
TEMA 8. USUARIOS Y PRIVILEGIOS en MYSQL

[Niveles de privilegios](#)

[Gestión de la seguridad](#)

[Nombres de usuarios y contraseñas](#)

[Crear usuarios. El comando CREATE USER.](#)

[Conceder privilegios: el comando GRANT](#)

[Cambiar la contraseña a una cuenta de usuario. Comando ALTER USER.](#)

[Cambiar el nombre a un usuario. Comando RENAME USER.](#)

[Limitar recursos a un usuario](#)

[Revocar privilegios](#)

[Borrar usuarios](#)

[Para saber más...](#)

1. Niveles de privilegios

Hasta ahora, todas las operaciones que hemos realizado con nuestro SGBD han usado sólo el usuario '**root**', que es el usuario administrador, y que tiene todos los privilegios disponibles en MySQL.

Hemos seguido esta forma de trabajar porque nuestro usuario es local, sólo lo usamos nosotros, pero, sin embargo, normalmente no será una buena práctica dejar que todos los usuarios con acceso al servidor tengan todos los privilegios.

Para conservar la integridad de los datos y de las estructuras será conveniente que sólo algunos usuarios puedan realizar determinadas tareas, y que otras, que requieren mayor conocimiento sobre las estructuras de bases de datos y tablas, sólo puedan realizarse por un número limitado y controlado de usuarios. Eso será imprescindible cuando se trata de un entorno en el que trabajan múltiples usuarios y sólo uno de ellos debe tener credenciales para administrar el servidor de bases de datos. Así, tendremos distintos usuarios, que, dependiendo de lo que vayan a realizar con la base de datos tendrán determinados permisos o privilegios, igual que ocurre con un sistema operativo, donde no todos los usuarios tienen los mismos permisos de acceso.

Los **conceptos de usuarios y privilegios** están **relacionados**. No se pueden crear usuarios sin asignarle al mismo tiempo privilegios. De hecho, la necesidad de crear usuarios está ligada a la necesidad de limitar las acciones que tales usuarios pueden llevar a cabo.

MySQL permite definir diferentes usuarios, y además, asignar a cada uno determinados privilegios en distintos niveles o categorías de ellos.

En el entorno MySQL, existen cinco niveles distintos de 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. Se referencia, como veremos más adelante como `*.*` → todos los objetos del servidor
- **De base de datos:** se refieren a bases de datos individuales, y por extensión, a todos los objetos que contiene cada base de datos. Se refieren nombre_base_de_datos.* → una determinada base de datos y todos sus objetos.
- **De tabla:** se aplican a tablas individuales, y por lo tanto, a todas las columnas de esas tabla. Se refieren nombre_base_de_datos.nombre_tabla.* → una determinada base de datos, una determinada tabla y todos sus objetos.

- **De columna:** se aplican a una columna en una tabla concreta.
- **De rutina:** se aplican a los procedimientos almacenados.

2. Gestión de la seguridad

El **sistema de privilegios** se encarga de establecer quién se puede conectar al servidor y que puede hacer una vez conectado.

Para conectarse es necesario conocer: servidor al que conectarse (IP), usuario y contraseña. El usuario está formado por el nombre de usuario y el equipo desde el que se conecta (IP).

Las cuentas y permisos que gestionan la seguridad se almacenan en el Diccionario de Datos. Algunas tablas implicadas son, que contienen información sobre usuarios y permisos son: (las veremos más adelante)

- ✓ user_privileges,
- ✓ schema_privileges,
- ✓ table_privileges, y
- ✓ column_privileges.

Funcionamiento del sistema de privilegios.

- **1º paso:** El servidor comprueba si debe permitir la conexión, localizando
 - Desde dónde se conecta el usuario (host)
 - El nombre del usuario (user)
 - Para ello se consulta la tabla **mysql.user** (host, user, password)

```
14 • describe user;
```

Field	Type	Null	Key	Default	Extra
Host	char(255)	NO	PRI		
User	char(32)	NO	PRI		
Select_priv	enum('N','Y')	NO		N	
Insert_priv	enum('N','Y')	NO		N	
Update_priv	enum('N','Y')	NO		N	
Delete_priv	enum('N','Y')	NO		N	
Create_priv	enum('N','Y')	NO		N	
Drop_priv	enum('N','Y')	NO		N	

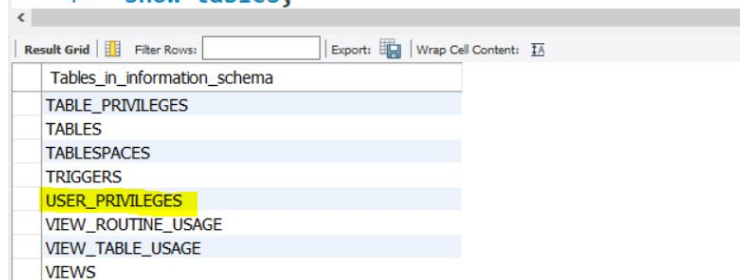
Algunos campos de la tabla user.

- **2º paso:** Asumiendo que el usuario se conecta, el servidor comprueba cada comando que ejecuta para ver si tiene suficientes permisos para hacerlo. Por cada petición en la conexión se comprueba si hay privilegios para efectuarla y para ello consulta a tablas `user`, `db`, `tables_priv`, `columns_priv`, `procs_priv`. Por ejemplo, si intenta seleccionar registros de una tabla en una base de datos o eliminar una tabla de la base de datos, el servidor verifica que tenga el permiso `SELECT` para la tabla o el permiso `DROP` para la base de datos.

TABLAS RELACIONADAS CON USUARIOS Y PERMISOS: Estas tablas están en la base de datos `information_schema` o `mysql` y sólo son accesibles a un usuario con permisos de administrador. Veamos qué contienen alguna de ellas (ya hemos visto antes la tabla `user` y seguiremos trabajando con ella más adelante)

- La tabla **`USER_PRIVILEGES`** determina si se rechazan o permiten conexiones entrantes. Para conexiones permitidas, cualquier privilegio otorgado en la tabla `user_privileges` indica los privilegios **globales** del usuario. Estos privilegios se aplican a todas las bases de datos en el servidor. Esta tabla toma sus datos de otra tabla que se encuentra en otra base de datos de administración de MySQL, la tabla **`mysql.user`** (de la que tb. hablaremos más adelante)

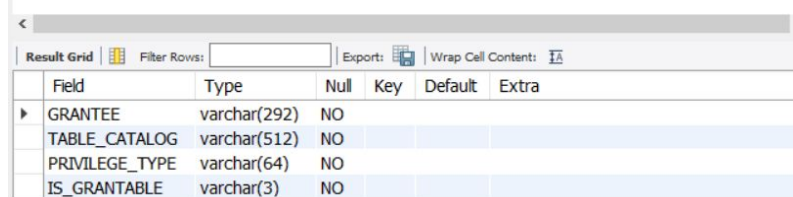
```
3 • use information_schema;
4 • show tables;
```



Tables_in_information_schema
TABLE_PRIVILEGES
TABLES
TABLESPACES
TRIGGERS
USER_PRIVILEGES
VIEW_ROUTINE_USAGE
VIEW_TABLE_USAGE
VIEWS

Esta tabla contiene los siguientes campos:

```
5 • describe user_privileges;
```



Field	Type	Null	Key	Default	Extra
GRANTEE	varchar(292)	NO			
TABLE_CATALOG	varchar(512)	NO			
PRIVILEGE_TYPE	varchar(64)	NO			
IS_GRANTABLE	varchar(3)	NO			

Si hacemos un select a esa tabla, vemos su contenido. Sólo podemos hacerlo si estamos conectados

al servidor como usuario administrador (root)

6 • `select * from user_privileges;`

GRANTEE	TABLE_CATALOG	PRIVILEGE_TYPE	IS_GRANTABLE
'root'@'localhost'	def	SELECT	YES
'root'@'localhost'	def	INSERT	YES
'root'@'localhost'	def	UPDATE	YES
'root'@'localhost'	def	DELETE	YES
'root'@'localhost'	def	CREATE	YES
'root'@'localhost'	def	DROP	YES
'root'@'localhost'	def	RELOAD	YES
'root'@'localhost'	def	SHUTDOWN	YES
'root'@'localhost'	def	PROCESS	YES

Estas mismas consultas podemos hacerlas desde la consola mysql, conectados con el usuario root y la contraseña 1234

```
C:\>cd program files

C:\Program Files>cd mysql

C:\Program Files\MySQL>cd mysql server 8.0

C:\Program Files\MySQL\MySQL Server 8.0>cd bin

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with \n.
```

```
mysql> use information_schema;
Database changed
mysql> describe user_privileges;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| GRANTEE        | varchar(81)   | NO   |     |          |       |
| TABLE_CATALOG | varchar(512)  | NO   |     |          |       |
| PRIVILEGE_TYPE | varchar(64)   | NO   |     |          |       |
| IS_GRANTABLE   | varchar(3)    | NO   |     |          |       |
+-----+-----+-----+-----+-----+-----+
```

```
mysql> select * from user_privileges;
```

GRANTEE	TABLE_CATALOG	PRIVILEGE_TYPE	IS_G
'root'@'localhost'	def	SELECT	YES
'root'@'localhost'	def	INSERT	YES
'root'@'localhost'	def	UPDATE	YES
'root'@'localhost'	def	DELETE	YES
'root'@'localhost'	def	CREATE	YES
'root'@'localhost'	def	DROP	YES
'root'@'localhost'	def	RELOAD	YES
'root'@'localhost'	def	SHUTDOWN	YES
'root'@'localhost'	def	PROCESS	YES
'root'@'localhost'	def	FILE	YES
'root'@'localhost'	def	REFERENCES	YES
'root'@'localhost'	def	INDEX	YES
'root'@'localhost'	def	ALTER	YES
'root'@'localhost'	def	SHOW DATABASES	YES
'root'@'localhost'	def	SUPER	YES
'root'@'localhost'	def	CREATE TEMPORARY TABLES	YES

En esta tabla, **Schema_privileges**, se determina qué usuarios pueden acceder a qué bases de datos desde qué equipo.

Las columnas de privilegios determinan qué operaciones se permiten. Un privilegio otorgado a nivel de base de datos se aplica a la base de datos y a todas sus tablas.

5 • `describe user_privileges;`

Field	Type	Null	Key	Default	Extra
GRANTEE	varchar(292)	NO			
TABLE_CATALOG	varchar(512)	NO			
PRIVILEGE_TYPE	varchar(64)	NO			
IS_GRANTABLE	varchar(3)	NO			

```
mysql> describe schema_privileges;
```

Field	Type	Null	Key	Default	Extra
GRANTEE	varchar(81)	NO			
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
PRIVILEGE_TYPE	varchar(64)	NO			
IS_GRANTABLE	varchar(3)	NO			

6 • `select * from user_privileges;`

GRANTEE	TABLE_CATALOG	PRIVILEGE_TYPE	IS_GRANTABLE
'root'@'localhost'	def	SELECT	YES
'root'@'localhost'	def	INSERT	YES
'root'@'localhost'	def	UPDATE	YES
'root'@'localhost'	def	DELETE	YES
'root'@'localhost'	def	CREATE	YES
'root'@'localhost'	def	DROP	YES
'root'@'localhost'	def	RELOAD	YES
'root'@'localhost'	def	SHUTDOWN	YES
'root'@'localhost'	def	PROCESS	YES

```
mysql> select * from user_privileges;
```

GRANTEE	TABLE_CATALOG	PRIVILEGE_TYPE	IS_GRANTABLE
'root'@'localhost'	def	SELECT	YES
'root'@'localhost'	def	INSERT	YES
'root'@'localhost'	def	UPDATE	YES
'root'@'localhost'	def	DELETE	YES
'root'@'localhost'	def	CREATE	YES
'root'@'localhost'	def	DROP	YES
'root'@'localhost'	def	RELOAD	YES
'root'@'localhost'	def	SHUTDOWN	YES
'root'@'localhost'	def	PROCESS	YES
'root'@'localhost'	def	FILE	YES
'root'@'localhost'	def	REFERENCES	YES
'root'@'localhost'	def	INDEX	YES

Las tablas **table_privileges** y **column_privileges** son similares a la tabla **schema_privileges**, pero son más detalladas: se aplican a nivel de tabla y de columna en lugar de a nivel de base de datos. Un privilegio otorgado a nivel de tabla se aplica a la tabla y a todas sus columnas. Un privilegio otorgado a nivel de columna se aplica sólo a la columna especificada.

A continuación se muestra la estructura de ambas tablas, desde MySQL Workbench y desde la consola mysql.

9 • `describe table_privileges;`

Field	Type	Null	Key	Default	Extra
GRANTEE	varchar(292)	NO			
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
TABLE_NAME	varchar(64)	NO			
PRIVILEGE_TYPE	varchar(64)	NO			
IS_GRANTABLE	varchar(3)	NO			

```
mysql> describe table_privileges;
```

Field	Type	Null	Key	Default	Extra
GRANTEE	varchar(81)	NO			
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
TABLE_NAME	varchar(64)	NO			
PRIVILEGE_TYPE	varchar(64)	NO			
IS_GRANTABLE	varchar(3)	NO			

9 • describe column_privileges;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Field	Type	Null	Key	Default	Extra
▶	GRANTEE	varchar(292)	NO			
	TABLE_CATALOG	varchar(512)	NO			
	TABLE_SCHEMA	varchar(64)	NO			
	TABLE_NAME	varchar(64)	NO			
	COLUMN_NAME	varchar(64)	NO			
	PRIVILEGE_TYPE	varchar(64)	NO			
	IS_GRANTABLE	varchar(3)	NO			

```
mysql> describe column_privileges;
```

Field	Type	Null	Key	Default	Extra
GRANTEE	varchar(81)	NO			
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
TABLE_NAME	varchar(64)	NO			
COLUMN_NAME	varchar(64)	NO			
PRIVILEGE_TYPE	varchar(64)	NO			
IS_GRANTABLE	varchar(3)	NO			

3. Nombres de usuarios y contraseñas

El **nombre de usuario** en MySQL no se limita a un nombre simple, sino que **tiene dos partes**.

La primera consiste en un nombre de usuario, por ejemplo 'usuario_1'. La segunda parte, que aparece separada de la primera por el carácter '@' es un nombre de máquina (host) y es opcional.

Este nombre puede ser bien el de una máquina, por ejemplo, 'localhost' para referirse al ordenador local, o cualquier otro nombre, o bien una dirección IP (IPv4 o IPv6). Si no se especifica nada, se asume que el usuario creado tendrá como segunda parte de su nombre '%', el carácter comodín que sustituye a cualquier host.

4. Crear usuarios. El comando CREATE USER.

En MySQL, al igual que en otros servidores de bases de datos, podemos crear cuentas de usuario con el comando **CREATE USER**. La forma de uso de esta sentencia es:

```
CREATE USER nombre_usuario [IDENTIFIED BY [PASSWORD]  
'contraseña'].
```

Todo lo que va entre [] son parámetros opcionales, que nos ayudan a crear el usuario.

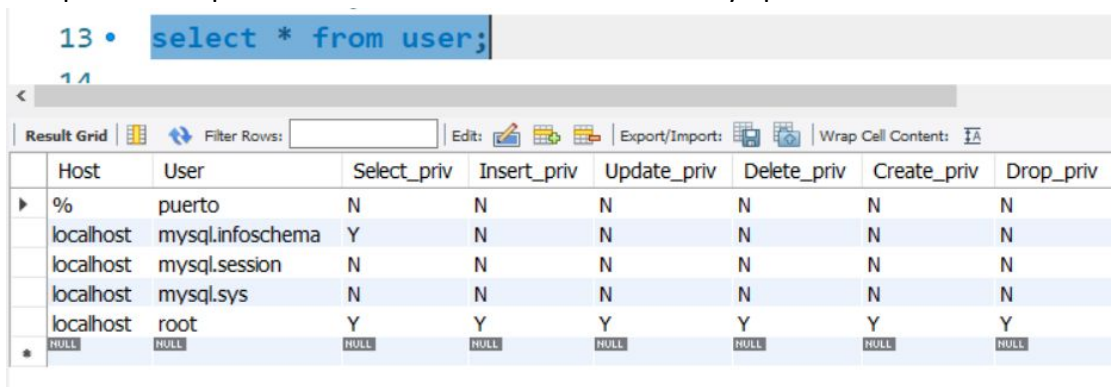
La contraseña debe ir entre comillas. Lo habitual es no incluir la palabra PASSWORD para que sea el sistema el que encripta la contraseña introducida.

Si no ponemos contraseña el usuario se creará sin ella, lo que nos dará problemas a la hora de la conexión. Podremos añadir una contraseña a un usuario o cambiar la que tenía modificando la cuenta de usuario posteriormente.

Ejemplo de creación de un usuario con contraseña:

```
create user puerto identified by "puerto";
```

Comprobamos que se ha creado consultando la tabla mysql.user

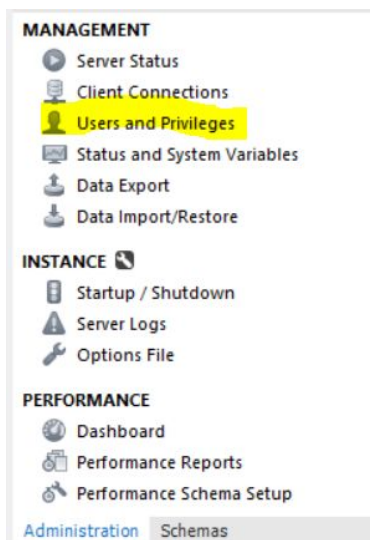


13 • select * from user;

14

Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv
%	puerto	N	N	N	N	N	N
localhost	mysql.infoschema	Y	N	N	N	N	N
localhost	mysql.session	N	N	N	N	N	N
localhost	mysql.sys	N	N	N	N	N	N
localhost	root	Y	Y	Y	Y	Y	Y
* NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

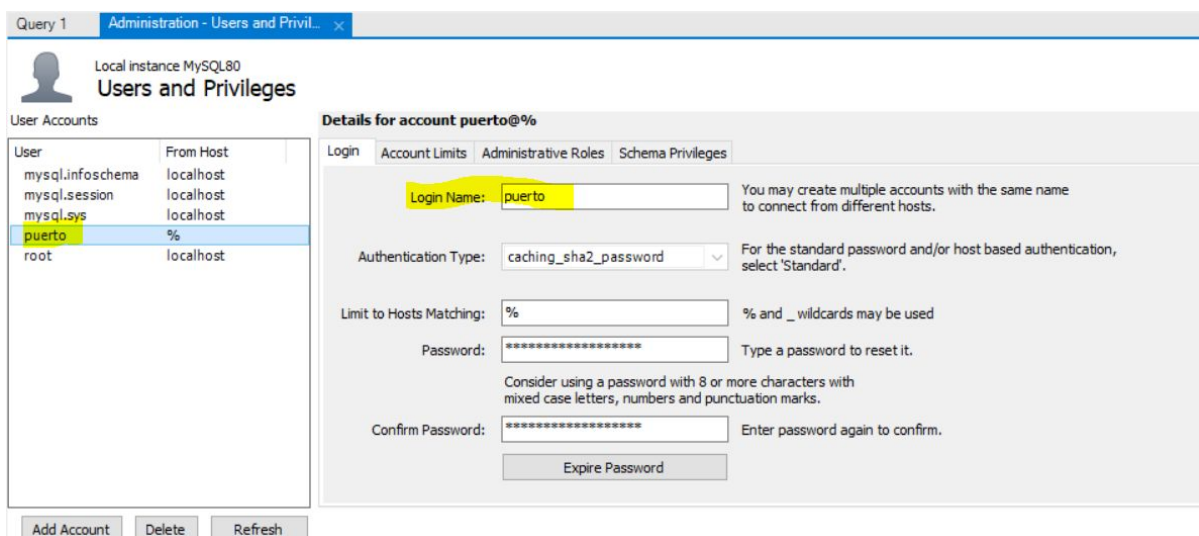
En el entorno gráfico, podemos comprobar los usuarios y sus privilegios:



Desde la consola,

```
mysql> create user puerto identified by 'puerto';
Query OK, 0 rows affected (0.00 sec)
```

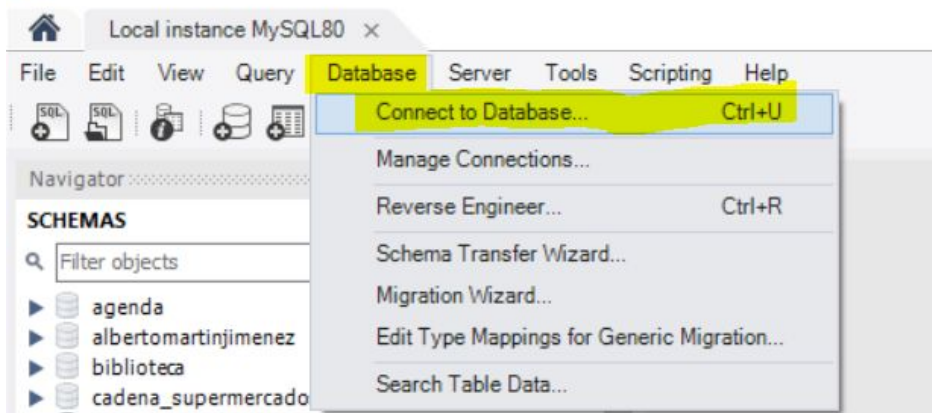
En el entorno gráfico de MySQL Workbench, veríamos los usuarios a través de esta opción de la pestaña “Administración”.



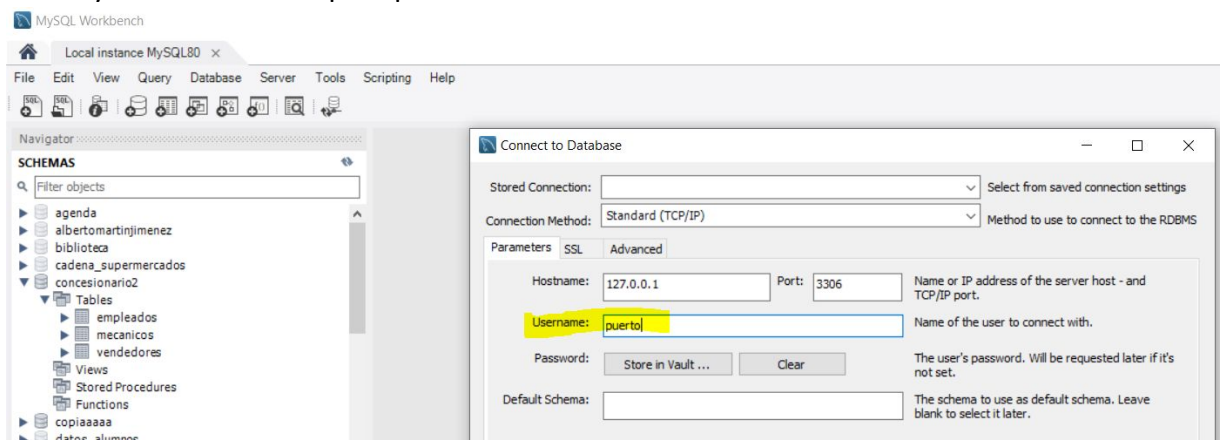
El usuario que acabamos de crear sería un usuario básico, que no tendría ningún permiso, sólo podría conectarse al servidor de bases de datos. Vamos a comprobarlo:

En el entorno gráfico, voy a conectarme con el usuario “puerto” y la contraseña “puerto”:

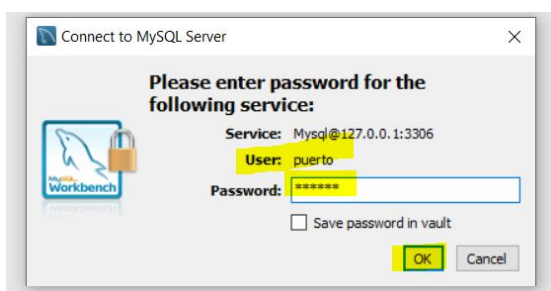
MySQL Workbench



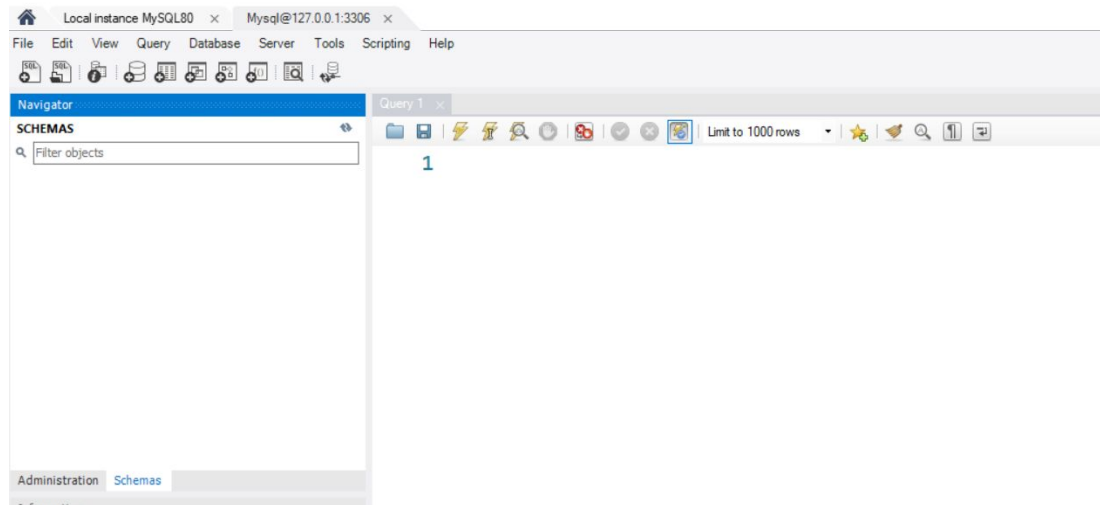
Sustituyo el usuario root por “puerto”:



Me solicita la contraseña:



Se conecta, pero no tengo ninguna base de datos para este usuario; no tiene permisos para ver bases de datos, tablas, etc. Sólo puede conectarse.



5. Conceder privilegios: el comando GRANT

EL comando **GRANT** se utiliza para asignar permisos a una cuenta de usuario que ya existe.

La sintaxis simplificada del comando GRANT es:

```
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']] ... [WITH GRANT OPTION]
```

Donde:

priv_type [(column_list)] se refiere al tipo de privilegio, que puede ser:

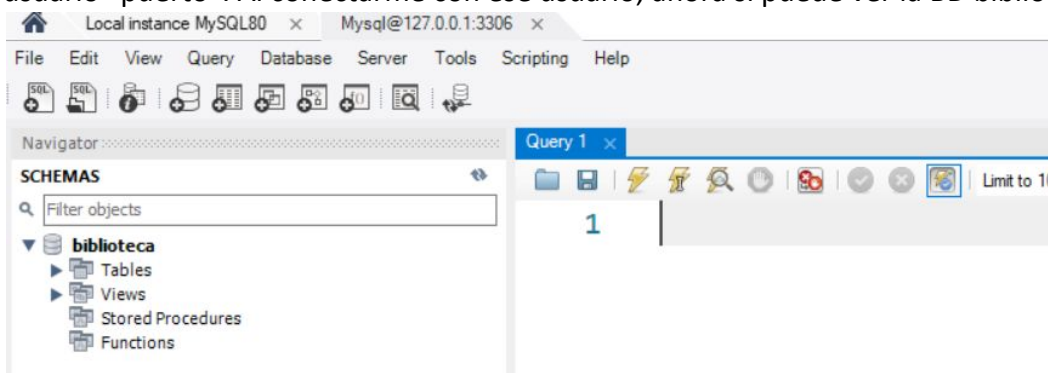
- ☐ **ALL PRIVILEGES**: se conceden todos los privilegios a este usuario.
- ☐ **USAGE**: sin ningún privilegio.
- ☐ **SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE_TMP_TABLE, LOCK_TABLES, CREATE_VIEW, SHOW_VIEW, CREATE_ROUTINE, ALTER_ROUTINE, EXECUTE y GRANT**.
- **ON**: hace referencia a los objetos a los que se aplican los privilegios. El formato es:
base_de_datos.tabla, *.* Por ejemplo: ventas.*, ventas.productos...,
- **TO**: hace referencia al usuario al que se le conceden los privilegios. El formato es:
usuario@'host'. Otros ejemplos: user1@'%', sergio@'localhost'

- **IDENTIFIED BY:** contraseña se indica en esta parte y se escribe en texto plano.
- **WITH GRANT OPTION:** opcional, indica que el usuario en cuestión puede a la vez otorgar privilegios a otros usuarios.
- **REQUIRE:** Opciones de seguridad en el acceso relacionadas con SSL.

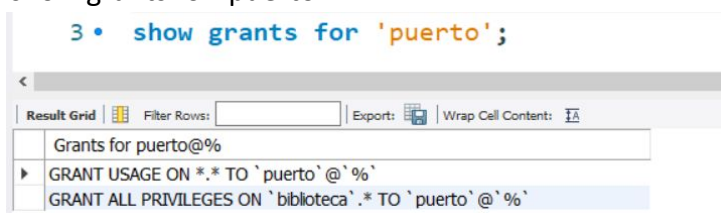
Ejemplo:

```
GRANT ALL ON biblioteca.* TO 'puerto';
```

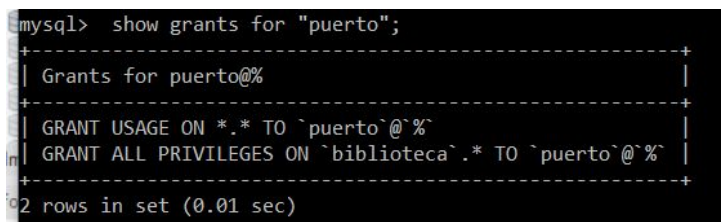
La ejecución de esta sentencia concede todos los permisos sobre la base de datos BIBLIOTECA al usuario "puerto". Al conectarme con ese usuario, ahora sí puede ver la BD biblioteca y sus tablas.



Para ver los permisos asociados al usuario "puerto" utilizamos el comando:
show grants for 'puerto'



desde la consola:



Para conceder privilegios globales se usa ON *.* , para indicar que los privilegios se conceden en todas las tablas de todas las bases de datos.

Para conceder privilegios en bases de datos concretas se usa ON nombre_db.* , indicando que los privilegios se conceden sobre todas las tablas de la base de datos 'nombre_db'.

Usando ON nombre_db.nombre_tabla, concedemos privilegios de nivel de tabla para la tabla y base de datos especificada. Si lo que queremos es conceder privilegios de columna, se usa la sintaxis tipo_privilegio (lista_de_columnas), [tipo_privilegio (lista_de_columnas)].

La opción **WITH GRANT OPTION** permite que el usuario al que le han concedido permisos pueda, a su vez, concederlos a otros usuarios.

Otros privilegios que se pueden conceder son:

- **CREATE:** permite crear nuevas tablas.
- **DELETE:** permite usar la sentencia DELETE.
- **DROP:** permite borrar tablas.
- **INSERT:** permite insertar datos en tablas.
- **UPDATE:** permite usar la sentencia UPDATE.

Se pueden conceder varios privilegios en una única sentencia. Por ejemplo:

```
mysql> grant select,update on biblioteca.libros to usuario_2 identified by '1234';  
Query OK, 0 rows affected (0.00 sec)
```

6. Cambiar la contraseña a una cuenta de usuario. Comando ALTER USER.

Otra operación que podemos realizar es **asignar una contraseña al usuario** o cambiar la actual. Para ello utilizaremos la sentencia **ALTER USER**. La forma de uso es:

ALTER USER 'user-name'@'localhost' IDENTIFIED BY 'NEW_USER_PASSWORD';

Después de ejecutar el comando, refrescamos el usuario para actualizar sus características.
FLUSH PRIVILEGES;

Por ejemplo, voy a cambiar la contraseña al usuario 'puerto'

```
4 • ALTER USER 'puerto' IDENTIFIED BY '1234';  
5 • FLUSH PRIVILEGES;
```

Y en la consola, podemos hacerlo igual:

```
mysql> ALTER USER 'puerto' IDENTIFIED BY '1234';FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.01 sec)  
  
Query OK, 0 rows affected (0.00 sec)
```


7. Cambiar el nombre a un usuario. Comando RENAME USER.

También podemos **cambiar el nombre de usuario** mediante la sentencia:

RENAME USER nombre_viejo TO nombre_nuevo

```
mysql> rename user puerto2 to rupertio;  
Query OK, 0 rows affected (0.00 sec)
```

Voy a cambiar el nombre al usuario 'puerto' por 'usuario1':

7 • **RENAME USER 'puerto' TO 'usuario1';**

Ahora compruebo los usuarios:

```
8 • use mysql;  
9 • select * from user;
```

Result Grid								
Filter Rows:								
Edit: Export/Import: Wrap Cell Content: IA								
	Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	
▶	%	usuario1	N	N	N	N	N	N
	localhost	mysql.infoschema	Y	N	N	N	N	N
	localhost	mysql.session	N	N	N	N	N	N
	localhost	mysql.sys	N	N	N	N	N	N

8. Limitar recursos a un usuario:

Es posible aplicar más restricciones de forma independiente para cada usuario, entre las que se encuentran:

- Número de **consultas** que un usuario pueda hacer cada hora.
- Número de **updates** que un usuario puede hacer cada hora.
- Número de veces que un usuario puede **acceder al servidor** a la hora.
- Número de **conexiones simultáneas** permitidas para cada usuario (como max_user_connections pero a nivel individual en lugar de global).

Estos límites tendrán un contador para cada acceso del usuario, excepto cuando su consulta sea servida a través de la cache. Este tipo de contextos se establecen en la tabla mysql.user y los podemos aplicar mediante **GRANT**.

En el siguiente ejemplo estableceremos todos estos límites para el usuario 'usuario2' cuando sus conexiones se realizan desde localhost:

```
CREATE USER 'usuario2'@'localhost' IDENTIFIED BY '1234'  
WITH MAX_QUERIES_PER_HOUR 20  
MAX_UPDATES_PER_HOUR 10  
MAX_CONNECTIONS_PER_HOUR 5  
MAX_USER_CONNECTIONS 2;
```

Para modificar los límites de una cuenta existente, se usa ALTER USER. La siguiente instrucción cambia el límite de consulta para usuario2 a 100:

```
ALTER USER 'usuario2'@'localhost' WITH MAX_QUERIES_PER_HOUR 100;
```

La declaración solo modifica el valor límite especificado y deja la cuenta sin cambios.

Para poner a 0 los contadores de límites, ejecutamos el comando:

```
ALTER USER 'usuario2'@'localhost' WITH MAX_CONNECTIONS_PER_HOUR 0;
```

9. Revocar privilegios

La sentencia **REVOKE** permite a los administradores del sistema revocar privilegios y roles, que pueden ser revocados de las cuentas y roles de los usuarios. Su sintaxis es la siguiente

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

La sintaxis, como vemos, es similar a la de GRANT.

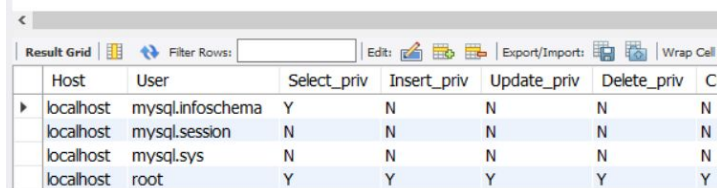
Por ejemplo, creo un usuario 'prueba'@'localhost' y le doy permisos de lectura en la tabla libros de la base de datos biblioteca

```
CREATE USER 'prueba'@'localhost' IDENTIFIED BY 'prueba';  
GRANT select ON biblioteca.libros TO 'prueba'@'localhost';
```

10. Borrar usuarios

La sentencia DROP USER declaración elimina una o más cuentas de MySQL y sus privilegios. Si queremos borrar al usuario1 que acabábamos de crear:

```
25 • DROP USER 'usuario1';  
26 • select * from user;
```



	Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Cr
▶	localhost	mysql.infoschema	Y	N	N	N	N
	localhost	mysql.session	N	N	N	N	N
	localhost	mysql.sys	N	N	N	N	N
	localhost	root	Y	Y	Y	Y	Y

11. Para saber más...

Si te han quedado dudas o quieres ampliar más cosas sobre el tema de usuarios y permisos en Mysql, puedes consultar los siguientes enlaces:

- [Manual de referencia de MySQL 8.0](#)
- <https://tutorialesenpdf.com/mysql/>
- [Videotutorial crear y eliminar usuarios en MySQL Workbench](#)
- [Videotutorial Otorgar privilegios a usuarios en MySQL Workbench](#)