CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNIDAD 7

USUARIOS Y EXTENSIONES (MYSQL)

BASES DE DATOS 22/23
CFGS DAW

PARTE 1 DE 2. GESTIÓN DE USUARIOS EN MYSQL

Revisado y ampliado por:
Abelardo Martínez y Pau Miñana
Autor:
Sergio Badal

ÍNDICE DE CONTENIDO

UD7.1. GESTIÓN DE USUARIOS EN MYSQL

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 por que 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

Veremos en esta unidad 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

Para agregar un "nuevo usuario" es tan simple como ejecutar esta sentencia:

```
CREATE USER 'nombre_de_usuario'@'host' IDENTIFIED BY 'claveentextoplano';
```

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

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 dónde está la base de datos instalada, deberíamos crear un **perfil de conexión de ese usuario para localhost.**

Por ejemplo, para agregar al usuario **andergarcia** con la clave 123456 conectado desde la misma máquina dónde está alojada la base de datos (localhost), se utilizaría este comando:

```
CREATE USER 'andergarcia'@'localhost' IDENTIFIED BY '123456';
```

En el caso anterior, estaríamos creando un perfil de conexión del usuario andergarcia para cuando se conecta desde la misma máquina de la base de datos (localhost).

No obstante, 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 'andergarcia'@'252.21.12.2' IDENTIFIED BY '123456';

En el caso anterior, estaríamos creando un perfil de conexión del usuario andergarcia para cuando se conecta desde su casa (252.21.12.2).

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

CREATE USER 'andergarcia'@'%' IDENTIFIED BY '123456';

En el caso anterior, estaríamos creando un perfil de conexión del usuario andergarcia para cuando se conecta desde cualquier máquina (%).

Piensa que 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. Con esto, cuando creamos un nuevo usuario estamos creando realmente un usuario y un host o, como lo hemos llamado al principio, un **perfil de conexión de un usuario (usuario+host)**.

Quizás lo entiendas mejor si accedes a la BD de tu equipo y muestras la **estructura** de la tabla que te comentamos. Recuerda: tabla ***user*** de la BD interna ***mysql***. Sería algo así:

```
mysql> desc mysql.user;
+-------------------------+-----------------------------------+------+-----+
| Field                   | Type                              | Null | Key |
+-------------------------+-----------------------------------+------+-----+
| Host                    | char(255)                         | NO   | PRI |
| User                    | char(32)                          | NO   | PRI |
| Select_priv             | enum('N','Y')                     | NO   |     |
```

Como ves, la clave primaria compuesta nos permite repetir cuantas veces queramos el nombre de usuario y el host pero nunca repetir la misma combinación de ambos. Si esto no te queda claro, revisa los temas anteriores antes de seguir.

IMPORTANTE

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

Para que lo veas más claro. Piensa que en estas tres sentencias estamos creando realmente TRES USUARIOS (**usuario+host**) distintos a los que asignaremos permisos distintos:

– Ander desde su casa:

CREATE USER 'andergarcia'@'254.68.2.15' IDENTIFIED BY '123456';

– Ander desde la oficina

CREATE USER 'andergarcia'@'214.168.42.60' IDENTIFIED BY '123456';

– Ander desde la misma máquina dónde está instalada la BD

CREATE USER 'andergarcia'@'localhost' IDENTIFIED BY '123456';

Si aún no lo acabas de entender, echa un vistazo al **contenido** de esa tabla interna que te comentamos tras ejecutar las tres sentencias anteriores con el usuario **sergiomarquez**. La tabla *mysql.user* tiene "mil campos", pero solo nos interesan los dos primeros. Sería algo así:

```
mysql>  SELECT user,host FROM mysql.user where user = 'sergiomarquez';
+---------------+-------------+
| user          | host        |
+---------------+-------------+
| sergiomarquez | %           |
| sergiomarquez | 192.168.2.5 |
| sergiomarquez | 192.168.2.6 |
| sergiomarquez | localhost   |
+---------------+-------------+
4 rows in set (0.00 sec)
```

1.2 CAMBIAR PASSWORD

Si más adelante quisieras 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'**;**

Por ejemplo, para cambiar la clave de acceso cuando Ander se conecta desde la oficina:

ALTER USER 'andergarcia'@'214.168.42.60' IDENTIFIED BY '987654';

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

1.3 ELIMINAR USUARIOS

Por otro lado, 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';

Por ejemplo, para eliminar el **usuario+host** de Ander conectado desde la oficina haremos:

DROP 'andergarcia'@'214.168.42.60';

En el caso anterior, estaríamos eliminando ÚNICAMENTE el perfil de conexión del usuario andergarcia, 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 usa este comando desde la consola de Windows (símbolo del sistema) o desde la terminal de Linux:

> mysql -u [nombredeusuario] -p

Por ejemplo, para probar los permisos del usuario **andergarcia** con clave 123 haríamos:

terminal> **mysql -u andergarcia -p** [pulsaríamos ENTER y nos pediría la clave]

Para probar con otro usuario, tendríamos que cerrar la sesión escribiendo lo siguiente:

mysql> **quit**

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u pruebas -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 41
Server version: 8.0.20 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

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 input statement.

mysql>
```

No te asustes si ves la palabra ORACLE en pantalla. 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 productos SGBD MySQL y SGBD Oracle son distintos y 100% independientes, ambos de tipo relacional.

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

Volviendo al punto 1, 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 ALL PRIVILEGES ON 'basededatos'.'tabla' **TO** 'nombre_de_usuario'@'host';

Por ejemplo, para agregar permisos al **usuario+host** andergarcia sobre todas las bases de datos y todas sus tablas se utilizaría este comando:

GRANT ALL PRIVILEGES ON *.* TO 'andergarcia'@'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 andergarcia sobre TODAS LAS BASES DE DATOS y TODAS SUS TABLAS cuando accede desde localhost.

 IMPORTANTE

Ten en cuenta que en este ejemplo estamos otorgando a andergarcia **acceso total** a toda nuestra base de datos cuando se conecta desde la misma máquina dónde 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 alto riesgo la seguridad de tu base de datos.

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.

Como dijimos, usar **ALL PRIVILEGES** puede ser demasiado amplio y puede poner en riesgo la seguridad de tu servidor de base de datos por lo que debes evitar conceder ese permiso a ningún **usuario+host** sin haberlo meditado y justificado antes.

Podríamos hacer una analogía con el uso del **SELECT * FROM:**

¿Podemos usarlo siempre? SÍ. ¿Es siempre la mejor opción? NO.

Estos son los permisos más comunes que puedes conceder a un **usuario+host**.

- **ALL PRIVILEGES**: Como vimos antes, esto le otorgaría a un usuario+host acceso completo a una o varias bases de datos.
- **CREATE**: Permite crear nuevas tablas o bases de datos.
- **CREATE VIEW**: Permite crear vistas.
- **DROP**: Permite eliminar tablas o bases de datos.
- **DELETE**: Permite eliminar filas de las tablas.
- **INDEX**: Permite crear y borrar índices sobre tablas existentes. Si se dispone del privilegio CREATE, pueden incluirse definiciones de índices en la sentencia CREATE TABLE.
- **INSERT**: Permite insertar filas en las tablas.
- **SELECT**: Les permite usar el comando SELECT para leer las bases de datos.
- **SHOW DATABASE**: Permite listar las bases de datos existentes.
- **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.

Para proporcionar un permiso específico, puede usarse esta sintaxis:

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

Si quieres dar a un **usuario+host** acceso a cualquier base de datos o a cualquier tabla, asegúrate de poner un asterisco (*) en el lugar del nombre de la base de datos o de la tabla.

Ah! y cada vez que actualices o cambies un permiso, asegúrate de usar **FLUSH PRIVILEGES**.

Por ejemplo, si queremos dar los permisos de 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 haríamos algo como:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ciclismo.ciclista TO 'andergarcia'@'localhost';
GRANT SELECT, INSERT, UPDATE ON ciclismo.ciclista TO 'andergarcia'@'192.168.2.6';
GRANT SELECT ON ciclismo.* TO 'andergarcia'@'%';
```

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 puedes hacer es asignar esos permisos a varios recursos al mismo tiempo. De esta manera:

```
GRANT permiso, permiso, permiso ...
ON basededatos.tabla, basededatos.tabla, basededatos.tabla
TO 'nombre_de_usuario'@'host' , 'nombre_de_usuario'@'host' , 'nombre_de_usuario'@'host';
```

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON ciclismo.*
TO 'andergarcia'@'localhost', 'andergarcia'@'214.168.42.60', 'andergarcia'@'254.68.2.15';
```

2.2 REVOCAR PERMISOS

Por otro lado, si necesitas revocar un permiso, esta es la estructura:

REVOKE permisos **ON** basededatos.tabla **FROM** 'nombre_de_usuario'@'host';

 IMPORTANTE

Ten en cuenta que, cuando revocas permisos, la sintaxis requiere que utilices FROM en lugar de TO, como se utiliza para otorgar permisos.

Por ejemplo, para revokar los permisos concedidos al usuario andergarcia, se utilizaría:

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

Cuando otorgas WITH GRANT OPTION a un usuario+host, le dejas dar (y luego revocar) los permisos que le das (todos o parte de ellos) a CUALQUIER USUARIO.

Por ejemplo, para estos 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, este se asume por defecto como @'%', y se refiere a la conexión desde cualquier dispositivo (PC, móvil, etc.) del mundo.

Ahora, con estos usuarios creados:

Si otorgas 3 permisos sobre 1 tabla concreta a un usuario (ej, musica.club), este usuario podrá dar (y luego revocar) cada uno de esos 3 permisos sobre esa tabla.

- [root] GRANT SELECT, INSERT, UPDATE ON musica.club TO pedro WITH GRANT OPTION;
- [root] QUIT
- [pedro] GRANT SELECT ON musica.club TO laura;   -- (se asigna a laura@'%')

- [pedro] GRANT INSERT ON musica.club TO sara; -- (se asigna a sara@'%')
- [pedro] GRANT INSERT ON musica.club TO luis; -- (se asigna a luis@'%')
- [pedro] REVOKE INSERT ON musica.club FROM luis; -- (se revoca a luis@'%')

Si otorgas 3 permisos sobre 1 base de datos entera a un usuario (ej, musica.*), 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 música.

- [root] GRANT SELECT, INSERT, UPDATE ON musica.* TO pedro WITH GRANT OPTION;
- [root] QUIT
- [pedro] GRANT SELECT ON musica.club TO laura;
- [pedro] GRANT INSERT ON musica.cancion TO sara;
- [pedro] GRANT INSERT ON musica.* TO luis;

Por último, si otorgas TODOS LOS PERMISOS sobre todas las bases de datos a un usuario (ej, *.*), 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 musica.club TO laura;
- [pedro] GRANT INSERT ON musica.* TO sara;
- [pedro] GRANT INSERT ON ciclismo.* TO luis;

2.3 MOSTRAR PERMISOS

Por último, puedes revisar los permisos actuales de un usuario ejecutando lo siguiente:

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

Por ejemplo, si queremos ver todos los permisos del usuario **andergarcia** tendremos que pedir los permisos de cada una de las combinaciones **usuario+host:**

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

2.4 ROLES DE USUARIO

MySQL 8.0 (19 April 2018) agrega la gestión de roles en la administración de usuarios, al igual que la usa Oracle. Al mismo tiempo, también 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.

No veremos gestión de roles este curso, al ser algo más propio de Oracle. Si quieres más información, puedes consultar este enlace:

https://wiki.cifprodolfoucha.es/index.php?title=Mysql_Gesti%C3%B3n_de_permisos#ROLES

3. 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, esta forma de acceso suele estar restringida a nivel administrador y es muy típica cuando estamos trabajando en el propio servidor donde se encuentra alojada la base de datos. No obstante, en la mayoría de ocasiones **el acceso por consola no es la forma más común a nivel de usuario estándar**; es decir, para usuarios con roles y funciones diferentes al administrador o "root" de la base de datos.

Estos son algunos de los canales (formas) de acceso a una base datos real más comunes.

3.1 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 pudiéndose tratar de una página web que accede mediante wordpress (**php**) o mediante dyango (**phyton**), o una aplicación de móvil para iPhone hecha en **Swift** o para Android hecha en **Kotlin**, o una aplicación de escritorio hecha en **Java** o en **C#**.

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

```
import java.sql.*;

public class EjemploAccesoBD1 {

    public static void main(String[] args) {

        Connection conexion = null;

        try {
            // Cargar el driver
            Class.forName("com.mysql.jdbc.Driver");

            // Se obtiene una conexión con la base de datos.
            // En este caso nos conectamos a la base de datos prueba
            // con el usuario root y contraseña 1daw
            conexion = DriverManager.getConnection("jdbc:mysql://localhost/prueba", "root", "1daw");

            // Se crea un Statement, para realizar la consulta
            Statement s = conexion.createStatement();

            // Se realiza la consulta. Los resultados se guardan en el ResultSet rs
            ResultSet rs = s.executeQuery("select * from persona");

            // Se recorre el ResultSet, mostrando por pantalla los resultados.
            while (rs.next()) {
                System.out.println(rs.getInt("Id") + " " + rs.getString(2) + " " + rs.getDate(3));
            }
```

driver

cadena de conexión:
driver:sgbd://servidor/bd

3.2 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 no confundir el término Sistema Gestores de Bases de Datos (SGBS) con el término Gestor de Bases de Datos y con la propia Base de Datos, observad esta tabla:

Sistema Gestor de Bases de Datos	Gestor de Bases de Datos	Base de datos (ejemplos)
SGBD MySQL	MySQL Workbench	ciclismo, ,musica, biblioteca
	Navicat	
	PhpMyAdmin	
	HeidiSQL	
SGBD Oracle	Toad	alumnos, clinicas, coches
SGBD SQLite	DB Browser	moviles, cajas, camiones

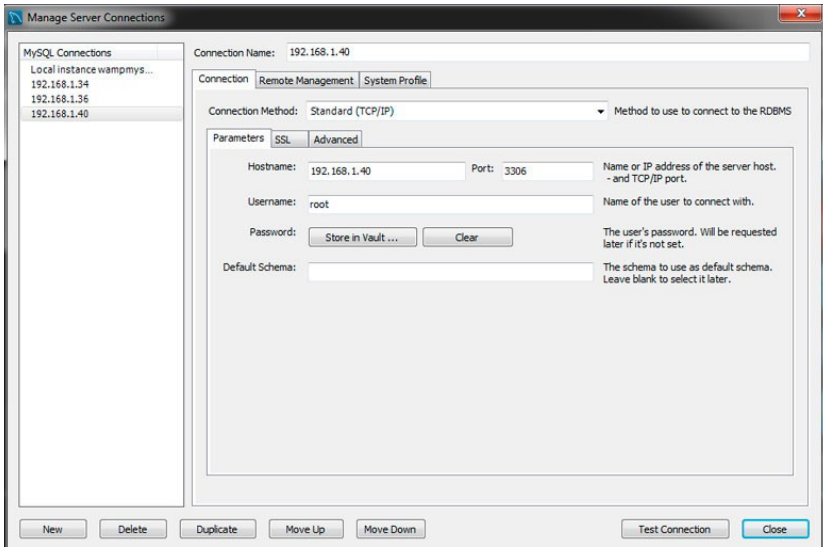
En este caso, tenemos que configurar el gestor para que acceda a cada base de datos con el usuario que más nos convenga. Debemos **evitar en la medida de lo posible usar el propio usuario root** (como vimos en el ejemplo anterior, puramente académico), aunque cuando necesitemos hacer cambios en los metadatos,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

Este sería un ejemplo de configuración de MySQL Workbench:



4. EJEMPLOS

4.1 EJEMPLO 1

Como ejemplo práctico, supongamos que tenemos dos tablas en la BD **weblibros** y esta consulta realizada por el usuario root (superadministador):

```
mysql> show tables from weblibros;
+---------------------+
| Tables_in_weblibros |
+---------------------+
| categoria           |
| libro               |
+---------------------+
2 rows in set (0.00 sec)
```

Después, ejecutamos esta consulta:

GRANT SELECT ON weblibros.categoria **TO** juanperez@localhost;

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

```
mysql> show tables from weblibros;
+---------------------+
| Tables_in_weblibros |
+---------------------+
| categoria           |
+---------------------+
1 row in set (0.00 sec)
```

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

```
mysql> select * from weblibros.libro;
ERROR 1142 (42000): SELECT command denied to user 'juanperez'@'localhost' for table
'libro'
```

4.2 EJEMPLO 2

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

CREATE DATABASE webcursos;

USE webcursos;

CREATE TABLE categoria(cat INT PRIMARY KEY);

CREATE TABLE libro(lib INT PRIMARY KEY);

Crearemos primero 3 usuarios, a los que solo se les permita conectarse de forma local:

CREATE USER 'vanina'@'localhost' **IDENTIFIED BY** 'v4n1n41974';

CREATE USER 'roxana'@'localhost' **IDENTIFIED BY** 'r0x4n4-1275';

CREATE USER 'silvana'@'localhost' **IDENTIFIED BY** '33885409sil';

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

GRANT SELECT, UPDATE ON weblibros.* **TO** 'vanina'@'localhost', 'roxana'@'localhost', 'silvana'@'localhost';

Ahora, al usuario vanina (desde localhost), le otorgaremos todos los permisos para la tabla categoria de la base de datos weblibro, permitiéndole conceder dichos permisos a cualquier otro usuario existente:

GRANT ALL PRIVILEGES ON weblibros.* **TO** 'vanina'@'localhost' **WITH GRANT OPTION;**