que se crea con la sentencia CREATE PROCEDURE y se invoca con la sentencia CALL. Un procedimiento puede tener cero o muchos parámetros de entrada y cero o muchos parámetros de salida.

Ejemplo B: Privilegios a Nivel de Base de Datos. Permite leer y escribir en todas las tablas de la base de datos escuela.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON escuela.* TO 'secretaria'@'localhost' IDENTIFIED BY 'secre_2024';
```

Ejemplo C: Privilegios a Nivel de Columna (Granularidad Máxima). Este es muy útil para seguridad. El usuario solo puede ver los nombres de los alumnos, pero solo puede modificar la columna nota.

```
GRANT SELECT (nombre, apellido), UPDATE (nota) ON escuela.alumnos TO 'profesor_ayudante'@'localhost';
```

Ejemplo D:

```
CREATE ROLE 'rol_lectura', 'rol_escritura';
```

- Asignar Privilegios al Rol

```
GRANT SELECT ON escuela.* TO 'rol_lectura';
GRANT INSERT, UPDATE, DELETE ON escuela.* TO 'rol_escritura';
```

- Asignar el Rol a los Usuarios (Vía GRANT)

```
GRANT 'rol_lectura' TO 'ana'@'localhost', 'pedro'@'localhost';
GRANT 'rol_escritura' TO 'admin_tienda'@'%';
```

5.2. Opciones Avanzadas de Seguridad (MySQL 8.0+).

Al crear usuarios (ya sea de forma individual o múltiple), puedes añadir cláusulas para fortalecer la seguridad de la cuenta:

A. Expiración de Contraseña. Obliga al usuario a cambiar su contraseña cada cierto tiempo o en su primer inicio de sesión.

PASSWORD EXPIRE: La contraseña expira inmediatamente.

PASSWORD EXPIRE INTERVAL 90 DAY: Expira cada 3 meses.

B. Bloqueo de Cuenta

ACCOUNT LOCK: Crea el usuario pero mantiene la cuenta deshabilitada.

FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 2: Bloquea la cuenta por 2 días tras 3 intentos fallidos.

C. Restricción de Recursos (WITH). Limita el impacto que un usuario puede tener en el rendimiento del servidor.

MAX_QUERIES_PER_HOUR 100: Limita las consultas por hora.

MAX_USER_CONNECTIONS 5: Limita conexiones simultáneas.

Ejemplo de usuario ultra-seguro:

```
CREATE USER 'app_analitica'@'192.168.1.50'
IDENTIFIED BY 'Pass_Complejo_2025'
WITH MAX_QUERIES_PER_HOUR 500
PASSWORD EXPIRE INTERVAL 30 DAY
FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME UNBOUNDED;
```

Ejemplo:

```
GRANT SELECT, INSERT ON ejemplo_db.* TO 'maria'@'localhost';
```

Este comando otorga a maria privilegios de SELECT y INSERT en todas las tablas de la base de datos ejemplo_db.

Ejemplo:

Para mejorar la seguridad, puedes crear usuarios con acceso limitado, como aquellos que solo pueden realizar ciertas operaciones en una base de datos específica:

```
CREATE USER 'lectura'@'localhost' IDENTIFIED BY 'lectura_password';
GRANT SELECT ON ejemplo_db.* TO 'lectura'@'localhost';
```

Ejemplo:

Crear un usuario llamado lectura que solo puede realizar consultas (SELECT) en la base de datos ejemplo_db.

Si necesitas revocar todos los privilegios de un usuario de manera global, puedes usar el comando REVOKE ALL PRIVILEGES:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'nombre_usuario'@'host';
```

Ejemplo:

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

Este comando revoca todos los privilegios de maria y elimina su capacidad para otorgar permisos a otros usuarios.

Ejemplo: El usuario ya no pertenece al grupo/rol de lectura

```
REVOKE 'rol_lectura' FROM 'ana'@'localhost';
```

Ejemplo: Cree una cuenta que use el complemento de autenticación predeterminado y la contraseña especificada. Marque la contraseña como caducada para que el usuario deba elegir una nueva al conectarse por primera vez al servidor.

```
CREATE USER 'jeffrey'@'localhost'
  IDENTIFIED BY 'new_password' PASSWORD EXPIRE;
```

Ejemplo: Cree una cuenta que use el caching_sha2_passwordcomplemento de autenticación y la contraseña especificada. Exija que se elija

una nueva contraseña cada 180 días y habilite el seguimiento de inicios de sesión fallidos, de modo que tres contraseñas incorrectas consecutivas provocan el bloqueo temporal de la cuenta durante dos días.

```
CREATE USER 'jeffrey'@'localhost'
  IDENTIFIED WITH caching_sha2_password BY 'new_password'
  PASSWORD EXPIRE INTERVAL 180 DAY

    FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 2;
```

Ejemplo: crear varias cuentas, especificando algunas propiedades por cuenta y algunas propiedades globales:

```
CREATE USER
  'jeffrey'@'localhost' IDENTIFIED WITH mysql_native_password
                                     BY 'new_password1',
  'jeanne'@'localhost' IDENTIFIED WITH caching_sha2_password
                                    BY 'new_password2'
  REQUIRE X509 WITH MAX_QUERIES_PER_HOUR 60
  PASSWORD HISTORY 5

    ACCOUNT LOCK;
```

Las propiedades restantes se aplican globalmente a todas las cuentas nombradas en la declaración, por lo que para ambas cuentas:

- Las conexiones deben realizarse utilizando un certificado X.509 válido.
- Se permiten hasta 60 consultas por hora.
- Los cambios de contraseña no pueden reutilizar ninguna de las cinco contraseñas más recientes.

- La cuenta está bloqueada inicialmente, por lo que efectivamente es un marcador de posición y no se puede usar hasta que un administrador la desbloquee.

6. Aplicación de los privilegios asignados

Después de ejecutar sentencias GRANT, es una buena práctica ejecutar el siguiente comando para refrescar las tablas de privilegios en la memoria de MySQL:

```
FLUSH PRIVILEGES;
```

`FLUSH PRIVILEGES` se usa para recargar los permisos en memoria después de hacer cambios manuales o usar comandos como `GRANT`/`REVOKE`, aunque las versiones modernas de MySQL a menudo no lo requieren con esos comandos, sí es crucial para aplicar privilegios a roles (o usuarios) recién creados o modificados, permitiendo que los cambios surtan efecto sin reiniciar el servidor

7. Ver Privilegios Asignados.

Para revisar los privilegios/permisos, usa el comando SHOW GRANTS:

SINTAXIS:

```
SHOW GRANTS [FOR user];
```

USER: 'nombre_usuario'@'host'. Si no se especifica el user, se mostrará los privilegios del usuario actual.

Desde MySQL 4.1.2, para listar los privilegios para la sesión actual, se puede usar cualquiera de las siguientes sentencias:

```
 - Muestra los privilegios del usuario actual
SHOW GRANTS;
SHOW GRANTS FOR CURRENT_USER;
SHOW GRANTS FOR CURRENT_USER();
```

Ejemplo:

```
mysql> SHOW GRANTS FOR 'root'@'localhost';
```

Muestra los privilegios del usuario root.

Resultado:

```
+-------------------------------------------------------------------
--+
|              Grants              for              root@localhost
|
+-------------------------------------------------------------------
--+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT
OPTION |
+-------------------------------------------------------------------
--+
```

Ejemplo:

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

Este comando muestra todos los privilegios que el usuario maria tiene en el host localhost.

8. Renombrar un usuario

SINTAXIS

```
RENAME USER old_user TO new_user [, old_user TO new_user] ...
```

NOTA: Es importante recordar que si el usuario que intentas renombrar no existe, o si el nombre nuevo ya está en uso, el comando fallará con un error.

Ejemplos

```
-- Cambiar nombre de 'antiguo' a 'nuevo' en el mismo host
RENAME USER 'pedro'@'localhost' TO 'pedro_admin'@'localhost';

-- Cambiar el host permitido (mover de local a cualquier IP)
RENAME USER 'ana'@'localhost' TO 'ana'@'%';

-- Renombrar varios usuarios a la vez
RENAME USER 'user1'@'localhost' TO 'u1'@'localhost',
            'user2'@'localhost' TO 'u2'@'localhost';
```

9. Cambiar la Contraseña de un Usuario

Puedes cambiar la contraseña de un usuario existente con el comando ALTER USER:

- `ALTER USER 'nombre_usuario'@'host' IDENTIFIED BY 'nueva_contraseña';`

Otras formas de cambiar la contraseña de un usuario, aunque están obsoletas en el caso de las dos primeras y desaconsejada la tercera porque puede corromper el sistema, incompatible entre versiones y requiere `FLUSH PRIVILEGES`

- `SET PASSWORD FOR 'user1'@'localhost' = PASSWORD('user11');`

  Sólo los usuarios tales como root con acceso de modificación para la base de datos mysql puede cambiar la contraseña de otro usuario

- `GRANT USAGE ON *.* TO 'user1'@'localhost' IDENTIFIED BY 'user12‘;`

  Puede usar el comando GRANT USAGE globalmente (ON *.*) para asignar una contraseña a una cuenta sin afectar los permisos actuales de la cuenta

- `UPDATE mysql.user SET Password = PASSWORD('user13') WHERE Host = 'localhost' AND User = 'user1';`

  `FLUSH PRIVILEGES;`

  Aunque generalmente es preferible asignar contraseñas usando uno de los métodos precedentes, se puede hacer modificando la tabla mysql.user directamente.

Ejemplo:

```
ALTER USER 'maria'@'localhost' IDENTIFIED BY 'nueva_maria_password';
```

- Preferiblemente no usar.

```
GRANT USAGE ON *.* TO 'maria'@'localhost' IDENTIFIED BY 'user12';
```

```
SET PASSWORD FOR 'maria'@'localhost' = PASSWORD('user11');
```

Este comando cambia la contraseña del usuario maria a nueva_maria_password.

10. ROLES

Un rol es una colección o grupo de privilegios que se asigna a un usuario o a otros roles, funcionando como una plantilla para simplificar la gestión de permisos, en lugar de otorgarlos uno por uno.

10.1. Creación de roles y concesión de privilegios a los mismos

SINTAXIS

```
CREATE ROLE [IF NOT EXISTS] rol;
```

rol puede ser:

- ‘nombre’: un nombre para el rol.
- 'nombre_rol'@'host': rol solo válido desde un host o ip determinados.

Ejemplos:

```
CREATE ROLE 'app_developer', 'app_read', 'app_write';
- Crear un rol solo válido desde localhost:
CREATE ROLE 'webapp'@'localhost';
```

Los nombres de rol son muy similares a los nombres de cuenta de usuario y constan de una parte de usuario y una parte de host . Si se omite la parte de host, el valor predeterminado es ‘%’.

Para asignar privilegios a los roles, ejecute **GRANT**, utilizando la misma sintaxis que para asignar privilegios a las cuentas de usuario:

```
GRANT ALL ON app_db.* TO 'app_developer';
GRANT SELECT ON app_db.* TO 'app_read';
GRANT INSERT, UPDATE, DELETE ON app_db.* TO 'app_write';
```

Para revocar los privilegios a los roles, se utiliza REVOKE, utilizando la sintaxis:

```
REVOKE [privilegio_o_privilegios] ON [objeto] FROM 'nombre_del_rol';
```

Ejemplos:

```
REVOKE SELECT ON nombre_base_de_datos.nombre_tabla FROM 'app_read';
REVOKE ALL PRIVILEGES ON nombre_base_de_datos.* FROM 'app_developer';
```

10.2. Eliminación de roles.

Sintaxis

```
DROP ROLE [IF EXISTS] role [, role ] ...
```

DROP ROLE. Elimina uno o más roles (conjuntos de privilegios con nombre). Para usar esta instrucción, se requiere el privilegio global DROP ROLE.

Ejemplo: Elimina los roles: admin, developer y webapp en local.

```
DROP ROLE 'admin', 'developer';
DROP ROLE 'webapp'@'localhost';
```

NOTA: en MySQL es necesario indicar el host del rol cuando se asignan, revocan privilegios o se elimina el rol, salvo en el caso en que el rol haya sido creado **sin host explícito** (y entonces será %).

Si el rol se creó así:

CREATE ROLE webapp;

MySQL lo crea como 'webapp'@'%'

Por eso parece que "no hace falta host", pero sí lo tiene.

Eliminar el rol creado anteriormente wepapp:

```
DROP ROLE 'webapp'@'%';
```

Si haces:

```
DROP ROLE webapp;
```

En versiones modernas de MySQL puede funcionar porque MySQL asume %

Pero NO es buena práctica.

% es un **comodín de host** (Desde **cualquier host/ip)**

10.3. Rol por defecto para un usuario.

Un usuario puede tener diferentes roles asignados.
En el último paso seleccionamos cual es el rol por defecto que va a utilitzar con la instrucción SET DEFAULT.

La sentencia **SET DEFAULT ROLE** sirve para indicar **qué roles se activan automáticamente** cuando un usuario inicia sesión.

SINTAXIS

```
SET DEFAULT ROLE
    { NONE | ALL | role [, role ] ... }
    TO user [, user ] ...
```

NONE: Indica que ningún rol se activará automáticamente cuando el usuario se conecte.

El usuario tendrá que activar manualmente los roles con SET ROLE.

Ejemplo: Al iniciar la sesión, Juan, no tendrá ninguno activo

```
SET DEFAULT ROLE NONE TO 'juan'@'localhost';
```

ALL: Activa **todos los roles que tenga asignados el usuario** automáticamente al iniciar sesión.

Ejemplo: Ana tiene varios roles**, todos** se activan automáticamente al conectarse

```
SET DEFAULT ROLE ALL TO 'ana'@'%';
```

role [, role]: Permite indicar **exactamente qué roles** se activarán por defecto.

Ejemplo: Pedro puede tener varios roles, solo rol_lectura se activa por defecto

```
SET DEFAULT ROLE rol_lectura TO 'pedro'@'localhost';
```

Ejemplo: María tiene al menos esos dos roles asignados. Ambos se activan automáticamente al iniciar sesión

```
SET DEFAULT ROLE rol_lectura, rol_edicion TO 'maria'@'%';
```

Ejemplo: Se puede aplicar la configuración a **más de un usuario en una sola sentencia**.

```
SET DEFAULT ROLE rol_consulta
TO 'alumno1'@'localhost', 'alumno2'@'localhost';
```

NOTA: SET DEFAULT ROLE Es una sintaxis alternativa para ALTER USER ... DEFAULT ROLE. Sin embargo, ALTER USER solo se puede establecer el valor predeterminado para un usuario, mientras que SET DEFAULT ROL lo puede establecer para varios.

10.4. Activar o desactivar roles temporalmente.

SET ROLE: Lo activa el propio usuario, después de conectarse, dentro de la sesión para **El usuario puede:**

- **Activar otros roles que tenga asignados. Al activar un rol se desactivan los demás.**
- **Desactivar los actuales**
- **Cambiar combinaciones de roles**

Siempre dentro de su lista de roles asignados.

NOTA:

SET DEFAULT ROLE decide con qué permisos entras.

SET ROLE decide con qué permisos trabajas en ese momento.

EJEMPLOS:

1. **Activar otro rol, en este caso el rol_ventas.**

   ```
   SET ROLE rol_ventas;
   ```

   Resultado:

   - `rol_consulta` **se desactiva**
   - `rol_ventas` **pasa a estar activo**

2. **Activar varios roles**

   ```
   SET ROLE rol_consulta, rol_ventas;
   ```

3. **Volver a los roles por defecto**

   ```
   SET ROLE DEFAULT;
   ```

4. **Desactivar todos los roles**

   ```
   SET ROLE NONE;
   ```

10.5. Comprobar los roles activos.

```
SELECT CURRENT_ROLE();
```

Esto te indicará qué roles se están utilizando en este momento para determinar permisos efectivos.

10.6. Ver roles existentes y configuraciones

```
SELECT * FROM mysql.role_edges;
```

Esta consulta es útil para ver la asignación de roles entre usuarios y roles.

- **mysql.role_edges: Esta tabla muestra qué usuarios tienen asignados qué roles. Es una tabla interna del sistema,**

útil para administración y aprendizaje, pero: ⚠ No es recomendable para uso habitual

10.7. Ver los permisos de un rol en MySQL, usa:

```
SHOW GRANTS FOR nombre_rol;
```

11. Eliminar un Usuario y Roles en MySQL

Para eliminar un usuario en MySQL, usa el comando DROP USER:

```
DROP USER [IF EXISTS] user [, user] ...
DROP ROLE [IF EXISTS] role [, role ] ...
```

Ejemplo:

```
DROP USER 'maria'@'localhost';
```

Este comando elimina completamente al usuario maria del servidor MySQL, incluyendo todos los privilegios asociados.

```
DROP ROLE 'admin', 'developer';
DROP ROLE 'webapp'@'localhost';
```

NOTA: Un rol eliminado se revoca automáticamente de cualquier cuenta de usuario (o rol) a la que se le haya otorgado. En cualquier sesión activa de dicha cuenta, sus privilegios modificados se aplican a partir de la siguiente instrucción ejecutada.

12. Más ejemplos

EJEMPLO: Privilegios Nivel de Base de Datos.

- Los permisos de base de datos se aplican a todos los objetos en una base de datos dada
- Estos permisos se almacenan en las tablas “mysql.db”

GRANT ALL ON db_name.* y REVOKE ALL ON db_name.*
Otorgan y quitan sólo permisos de bases de datos.

```
GRANT all privileges on cdcol.* to user1@'localhost' identified by 'user1' with grant option;
REVOKE all privileges on cdcol.* FROM user1@'localhost' identified by 'user1';
```

EJEMPLO: Privilegios Nivel de Base de Datos

```
GRANT all privileges on cdcol.* to user1@'localhost' identified by
'user1' with grant option;
SHOW GRANTS FOR user1@localhost;

REVOKE all privileges on cdcol.* FROM user1@'localhost' identified
by 'user1';
SHOW GRANTS FOR user1@localhost;
```

EJEMPLO: Privilegios Nivel de Tabla

Los permisos de tabla se aplican a todas las columnas en una tabla dada

- Estos permisos se almacenan en la tabla mysql.tables_priv

GRANT ALL ON db_name. tbl_name y **REVOKE ALL ON** db_name. tbl_name

Otorgan y quitan permisos sólo de tabla.

```
GRANT all privileges on cdcol.cds to user1@'localhost' identified by
'user1' with grant option;
REVOKE all privileges on cdcol.cds FROM user1@'localhost' identified
by 'user1';

GRANT all privileges on cdcol.cds to user1@'localhost' identified by
'user1' with grant option;
SHOW GRANTS FOR user1@localhost;
```

EJEMPLO: Privilegios Nivel de Columna.

Los permisos de columna se aplican a columnas en una tabla dada.
Estos permisos se almacenanen la tabla “mysql.columns_priv”
Usando REVOKE, debe especificar las mismas columnas que se otorgaron los permisos.

```
GRANT SELECT(titel,interpret) on cdcol.cds to user1@'localhost'
identified by 'user1' with grant option;
REVOKE SELECT(titel,interpret) on cdcol.cds FROM user1@'localhost'
identified by 'user1';
```

EJEMPLO: Privilegios a nivel de rutina.

```
GRANT EXECUTE ON PROCEDURE TablasFecha3.tabla_fecha TO
user1@'localhost'; SHOW GRANTS FOR user1@localhost;
```