

**BASES DE DATOS**

**Desarrollo de Aplicaciones Web  
GESTIÓN DE BASES DE DATOS**

**Administración de Sistemas Informáticos en Red**

**USUARIOS Y PERMISOS**

**Apuntes**

**Luis Dorado Garcés**

**Basado en el trabajo de Ángel D. Fernández González**

Contenido

[1 INTRODUCCIÓN 2](#_Toc72477094)

[2 LENGUAJE DE CONTROL DE DATOS (DCL) 3](#_Toc72477095)

[3 NIVELES DE SEGURIDAD 4](#_Toc72477096)

[3.1 Primer Nivel: Servidor MySQL (tabla ‘mysql.user’) 4](#_Toc72477097)

[3.2 Segundo Nivel: Bases de datos 6](#_Toc72477098)

[3.2.1 Permisos a nivel de todas las BD (tabla ‘mysql.user’) 6](#_Toc72477099)

[3.2.2 Permisos a nivel BD concretas (tabla ‘mysql.db’) 6](#_Toc72477100)

[3.3 Tercer Nivel (objetos de una BD) 8](#_Toc72477101)

[3.3.1 Permisos sobre tablas (tabla ‘mysql.tables\_priv’) 8](#_Toc72477102)

[3.3.2 Procedimientos y funciones (tabla ‘mysql.procs\_priv’) 10](#_Toc72477103)

[3.4 Sistema de comprobación de permisos 11](#_Toc72477104)

[3.5 AMPLIACIÓN 12](#_Toc72477105)

[4 GESTIÓN DE USUARIOS 13](#_Toc72477106)

[4.1 Introducción 13](#_Toc72477107)

[4.2 SQL 13](#_Toc72477108)

[4.2.1 Creando nuevos usuarios 13](#_Toc72477109)

[4.2.2 Modificando usuarios 15](#_Toc72477110)

[4.2.3 Bloqueo / Desbloqueo de cuentas 17](#_Toc72477111)

[4.2.4 Borrando usuarios 18](#_Toc72477112)

[4.3 MySQL WorkBench 19](#_Toc72477113)

[4.4 AMPLIACIÓN 20](#_Toc72477114)

[4.4.1 Consulta de lista de usuarios del sistema 21](#_Toc72477115)

[4.4.2 Consulta de usuario conectado 21](#_Toc72477116)

[4.4.3 Consulta de estadísticas de conexiones 21](#_Toc72477117)

[4.4.4 Conexiones seguras 21](#_Toc72477118)

[4.4.5 Opciones de limitación de recursos 21](#_Toc72477119)

[4.4.6 Conexiones simultáneas 21](#_Toc72477120)

[4.4.7 Opciones de autentificación 21](#_Toc72477121)

[5 GESTIÓN DE PERMISOS 21](#_Toc72477122)

[5.1 GRANT y REVOQUE 21](#_Toc72477123)

[5.2 Permisos en MySQL 21](#_Toc72477124)

[5.3 Consulta de permisos de un usuario 22](#_Toc72477125)

[5.4 GRANT 23](#_Toc72477126)

[5.4.1 Permiso para crear usuarios 24](#_Toc72477127)

[5.4.2 Permisos a nivel global 24](#_Toc72477128)

[5.4.3 Permisos a nivel de bases de datos 28](#_Toc72477129)

[5.4.4 Permisos a nivel de tabla 31](#_Toc72477130)

[5.4.5 Permisos sobre rutinas (procedimientos almacenados/funciones) 32](#_Toc72477131)

[5.4.6 Ampliación: Permisos a nivel de columnas 33](#_Toc72477132)

[5.5 REVOKE 33](#_Toc72477133)

[5.6 Opción WITH GRANT OPTION 34](#_Toc72477134)

[5.7 Roles 35](#_Toc72477135)

[5.7.1 Roles en MySQL 8 36](#_Toc72477136)

# INTRODUCCIÓN

Si queremos acceder a información guardada en una base de datos, debemos tener establecido algún sistema de permisos que especifique que operaciones van a poderse realizar con los datos guardados.

Además, el servidor MySQL también va a permitir gestionar ciertos permisos administrativos sobre el propio gestor. Este manual nos servirá para profundizar en ello.

# LENGUAJE DE CONTROL DE DATOS (DCL)

Las operaciones que vamos a poder realizar las podemos agrupar en las siguientes categorías:

1. DATOS:Conformaría el Lenguaje de Manipulación de Datos o DML (Data ManipulationLanguage):

* Manipulación de datos: **INSERT**, **UPDATE**, **DELETE** aplicadas a tablas/vistas dentro de una base de datos.
* Consulta de datos: **SELECT** aplicada a tablas/vistas dentro de una base de datos.
* Vistas: **CREATE VIEW**, **ALTER VIEW**, **DROP VIEW**.
* [Resto de órdenes **DML**](https://dev.mysql.com/doc/refman/8.0/en/sql-syntax-data-manipulation.html).

1. **ESTRUCTURA Y RESTRICCIONES**: conformaría el Lenguaje de Definición de Datos O DDL (Data DefinitionLanguage):

* Base de datos: **CREATE DATABASE**, **ALTER DATABASE**, **DROPDATABASE.**
* Tablas: **CREATE TABLE**, **ALTER TABLE**, **DROP TABLE.**
* Procedimientos almacenados: **CREATE PROCEDURE**, **ALTER PROCEDURE**, **DROPPROCEDURE**, **EXECUTE**
* Triggers: **CREATE TRIGGER**, **ALTER TRIGGER**, **DROP TRIGGER**
* [Resto de órdenes **DDL**](https://dev.mysql.com/doc/refman/8.0/en/sql-syntax-data-definition.html).

1. **SEGURIDAD:** conformaría un sublenguaje del Lenguaje de Definición de Datos, denominado Lenguaje de Control de Datos O dcl(Data Control Language):

* Usuarios: **CREATE USER**, **ALTER USER**, **DROP USER**
* Gestión de permisos: **GRANT**,**REVOKE**
* [Resto de órdenes **DCL**](https://dev.mysql.com/doc/refman/8.0/en/sql-syntax-server-administration.html).

MySQL incorpora un sistema de seguridad en el que, con las órdenes definidas en el DCL, especifica qué permisos DDL y DML va a tener un usuario.

# NIVELES DE SEGURIDAD

## Primer Nivel: Servidor MySQL (tabla ‘mysql.user’)

El primer nivel lo compone el propio servidor MySQL.

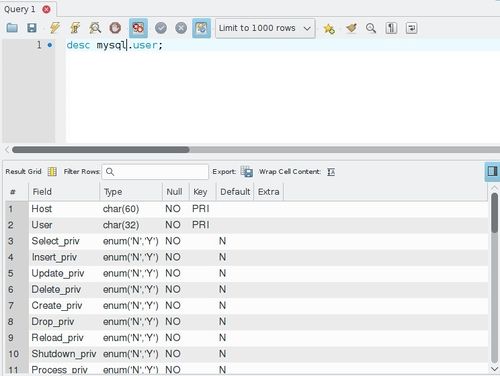
Debemos de determinar qué usuarios tienen permiso de conexión al servidor MySQL.

La información que va a guardar MySQL para determinar si un usuario tiene permiso de conexión es:

* Nombre del usuario
* Password del usuario
* Dirección IP/Nombre del host desde el que el usuario está intentando conectarse al servidor.

Toda esa información está guardada en la tabla user de la base de datos MySQL.

Si ejecutamos la orden SQL: descmysql.user podemos observar lo siguiente:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_1.jpg)

1. **Host**: Indica la dirección ip/nombre del host desde el cual el usuario va a poder conectarse[[1]](#footnote-2).

Los valores más comunes para este campo suelen ser:

* localhost/127.0.0.1: esto implica que el usuario sólo puede conectarse vía ssh o estando físicamente en la misma máquina donde se encuentra el servidor MySQL.
* %: Cualquier dirección IP.
* A.B.C.D: Una dirección ip concreta. Por ejemplo: 192.168.1.10

Es posible utilizar el símbolo '%' como cualquier parte de una dirección ip de la forma: 192.168.1.%

Es posible utilizar el símbolo '%' como cualquier parte de un nombre de host de la forma: '%.mysql.com'

Más información [**en este enlace**](https://dev.mysql.com/doc/refman/8.0/en/account-names.html).

1. **User**: El nombre del usuario que tiene acceso al servidor.
2. **Password**: El password del usuario. Está encriptada.

**Importante**: Recordad que los usuarios del servidor MySQL no tienen nada que ver con los usuarios creados a nivel del sistema operativo.

Una vez creado un usuario, éste puede conectarse al MySQL, pero no puede realizar ninguna acción ya que no tiene establecido ningún tipo de permiso.

Podemos observar cómo hay permisos que se sitúan exclusivamenteen este nivel (al nivel de MySQL) y no contaremos con ellos en el resto de niveles de seguridad.

Son permisos que afectan al servidor MySQL y no a sus objetos (bases de datos, tablas, ...).

Por ejemplo, el permiso para poder parar el servicio de MySQL está en la columna **shutdown\_priv**y es un permiso que solo tiene sentido (y solo existe) en el nivel del servidor MySQL.

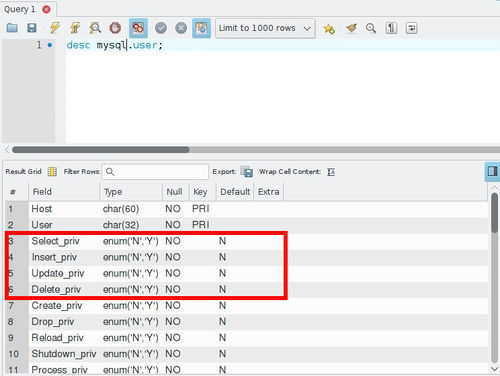
## Segundo Nivel: Bases de datos

### Permisos a nivel de todas las BD (tabla ‘mysql.user’)

Los permisos más comunes que se suelen gestionar son los que le van a permitir realizar operaciones sobre una base de datos.

Cuando creamos un usuario podemos otorgarle permisos a nivel de 'todas las bases de datos' o a nivel de 'una base de datos en particular'.

Los permisos de BBDD globales (los que se aplican a todas las bases de datos) están definidos en la propia tabla 'user' de la base de datos 'mysql'.

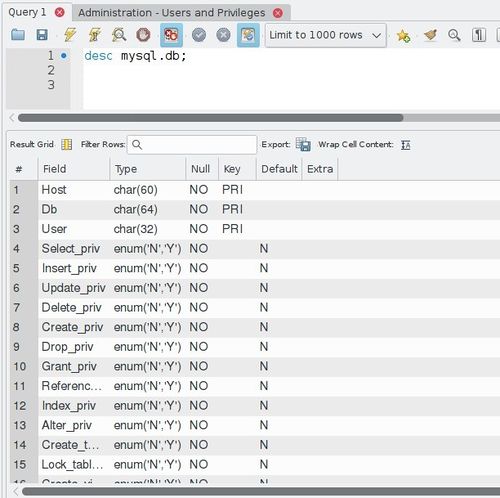
[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_2.jpg)

* En la imagen podemos observar las columnas de la tabla user: 'select\_priv','update\_priv','delete\_priv','insert\_priv'.
* Estas columnas son de tipo ENUM y sólo pueden tener un valor 'Y' o 'N' y por defecto, el valor predeterminado es 'NO'. Un usuario que tenga puesto a 'YES' el valor de estas columnas, tendría como consecuencia que podría realizar operaciones SQL de SELECT, UPDATE, DELETE e INSERT **sobre todas las tablas de todas las bases de datos** de MySQL.

### Permisos a nivel BD concretas (tabla ‘mysql.db’)

Los permisos locales a nivel de una base de datos en particular se encuentran guardados en la tabla **db** de la base de datos **mysql**.

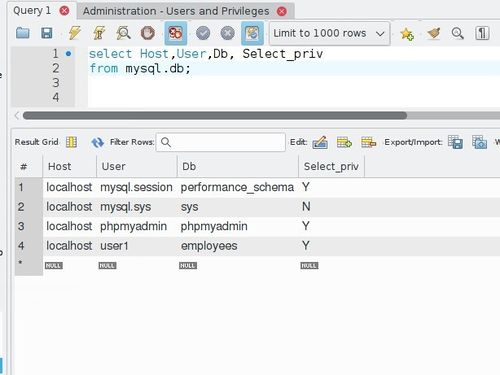
En el caso de que un usuario intentara realizar alguna operación en una base de datos, y no tuviera permiso 'global' para realizarla, iría a esta tabla a buscar si tiene otorgado el permiso.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_4.jpg)

Si vemos la imagen podemos comprobar como en esta tabla:

* Las columnas Host,Db, User conforman la clave primaria de la tabla y por tanto no puede haber dos filas con los mismos valores.
* **Host**: nombre del host cuyos permisos se definen en la BD.
* **Db**: nombre de la base de datos sobre la cual se definen los permisos del usuario.
* **User**: nombre del usuario que tiene permiso de acceso a la base de datos.

Igual que en los permisos a nivel global, un permiso a 'YES' en esta tabla, implicaría que el usuario tendría ese permiso en todas las tablas creadas **dentro de la base de datos**.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_5.jpg)

Podemos comprobar en la imagen superior como el usuario 'user1' tiene permiso de selección (select\_priv) desde 'localhost' en la base de datos 'employees' y por lo tanto va a poder ejecutar la orden SQL SELECT sobre cualquier tabla de esa base de datos.

Si aún no habéis creado ningún usuario y estáis trabajando con root observaréis que no tenéis ningún permiso para ninguna de vuestras BDs en la tabla **db** de la base de datos **mysql. Esto se debe a que ya contáis en root con permisos globales sobre todas las tablas del servidor.**

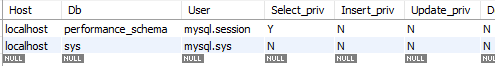


Tabla **db**

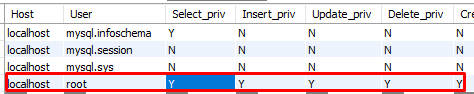


Tabla **user**

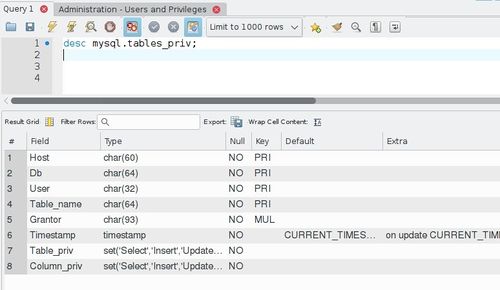
## Tercer Nivel (objetos de una BD)

### Permisos sobre tablas (tabla ‘mysql.tables\_priv’)

A nivel de tabla, un usuario que tenga permiso a este nivel lo tendrá sobre toda la tabla, es decir, **sobre todas sus columnas**.

Es decir, si tengo permiso de selección, podré seleccionar todas las columnas de la tabla.

Los permisos en este nivel se guardan en la tabla **tables\_priv** de la base de datos **mysql**.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_8.jpg)

Como observamos, las columnas de la tabla tienen:

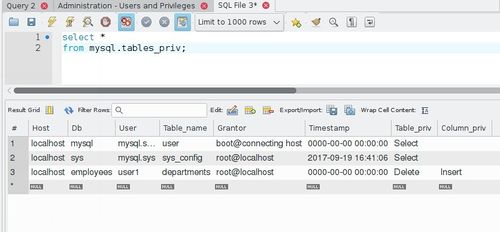
* Clave primaria formada por las columnas (**User,Db,Host y Table\_name**).
* **Grantor**: Indica el usuario que otorgó este permiso.
* **TimeStamp**: El momento en que se otorgó el permiso.
* **Table\_Priv**: Permisos a aplicar en toda la tabla.

Debemos de tener en cuenta que si otorgamos permisos a nivel de tabla, aquí aparecerá el tipo de permiso (por ejemplo, delete,insert).

Es de tipo **SET** y a diferencia de los de tipo 'ENUM' pueden tener varios valores, dentro de las opciones posibles,

* **Column\_priv**: Esta columna es la más confusa. Sencillamente, cada vez que se concede un permiso a nivel de columna (tabla columns\_priv) se creará un registro en esta tabla indicando que hay un permiso concedido a nivel de columna (sin especificar cuál). En caso de que este campo tenga un valor específico, el campo table\_priv no lo tendrá.

Es de tipo **SET** y a diferencia de los de tipo 'ENUM' pueden tener varios valores, dentro de las opciones posibles,

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_10.jpg)

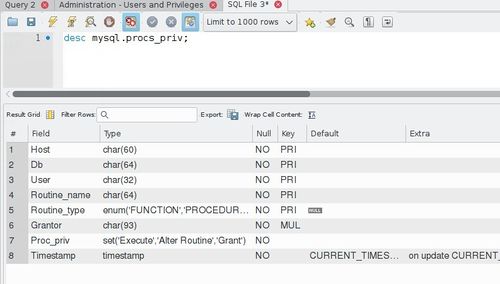
En la imagen se puede ver como a nivel de tabla, el usuario 'user1' tiene permiso de ‘delete’ sobre toda la tabla mientras que tiene un permiso de 'insert' sobre alguna columna de la tabla.

### Procedimientos y funciones (tabla ‘mysql.procs\_priv’)

Son permisos que afectan a los procedimientos y funciones que podamos tener creados en MySQL.

A estas alturas sólo debemos saber que un procedimiento/función es un conjunto de instrucciones SQL agrupadas y que se ejecutan todas juntas cuando se llama al nombre del procedimiento/función.

Por lo tanto, uno de los permisos que vamos a tener es el de ejecución (EXECUTE) para que un usuario pueda llamar al procedimiento/función.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_13.jpg)

Las columnas TimeStamp y Grantor no se utilizan.

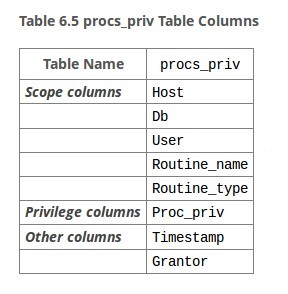
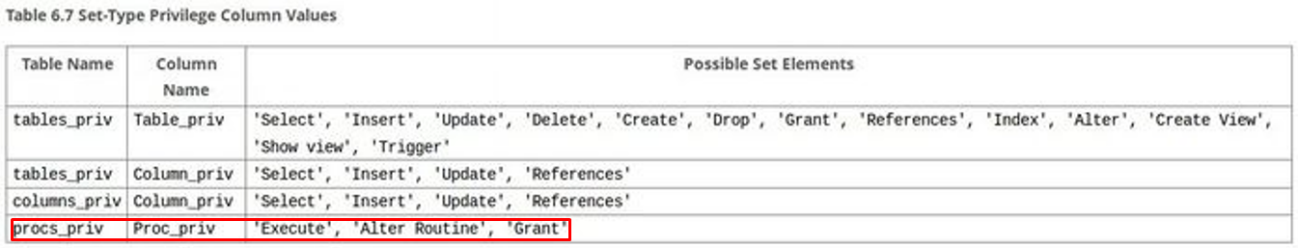
[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_14.jpg)

Imagen obtenida [**de este enlace**](https://dev.mysql.com/doc/refman/5.7/en/grant-tables.html)

Los permisos que podemos otorgar a este nivel son los indicados en la siguiente figura en la última fila (proc\_priv):

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_19.jpg)

Como podemos ver, los permisos que podemos aplicar a nivel de un procedimiento/función concretos son los de poder ejecutarlo (EXECUTE), poder modificarlo o borrarlo (ALTER ROUTINE) y que tengamos el permiso para poder otorgar los dos permisos anteriores a otros usuarios (GRANT). Estos permisos se registran en la tabla **mysql.provs\_priv**.

Disponemos de los mismos permisos en los otros niveles vistos anteriormente, a nivel general (mysql.user) y a nivel de base de datos concreta (mysql.db). En estos niveles, a mayores disponemos del permiso CREATE ROUTINE que permite a un usuario que tenga dicho permiso crear procedimientos/funciones.

## Sistema de comprobación de permisos

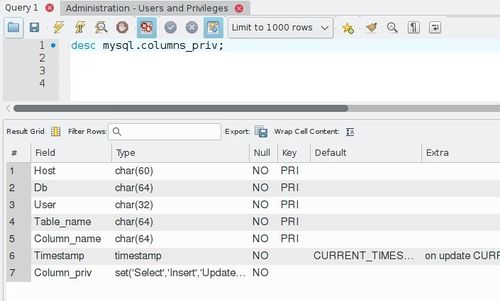
Supongamos que el usuario PEPE del host ejemplo.com quiere ejecutar un UPDATE en la tabla tb\_discosde la base de datosbd\_discografica.

1. Una vez que se inicia la solicitud, MySQL comienza revisando la tabla de usuarios para ver si pepe@ejemplo.com posee privilegios globales UPDATE (tabla user).
2. Si este no es el caso, se revisan a continuación los privilegios de actualización específicos de la base de datos bd\_discografica(tabla db).
3. Si estas tablas no satisfacen la solicitud, MySQL buscará entonces en la tabla tables\_priv para verificar si el usuario pepe@ejemplo.com posee el privilegio de actualización para la tabla tb\_discos.
4. Si aun no se hubieran encontrado los privilegios necesarios, MySQL probará a buscar registros en la tablacolumns\_priv correspondientes a las columnas involucradas en la actualización.

## AMPLIACIÓN

#### Permisos sobre columnas(tabla ‘mysql.columns\_priv’)

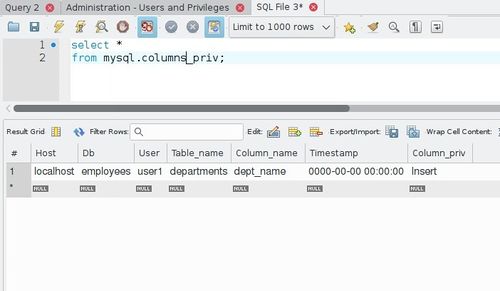
A nivel de columnas, en la tabla **columns\_priv** de la base de datos **mysql** indicamos sobre qué columnas de una tabla voy a tener permiso.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_9.jpg)

Como observamos, las columnas de la tabla tienen:

* Clave primaria formada por las columnas (**User,Db,Host, Table\_name y Column\_name**).
* **TimeStamp**: No se usa.
* **Column\_priv**: permiso que tenemos sobre esta columna de esta tabla.

Es de tipo **SET** y a diferencia de los de tipo 'ENUM' pueden tener varios valores, dentro de las opciones posibles,

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_11.jpg)

En la imagen superior se puede ver como a nivel de columna, el usuario 'user1' tiene permiso de 'insert' sobre la tabla 'departments' de la base de datos 'employees' pero sólo en la columna 'dept\_name'. Por eso la tabla anterior indicaba en la columna 'column\_priv' el valor 'insert'

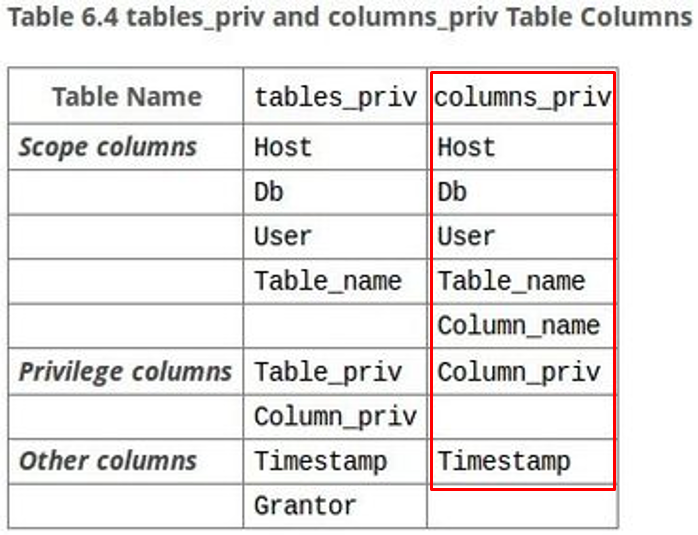
[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_12.jpg)

Imagen obtenida de[**este enlace**](https://dev.mysql.com/doc/refman/5.7/en/grant-tables.html)

El listado de los posibles permisos que podemos otorgar a nivel de tabla/columna se muestran en la siguiente imagen (también están los permisos a nivel de procedimientos y funciones que se comenta en el siguiente punto).

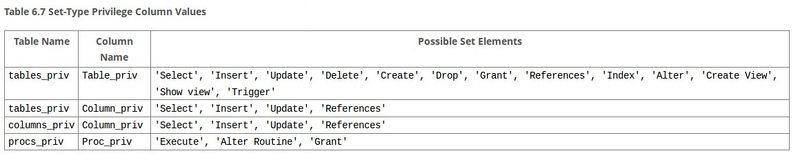
[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_19.jpg)

Imagen obtenida de [**este enlace**](https://dev.mysql.com/doc/refman/5.7/en/grant-tables.html)

#### Lista de privilegios en MySQL

En base a lo visto anteriormente, podemos detallar la lista de privilegios que disponemos en MySQL:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_20.jpg)

Imagen obtenida de [**este enlace**](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html)

#### Modificación directa de tablas de permisos

Cualquier modificación que hagamos **directamente** sobre las tablas del sistema vistas anteriormente (user, db, table\_priv, column\_priv) **no tendrán efecto hasta que**:

1. Reiniciemos el servicio
2. Ejecutemos la orden SQL: FLUSHPRIVILEGES
3. Ejecutemos la herramienta cliente mysqladmin con permisos root:
   * **mysqladmin -u root -p flush-privileges**
4. Ejecutemos la herramienta cliente mysqladmincon permisos root:
   * **mysqladmin -u root -p reload**

# GESTIÓN DE USUARIOS

## [Introducción](https://wiki.cifprodolfoucha.es/index.php?title=Mysql_Creaci%C3%B3n_de_usuarios#Introducci.C3.B3n)

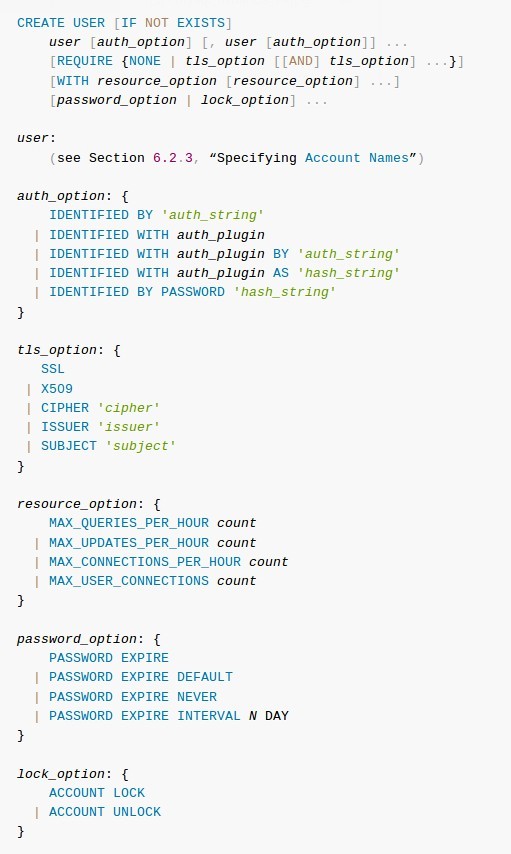
Vimos en el punto anterior cómo es posible manipular la tabla **user** directamente con las órdenes SQL INSERT, UPDATE, DELETE.

Pero a la hora de crear nuevos usuarios, normalmente haremos uso de órdenes SQL pertenecientes al lenguaje de control de datos.

## SQL

### Creando nuevos usuarios

La orden SQL que permite añadir nuevos usuarios es: [**CREATE USER**](https://dev.mysql.com/doc/refman/8.0/en/create-user.html).

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_13.jpg)

Su forma más simple:

CREATEUSERnombre\_usuarioIDENTIFIEDBY'password';

nombre\_usuario tiene el formato: 'usuario'@'host'

Por ejemplo:

CREATEUSER'angel'@'localhost'IDENTIFIEDBY'12345678';

Estaríamos añadiendo una nueva fila a la tabla 'user' de la base de datos 'mysql', con nombre de usuario 'angel' y password '12345678'

Si no podemos la palabra clave 'IDENTIFIED BY' estaríamos creando un usuario sin contraseña, lo cual no es recomendado.

Por ejemplo:

CREATEUSER'anonimo'@'localhost';

**CUIDADO**: Si no indicamos el host, MySQL asignará '%', es decir, cualquier dirección IP.

Por ejemplo:

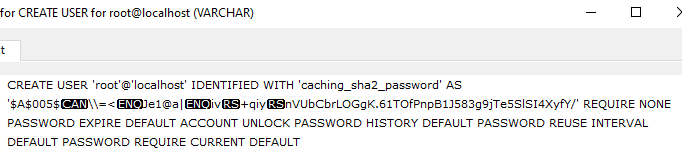
CREATEUSER'anonimo';

**Nota**: Es importante tener en cuenta el formato del usuario de la forma: **'**usuario**'**@**'**host**'**, ya que si ponemos esto: 'usuario@host' estaríamos creando un usuario de nombre 'usuario@host' que puede conectarse desde cualquier PC.

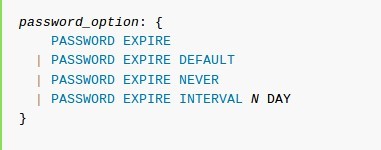
Para ver las sentencias 'CREATE USER' utilizada para crear un determinado usuario, debemos ejecutar la orden SQL: [**SHOW CREATE USER usuario**](https://dev.mysql.com/doc/refman/5.7/en/show-create-user.html)

Por ejemplo, para ver la sentencia 'create user' del usuario actual podemos poner:

SHOWCREATEUSERCURRENT\_USER();

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_1.jpg)

#### Opciones para el password

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_14.jpg)

Más información [**en este enlace**](https://dev.mysql.com/doc/refman/5.7/en/create-user.html#create-user-password-management).

Podemos 'obligar' a que un usuario cambie su contraseña cuando se conecta por primera vez:

CREATEUSER'angel'@'localhost'IDENTIFIEDBY'12345678'PASSWORDEXPIRE;

Poniendo la palabra 'PASSWORD EXPIRE'.

También podemos 'obligar' al usuario a que cambie de password cada cierto número de días:

CREATEUSER'angel'@'localhost'IDENTIFIEDBY'12345678'

PASSWORDEXPIREINTERVAL60DAY;

En el ejemplo, deberá cambiar de password cada 60 días.

Otras opciones indican:

* **NEVER**: que el password nunca 'caduca'.
* **DEFAULT**: El password caducaría en el número de días indicado por la variable del sistema **[default\_password\_lifetime](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \l "sysvar_default_password_lifetime)** que es de 360 días en versiones anteriores a la 5.7.11 y 0 (no caduca) a partir de la versión anterior.

### Modificando usuarios

#### ALTER USER

La orden SQL que permite modificar usuarios es: [**ALTER USER**](https://dev.mysql.com/doc/refman/5.7/en/alter-user.html).

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_22.jpg)

Imagen obtenida de [**este enlace**](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html)

Como vemos la sintaxis es casi idéntica a la del **CREATE USER**, con las mismas opciones, lo que pasa es que estaríamos modificando los valores de dichas opciones para un usuario existente.

Lo único diferente es la opción **IF EXISTS**. Dicha opción sirve para que el gestor MySQL realice la orden SQL 'ALTER USER' sólo en el caso de que el usuario exista. Si no existe no hace nada.

Si no ponemos esta opción a la hora de modificar un usuario, tendríamos un mensaje de error.

Si estamos ejecutando un conjunto de instrucciones MySQL, pararía al llegar a esta instrucción.

Un ejemplo:

ALTERUSERIFEXISTS'angel'@'%'IDENTIFIEDBY'nueva\_password';

#### SET PASSWORD

En el caso del password también disponemos de la orden [SET PASSWORD](https://dev.mysql.com/doc/refman/5.7/en/set-password.html), pero que está **deprecated** (obsoleto) desde la versión 5.7.6 de MySQL.

Un ejemplo:

SETPASSWORDFOR'angel'@'%'=PASSWORD('nuevaPASSWORD');

#### RENAME USER

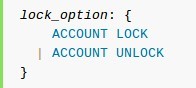
Para **modificar el nombre del usuario** debemos hacer uso de la orden SQL: [RENAME USER](https://dev.mysql.com/doc/refman/5.7/en/rename-user.html).

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_46.jpg)

Un ejemplo:

RENAMEUSER'angel'@'%'to'angel2'@'localhost';

### Bloqueo / Desbloqueo de cuentas

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_15.jpg)

Más información [**en este enlace**](https://dev.mysql.com/doc/refman/8.0/en/create-user.html#create-user-account-locking).

Cuando creamos (o modificamos) una cuenta, podemos indicar si dicha cuenta está bloqueada.

En caso de estarlo, no estaría permitida la conexión al servidor MySQL haciendo uso de dicha cuenta.

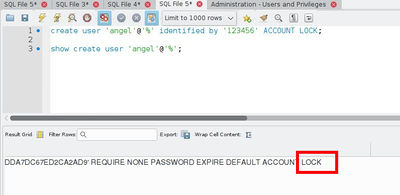
Por ejemplo:

CREATEUSER'angel'@'localhost'IDENTIFIEDBY'12345678'ACCOUNTLOCK;

Por defecto las cuentas, si no indicamos nada, son creadas con el estado desbloqueado, sería lo mismo que poner:

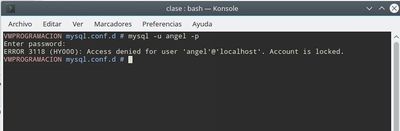
CREATEUSER'angel'@'localhost'IDENTIFIEDBY'12345678'ACCOUNTUNLOCK;

Para saber si una cuenta está bloqueada podemos hacer uso de la orden SQL: **SHOW CREATE USER usuario**.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_16.jpg)

Cada vez que un usuario intenta conectarse haciendo uso de una cuenta bloqueada, la [**variable de estado locked\_connects**](https://dev.mysql.com/doc/refman/5.7/en/server-status-variables.html#statvar_Locked_connects) se incrementa.

Si un usuario intenta conectarse haciendo uso de una cuenta bloqueada, recibirá el siguiente aviso:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_17.jpg)

Y se escribirá una entrada en el fichero de log **error.log**.

### Borrando usuarios

La orden SQL que permite dar de baja a usuarios es: [**DROP USER**](https://dev.mysql.com/doc/refman/5.7/en/drop-user.html)**.**

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_23.jpg)

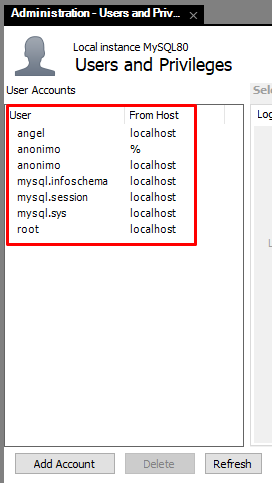
Un ejemplo:

DROPUSERIFEXISTS'angel'@'%';

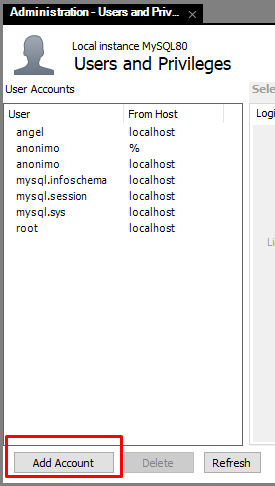
## MySQL WorkBench

Podemos dar de alta, modificar o dar de baja a los usuarios desde la herramienta cliente MySQLWorkBench.

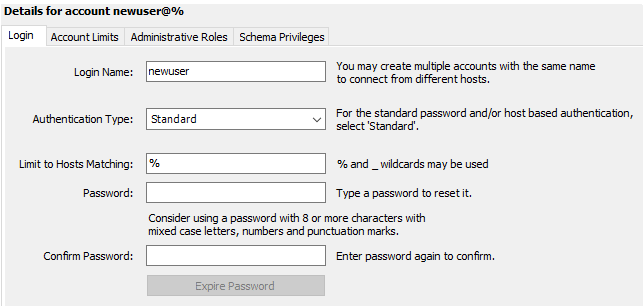
Escogiendo la opción **Users and privileges** podemos ver la lista de usuarios actualmente.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_15.jpg)

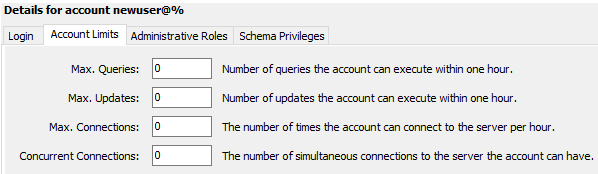
Pulsamos el botón **AddCount** para añadir un nuevo usuario.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_24.jpg)

Introducimos los datos para el nuevo usuario.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_25.jpg)

Si queremos establecer alguna de las limitaciones de los recursos que se describen en la ampliación, debemos pulsar la pestaña **Accountlimits**. Una vez tengamos los datos introducidos **debemos de presionar el botón Apply.**

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_26.jpg)

Las operaciones de baja y modificación son bastantes obvias de ver.

* Para modificar sólo debemos de seleccionar el usuario, hacer los cambios que queramos, y pulsar el botón **Apply**.
* Para borrar un usuario sólo debemos de seleccionar el usuario y pulsar el botón **Delete**.

Más información [**en este enlace**](https://dev.mysql.com/doc/workbench/en/wb-mysql-connections-navigator-management-users-and-privileges.html).

## AMPLIACIÓN

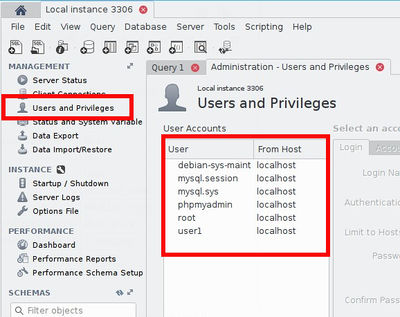
### Consulta de lista de usuarios del sistema

Para mostrar la lista de usuarios del sistema:

1. Orden SQL:

SELECTuser,hostFROMmysql.user;

1. Gráficamente desde 'MySQLWorkBench:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_15.jpg)

Vimos anteriormente que podemos tener varias entradas en la tabla 'user' que pueden 'concordar' con un usuario que se quiera conectar al MySQL.

### Consulta de usuario conectado

Para saber el usuario-host que está conectado disponemos de la orden SQL [CURRENT\_USER()](https://dev.mysql.com/doc/refman/5.7/en/information-functions.html#function_current-user):

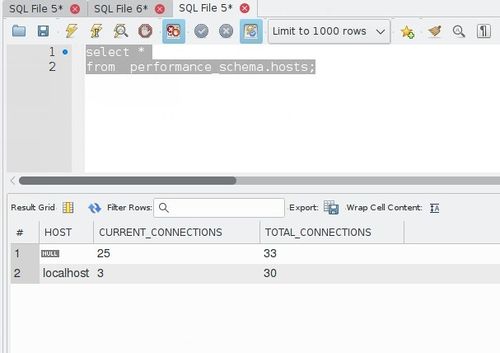
selectCURRENT\_USER()

### Consulta de estadísticas de conexiones

Para mostrar información en forma de estadística debemos acudir a la base de datos **performance\_schema**:

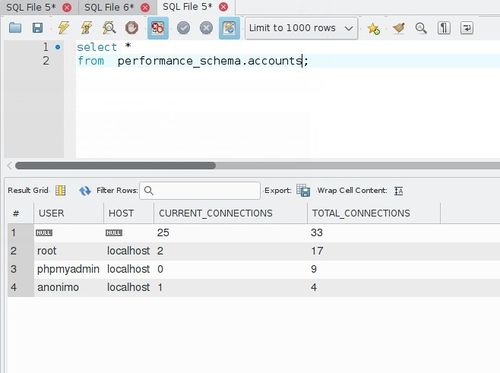
* [Tabla **hosts**](https://dev.mysql.com/doc/refman/5.7/en/hosts-table.html): Información sobre el número de conexiones totales/activas por cada host.

Por ejemplo:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_21.jpg)

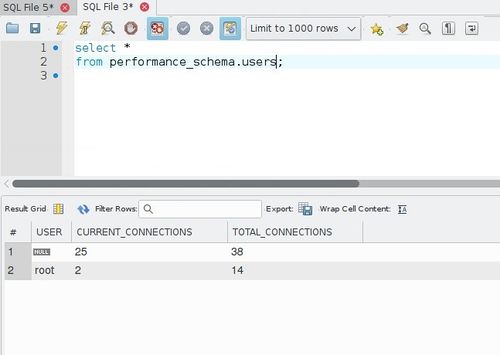
* [Tabla **accounts**](https://dev.mysql.com/doc/refman/5.7/en/accounts-table.html): Información sobre el número de conexiones totales/activas por cada host y usuario.

Por ejemplo:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_22.jpg)

* [Tabla **users**](https://dev.mysql.com/doc/refman/5.7/en/users-table.html): Información sobre el número de conexiones totales/activas por cada usuario.

Por ejemplo:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_23.jpg)

Lógicamente esta información sólo tiene sentido en entornos de trabajo en los que múltiples usuarios están conectándose a MySQL con su propio usuario/password. En entornos de aplicaciones web en los que la conexión siempre utiliza el mismo usuario y de forma local (lo hace el servidor web instalado en el equipo) no tiene sentido, sólo nos servirá para saber cuántas conexiones activas estuvieron activas al mismo tiempo.

### Conexiones seguras

**Conexiones seguras en el servidor**

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_18.jpg)

Más información [**en este enlace**](https://dev.mysql.com/doc/refman/5.7/en/create-user.html#create-user-tls).

Queda fuera de este manual explicar los tipos de certificados y cómo funcionan para garantizar una conexión segura.

Decir que se basan en el uso de 'claves públicas' y 'claves privadas'.

Podemos consultar [**la wikipedia**](https://es.wikipedia.org/wiki/Transport_Layer_Security)para aclarar los conceptos.

Deberíamos conocer los siguientes conceptos:

* **Certificado digital**: identidad electrónica firmada por una **autoridad certificadora** o **CA**
* **Firma digital**: es un sistema para demostrar la autenticidad de un mensaje electrónico.
* **Clave pública**: es una clave usada para la autenticación en una comunicación segura TLS (es el sucesor de SSL) o cifrar mensajes destinados al dueño de la clave.

En MYSQL podemos hacer que el servidor acepte conexiones SSL, es decir, conexiones cifradas.

Para ello debemos seguir los siguientes pasos:

**Habilitando conexiones seguras en MySQL**

|  |  |
| --- | --- |
|  |  |
| Comprobamos si el servidor MySQL acepta conexiones seguras. Para ello debemos mirar las variables del sistema [have\_open\_ssl](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \l "sysvar_have_openssl) y [have\_ssl](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \l "sysvar_have_ssl). Como vemos en la imagen, están deshabilitados. | Podemos comprobar como desde la herramienta cliente 'mysql', si ejecutamos la orden **\s** nos indica que la conexión no es segura. |
|  |  |
| Ahora necesitamos generar los archivos que conformarán el certificado y que serán utilizados por los clientes y por el servidor. Para ello debemos hacer uso de la orden [mysql-ssl-rsa-setup](https://dev.mysql.com/doc/refman/5.7/en/mysql-ssl-rsa-setup.html) que viene con MySQL y que generá los archivos necesarios en el directorio indicado por la variable del sistema **datadir** y que por defecto es **/var/lib/mysql/**. Como a estos archivos tiene que acceder el servidor mysql vamos a indicar con la opción **--uid=mysql** que el propietario de los archivos generados será el usuario que utiliza mysql para acceder al sistema operativo y que por defecto es **mysql**. | Podemos ver los archivos generados con la orden: **sudo find /var/lib/mysql -name '\*.pem' -ls** |
|  |  |
| En versiones anteriores de MySQL debíamos editar el archivo de configuración y añadir las líneas que le indicaban a MySQL donde buscar el certificado (más información [en este enlace](https://www.percona.com/blog/2013/06/22/setting-up-mysql-ssl-and-secure-connections/). Pero en las versiones actuales solamente necesitamos reiniciar el servicio. | Ahora los clientes que intenten conectarse al MySQL, si lo haces desde la propia máquina, intentarán utilizar la conexión SSL por defecto. Podemos comprobarlo conectándose con la orden: **mysql -u root -p -h 127.0.0.1**. Fijarse que debemos de poner el parámetro -h (de host) ya que de esta forma estamos conectándonos por TCTP/IP. Si no lo ponemos estaríamos conectándonos a través del socket. Además debemos poner la IP de la máquina, ya que si ponemos 'localhost' seguiríamos conectándonos a través del socket. |
|  | |
| Si hubiéramos generado los archivos SSL en otro directorio diferente al directorio de datos de MySQL, tendríamos que editar el archivo de configuración del servidor MySQL, indicar la ruta donde se encuentran los archivos (como muestra la imagen), editar el archivo de configuración de las herramientas clientes (en la sección [] correspondiente) y poner la misma información que en el servidor, pero sólo las entradas: **ssl-cert** y **ssl-key**. Recuerda que en las distribuciones basadas en Ubuntu tenemos que dar permiso de acceso a esos archivos si están en un directorio diferente como vimos en el punto de 'Casos especial: los directorios' durante [la instalación](http://wiki.cifprodolfoucha.es/index.php?title=Mysql_Instalaci%C3%B3n#Del_servidor). | |

* En este momento el servidor MySQL por defecto va a intentar siempre establecer una conexión segura cuando un cliente intente conectarse.
* Pero no impedimos que un cliente pueda conectarse de forma insegura.

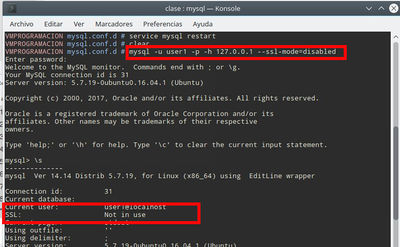
Veamos un ejemplo:

Si ejecutamos la orden

mysql -u user1 -p -h 127.0.0.1 **--ssl-mode=disabled**

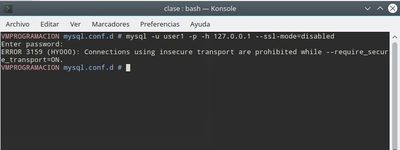
Estaríamos conectando con un usuario creado previamente, pero sin utilizar la conexión SSL.

Podemos comprobarlo:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_10.jpg)

Si quisiéramos que el servidor MYSQL **sólo aceptara conexione seguras**, debemos de escribir en su fichero de configuración: **require\_secure\_transport = ON y r**einiciar el servicio.

Podemos comprobar ahora como las conexiones inseguras no está permitidas.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_11.jpg)

**Nota:** En algunos sitios recomienda poner la opción de configuración en el archivo **/etc/mysql/my.cnf** para evitar conflictos con otras opciones puestas en los diferentes archivos de configuración.

De esta forma siempre cargará, después de cargar las configuraciones indicadas en los include, la orden indicada.

!includedir/etc/mysql/conf.d/

!includedir/etc/mysql/mysql.conf.d/

[mysqld]

require\_secure\_transport=ON

Una vez tenemos configurado el servidor MySQL para que acepte conexiones seguras, podemos **obligar** a un determinado usuario a que sólo se pueda conectar si utiliza una conexión cifrada mediante un certificado SSL:

**Nota:** Para hacer esta práctica deberéis permitir que el servidor MySQL acepte conexiones inseguras, comentando la orden anterior y reiniciando el servicio...

1 CREATEUSER'usuario'@'localhost'IDENTIFIEDBY'password'REQUIRESSL;

Como vemos al final de la orden CREATE USER debemos escribir **REQUIRE SSL**.

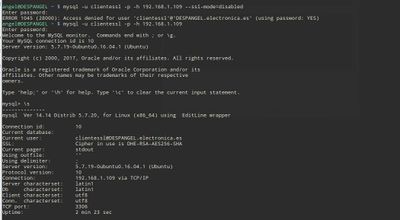
**Nota:** Un certificado SSL es el tipo de certificado más conocidos de los [certificados X.509](https://en.wikipedia.org/wiki/X.509).

Por ejemplo:

CREATEUSER'clientessl'@'192.168.1.103'IDENTIFIEDBY'clientessl'REQUIRESSL;

Estoy creando un usuario de nombre 'clientessl' que va a conectarse desde mi máquina real que tiene ip 192.168.1.103 hacia el servidor MySQL que está en una máquina virtual en modo bridge con la ip 192.168.1.109.

Si intento conectarme deshabilitando las conexiones seguras me mostrará un mensaje de error, pero sí me dejará con la conexión SSL como podemos comprobar en la siguiente imagen:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_12.jpg)

**IMPORTANTE:** Recordar que es necesario comentar la línea del archivo de configuración de MySQL, **bind\_address** y que si nuestro equipo tiene un firewall deberemos habilitar una regla para permite el tráfico desde los ordenadores que intentan conectar al servidor MySQL y a su puerto 3306.

Información obtenida de:

* [Digitalocean](https://www.digitalocean.com/community/tutorials/how-to-configure-ssl-tls-for-mysql-on-ubuntu-16-04).
* [Percona Blog](https://www.percona.com/blog/2013/06/22/setting-up-mysql-ssl-and-secure-connections/).

#### Conexiones seguras en MySQL WorkBench

**Opción 1**

Podemos hacer uso de una conexión SSH para realizar un 'puente' hacia el puerto 3306 del servidor remoto.

En este caso, no sería necesario modificar el archivo de configuración del servidor para aceptar conexiones remotas, ya que la conexión la haríamos realmente en el servidor, como si estuviéramos trabajando localmente, pero a través de una conexión ssh.

Por lo tanto, con esta opción no necesitamos comentar la línea 'bind\_address' del archivo de configuración del servidor.

Los pasos a seguir son:

Que el servidor donde está instalado el MySQL acepte conexiones ssh, instalando el servicio con la orden de consola: **sudo apt-getinstallssh**.

Podemos cambiar el puerto por defecto en el archivo de configuración: /etc/ssh/sshd\_config

Debemos de crear una conexión ssh desde el equipo cliente al equipo donde está el servidor MySQL y creando un 'túnel' para que la herramienta MySQLWorkBench se conecte a través de la conexión ssh establecida. Para hacer eso seguiremos los pasos indicados [en este enlace](https://hostpresto.com/community/tutorials/how-to-connect-to-a-remote-mysql-server-via-an-ssh-tunnel/) o de forma offline en este archivo: [Media:Conectar\_via\_SSH\_a\_MYsql.pdf](https://wiki.cifprodolfoucha.es/images/3/34/Conectar_via_SSH_a_MYsql.pdf" \o "Conectar via SSH a MYsql.pdf)

**Atención:** En esta forma de conexión, los usuarios se están conectando al servidor MySQL desde **localhost** ya que estamos estableciendo una conexión ssh.

**Opción 2**

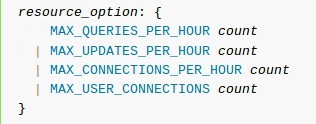
Esta opción es menos segura que la anterior ya que necesitamos que el servidor acepte conexiones desde fuera, dando lugar a posibles ataques.

En este caso, una vez instalado el certificado de seguridad, como vimos en el punto “Conexiones seguras del servidor”, solamente tenemos que ir a las propiedades de una conexión en MySQL WorkBench y escoger la opción **SSL Required**.

**SSL en MySQLWorkbench**

|  |  |
| --- | --- |
|  |  |
| Vamos a las propiedades de una conexión y en la pestaña SSL escogemos **Required.** | Presionando el botón **Test Connection** se nos informará de que la conexión es segura. |

### Opciones de limitación de recursos

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_19.jpg)

Más información [**en este enlace**](https://dev.mysql.com/doc/refman/5.7/en/create-user.html#create-user-resource-limits).

Cuando creamos una cuenta podemos establecer ciertos límites al uso que un usuario pueda hacer de los recursos del servidor MySQL.

Un ejemplo:

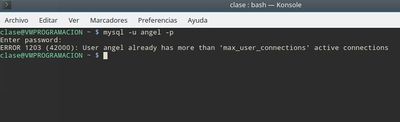
CREATEUSER'usuario'@'localhost'IDENTIFIEDBY'password'

WITHMAX\_QUERIES\_PER\_HOUR10MAX\_USER\_CONNECTIONS5;

Los recursos que podemos limitar son:

* MAX\_QUERIES\_PER\_HOUR count, MAX\_UPDATES\_PER\_HOUR count, MAX\_CONNECTIONS\_PER\_HOUR count: Indicancuantasconsultas, actualizaciones y conexiones por hora son permitidas. Si no se indica o se pone el valor 0 no existen limitaciones.
* MAX\_USER\_CONNECTIONScount: Indica cuantas conexiones simultáneas están permitidas para el usuario. Si no se pone nada, se utilizará el valor de la [**variable del sistema max\_user\_connections**](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html) que por defecto está sin limitación.

Si un usuario intenta hacer uso de un número de recursos superior a los indicados, tendrá un mensaje de error al intentar conectarse como el de la siguiente imagen:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_20.jpg)

### Conexiones simultáneas

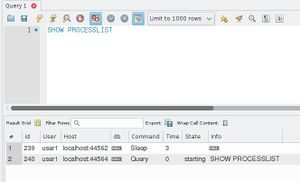
En lo referente al número de conexiones máximas de un usuario (simultáneas) debemos de tener en cuenta que cuando nos conectamos:

* Al MySqlWorkBench se crean dos conexiones por cada usuario.

Para comprobarlo podemos ejecutar la orden SQL: **SHOW PROCESSLIST**

Esta orden realiza una consulta select en la tabla del sistema: information\_schema.**processlist** y devuelve las conexiones activas al servidor MySQL de todos los usuarios.

* El usuario tiene que tener el permiso **PROCESS** otorgado para ver las conexiones de todos los usuarios, sino sólo ve las suyas (columna Process\_priv de la tabla mysql.users).

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_50.jpg)

* A través de herramientas clientes, tipo **mysql**, **mysqladmin**,...se crea una única conexión al servidor y se mantiene hasta que salgamos.

### Opciones de autentificación

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_usuarios_21.jpg)

Más información [**en este enlace**](https://dev.mysql.com/doc/refman/5.7/en/create-user.html#create-user-authentication).

Esta sección ya la vimos al principio para crear un usuario con password.

En la creación de un usuario podemos especificar el plugin utilizado para guardar la contraseña.

Si no se indica el nombre del plugin, se utiliza por defecto el plugin con nombre 'mysql\_native\_password' a no ser que se indique otro diferente en la [**variable del sistema default\_authentication\_plugin**](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html).

Podéis consultas los diferentes plugins en [**siguiente enlace**](https://dev.mysql.com/doc/refman/8.0/en/authentication-plugins.html).

Recordar que la contraseña se guarda 'encriptada' haciendo uso del plugin indicado, en la columna **authentication\_string** de la tabla **mysql.user**.

Podemos enviar o bien:

* 'auth\_string': Que sería la contraseña en texto sin encriptar.
* 'hash\_string': Que sería la contraseña en formato 'hash' después de aplicarle la función de hash indicada por el plugin.

Un ejemplo:

CREATEUSER'angel'@'%'IDENTIFIEDBY'12345678';

# GESTIÓN DE PERMISOS

## GRANT y REVOQUE

Las dos órdenes SQL que nos van a permitir establecer y quitar permisos son: GRANT y REVOKE.[[2]](#footnote-3)

En general **podemos afirmar lo siguiente**:

* Los cambios en los privilegios de tabla y columna tienen efecto inmediatamente en el usuario, aunque ya se encuentre conectado.
* Los cambios en los privilegios a nivel de base de datos no tienen efecto hasta que se ejecute la sentencia SQL: USE *nombre\_db*;

A nivel gráfico si estamos utilizando el Workbench, se produce al escoger la base de datos presionando dos veces sobre ella, quedando en negrilla.

* Los cambios en los privilegios globales no tienen efecto **hasta que el cliente vuelve a conectarse**.

## Permisos en MySQL

Los permisos que vamos a otorgar estarán en uno de los niveles de seguridad que vimos en un apartado NIVELES DE SEGURIDAD y que por lo tanto se traducirá en la inserción, modificación y borrado de filas en las tablas: **user,db,tables\_priv,columns\_privyprocs\_priv** dentro de la base de datos **mysql**.

La lista de todos los permisos que se pueden 'otorgar' o 'denegar' a un determinado usuario son los siguientes (están todos los niveles: servidor, base de datos y tablas):

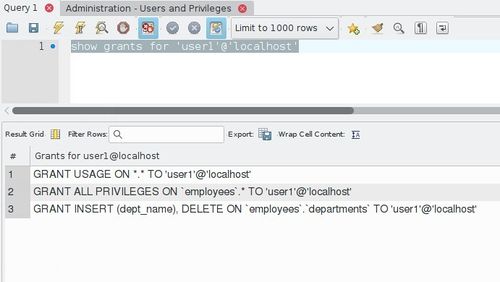
[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_grantrevoke_7.jpg)

## Consulta de permisos de un usuario

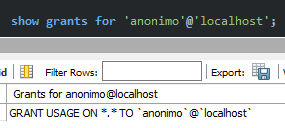
Para mostrar los permisos que posee un determinado usuario:

SHOWGRANTS**FOR**'user'@'host'

Por ejemplo:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_18.jpg)

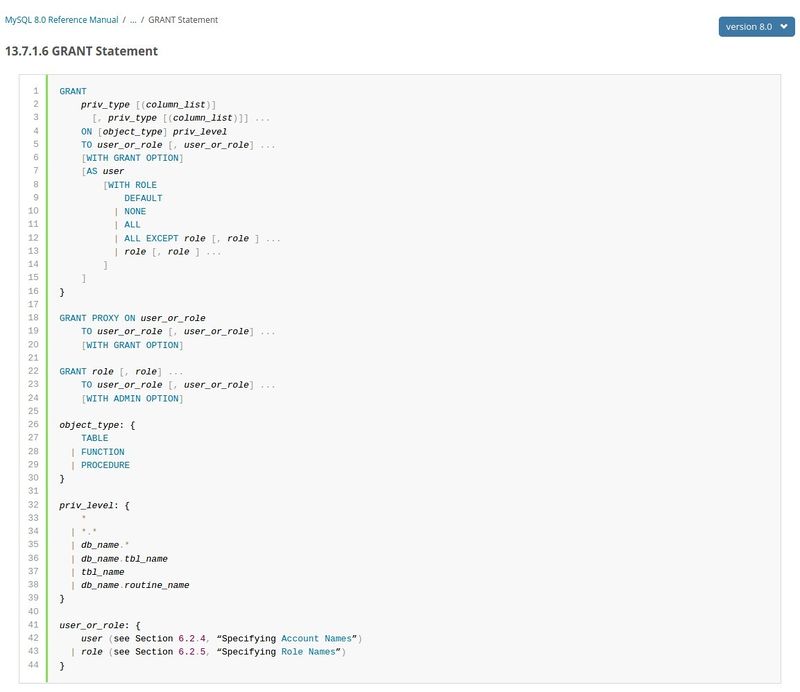
Cuando se crea un usuario el único permiso que tiene es el de USAGE que le da derecho a conectarse a MySQL.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_grantrevoke_1.jpg)

## GRANT

Esta es la orden SQL que concede permisos.

Podemos consultar su sintaxis completa [**en este enlace**](https://dev.mysql.com/doc/refman/8.0/en/grant.html)**.**



Para ver los permisos que tiene otorgado un usuario, debemos de ejecutar la orden SQL: [**SHOW GRANTS**](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html).

* Todo usuario tiene permiso para ver sus privilegios.
* Si queremos ver los privilegios de otro usuario debemos de tener permiso de **SELECT** sobre la base de datos **mysql**.

### Permiso para crear usuarios(tabla ‘mysql.user’)

Vimos en el apartado GESTIÓN DE USUARIOS como podíamos crear usuario con la orden SQL: **CREATE USER**

Para que otros usuarios puedan crear usuarios, necesitan tener el permiso: [**CREATE USER**](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html) (lo veremos a continuación)

### Permisos a nivel global(tabla ‘mysql.user’)

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_6.jpg)

Imagen obtenida [**de este enlace**](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html)

Para que otorgar un permiso a este nivel, hay que indicar: ON \*.\* en la cláusula **GRANT**.

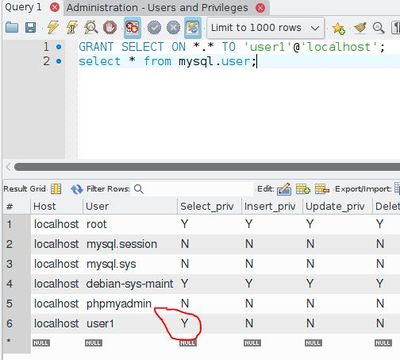
La sintaxis básica para dar permisos a nivel de servidor MySQL, incluyendo los permisos administrativos sería:

GRANTSELECTON\*.\*TO'user1'@'localhost'

**Nota**: Si el usuario ya está conectado cuando se realiza el cambio de permisos, no serán aplicados. El usuario necesita cerrar la conexión y volver a abrirla.

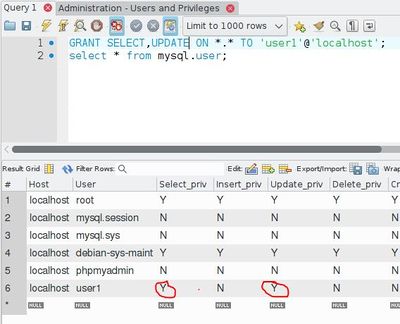
En este ejemplo, estaríamos dando permiso de selección sobre todas las tablas de todas las bases de datos, al usuario 'user1' conectado desde host 'localhost'.

Recordar que esta sentencia se traducirá en una orden UPDATE sobre la tabla **mysql.user**.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_grantrevoke_3.jpg)

Podemos otorgar varios permisos en una única operación, separándolos con comas:

GRANTSELECT,UPDATEON\*.\*TO'user1'@'localhost'[[3]](#footnote-4)

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_grantrevoke_4.jpg)

Como ya comentamos en un punto anterior, existen ciertos permisos que permiten administrar el servidorMySQL y que no tienen sentido en niveles inferiores. El caso contrario es, por ejemplo, el permiso SELECT ya que podemos tener un SELECT a nivel de base de datos, a nivel de tabla, a nivel de columna de una tabla.

**Nota**: Lógicamente para que un usuario pueda otorgar estos permisos, debe de tener permiso para poder otorgarlo. Esto lo veremos posteriormente, pero por ahora hay que indicar que el usuario que ejecuta las órdenes que otorgan los permisos es 'root'.

Todos los permisos que vamos a indicar a continuación tienen su correspondiente 'columna' en la tabla **mysql.user**.

Vamos a analizar alguno de estos permisos a nivel exclusivamente global:

* **CREATE USER**: Permite crear, modificar, borrar, renombrar y quitar los permisos de los usuarios. Esto último lo conseguimos ejecutando la orden SQL: REVOKE ALL PRIVILEGES, GRANT OPTION FROM user.
* **SHOW DATABASES**: La cuenta que tenga este privilegio puede ver las bases de datos del MySQL ejecutando la orden SQL SHOW DATABASES. En caso de no tener este permiso, un usuario solo va a poder ver las bases de datos sobre las que tenga algún permiso de acceso.
* **SHUTDOWN**: Quien tenga este permiso podrá ejecutar la orden SQL SHUTDOWN o ejecutar la orden de consola: mysqladminshutdown
* **USAGE**: Este permiso sólo indica que el usuario puede conectarse al MySQL. No da ningún tipo de privilegio.
* **SUPER**: Habilita diferentes permisos relacionados con la configuración del servidor, los cuales estén indicados en [este enlace](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super). Entre ellos puede ‘matar’ procesos de otras cuentas, modificar parámetros de encriptación, roles, replicación, transacciones, variables de sistema, etc.

Entre los más interesantes:

* **KILL**: Tenemos su equivalente con la orden ‘mysqladminkill’. Lo que hace esta orden es 'matar' o 'detener' un determinado hilo de ejecución. Eso se traduce en que podemos cerrar la conexión de un determinado usuario o cancelar la ejecución de una orden que pueda estar bloqueando el servidor MySQL. Para saber el identificar del proceso, podemos ejecutar la orden sql: **SHOW PROCESSLIST**. Un usuario siempre puede detener sus propias conexiones.
* Existen ciertas variables globales del sistema que necesitan que el usuario tenga el permiso SUPER para modificar su valor, como binlog\_format, sql\_log\_bin, y sql\_log\_off.
* Quien tenga el permiso SUPER puede conectarse al servidor MySQL (una sóla vez) aunque se alcance el valor de [max\_connections](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \l "sysvar_max_connections).
* Podrá parar o iniciar los servidores esclavos (slave servers) en un entorno de replicación.

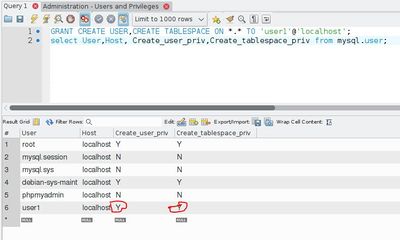
#### Ampliación: Otros permisos a nivel global

[**CREATE TABLESPACE**](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html): El usuario tiene permiso para crear (CREATE), borrar (DROP) o modificar (ALTER) 'tablespace'. Un 'tablespace' es una parte de disco donde podemos guardar la información de una o varias tablas. La principal ventaja es que nos va a permitir guardar los datos de una tabla 'fuera' del directorio de datos de MySQL. Existen otras funcionalidades que se pueden consultar en el enlace anterior.

Un ejemplo:

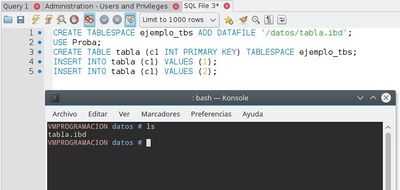
GRANTCREATETABLESPACEON\*.\*TO'user1'@'localhost'

Por defecto se crea en el mismo director que donde MySQL guarda los datos, indicado por la variable del sistema **datadir**.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_grantrevoke_5.jpg)

Una vez creado un *tablespace* podemos crear una tabla o varias tablas y hacer que se guarde en ese *tablespace*.

Un ejemplo:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_grantrevoke_6.jpg)

En este ejemplo estamos creando un tablespace en el directorio '/datos/', creado previamente y con permisos **por**[**parte de apparmor**](http://wiki.cifprodolfoucha.es/index.php?title=Mysql_Instalaci%C3%B3n#Caso_especial:_Los_directorios) (tenéis que darle los mismos permisos que los de datos) y posteriormente creamos una tabla en ese *tablespace*, comprobando como después de añadir dos filas, físicamente aparece el 'archivo ibd' en el directorio.

Más información:[**Enlace**](https://dev.mysql.com/doc/refman/8.0/en/general-tablespaces.html).

[**FILE**](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_file): Quien tenga otorgado este privilegio puede leer y escribir en ficheros del servidor utilizando las intrucciones [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html), [SELECT INTO ...OUTFILE](https://dev.mysql.com/doc/refman/5.7/en/select-into.html) y la función [LOAD\_FILE()](https://dev.mysql.com/doc/refman/5.7/en/string-functions.html#function_load-file).

Este permiso puede ser muy peligroso por lo que no debería otorgarse a ningún usuario.

Con la función LOAD\_FILE() cargamos archivos que se encuentran en el servidor. En la versión 5.7, solamente deja 'cargar' los archivos que se encuentren en el directorio /var/lib/mysql-files/, el cual está establecido en la variable del sistema [secure\_file\_priv](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \l "sysvar_secure_file_priv). Recordar que en distribuciones Ubuntu es necesario otorgar permisos de acceso si utilizamos un directorio diferente, como vimos [anteriormente en la instalación](http://wiki.cifprodolfoucha.es/index.php?title=Mysql_Instalaci%C3%B3n#Caso_especial:_Los_directorios).

Veamos un ejemplo:

SELECTLOAD\_FILE('/var/lib/mysql-files/prueba.txt');

Suponemos que tenemos creado un archivo de texto de nombre 'prueba.txt' en el directorio indicado.

La función **LOAD DATA INFILE** puede cargar tanto ficheros locales (**LOAD DATA LOCAL INFILE**) como del servidor.

La función SELECT INTO hace el proceso contrario, y guarda una consulta en un archivo.

**PROCESS**: Permite ver los threads del servidor. Es utilizado al ejecutar las órdenes SQL:

show processlist o la orden mysqladminprocesslist: Muestra las conexiones al servidor. Todo usuario puedo ejecutarla para ver sus propias conexiones, pero la forma: show **full** processlist, muestra todas las conexiones y esto sólo lo podrá ver el usuario que tenga el permiso PROCESS.

**SHOW ENGINE**: Muestra información sobre los motores de almacenamiento.

**RELOAD**: Permite realizar operaciones de **FLUSH** en el servidor. Dichas operaciones las podemos realizar desde SQL con el comando sqlFLUSH o con la herramienta mysqladmin y las operaciones: flush-hosts, flush-logs, flush-privileges, flush-status, flush-tables, flush-threads, refresh, y reload (hace lo mismo que flush-privelges).

**REPLICATION CLIENT**: Permiteejecutar las órdenes SQL SHOW MASTER STATUS, SHOW SLAVE STATUS y SHOW BINARY LOGS.

**REPLICATION SLAVE**: Permite que una cuenta que tenga este permiso, desde el servidor esclavo notifique los cambios al servidor maestro para su actualización.

### Permisos a nivel de bases de datos(tabla ‘mysql.db’)

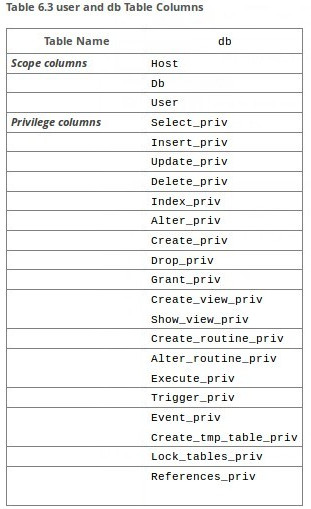
[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_7.jpg)

Imagen obtenida [de este **enlace**](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html)

#### Principales permisos

Debemos recordar que, a este nivel, los permisos son filas que se añaden, borran o se modifican en la tabla **mysql.db**, y se aplican en una base de datos concreta.

A este nivel, la sintaxis para otorgar un permiso es la siguiente:

GRANTALLONmydb.\*TO'someuser'@'somehost';

GRANTSELECT,INSERTONmydb.\*TO'someuser'@'somehost';

Con **GRANT ALL** estamos otorgando todos los privilegios a este nivel.

Fijarse como se indica el nombre de la base de datos:

GRANT SELECT ON **mydb**.\* TO 'someuser'@'somehost';

Los privilegios que pueden ser especificados a nivel de base de datos:

1. **SELECT, UPDATE, INSERT, DELETE**
2. **CREATE**: Permite crear tablas.
3. **DROP**: Permite eliminar tablas.
4. **ALTER**: Permite modificar tablas. Es necesario tener el privilegio 'CREATE'.
5. **EVENT**: Permite crear, eliminar o modificar eventos que pueden ser programados en el servidor MySQL. Más información [en este enlace](https://dev.mysql.com/doc/refman/8.0/en/create-event.html).
6. **LOCK TABLES**: Permite 'bloquear' una tabla impidiendo que ningún usuario pueda realizar operaciones sobre la misma e incluso impidiendo que puedan leerla. Es necesario tener el permiso SELECT sobre la misma tabla para poder bloquearla.
7. **REFERENCES**: Para poder crear una regla de clave foránea en una tabla.
8. **GRANT OPTION**: Otorgamos privilegios a un usuario y dicho usuario puede otorgar esos mismos privilegios a otros usuarios.

GRANTCREATEROUTINE,ALTERROUTINEONmydb.\*

TO'someuser'@'somehost'WITHGRANTOPTION;

#### Permisos a nivel de base de datos sobre procedimientos y funciones

Existen privilegios relacionados con la gestión de procedimientos almacenados y funciones (routines) pueden ser aplicados a nivel general y a nivel de base de datos[[4]](#footnote-5). En este último caso, sólo se permitiría crear/modificar (incluye borrado) procedimientos/funciones dentro de la base de datos indicada.

GRANTCREATEROUTINE,ALTERROUTINEONmydb.\*TO'someuser'@'somehost';

* [**EXECUTE**](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_execute): El usuario tiene permiso para ejecutar procedimientos y funciones.
* ALTER ROUTINE(incluye borrado)
* CREATE ROUTINE
* EXECUTE
* GRANT OPTION

Debo indicar que, en el caso de las rutinas, no existe el permiso DROPROUTINE. Si posee el permiso ALTER ROUTINE puede modificar o borrar una rutina.

* **Aunque se tengan permisos para ejecutar la función o el procedimiento, la rutina solo tendrá permisos para ejecutar instrucciones permitidas para el usuario que la ejecutó.**

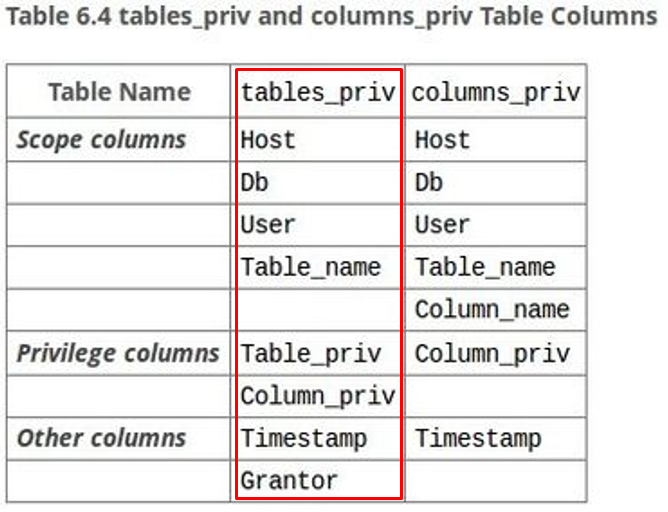
### Permisos a nivel de tabla(tabla ‘mysql.tables\_priv’)

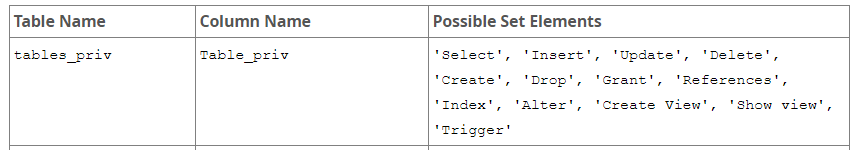
Recordar que a este nivel, los permisos son filas que se añaden, borran o se modifican en la tabla **mysql.tables\_priv**, y se aplican en una base de datos concreta y en una tabla concreta.

GRANTALLONmydb.mytblTO'someuser'@'somehost';

GRANTSELECT,INSERTONmydb.mytblTO'someuser'@'somehost';

Los privilegios que pueden ser aplicados a este nivel:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_seguridad_12.jpg)



Privilegios que gestionan tablas:

* **CREATE**: MySQL permite dar permiso para crear una tabla concreta, incluso si esta no existe.
* **ALTER**: MySQL permite dar permiso para modificar una tabla concreta, incluso si esta no existe.
* **DROP**: MySQL permite dar permiso para eliminar una tabla concreta, incluso si esta no existe.

Privilegios que manejan datos de la tabla:

* **SELECT, UPDATE, INSERT, DELETE**

Estos privilegios se aplican a todas las columnas de una tabla determinada.

Privilegios que manejan vistas (visiones externas de una base de datos):

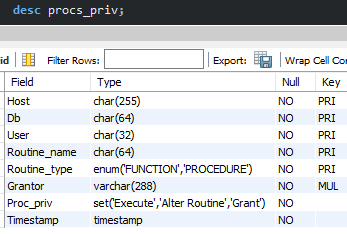
* **CREATE VIEW**
* **SHOW VIEW**

Para borrar una vista o modificarla, se necesita el permiso DROP.

Otros privilegios:

* **GRANT OPTION**: Otorgamos privilegios a un usuario y dicho usuario puede otorgar esos mismos privilegios a otros usuarios.
* **INDEX**: Permite crear o borrar índices sobre una tabla.
* **REFERENCES**: Para poder crear una regla de clave foránea al crear una tabla.
* **TRIGGER**: Permite crear, borrar, modificar y mostrar triggers.

### Permisos sobre rutinas (tabla ‘mysql.procs\_priv’)



Recordad que a este nivel, los permisos son filas que se añaden, borran o se modifican en la tabla **mysql.user** (todas las rutinas de todas las bases de datos),**mysql.db** (todas las rutinas de una base de datos concreta) y **mysql.procs\_priv**(cuando otorgamos permiso sobre una rutina concreta).

// A nivel de tabla

**GRANTCREATEROUTINEON**mydb.\***TO**'someuser'@'somehost';

//A nivel de rutina concreta

**GRANTEXECUTEONPROCEDURE**mydb.myproc**TO**'someuser'@'somehost';

Los privilegios que pueden ser aplicados a este nivel (rutina concreta):

* ALTER ROUTINE (incluye borrado).
* EXECUTE
* GRANT OPTION

Debo indicar que, en el caso de las rutinas, no existe el permiso DROPROUTINE. Si posee el permiso ALTER ROUTINE puede modificar o borrar una rutina.

**Ojo**: A diferencia de algunos permisos a nivel de tabla (tables\_priv) la rutina DEBE existir para aplicarle un permiso.

### Ampliación: Permisos a nivel de columnas

Recordar que a este nivel, los permisos son filas que se añaden, borran o se modifican en la tabla **mysql.columns\_priv**, y se aplican en una base de datos concreta, en una tabla concreta y una columna concreta de esa tabla. Recordar que también se modifica la tabla **mysql.tables\_priv**.

GRANTSELECT(col1),INSERT(col1,col2)

ONmydb.mytblTO'someuser'@'somehost';

Los privilegios que pueden ser aplicados a este nivel:

* **INSERT**
* **REFERENCES**: Para poder crear una regla de clave foránea al crear una tabla.
* **SELECT**
* **UPDATE**

## REVOKE

La orden SQL contraria a GRANT es REVOKE.

Podéis consultar la sintaxis completa [en este enlace](https://dev.mysql.com/doc/refman/8.0/en/revoke.html).

Un ejemplo sería el siguiente:

REVOKEINSERTON\*.\***FROM**'usuario'@'localhost';

En este caso estaríamos quitando el permiso de inserción a nivel global (tabla mysql.user) al usuario 'usuario'.

Los permisos que podemos quitar son los que ya vimos con la orden GRANT.

Para poder quitar un permiso a un usuario debes de poseer dicho permiso y haberlo obtenido con la opción 'WITH GRANT OPTION'.

Otra opción para poder quitar permisos (todos ellos) es que un usuario tenga el permiso 'CREATE USER'.

En este caso, va a poder ejecutar la orden:

REVOKEALLPRIVILEGES,GRANTOPTIONFROM'user'@'localhost'[[5]](#footnote-6)

Aclarar que esta orden quita permisos a un usuario, pero no lo elimina.

## Opción WITH GRANT OPTION

Si observamos la [sintaxis de la orden GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html), podemos ver cómo se puede utilizar una opción **WITH GRANT OPTION** al final de la misma.

Esta opción lo que conlleva es que el permiso que estamos concediendo a un usuario con la orden GRANT, lo va a poder gestionar. Es decir, que va a poder otorgar el mismo permiso a otros usuarios.

Si otorga el permiso a otros usuarios con la opción **WITH GRANT OPTION** esos otros usuarios van a poder otorgar el mismo permiso a terceros. Y así sucesivamente.

Por ejemplo:

GRANTSELECT,INSERTON\*.\*TO'matador'@'%'WITHGRANTOPTION;

Ahora si nos conectamos como 'matador', dicho usuario va a poder otorgar el permiso SELECT o INSERT a otro usuario.

* Indicar que, si el permiso se puede aplicar a diferentes niveles, el usuario que tengo el permiso en un nivel con opción 'WITH GRANT OPTION' va a poder otorgar el mismo permiso en los niveles de seguridad igual o inferiores al que posee.

Es decir, si tengo el permiso SELECT a nivel global con la opción 'WITH GRANT OPTION' voy a poder otorgar el permiso SELECT a nivel global (tabla mysql.user), a nivel de una base de datos en concreto (tabla mysql.db) o a nivel de una tabla de una base de datos en concreto (tabla mysql.tables\_priv).

* Si eliminamos un usuario (DROP USER) que haya otorgado permisos a otros usuarios, los permisos otorgados por el usuario **se mantienen**.
* Si queremos eliminar el permiso 'WITH GRANT OPTION' de un usuario que lo tenga otorgado, tendremos que ejecutar la siguiente orden REVOKE:

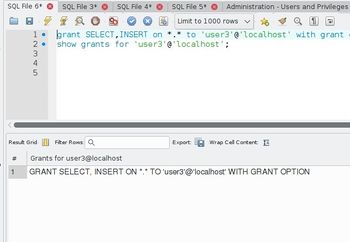
REVOKEGRANTOPTIONON\*.\*FROM'user'@'host'

Esto eliminará **la opción WITH GRANT OPTION** de todos los permisos otorgados con esa opción en el **NIVEL CORRESPONDIENTE** (en el ejemplo a nivel global).

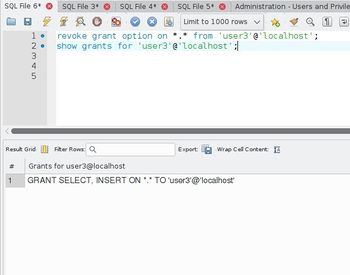
**LOS PERMISOS SE MANTIENEN**.

Veamos un ejemplo:

**Elimandowithgrantoption**

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_permisos_revoke_1.jpg)

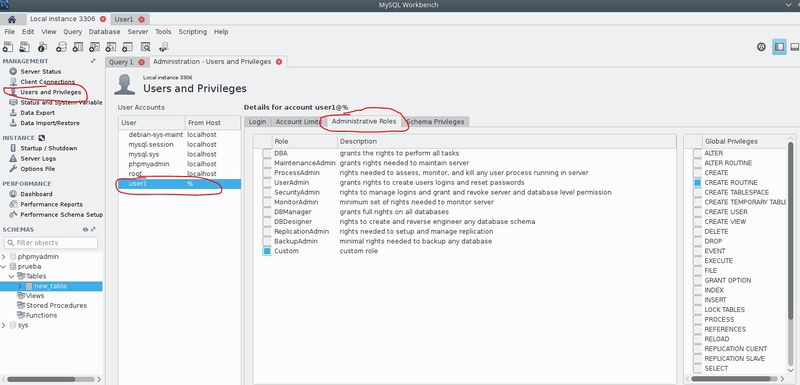
Otorgamos permisos SELECT e INSERT a nivel global con la opción de gestionarlos al usuario 'user3'.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_permisos_revoke_2.jpg)

Eliminamos la opción **WITH GRANT OPTION** a dicho usuario en el mismo nivel de seguridad. Podemos comprobar como mantiene los permisos otorgados.

## Roles

* Un rol es un identificar que engloba un conjunto de permisos.
* Si un usuario tiene permisos específicos, aparecerá una entrada 'CUSTOM' en la lista de roles.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_roles_1.JPG)

* **DBA**: Grants all privileges
* **MaintenanceAdmin**: Grants privileges to maintain the server
* **ProcessAdmin**: Grants privileges to monitor and kill user processes
* **UserAdmin**: Grants privileges to create users and reset passwords
* **SecurityAdmin**: Grants privileges to manage logins and grant and revoke server privileges
* **MonitorAdmin**: Grants privileges to monitor the server
* **DBManager**: Grants privileges to manage databases
* **DBDesigner**: Grants privileges to create and reverse engineer any database schema
* **ReplicationAdmin**: Grants privileges to set up and manage replication
* **BackupAdmin**: Grants privileges required to back up databases
* **Custom**: Lists other (custom) privileges that are assigned to the user account

### Rolesen MySQL 8

* A partir de MySQL 8 es posible definir nuestros propios roles con un conjunto de permisos y asignarlo a usuarios.
* Es importante entender que un rol se crea como un usuario (desactivado) con unos permisos determinados, que pueden ser adquiridos por otro usuario.
* Más información [en este enlace](https://dev.mysql.com/doc/refman/8.0/en/roles.html).

#### Crear Roles

Más información en [este enlace](https://dev.mysql.com/doc/refman/8.0/en/create-role.html).

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_ud8_roles_1.jpg)

El proceso para trabajar con roles debería seguir estos pasos:

1. Crear los roles.
2. Asignar a cada ROL los permisos correspondientes.
3. Asignar a cada usuario el/los rol/es correspondientes.
4. **Activar el/los rol/es asignados**

Veamos un ejemplo:

**CREATEROLEIFNOTEXISTS**

mantenimiento\_basico,mantenimiento\_total,consulta\_basica;

Es posible emplear el mismo formato de nombre que para el caso de nombres de usuarios, en los que el nombre está conformado por dos partes: nombre@host

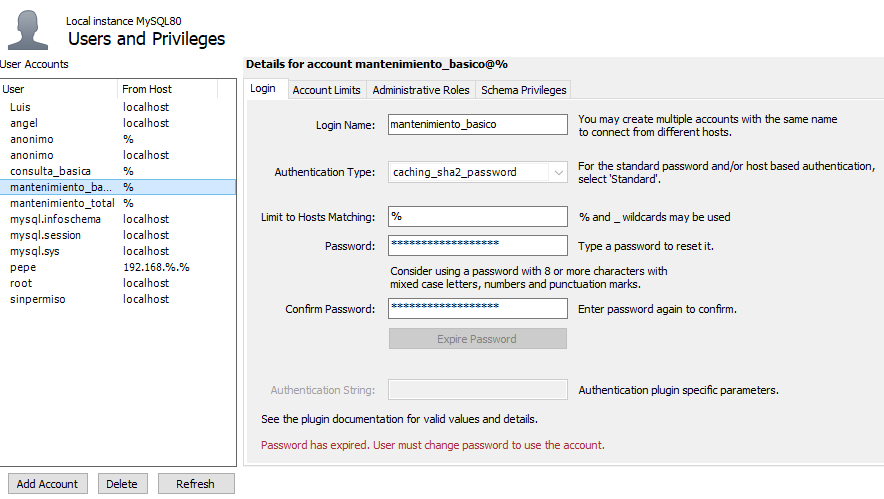
A diferencia de la creación de usuarios, si empleamos el carácter % como identificador de host, este forma parte del identificador del ROL como un **literal** y no como un comodín.

**CREATEROLEIFNOTEXISTS**'miRol'@'%';

Por defecto es el carácter que asigna a cualquier rol que no lleve la parte host indicada.

Indicar que cuando se crea un ROL, en el MySQLWorkBench aparece un usuario con dicho nombre y parte de host % si no se indica nada.

Dicho 'usuario' tiene la cuenta bloqueada y el password está expirado. Se comporta como un usuario, PERO NO LO ES.

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_ud8_roles_3.jpg)

Una vez tenemos el Rol creado ya podemos asignar los diferentes permisos sobre el mismo, tal cual vimos en los puntos anteriores:

**GRANTEXECUTEON**\*.\***TO**mantenimiento\_total;

*-- Permiso a nivel global*

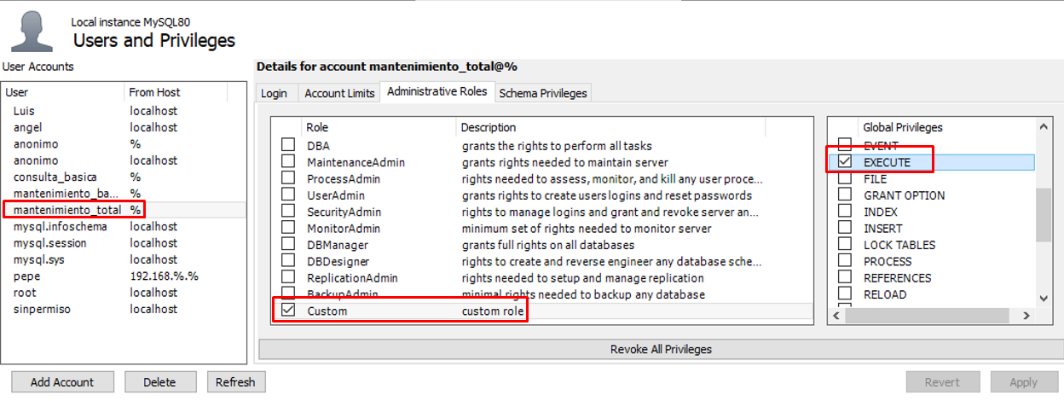
**GRANTSELECTON**bdempresa.\***TO**mantenimiento\_basico;

*-- Permiso a nivel de base de datos*

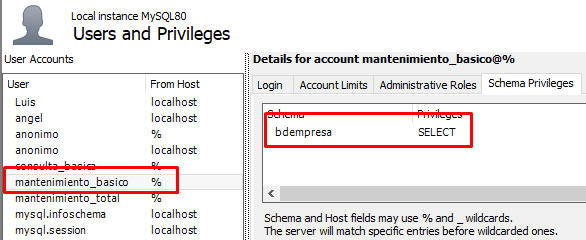
**GRANTSELECTON**bdempresa.empleados**TO**consulta\_basica;

*-- Permiso a nivel de tabla*

Podemos gestionar los permisos a nivel de servidor desde la pestaña **Administrative Roles** en la parte inferior.

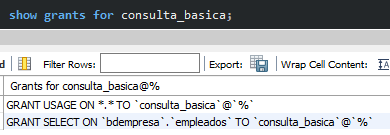


Podemos gestionar los permisos a nivel de BD desde la pestaña **SchemaPrivileges** en la parte inferior.



Sin embargo, no podemos ver de manera gráfica los permisos a nivel de tabla o de rutina.

Para consultar los permisos asignados a un rol debemos usar SHOW GRANTS de manera similar a como lo hacemos con usuarios (recuerda que un rol, es en esencia, un usuario desactivado).



Ahora debemos asignar los roles a los usuarios:

**GRANT**mantenimiento\_basico,consulta\_basica**TO**'user1'@'%';

*--Ejemplo suponiendo que tenemos un usuario user1@% previamente creado*

**GRANT**mantenimiento\_total**TO**'user2'@'%';

*--Ejemplo suponiendo que tenemos un usuario user1@% previamente creado*

**Nota:**Es posible definir roles que sean 'asignados' por defecto a todos los usuarios. Son los llamados 'mandatory roles'. Más información [en este enlace](https://dev.mysql.com/doc/refman/8.0/en/roles.html#mandatory-roles).

Debemos fijarnos que dicha asignación no lleva consigo que el usuario tenga los permisos asociados a los roles que ahora tiene.

Es necesario **activar** dichos roles.

Se puede comprobar como al acceder como el usuario 1, si ejecutamos la [función CURRENT\_ROLE()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-role) devuelve NONE:

**SELECTcurrent\_role**();

La activación de un rol la podemos implementar de dos formas:

1. Dejarla activada por defecto y cada vez que el usuario se conecta, los permisos del rol están activos para ese usuario.
2. Que sea el usuario (o el administrador) el que active/desactive el rol.

Para activar por defecto un rol, necesitamos emplear la orden [SET DEFAULT ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-default-role.html).

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_ud8_roles_4.jpg)

**El mismo usuario puede activar o desactivar los roles por defecto** o bien un administrador con permisos CREATE USER, o UPDATE en la tabla del sistema mysql.default\_roles.

Por ejemplo, para activar todos los roles asignados a un usuario podemos poner:

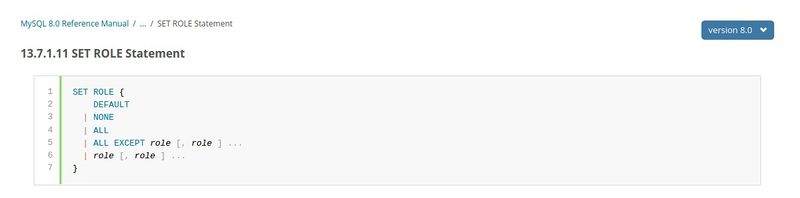
**SETDEFAULTROLEALLTO**'user1'@'%','user2'@'%';

Para activar un rol concreto:

**SETDEFAULTROLE**mantenimiento\_total**TO**'user2'@'%';

Esto hará que cuando el usuario user2 se conecte tenga activados todos los roles asignados (mantenimiento\_basico,consulta\_basica) y que el usuario user2 tenga activado por defecto el rol mantenimineto\_total.

Si queremos activar/desactivar un rol mientras estamos conectados, es el propio usuario que deberá de ejecutar la orden [SET ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-role.html).

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_ud8_roles_5.jpg)

Por ejemplo:

**SETROLE**mantenimineto\_total;

*-- Activar el role a user2 ESTANDO CONECTADO COMO user2*

**SETROLENONE**;

*-- Deactiviaría todos los roles a user2 ESTANDO CONECTADO COMO user2*

**Nota:** Podéis comprobar en la sintaxis de SET ROL que existen diferentes formas de activar y desactivar múltiples roles.

1. Recordad que, por defecto, MySQL no permite conexiones remotas utilizando TCP/IP, por lo que tendríais que comentar la bind-address variable del sistema bind-address] en su archivo de configuración. [↑](#footnote-ref-2)
2. Cada vez que ejecutemos una de estas órdenes **NO SERÁ** necesario ejecutar la orden **FLUSHPRIVILEGES** para que los cambios tengan efecto ya que el gestor informa del cambio a Mysql para que recargue los permisos a memoria. [↑](#footnote-ref-3)
3. Nota: Si el usuario ya está conectado cuando se realiza el cambio de permisos, no serán aplicados. El usuario necesita cerrar la conexión y volver a abrirla. [↑](#footnote-ref-4)
4. Recuerda que procedimientos y funciones son objetos hijos de las bases de datos, al mismo nivel que las tablas. Por ello no estarán disponibles a nivel de tabla. Sin embargo, los permisos sobre procedimientos y funciones específicos sí estarán disponibles en la tabla ‘procs\_priv’ [↑](#footnote-ref-5)
5. ‘ALL PRIVILEGES, GRANT OPTION’ no son dos cláusulas sino una sola. La propia sintaxis de REVOKE obliga a especificar ‘, GRANT OPTION’ siempre que usamos ‘ALL PRIVILEGES’. [↑](#footnote-ref-6)