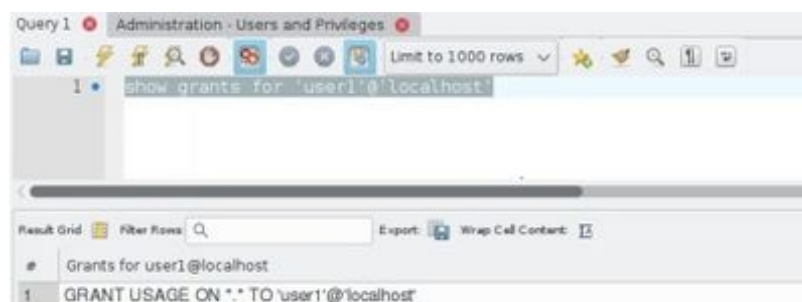


Introducción

- Las dos órdenes SQL que nos van a permitir establecer y quitar permisos son: GRANT y REVOKE.
- Cada vez que ejecutemos una de estas órdenes **NO SERÁ necesario ejecutar la orden FLUSH PRIVILEGES** para que los cambios tengan efecto ya que el gestor informa del cambio a Mysql para que recargue los permisos a memoria.
- En general **podemos afirmar lo siguiente:**
 - Los cambios en los privilegios de tabla y columna tiene 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.
- Los permisos que vamos a otorgar estarán en uno de los 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_priv y procs_priv** dentro de la **base de datos mysql**.
- Cuando se crea un usuario el único permiso que tiene es el de **USAGE** que le da derecho a conectarse a Mysql.



- La lista de todos los permisos que se pueden 'otorgar' o 'denegar' a un determinado usuario son los siguientes (están todos los niveles: general, db, table, column):

Table 6.2 Permissible Privileges for GRANT and REVOKE

Privilege	Column	Context
<u>ALL [PRIVILEGES]</u>	Synonym for "all privileges"	Server administration
<u>ALTER</u>	Alter_priv	Tables
<u>ALTER ROUTINE</u>	Alter_routine_priv	Stored routines
<u>CREATE</u>	Create_priv	Databases, tables, or indexes
<u>CREATE ROUTINE</u>	Create_routine_priv	Stored routines
<u>CREATE TABLESPACE</u>	Create_tablespace_priv	Server administration
<u>CREATE TEMPORARY TABLES</u>	Create_tmp_table_priv	Tables
<u>CREATE USER</u>	Create_user_priv	Server administration
<u>CREATE VIEW</u>	Create_view_priv	Views
<u>DELETE</u>	Delete_priv	Tables
<u>DROP</u>	Drop_priv	Databases, tables, or views
<u>EVENT</u>	Event_priv	Databases
<u>EXECUTE</u>	Execute_priv	Stored routines
<u>FILE</u>	File_priv	File access on server host
<u>GRANT OPTION</u>	Grant_priv	Databases, tables, or stored routines
<u>INDEX</u>	Index_priv	Tables
<u>INSERT</u>	Insert_priv	Tables or columns
<u>LOCK TABLES</u>	Lock_tables_priv	Databases
<u>PROCESS</u>	Process_priv	Server administration
<u>PROXY</u>	See proxies_priv table	Server administration
<u>REFERENCES</u>	References_priv	Databases or tables
<u>RELOAD</u>	Reload_priv	Server administration
<u>REPLICATION CLIENT</u>	Repl_client_priv	Server administration
<u>REPLICATION SLAVE</u>	Repl_slave_priv	Server administration
<u>SELECT</u>	Select_priv	Tables or columns
<u>SHOW DATABASES</u>	Show_db_priv	Server administration
<u>SHOW VIEW</u>	Show_view_priv	Views
<u>SHUTDOWN</u>	Shutdown_priv	Server administration
<u>SUPER</u>	Super_priv	Server administration
<u>TRIGGER</u>	Trigger_priv	Tables
<u>UPDATE</u>	Update_priv	Tables or columns
<u>USAGE</u>	Synonym for "no privileges"	Server administration

GRANT

- Esta es la orden SQL que concede permisos.
- Para ver los permisos que tiene otorgado un usuario, debemos de ejecutar la orden SQL: [SHOW GRANTS](#).

Todo usuario tiene permiso para ver sus privilegios.

Si queremos ver los privilegios de otro usuario (mirar la sintaxis de la orden SQL) debemos de tener permiso de **SELECT** sobre la base de datos **mysql**.

Permiso para crear usuarios

- Para que otros usuarios puedan crear usuarios, necesitan tener el permiso: [CREATE USER](#) (lo veremos a continuación)

Permisos a nivel global

- Para que otorgar un permiso a este nivel, hay que indicar: **ON *.*** en la cláusula grant.
- La sintaxis básica para dar permisos a nivel del gestor Mysql, incluyendo los permisos administrativos sería:

```
1 GRANT SELECT ON *.* 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 INSERT sobre la tabla **mysql.user**.

Query 1 Administration - Users and Privileges

```

1 • GRANT SELECT ON *.* TO 'user1'@'localhost';
2 • select * from mysql.user;
  
```

Limit to 1000 rows

Result Grid

#	Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv
1	localhost	root	Y	Y	Y	Y
2	localhost	mysql.session	N	N	N	N
3	localhost	mysql.sys	N	N	N	N
4	localhost	debian-sys-maint	Y	Y	Y	Y
5	localhost	phpmyadmin	N	N	N	N
6	localhost	user1	Y	N	N	N
*	NULL	NULL	NULL	NULL	NULL	NULL

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

```
1 GRANT SELECT,UPDATE ON *.* 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.

Query 1 Administration - Users and Privileges

```

1 • GRANT SELECT,UPDATE ON *.* TO 'user1'@'localhost';
2 • select * from mysql.user;
  
```

Limit to 1000 rows

Result Grid

#	Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv
1	localhost	root	Y	Y	Y	Y	Y
2	localhost	mysql.session	N	N	N	N	N
3	localhost	mysql.sys	N	N	N	N	N
4	localhost	debian-sys-maint	Y	Y	Y	Y	Y
5	localhost	phpmyadmin	N	N	N	N	N
6	localhost	user1	Y	N	Y	N	N
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Como ya comentamos en un [punto anterior](#), existen ciertos permisos que permiten administrar el gestor Mysql y que no tienen 'otros niveles' como sucede con el permiso SELECT por ejemplo (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 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**.

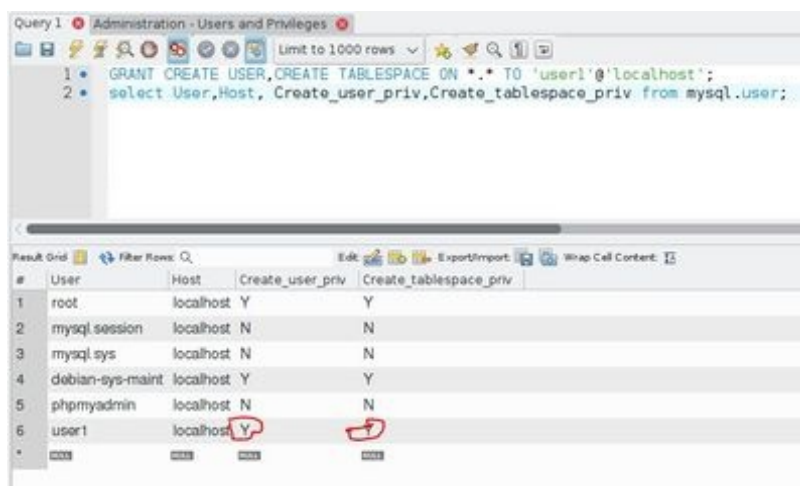
Estos permisos son los siguientes:

- **CREATE TABLESPACE**: 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:

```
1 GRANT CREATE TABLESPACE ON *.* TO 'user1'@'localhost'
```

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



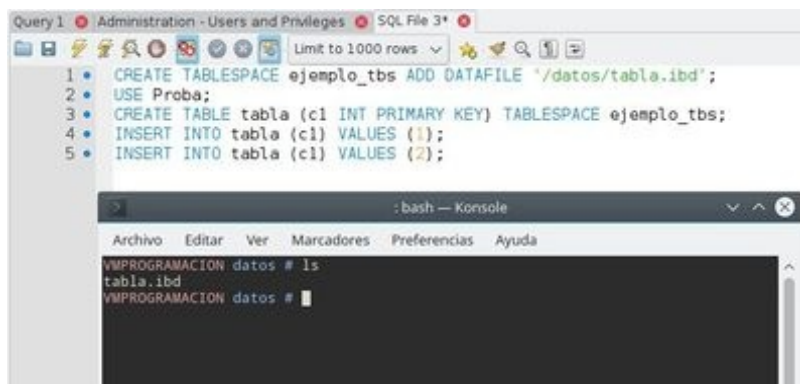
Query 1 Administration - Users and Privileges

```
1 GRANT CREATE USER,CREATE TABLESPACE ON *.* TO 'user1'@'localhost';
2 select User,Host, Create_user_priv,Create_tablespace_priv from mysql.user;
```

#	User	Host	Create_user_priv	Create_tablespace_priv
1	root	localhost	Y	Y
2	mysql.session	localhost	N	N
3	mysql.sys	localhost	N	N
4	debian-sys-maint	localhost	Y	Y
5	phpmyadmin	localhost	N	N
6	user1	localhost	Y	Y

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

Un ejemplo:



Query 1 Administration - Users and Privileges SQL File 3*

```
1 CREATE TABLESPACE ejemplo_tbs ADD DATAFILE '/datos/tabla.ibd';
2 USE Proba;
3 CREATE TABLE tabla (c1 INT PRIMARY KEY) TABLESPACE ejemplo_tbs;
4 INSERT INTO tabla (c1) VALUES (1);
5 INSERT INTO tabla (c1) VALUES (2);
```

: bash — Konsole

```
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
VMPROGRAMACION datos # ls
tabla.ibd
VMPROGRAMACION datos #
```

En este ejemplo estamos creando un tablespace en el directorio '/datos/', creado previamente y con permisos por [parte de apparmor](#) (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.

- **EXECUTE**: El usuario tiene permiso para ejecutar procedimientos y funciones.
- **FILE**: Quien tenga otorgado este privilegio puede leer y escribir en ficheros del servidor utilizando las intrucciones **LOAD DATA INFILE**, **SELECT INTO ...OUTFILE** y la función **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**. Recordar que en distribuciones Ubuntu es necesario otorgar permisos de acceso si utilizamos un directorio diferente, como vimos anteriormente en la instalación.
Veamos un ejemplo:

```
1 SELECT LOAD_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.

- **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**.
- **PROCESS**: Permite ver los threads del servidor. Es utilizado al ejecutar las órdenes SQL:
 - **show processlist** o la orden **mysqladmin processlist**: Muestra las conexiones al servidor. Todo usuario puede 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 **sql FLUSH** 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-privileges**).
- **REPLICATION CLIENT**: Permite ejecutar 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.
- **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 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: `mysqladmin shutdown`
 - **SUPER**: Disponemos de diferentes permisos, los cuales estén indicados en el enlace.
Entre los más interesantes:
 - **KILL**: Tenemos su equivalente con la orden `mysqladmin kill`. 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 sólo vez) aunque se alcance el valor de `max_connections`.
 - Podrá parar o iniciar los servidores esclavos (slave servers) en un entorno de replicación.
 - **USAGE**: Este permiso sólo indica que el usuario puede conectarse al Mysql. No da ningún tipo de privilegio.
-

Permisos a nivel de bases de datos

- 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:

```
1 GRANT ALL ON mydb.* TO 'someuser'@'somehost';
2 GRANT SELECT, INSERT ON mydb.* 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:
 - SELECT, UPDATE, INSERT, INSERT
 - CREATE: Permite crear tablas.
 - DROP: Elimina una tabla.
 - ALTER: Modifica una tabla. Es necesario tener el privilegio 'CREATE'.
 - EVENT: Permite crear, eliminar o modificar eventos que pueden ser programados en el servidor Mysql. Más información [en este enlace](#).
 - 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.
 - REFERENCES: Para poder crear una regla de clave foránea al crear una tabla.
 - Los privilegios relacionados con la gestión de procedimientos almacenados y funciones (routines) pueden ser aplicados a nivel general y a nivel de base de datos. En este último caso, sólo se permitiría crear/modificar (incluye borrado) procedimientos/funciones dentro de la base de datos indicada.

```
1 GRANT CREATE ROUTINE, ALTER ROUTINE ON mydb.* TO
'someuser'@'somehost';
```

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

- GRANT OPTION: Otorgamos privilegios a un usuario y dicho usuario puede otorgar esos mismos privilegios a otros usuarios.

```
1 GRANT CREATE ROUTINE, ALTER ROUTINE
2 ON mydb.*
3 TO 'someuser'@'somehost'
4 WITH GRANT OPTION;
```


Permisos a nivel de tabla

- 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.

```
1 GRANT ALL ON mydb.mytbl TO 'someuser'@'somehost';  
2 GRANT SELECT, INSERT ON mydb.mytbl TO 'someuser'@'somehost';
```

- Los privilegios que pueden ser aplicados a este nivel:
 - 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:
 - DELETE
 - SELECT
 - UPDATE
 - INSERTEstos 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 VIEWPara 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 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**.

```
1 GRANT SELECT (col1), INSERT (col1,col2) ON mydb.mytbl TO  
'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

Permisos sobre rutinas (proc.almacenados/funciones)

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

```
1 GRANT CREATE ROUTINE ON mydb.* TO 'someuser'@'somehost';  
2 GRANT EXECUTE ON PROCEDURE mydb.myproc TO  
'someuser'@'somehost';
```

•Los privilegios que pueden ser aplicados a este nivel:

- Aplicables a cualquier rutina:
 - ALTER ROUTINE (incluye borrado)
 - CREATE ROUTINE
 - EXECUTE
 - GRANT OPTION
- Aplicados a una rutina concreta:
 - ALTER ROUTINE
 - EXECUTE
 - GRANT OPTION

REVOKE

- La orden SQL contraria a grant es REVOKE.
- Podéis consultar la sintaxis completa [en este enlace](#).

- Un ejemplo sería el siguiente:

```
1 REVOKE INSERT ON *.* 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:

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

- 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](#), podemos ver como 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:

```
1 GRANT SELECT, INSERT ON *.* TO 'matador'@'%' WITH GRANT OPTION;
```


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:

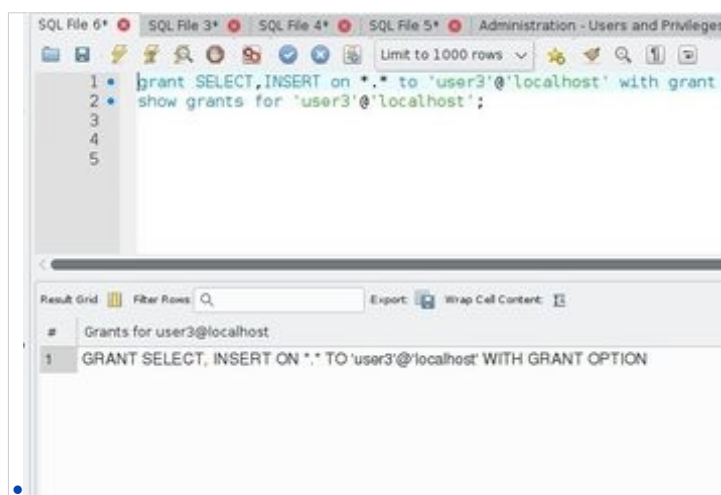
```
1 REVOKE GRANT OPTION ON *.* FROM 'user'@'host'
```

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).

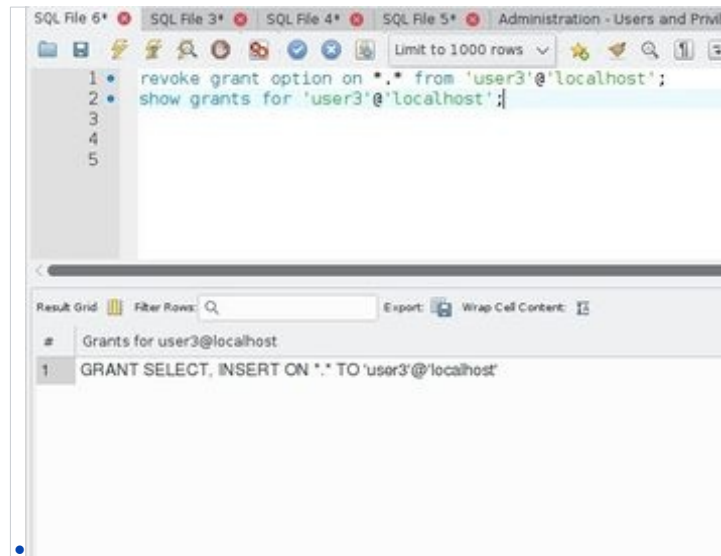
LOS PERMISOS SE MANTIENEN.

Veamos un ejemplo:

•Eliminando with grant option



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



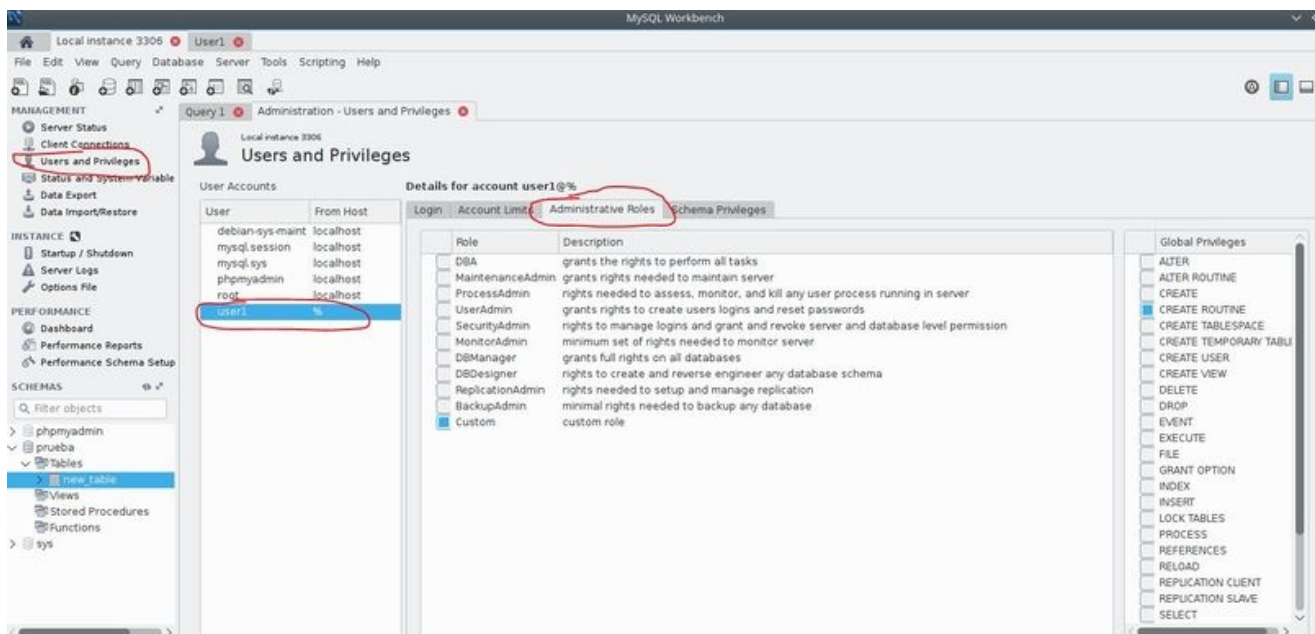
```
SQL File 6* SQL File 3* SQL File 4* SQL File 5* Administration - Users and Privile
Limit to 1000 rows
1 • revoke grant option on *.* from 'user3'@'localhost';
2 • show grants for 'user3'@'localhost';
3
4
5

Result Grid Filter Rows: Export: Wrap Cell Content:
# Grants for user3@localhost
1 GRANT SELECT, INSERT ON *.* TO 'user3'@'localhost'
```

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.
 - A partir del Mysql v8 ya disponemos de roles.
 - En la versión actual (5.7) lo más parecido lo tenemos en el MysqlWorkBench, donde existen un conjunto de roles 'predefinidos' que al pulsar sobre ellos aparecen en la parte derecha los permisos asociados a cada rol.
- Si un usuario tiene permisos específicos, aparecerá una entrada 'CUSTOM' en la lista de roles.



- 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