DAW/DAM. UD 7. USUARIOS Y EXTENSIONES PARTE 1. GESTIÓN DE USUARIOS EN MYSQL

DAW/DAM. Bases de datos (BD)

UD 7. USUARIOS Y EXTENSIONES

Parte 1. Gestión de usuarios en MySQL

Abelardo Martínez y **Pau Miñana**
Basado y modificado de Sergio Badal (www.sergiobadal.com) y Raquel Torres.
Curso 2023-2024

1. Gestión de usuarios

MySQL presenta un esquema de seguridad refinado, flexible y basado en lista de control de accesos. Un usuario en MySQL se identifica por el login e identificador del servidor cliente (IP, nombre), esto se basa en el principio de que el usuario que se conecta desde la oficina no tiene porqué ser el mismo que se conecta desde la casa.

En gestores como Oracle, la cosa se complica un poco más. Si quieres saber cómo se realiza en Oracle puedes consultar este manual:

https://orasite.com/tutoriales/administracion/administracion-de-usuarios-en-oracle

En esta unidad veremos cómo crear y manipular usuarios y permisos. Recuerda que, para probar todas las opciones en tu equipo, es recomendable que trabajes directamente sobre la consola de comandos de MySQL.

1.1. Crear usuarios

A partir de la versión de MySQL 5.7, se ha ajustado el método de cifrado de contraseña predeterminado SHA1 y se ha cambiado a SHA2. Con la función de MySQL 5.7 de deshabilitar a los usuarios y la caducidad de los usuarios, las funciones de administración de usuarios y la seguridad de MySQL se mejoran enormemente en comparación con la versión anterior.

En MySQL la sentencia para agregar un "nuevo usuario" es la siguiente:

> **Versión 5.7 o anterior**
>
> **CREATE USER** 'nombre_de_usuario'@'host' **IDENTIFIED BY** 'claveentextoplano';
>
> **Versión 8 o posterior**
>
> **CREATE USER** 'nombre_de_usuario'@'host' **IDENTIFIED WITH mysql_native_password BY** 'claveentextoplano';

donde

- "**nombre de usuario**" es el identificador del perfil o de la persona que se quiere conectar a la BD
- "**host**" es la máquina/equipo/servidor (dirección IP) desde la que se quiere conectar, por lo que realmente no estamos creando un usuario, sino un **perfil de conexión de un usuario (usuario+host)**.
- **"claveentextoplano"** es la contraseña del usuario desde ese host.

> Todos los usuarios (perfiles de conexión de usuarios) y los permisos, que veremos más adelante, se almacenan en una misma tabla llamada user de la base de datos interna mysql, en el servidor MySQL.

Cada **nombre de usuario** se almacena junto a su **host** en una tabla llamada ***user*** de una de las bases de datos del sistema llamada ***mysql***, y ambos campos son clave primaria. Por ello, cuando creamos un nuevo usuario estamos creando realmente un usuario y un host o, como lo hemos denominado anteriormente, un perfil de conexión de un usuario (usuario+host). Para ver la estructura de esta tabla accedemos a la BD y tecleamos DESC:

```
mysql> desc mysql.user;
+-------------------------+-----------------------------------+-
| Field                   | Type                              |
+-------------------------+-----------------------------------+-
| Host                    | char(255)                         |
| User                    | char(32)                          |
| Select_priv             | enum('N','Y')                     |
```

Como podemos comprobar, la **clave primaria compuesta (host+user)** nos permite repetir cuantas veces queramos el nombre de usuario y el host pero nunca repetir la misma combinación de ambos.

Generalmente, al crear usuarios en MySQL especificaremos el host del usuario como localhost para prevenir conexiones desde hosts no deseados. Solo en casos muy específicos cambiaremos ese “localhost” por una IP de una máquina concreta o el comodín “%” que veremos más adelante.

Es importante que entiendas que el host se refiere a la máquina desde donde se conecta el usuario a la base de datos. Por ejemplo, si el usuario se va a sentar físicamente en el mismo portátil o PC donde está la base de datos instalada, deberíamos crear un perfil de conexión de ese usuario para localhost.

1.1.1. Ejemplos

En las siguientes sentencias estamos creando realmente CUATRO USUARIOS (usuario+host) distintos a los que asignaremos permisos distintos:

Usuario 1

Agregar el usuario 'pepegarcia' con la clave 123456 **conectado desde la misma máquina** donde está alojada la base de datos (***localhost***).

```
CREATE USER 'pepegarcia'@'localhost' IDENTIFIED WITH mysql_native
```

En este caso estamos creando un perfil de conexión del usuario 'pepegarcia' para cuando se conecta desde la misma máquina de la base de datos (*localhost*).

Usuario 2

Si queremos especificar permisos para cuando ese mismo usuario se conecta desde una IP determinada -por ejemplo desde su casa- podemos especificarlo de la siguiente manera:

```
CREATE USER 'pepegarcia'@'252.21.12.2' IDENTIFIED WITH mysql_nati
```

En este caso, estamos creando un perfil de conexión del usuario 'pepegarcia' para cuando se conecta desde su casa (252.21.12.2).

Usuario 3

Usuario Pepe García conectado desde la oficina:

```
CREATE USER 'pepegarcia'@'214.168.42.60' IDENTIFIED WITH mysql_na
```

Usuario 4

También podemos asignar permisos a un usuario concreto **cuando se conecte desde cualquier parte**, desde cualquier máquina, usando el **comodín %**:

```
CREATE USER 'pepegarcia'@'%' IDENTIFIED WITH mysql_native_passwor
```

En este caso, estamos creando un perfil de conexión del usuario 'pepegarcia' para cuando se conecta desde cualquier máquina (%).

Para entenderlo mejor, podemos echar un vistazo al contenido de la tabla interna "user" tras ejecutar las sentencias anteriores con el usuario 'pepegarcia'. La tabla mysql.user tiene muchos campos, pero solo nos interesan los dos primeros.

```
mysql> SELECT user, host
    -> FROM mysql.user
    -> WHERE USER = 'pepegarcia';
+------------+---------------+
| user       | host          |
+------------+---------------+
| pepegarcia | %             |
| pepegarcia | 214.168.42.60 |
| pepegarcia | 252.21.12.2   |
| pepegarcia | localhost     |
+------------+---------------+
4 rows in set (0,01 sec)
```

1.2. Cambiar contraseña

Si queremos cambiar (o asignar en caso de inexistencia) una contraseña a un determinado usuario+host, se utilizará ALTER USER como se muestra a continuación:

> **ALTER USER** 'nombre_de_usuario'@'host' **IDENTIFIED BY** 'nueva clave en texto plano';

donde

- "**nombre de usuario**" es el identificador del perfil o de la persona que se quiere conectar a la BD.
- "**host**" es la máquina/equipo/servidor (dirección IP) desde la que se quiere conectar, por lo que realmente es un **perfil de conexión de un usuario (usuario+host)**.
- **"nueva clave en texto plano"** es la nueva contraseña del usuario desde ese host.

Ejemplo

Para cambiar la clave de acceso cuando Pepe se conecta desde la oficina:

```
ALTER USER 'pepegarcia'@'214.168.42.60' IDENTIFIED BY '987654';
```

En el caso anterior, **cambiaríamos ÚNICAMENTE la clave** del usuario 'pepegarcia', pero **solo cuando se conecta desde esa dirección IP**, dejando el resto de claves (desde otros host) intactas.

1.3. Eliminar usuarios

Al igual que puedes eliminar bases de datos y tablas con DROP, también puedes usar DROP USER para eliminar un usuario+host por completo:

```
DROP USER 'nombre_de_usuario'@'host';
```

donde

- "**nombre de usuario**" es el identificador del perfil o de la persona que se quiere conectar a la BD.
- "**host**" es la máquina/equipo/servidor (dirección IP) desde la que se quiere conectar, por lo que realmente es un **perfil de conexión de un usuario (usuario+host)**.

Ejemplo

Para eliminar el **usuario+host** de Pepe conectado **desde la oficina** haremos:

```
DROP USER 'pepegarcia'@'214.168.42.60';
```

En el caso anterior, estaríamos eliminando ÚNICAMENTE el perfil de conexión del usuario 'pepegarcia', pero solo cuando se conecta desde esa dirección IP, dejando el resto de perfiles de conexión (desde otros host) intactos.

1.4. Iniciar sesión con un nuevo usuario

Para iniciar sesión con un usuario concreto, usaremos este comando desde la terminal de Linux o desde la consola de Windows (símbolo del sistema):

```
terminal: mysql -u [nombredeusuario] -p
```

Ejemplo

Para probar los permisos del usuario 'pepegarcia' con clave 123456 haríamos:

```
terminal:~$ mysql -u pepegarcia -p [pulsaríamos ENTER y nos pedir
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.30-0ubuntu0.20.04.2 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current inp
```

Para probar con otro usuario, tendríamos que cerrar la sesión escribiendo quit o exit:

```
mysql> quit
```

Como puedes comprobar, en pantalla aparece la palabra ORACLE. La empresa MySQL AB fue adquirida por la empresa Sun Microsystems en 2008 y ésta fue comprada por la empresa Oracle Corporation en 2010. Pese a ello, los SGBD relacionales **MySQL y Oracle son distintos y 100% independientes**. Recuerda pues que Oracle es una empresa y el SGBD MySQL y el SGBD Oracle son productos de esa misma empresa.

Existe una versión “libre” del SGBD MySQL llamada **MariaDB**, que fue creada por uno de los desarrolladores de MySQL. Es un SDBD también relacional y muy similar al SGBD MySQL. El objetivo de su desarrollo fue el de mantener el software de gestión de base de datos en un modelo de **software libre**. Además, MariaDB incluye prestaciones para soportar grandes volúmenes de datos.

Más info:

https://www.hostingplus.com.es/blog/que-es-mariadb-y-cuales-son-sus-caracteristicas/

2. Permisos de usuarios sobre recursos

2.1. Otorgar permisos

Cuando creamos un usuario+host con la orden CREATE USER, ese usuario no tiene acceso a nada. De hecho, incluso si ese usuario intenta iniciar sesión (con la contraseña indicada), ¡no podrá siquiera acceder a la consola de MySQL!

Por lo tanto, lo siguiente que haremos tras el **CREATE USER** es dar a ese **usuario+host** acceso a la información que necesitará, con esta sintaxis:

> **GRANT** permisos **ON** basededatos.tabla **TO** 'nombre_de_usuario'@'host' [**WITH GRANT OPTION**];

donde

- "**permisos**" son las posibles opciones de permisos comunes que se pueden conceder a un usuario+host. Son las siguientes:

Opción	Funcionalidad
ALL PRIVILEGES	Otorga a un usuario+host acceso completo a una o varias bases de datos.
CREATE	Permite crear nuevas tablas o bases de datos.
DROP	Permite eliminar tablas o bases de datos.
DELETE	Permite eliminar filas de las tablas.
EXECUTE	Otorga permiso para ejecutar una función almacenada o un procedimiento en la base de datos.
INSERT	Permite insertar filas en las tablas.
SELECT	Permite usar el comando SELECT para leer las bases de datos.
UPDATE	Permite actualizar las filas de las tablas.
WITH GRANT OPTION	Permite otorgar o eliminar privilegios de otros usuario+host. Cuando otorgamos permisos a un determinado usuario+host, es posible además, conceder un permiso extra que le permita asignar sus mismos privilegios a otros usuario+host. El perfil ALL PRIVILEGES no incluye GRANT OPTION, por lo que habría que dárselo posteriormente.

- "**basededatos**" representa la base de datos para la que damos permisos. El **comodín** * significaría todas las bases de datos, es decir, dar acceso a cualquier base de datos.
- "**tabla**" es la tabla a la que damos permiso. El **comodín** * significaría todas las tablas, es decir, dar acceso a cualquier tabla.
- "**nombre de usuario**" es el identificador del perfil o de la persona que se quiere conectar

a la BD.

- "**host**" es la máquina/equipo/servidor (dirección IP) desde la que se quiere conectar, por lo que realmente es un **perfil de conexión de un usuario (usuario+host)**.
- **WITH GRANT OPTION** permite que un usuario+host pueda dar (y luego revocar) los permisos que le damos (todos o parte de ellos) a CUALQUIER USUARIO.

Aunque no es tan común y quizás no tan legible, también podemos aplicar varios permisos a varios perfiles, todo en una misma orden. Lo que no podemos hacer es asignar esos permisos a varios recursos al mismo tiempo. La sentencia es la siguiente:

> **GRANT** permiso, permiso, permiso ...
>
> **ON** basededatos.tabla, ~~basededatos.tabla, basededatos.tabla~~
>
> **TO** 'nombre_de_usuario'@'host' , 'nombre_de_usuario'@'host' , 'nombre_de_usuario'@'host';

Una vez que has finalizado los permisos que quieres configurar para tus nuevos usuarios, asegúrate siempre de "refrescar" todos los privilegios mediante este comando:

> **FLUSH PRIVILEGES**;

En algunos escenarios no es necesaria la orden anterior, pero si lo ejecutas siempre te aseguras de que los cambios se aplican correctamente (buena praxis). Tras aplicar esa orden, los cambios ahora estarán vigentes.

2.1.1. Ejemplos

Ejemplo 1

Para agregar permisos al usuario+host 'pepegarcia' sobre todas las bases de datos y todas sus tablas se utilizaría este comando:

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

Los asteriscos en este comando se refieren a la base de datos y las tablas (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 el caso anterior, estaríamos dando TODOS LOS PRIVILEGIOS al usuario 'pepegarcia' sobre TODAS LAS BASES DE DATOS y TODAS SUS TABLAS cuando accede desde localhost.

Ten en cuenta que en este ejemplo estamos otorgando a 'pepegarcia' acceso total a toda nuestra base de datos cuando se conecta desde la misma máquina donde está instalada la base de datos. Si bien esto es útil para explicar algunos conceptos de MySQL, puede ser poco práctico para la mayoría de casos de uso y podría poner en serio riesgo la seguridad de tu base de datos. En la medida de lo posible, debes evitar conceder ese permiso a ningún usuario+host sin haberlo meditado y justificado antes. Lo mismo ocurre con @'%'.

Ejemplo 2

Si queremos dar los permisos de inserción, selección y actualización sobre una determinada tabla a un determinado usuario cuando se conecta desde la propia máquina de la base de datos, y otros diferentes si se conecta desde fuera:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON DBJardineria.Empleados TO
GRANT SELECT, INSERT, UPDATE ON DBJardineria.Empleados TO 'pepega
GRANT SELECT ON DBJardineria.* TO 'pepegarcia'@'%'; --
```

Ejemplo 3

Si queremos dar los permisos de inserción, selección y actualización (a la vez) sobre todas las tablas de una determinada base de datos a un determinado usuario cuando se conecta desde la propia máquina de la base de datos o desde fuera:

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON DBJardineria.*
TO 'pepegarcia'@'localhost', 'pepegarcia'@'214.168.42.60', 'pepeg
```

2.2. Revocar permisos

Para revocar un permiso usamos la sentencia:

> **REVOKE** permisos **ON** basededatos.tabla **FROM** 'nombre_de_usuario'@'host' [**WITH GRANT OPTION**];
>
> **Ten en cuenta que, cuando revocas permisos, la sintaxis requiere que utilices FROM en lugar de TO, como se utiliza para otorgar permisos.**

donde

- "**permisos**" son las posibles opciones de permisos comunes que se pueden revocar a un usuario+host. Son las mismas opciones vistas en el otorgamiento de permisos del apartado anterior.
- "**basededatos**" representa la base de datos para la que damos permisos. El **comodín** * significaría todas las bases de datos, es decir, quitar acceso a cualquier base de datos.
- "**tabla**" es la tabla a la que damos permiso. El **comodín** * significaría todas las tablas, es decir, quitar acceso a cualquier tabla.
- "**nombre de usuario**" es el identificador del perfil o de la persona que se quiere conectar a la BD.
- "**host**" es la máquina/equipo/servidor (dirección IP) desde la que se quiere conectar, por lo que realmente es un **perfil de conexión de un usuario (usuario+host)**.
- **WITH GRANT OPTION** permite que un usuario+host pueda revocar los permisos (todos o parte de ellos) a CUALQUIER USUARIO.

Ejemplo

Para revokar los permisos concedidos al usuario 'pepegarcia', se utilizaría:

```
REVOKE ALL PRIVILEGES ON *.* FROM 'pepegarcia'@'localhost';
```

2.2.1. Ejemplos

Creamos los siguientes usuarios:

```
[root] CREATE USER pedro; -- (se crea pedro@%' sin clave)
[root] CREATE USER luis; -- (se crea luis@'%' sin clave)
[root] CREATE USER laura; -- (se crea laura@'%' sin clave)
[root] CREATE USER sara; -- (se crea sara@'%' sin clave)
```

> Debes tener en cuenta que, **cuando no se especifica el host, éste se asume por defecto como @'%'**, y se refiere a la conexión desde cualquier dispositivo (PC, móvil, etc.) del mundo.

Ahora, vamos a realizar varias operaciones con estos usuarios creados.

Ejemplo 1

Si otorgas 3 permisos sobre 1 tabla concreta a un usuario (por ejemplo, DBJardineria.Empleados), este usuario podrá dar (y luego revocar) cada uno de esos 3 permisos sobre esa tabla.

```
[root] GRANT SELECT, INSERT, UPDATE ON DBJardineria.Empleados TO
[root] quit
[pedro] GRANT SELECT ON DBJardineria.Empleados TO laura; -- (se a
[pedro] GRANT INSERT ON DBJardineria.Empleados TO sara; -- (se as
[pedro] GRANT INSERT ON DBJardineria.Empleados TO luis; -- (se as
[pedro] REVOKE INSERT ON DBJardineria.Empleados FROM luis; -- (se
```

Ejemplo 2

Si otorgas 3 permisos sobre 1 base de datos entera a un usuario (por ejemplo, DBJardineria.*), este usuario podrá dar (y luego revocar) cada uno de esos 3 permisos sobre todas las tablas de esa base de datos o, de manera pormenorizada, sobre cada una

de las tablas de DBJardineria.

```
[root] GRANT SELECT, INSERT, UPDATE ON DBJardineria.* TO pedro WI
[root] quit
[pedro] GRANT SELECT ON DBJardineria.Empleados TO laura;
[pedro] GRANT INSERT ON DBJardineria.Clientes TO sara;
[pedro] GRANT INSERT ON DBJardineria.* TO luis;
```

Ejemplo 3

Por último, si otorgas TODOS LOS PERMISOS sobre todas las bases de datos a un usuario (por ejemplo, *.*), este usuario podrá dar (y luego revocar) TODOS LOS PERMISOS sobre todas las bases de datos o, de manera pormenorizada, sobre cada una de las bases de datos y sus tablas.

```
[root] GRANT ALL PRIVILEGES ON *.* TO pedro WITH GRANT OPTION;
[root] quit
[pedro] GRANT SELECT ON DBJardineria.Empleados TO laura;
[pedro] GRANT INSERT ON DBJardineria.* TO sara;
[pedro] GRANT INSERT ON DBSuministros.* TO luis;
```

2.3. Mostrar permisos

Podemos revisar los permisos actuales de un usuario mediante la sentencia:

```
SHOW GRANTS FOR 'nombre_de_usuario'@'host';
```

donde

- "**nombre de usuario**" es el identificador del perfil o de la persona que se quiere conectar a la BD.
- "**host**" es la máquina/equipo/servidor (dirección IP) desde la que se quiere conectar, por lo que realmente es un **perfil de conexión de un usuario (usuario+host)**.

Ejemplo

Si queremos ver todos los permisos del usuario 'pepegarcia' tendremos que pedir los permisos de cada una de las combinaciones usuario+host:

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

2.4. Roles de usuario

MySQL 8.0 agrega la gestión de roles en la administración de usuarios (al igual que la usa Oracle), lo que simplifica el trabajo de administración de la base de datos. Un **rol** en MySQL es una **colección de privilegios a la que se le ha dado un nombre y que se puede otorgar a un usuario o a otro Rol**. Al igual que sucede con las cuentas de usuario, a los roles se les puede conceder y revocar privilegios. A una cuenta de usuario se le pueden conceder roles, los cuales conceden a la cuenta los privilegios asociados con esos roles. Esto permite asociar un conjunto de permisos a una serie de cuentas en lugar de hacerlo de manera individual.

Los privilegios del rol pueden ser de:

- Nivel global
- Base de datos
- Tabla y columnas

Para crear un rol utilizamos la instrucción:

> **CREATE ROLE** [**IF NOT EXISTS**] nombrerol;
>
> Podemos asignar diferentes roles a un mismo usuario, pero solo puede tener activado uno solo por defecto.

donde

- "**nombrerol**" es el nombre identificador del rol.

Una vez que el rol ha sido creado, será necesario añadirle permisos a través de instrucción **GRANT**, puesto que **un rol es una cuenta de usuario que no puede hacer login**. Para asignar y retirar privilegios a los roles utilizamos la misma orden GRANT vista anteriormente. A los roles se les asignar privilegios igual que a los usuarios. Para eliminar los privilegios de un rol utilizamos REVOKE.

Para borrar un rol, tenemos la sentencia:

> **DROP ROLE** [**IF EXISTS**] nombrerol;

Diferencias con Oracle

MySQL presenta algunas **diferencias respecto a Oracle**:

- No existen roles por defecto.
- Los roles no están activos por defecto y es necesario activarlos con la sentencia:

```
SET DEFAULT ROLE nombrerol;
```

2.4.1. Ejemplos

Ejemplo 1

Podemos crear un rol llamado 'creadorCuentas' que solo pueda crear usuarios y no pueda realizar ninguna otra operación. Las sentencias que permiten hacer esto son las siguientes (podemos ver que le asignamos el privilegio de CREATE USER sobre todos los objetos de la base de datos):

```
CREATE ROLE IF NOT EXISTS 'creadorCuentas';
GRANT CREATE USER ON *.* TO 'creadorCuentas';
```

Si queremos borrar el rol creado anteriormente:

```
DROP ROLE IF EXISTS 'creadorCuentas';
```

Ejemplo 2

Crear un rol llamado 'permisosPepe':

```
CREATE ROLE IF NOT EXISTS 'permisosPepe';
```

Asignamos al rol anterior, el *select* de todas las tablas de la base de datos DBJardineria, el *insert* en la tabla Empleados y el *update* y *delete* en la tabla Oficinas.

```
GRANT SELECT ON DBJardineria.* TO 'permisosPepe';
GRANT INSERT ON DBJardineria.Empleados TO 'permisosPepe';
GRANT UPDATE, DELETE ON DBJardineria.Oficinas TO 'permisosPepe';
```

El siguiente paso es asignar el rol a los usuarios con la instrucción GRANT. En este caso, lo asignamos al usuario+host de Pepe que se conecta desde localhost y desde la oficina:

```
GRANT 'permisosPepe' TO 'pepegarcia'@'localhost', 'pepegarcia'@'2
```

Como se ha explicado anteriormente, un usuario puede tener diferentes roles asignados pero solo puede tener activado uno por defecto. En el último paso seleccionamos cuál es el rol por defecto que va a utilizar con la instrucción SET DEFAULT ROLE.

```
SET DEFAULT ROLE 'permisosPepe' TO 'pepegarcia'@'localhost', 'pep
```

3. Ejemplo DBLibros

Suponiendo que en el sistema existen las siguientes bases de datos y tablas:

```
CREATE DATABASE DBLibros;
USE DBLibros;

CREATE TABLE categoria(cat INT PRIMARY KEY);
CREATE TABLE libro(lib INT PRIMARY KEY);
```

El usuario root (superadministador) puede consultar todas las tablas de la base de datos:

```
mysql> show tables from DBLibros;
+--------------------+
| Tables_in_DBLibros |
+--------------------+
| categoria          |
| libro              |
+--------------------+
2 rows in set (0,01 sec)
```

Ejemplo 1

Si otorgamos permisos de *select* al usuario 'pepegarcia' desde el servidor local:

```
GRANT SELECT ON DBLibros.categoria TO 'pepegarcia'@'localhost';
```

Si en vez del usuario root (superaministrador) la consulta “show tables” fuese realizada por el usuario 'pepegarcia', obtendríamos lo siguiente al no tener permiso SELECT para ver todas las tablas:

```
mysql> show tables from DBLibros;
+--------------------+
| Tables_in_DBLibros |
+--------------------+
| categoria          |
+--------------------+
1 row in set (0,01 sec)
```

Si el usuario 'pepegarcia' intentara acceder a una tabla no permitida, el acceso le sería negado:

```
mysql> SELECT *
    -> FROM DBLibros.libro;
ERROR 1142 (42000): SELECT command denied to user 'pepegarcia'@'l
```

Ejemplo 2

Creamos 3 usuarios, a los que solo se les permite conectarse de forma local:

```
CREATE USER 'paco'@'localhost' IDENTIFIED WITH mysql_native_passw
CREATE USER 'ana'@'localhost' IDENTIFIED WITH mysql_native_passwo
CREATE USER 'carmen'@'localhost' IDENTIFIED WITH mysql_native_pas
```

Y ahora, otorgamos permisos de selección y actualización, a todos los usuarios recién creados, para todas las tablas de la base de datos DBLibros:

```
GRANT SELECT, UPDATE ON DBLibros.* TO 'paco'@'localhost', 'ana'@'
```

Ahora, al usuario ana (desde localhost), le otorgaremos todos los permisos para la tabla

"categoria" de la base de datos DBLibros, permitiéndole conceder dichos permisos a cualquier otro usuario existente:

```
GRANT ALL PRIVILEGES ON DBLibros.* TO 'ana'@'localhost' WITH GRAN
```

4. Canales de acceso más comunes

El acceso a una base de datos no solo se realiza desde la consola o terminal de comandos. De hecho, el acceso por consola es la forma de acceso menos utilizada en entornos “reales” (no académicos), excepto para el administrador de la base de datos.

A continuación vamos a ver algunos de los canales (formas) de acceso a una base datos real más comunes.

4.1. Acceso desde una aplicación

En este caso, es una aplicación (un programa informático) el que accede y se comunica con una o varias bases de datos, como puede ser:

- Página web que accede mediante Wordpress (php) o mediante Dyango (Phyton)
- Aplicación de móvil para Android hecha en Kotlin o para iPhone hecha en Swift
- Aplicación de escritorio hecha en Java o en C#
- Etc.

Por ejemplo, éste sería el código para acceder a una base de datos desde Java mediante JDBC:

```
import java.sql.*;

/**
 * ================================================================
 * @author Abelardo Martínez. Based and modified from Sergio Bada
 * ================================================================
 */

public class DBManagementExample {

    /*
     * ----------------------------------------
     * CONSTANTS AND VARIABLES
     * ----------------------------------------
     */
    /*
     * Database name, user and password
     */
    private static final String DBNAME = "DBExample";
    private static final String DBUSER = "pepegarcia";
    private static final String DBPASSWORD = "123456";
```

```
    /*
     * Connection string for selected database
     */
    private static final String CONN_URL = "jdbc:mysql://localhos
            + "?useSSL=false&useTimezone=true&serverTimezone=UTC&
    private Connection cnDB = null;


    public static void main(String[] stArgs) throws ParseExceptic

        try {
            //Establish the connection to the DB
            // This will load the MySQL driver, each DB has its c
            Class.forName("com.mysql.cj.jdbc.Driver");
            // Setup the connection with the DB
            cnDB = DriverManager.getConnection(CONN_URL,DBUSER,DE
            System.out.println("Connected to the database.");

            String stSQLquery = "SELECT * FROM example";
            //create the MySQL select statement
            Statement staSQLSelect = objExample.cnDB.createStatem
            ResultSet rsExample = staSQLSelect.executeQuery(stSQL
            while (rsExample.next()) {
                System.out.println("Id: " + rsExample.getInt("id"
            }
```

4.2. Acceso desde un gestor de bases de datos

Otra manera de acceder a la base de datos más usada y más “amigable” que la consola o el terminal, es el software especializado en bases de datos o gestores de bases de datos.

Para que no confundas el término Sistema Gestores de Bases de Datos (SGBS) con el término Gestor de Bases de Datos y con la propia Base de Datos observa esta tabla:

Sistema Gestor de Bases de Datos	Gestor de Bases de Datos	Base de datos (ejemplos)
SGBD MySQL	MySQL Workbench Navicat PhpMyAdmin HeidiSQL	DBSuminstros, DBJardineria, DBBiblioteca
SGBD Oracle	Toad	DBEducacion, DBClinicas, DBConcesionario
SGBD SQLite	DB Browser	DBmoviles, DBEmpresa, DBMusica

En este caso, tenemos que configurar el gestor para que acceda a cada base de datos con el usuario que más nos convenga. Debemos **evitar en la medida de lo posible usar el propio usuario root** (como vimos en el ejemplo anterior, puramente académico), aunque cuando necesitemos hacer cambios en los metadatos, gestionar usuarios u otras operaciones similares, debemos usar root para tener todos los permisos necesarios.

Respecto a los **perfiles de conexión que incluyen el comodín % en la columna de host**, **debemos usarlos con precaución**, ya que daremos acceso (potencial) a la base de datos desde ese gestor, desde cualquier máquina del mundo. Por tanto, lo más lógico es dar pocos permisos con el perfil de comodín % y más permisos desde IP’s concretas o localhost.

Ahora bien, es muy común utilizar este comodín para usuarios no administradores y en determinados puestos de trabajo, como los comerciales. Una persona comercial debe poder conectarse a la base de datos desde cualquier lugar y poder mostrar información completa del catálogo de productos de la empresa y poder realizar pedidos. Por ello, daremos acceso de lectura (consultas) e inserción (INSERT) con el perfil de conexión del comodín %.

Las dos opciones más seguras son: crear un usuario para dar acceso solo desde nuestra IP o conectarnos directamente al gestor instalado en el servidor mediante un usuario con localhost como host, aunque, como hemos visto antes, el comodín es necesario en muchas situaciones.

5. Bibliografía

- MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/refman/8.0/en/
- Oracle Database Documentation. https://docs.oracle.com/en/database/oracle/oracle-database/index.html
- MySQL Tutorial. https://www.w3schools.com/mysql/
- Mysql Gestión de permisos. https://wiki.cifprodolfoucha.es/index.php?title=Mysql_Gesti%C3%B3n_de_permisos#ROLES
- Tema 1: Lenguaje DCL: Usuarios y permisos en MYSQL. http://linkiafp.duckdns.org/M10%20-%20Administraci%C3%B3n%20de%20sistemas%20gestores%20de%20bases%20de%20datos/UF1/Temario/ASIX_M10_T01_IMPR.pdf