GESTIÓN DE CUENTAS DE USUARIOS Y PERMISOS

Índice de contenido

1. Gestión de usuarios y permisos en MySQL

1.1. Sistema de permisos

El sistema de privilegios de MySQL se encarga de establecer quién puede conectar con el servidor y de asegurar que cada usuario ejecute solamente operaciones para las que tenga permisos.
La identidad de un usuario que se conecta a una B.D. viene determinada por el equipo desde el que se conecta (**host**) y el nombre del **usuario**. Una vez conectado, dicho usuario enviará peticiones y el sistema se encarga de comprobar si tiene o no permiso para ejecutarlas.

Las cuentas (usuarios) y permisos están organizadas en una base de datos llamada "mysql" que describimos en este capítulo.

El acceso al servidor tiene lugar en dos etapas:

- Comprobar nuestra identidad (host/usuario/password)
- Comprobar nuestros permisos para hacer operaciones sobre los objetos

1.1.1. Tablas de permisos

Las tablas que se usan para el control de acceso son:

- user: Una fila por cada usuario/máquina permitido
- db: Una fila por cada usuario/máquina/bd permitido
- host: En conjunción con la anterior, una para cada fila de la tabla "host" con la columna "host" en blanco, contiene el usuario/host permitidos
- tables_priv:Una fila por cada usuario/bd/host/tabla permitidos
- columns_priv:Una fila por cada usuario/bd/host/tabla/columna permitidos

Las 3 primeras tablas se usan en ambas fases del control de acceso. Las dos últimas solo en la segunda.

Tabla "user"	Permisos de host origen/usuario	**Tabla "db"**	Permisos de host/usuario/bd
Host	Nombre de la máquina, IP o dominio. Puede contener una red o subred entera, o el valor "localhost"	***Host***	Nombre de la máquina, IP o dominio. Puede contener una red o subred entera, o el valor "localhost"
User	Nombre de usuario. No se permiten comodines, pero sí un valor en blanco que indica que se se permiten conexiones anónimos	***Db***	Nombre de la base de datos.
Password	Contraseña. Si está vacía indica que no hay contraseña.	***User***	Nombre de usuario.No se permiten comodines, pero sí un valor en blanco que indica que se se permiten conexiones anónimos
Select_priv	Y/N (tiene o no permiso para SELECT)	Select_priv	Y/N (tiene o no permiso para SELECT)
Insert_priv	Y/N (tiene o no permiso para INSERT)	Insert_priv	Y/N (tiene o no permiso para INSERT)
Update_priv	Y/N (tiene o no permiso para UPDATE)	Update_priv	Y/N (tiene o no permiso para UPDATE)

Delete_priv	Y/N (tiene o no permiso para DELETE)	Delete_priv	Y/N (tiene o no permiso para DELETE)
Index_priv	Y/N (tiene o no permiso para manipular índices)	Index_priv	Y/N (tiene o no permiso para manipular índices)
Create_priv	Y/N (tiene o no permiso para crear objetos)	Create_priv	Y/N (tiene o no permiso para crear objetos)
Drop_priv	Y/N (tiene o no permiso para borrar objetos)	Drop_priv	Y/N (tiene o no permiso para borrar objetos)
Shutdown_priv	Y/N (tiene o no permiso para parar la sesión)	...	** No tiene este campo
...			

Tabla "host"	
Host	Nombre de la máquina, IP o dominio. Puede contener una red o subred entera.
Db	Nombre de base de datos
Select_priv	Y/N (tiene o no permiso para SELECT)
Insert_priv	Y/N (tiene o no permiso para INSERT)
Update_priv	Y/N (tiene o no permiso para UPDATE)
Delete_priv	Y/N (tiene o no permiso para DELETE)
Index_priv	Y/N (tiene o no permiso para manipular índices)
Create_priv	Y/N (tiene o no permiso para crear objetos)
Drop_priv	Y/N (tiene o no permiso para borrar objetos)
...	

Tabla "Tables_priv"		**Tabla "Colums_priv"**	
Host	Máquina	***Host***	Máquina
Db	Base de datos	***Db***	Base de datos
User	Usuario	***User***	Usuario
Table_name	Nombre de la tabla	***Table_name***	Nombre de la tabla
		Column_name	Nombre de la columna de esa tabla
Table_priv	Privilegios que se asignan sobre la tabla *		
Column_priv	Privilegios que se asignan sobre la columna **	Column_priv	Privilegios que se asignan sobre la columna **
Timestamp		Timestamp	
Grantor			

***** *'select', 'insert', '', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter'*
****** *'select', 'insert', '', 'Update', 'References'*

- Las tablas "**user**" y "**host**" determinan los permisos en la primera fase de la conexión, en que se comprueba si ese usuario puede acceder al servidor y sus permisos globales. Las tablas "**db**", "**tables_priv**", y "**column_priv**" proporcionan permisos a nivel de base de datos, tabla y columna respectivamente.
- Cada tabla contiene dos tipos de columna:
  - **Columnas de "alcance"**, que determinan a quien afectan los permisos (*host/user* en la tabla "*user*", *host/user/db* para la tabla "*db*"). P.e., la tabla "user" puede contener los permisos para el host "guara.org" y el usuario "antonio", de forma que cuando este usuario se conecte desde ese equipo, se comprobarán sus permisos en esa fila de la tabla "user".
  - **Columnas de "privilegios"**, que indican qué privilegios se otorgan a la cuenta, base de datos, tabla y/o columna, según la tabla de qué se trate. En las tablas "user", "db" y "host" hay una columna para cada tipo de privilegio, que toma el valor "Y" si el privilegio está otorgado o el valor "N" si no lo está. Por defecto tienen el valor "N". En las tablas "tables_priv" y "columns_priv" solo hay un campo para privilegios a nivel de fila y/o columna. Ese campo puede contener cualquier combinación de los privilegios(* y **).
- Los permisos contenidos en las tablas "**user**", "**host**" y "**bd**" son globales (afectan a todos los objetos, es decir, tablas, columnas, índices, etc, de la base de datos).
- Los permisos de la tabla "**tables_priv**" afectan a la tabla de que se trate.
- Los permisos de la tabla "**column_priv**" afectan a la columna de que se trate.

Utilizando phpmyadmin podemos ver un resumen de los privilegios asignados a usuarios:

El servidor usa las tablas así:

- La tabla "**user**" determina los permisos de un usuario que se conecta desde un host concreto. Si se intenta conectar un usuario/host para el que NO existe fila en esta tabla, la conexión será rechazada. Si se conecta un usuario/host para el que SÍ existe fila en esta tabla, se comprueba la contraseña y, si es correcta, se permite la conexión. Las columnas de privilegios indican los privilegios globales del usuario sobre toda la base de datos. Si un permiso en esta tabla está a "Y" significa que el usuario tiene ese permiso sobre todas las bases de datos. Si lo tiene a "N" habrá que

consultar otras tablas. Cuando el usuario intente hacer una operación sobre la BD, el sistema el permitirá o denegará el acceso en función del valor de las columnas de permisos.

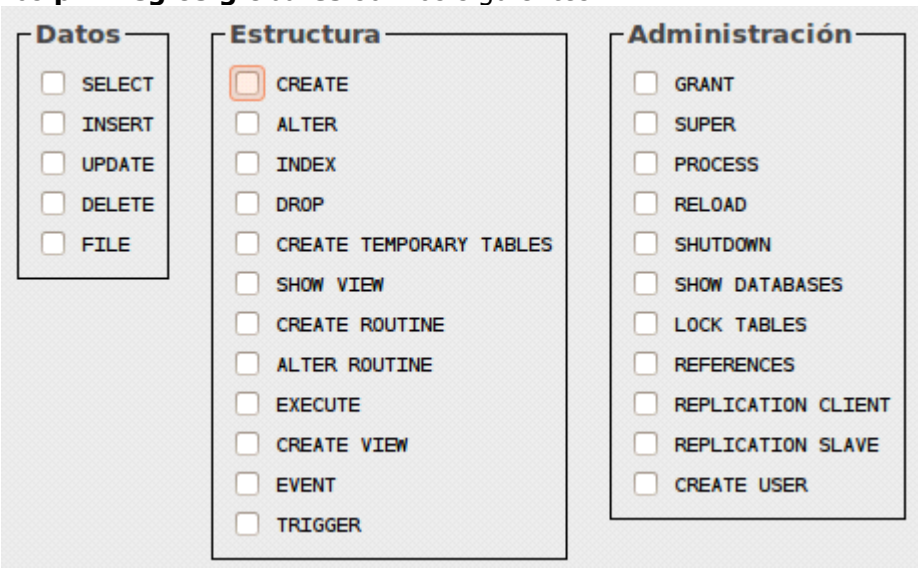
- La tabla "**db**" determina qué usuarios pueden acceder a qué bases de datos desde qué equipos, es decir, contiene los permisos del usuario/host sobre una base de datos concreta. Si un permiso está a "Y" significa que el usuario puede ejercer ese permiso sobre la base de datos indicada.

  ** *Cuando creamos un usuario y le asignamos permisos para TODAS las bases de datos, se crea una fila en la tabla "**user**" con los privilegios asignados a "Y", pero no se crea ningún registro en la tabla "db"*
  ** *Cuando creamos un usuario y le asignamos permisos para una única base de datos, se crea una fila en la tabla "**user**" con los privilegios a "N" y se crea otro registro en la tabla "db" con los privilegios asignados a "Y"*
- La tabla "**host**" se usa conjuntamente con la tabla "db" en casos muy concretos.Si queremos que un usuario pueda utilizar una base de datos desde ciertos equipos de la red, podemos dejar el valor "host" de la tabla "bd" vacío y rellenar la tabla "host" con una fila para cada uno de estos equipos (el campo "Bd" tendrá siempre el mismo valor").
  Para que el servidor MySQL admita la conexión remota de clientes debe tener en la propiedad **bind-address** del fichero **my.cnf** con el valor **0.0.0.0**.(Solo podría tener una única dirección IP de un equipo o bien la de "todos los equipos". Para permitir el acceso desde solo algunas IP se necesita configurar el firewall con las reglas oportunas).
- Las tablas "**tables_priv**" y "**columns_priv**" son similares a la tabla "db", pero más detalladas, ya que contienen los permisos de un usuario/host sobre una *tabla concreta de una base de datos (en el caso de "tables_priv")* o *sobre una columna de una tabla de una base de datos ("columns_priv")*. Un privilegio otorgado a nivel de tabla se aplica para la tabla y todas sus columnas, pero un privilegio otorgado a nivel de columna se aplica solo a esa columna.

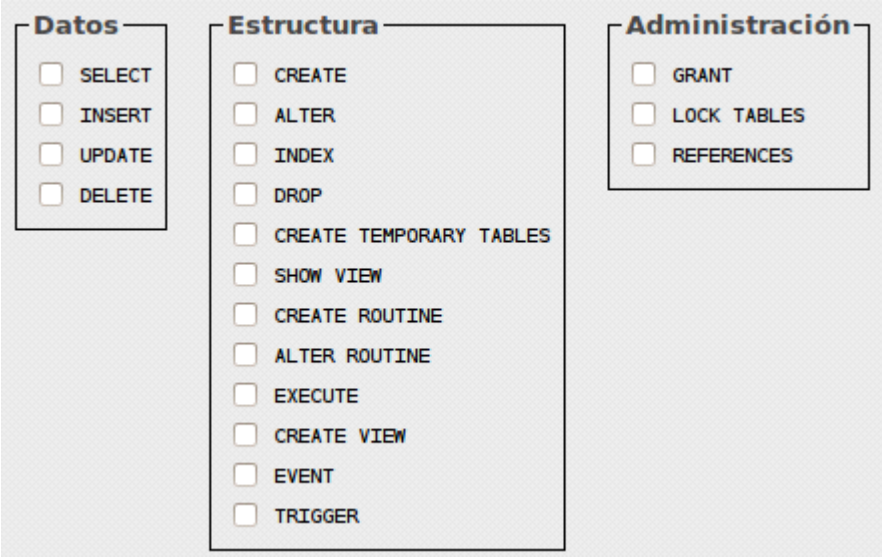
1.1.2. Privilegios y su significado

Los privilegios que se pueden asignar dependen del nivel, no es lo mismo los privilegios a nivel de la base de datos, que a nivel de una tabla.

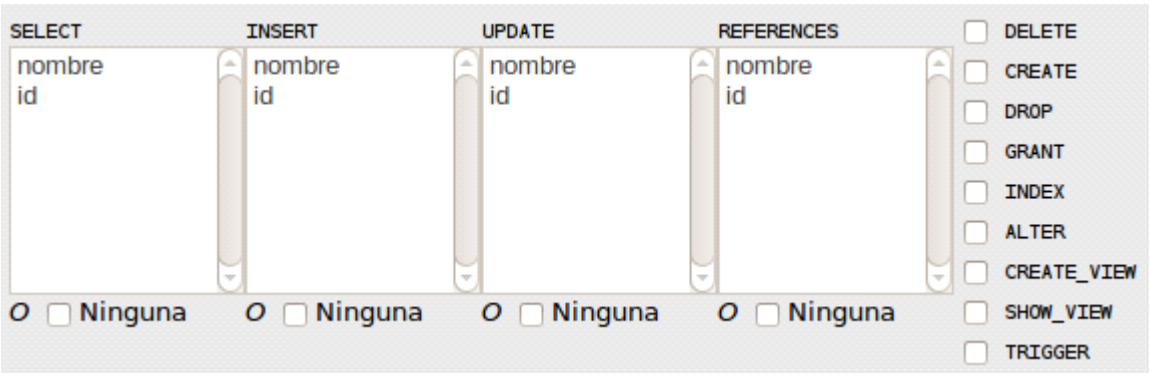
Los **privilegios globales** son los siguientes:



Así para los datos, entre los permisos disponibles están "SELECT", "INSERT",... que permiten al usuario hacer búsquedas o insertar datos.
A nivel de una **base de datos** los permisos disponibles son:



Finalmente, quedan los permisos dentro de una **tabla**. Supongamos una tabla con dos columnas "nombre" e "id":



Se pueden aplicar los permisos por columna, si es necesario.

En cuanto al significado de cada privilegio:

- *Permisos de datos:*
    - SELECT: Permite leer los datos.
    - INSERT: Permite insertar y reemplazar datos.
    - UPDATE: Permite cambiar los datos.
    - DELETE: Permite borrar datos.
    - FILE: Permite importar y exportar datos de y hacia archivos.
- *Permisos de estructura:*
    - CREATE: Permite crear nuevas bases de datos y tablas.
    - ALTER: Permite alterar la estructura de las tablas existentes.
    - INDEX: Permite crear y eliminar índices.

- DROP: Permite eliminar bases de datos y tablas.
- CREATE TEMPORARY TABLES: Permite la creación de tablas temporales.
- SHOW VIEW: Permite llevar a cabo las consultas SHOW CREATE VIEW (MOSTRAR CREAR VER).
- CREATE ROUTINE: Permite crear el almacenamiento de rutinas.
- ALTER ROUTINE: Permite alterar y eliminar las rutinas almacenadas.
- EXECUTE: Permite ejecutar las rutinas almacenadas.
- CREATE VIEW: Permite crear nuevas vistas.
- EVENT: Permite organizar los eventos para el gestor de eventos.
- TRIGGER: Permite crear y eliminar un evento desencadenante.

• *Permisos de administración:*

- GRANT: Permite añadir usuarios y privilegios sin conectarse nuevamente a la tabla de privilegios.
- SUPER: Permite la conexión, incluso si el número máximo de conexiones ha sido alcanzado. Necesario para la mayor parte de operaciones administrativas tales como montar parámetros de variables globales o matar procesos de otros usuarios.
- PROCESS: Permite ver los procesos de todos los usuarios.
- RELOAD: Permite volver a cargar los parámetros del servidor y depurar los cachés del servidor.
- SHUTDOWN: Permite desconectar el servidor.
- SHOW DATABASES: Concede acceso a la lista completa de bases de datos.
- LOCK TABLES: Permite poner candados a las tablas para el proceso actual.
- REPLICATION CLIENT: Da el derecho al usuario para preguntar dónde están los "esclavos / masters".
- REPLICATION SLAVE: Necesario para los "esclavos" de replicación.
- CREATE USER: Permite crear, eliminar y cambiar el nombre de las cuentas de usuario.
- Comodín, se usa como comodín para pasar otras opciones, sirve para no poner nada en el apartado de permisos y MySQL no genere un error:
- USAGE: Sin especificar permisos.

1.1.3. Control de acceso detallado

a) Control de acceso nivel 1: Comprobación de la conexión
Cuando intentamos conectarnos con un servidor MySQL, éste aceptará o no la conexión según el usuario/host (validando también la contraseña).
El usuario/host se comparan con las filas de la tabla "user". Si existe una fila para ese usuario/host de conexión y la contraseña es correcta, la conexión se acepta.
Las contraseñas se almacenan cifradas, de forma que nunca viajan a través de la red en texto plano.
Podría darse el caso de que el host/usuario de conexión coincida con más de un registro de la tabla "user". Si esto sucede, la table "user" se ordena en primer lugar con los valores de "host" más específicos, y en segundo lugar con los valores de "user" más específicos. Los valores "%" o blancos no son específicos. P.e, para la siguiente tabla:

Host	User
%	antonio
%	admin
localhost	admin
localhost	

Los valores de host más específicos son "localhost", frente a % que es más genérico, por lo que la dos últimas filas serán las primeras. De ellas, la que tiene valor para el campo "user" será la primera:

Host	User
localhost	admin
localhost	
%	antonio
%	admin

Cuando un usuario se intenta conectar, el servidor mira los registros ordenados y usa la primera concordancia.

b) Control de acceso nivel 2: Verificación de permisos

Una vez aceptada una conexión, por cada operación que se intente hacer dentro de esa conexión, el servidor determina la operación a realizar y si la cuenta tiene suficientes privilegios para hacerla. Aquí es donde intervienen las columnas de privilegios de las 5 tablas.

- La tabla "**user**" otorga privilegios de forma global, sin importar la BD, tabla o columna sobre la que se realice la operación. Por ejemplo, si la tabla "user" otorga el privilegio *DELETE*, se podrá borrar registros de cualquier tabla de cualquier BD del servidor. *Es aconsejable otorgar privilegios en la tabla "user" solamente a superusuarios*. Para el resto de usuarios se deberían dejar los privilegios de esta tabla con el valor "N" y otorgar los privilegios en niveles más específicos.
- Las tablas "db" y "host" otorgan privilegios específicos para una base de datos. Igual que antes, estas dos tablas se ordenan de valores más específicos a menos específicos primero por *host*, después por d*b* y, en el caso de la tabla "db", por último por "*user*". El campo "db" no puede contener valores vacíos ni comodines.
- Las tablas "tables_priv" y "columns_priv" se ordenan también basándose en las columnas host, db y user.

El proceso de verificación de peticiones funciona así:

- Para peticiones que requieren privilegios de administrador, como SHUTDOWN o RELOAD, el servidor comprueba únicamente el registro de la tabla "user" porque es la única tabla que especifica estos privilegios. La operación se puede realizar si el privilegio solicitado tiene el valor "Y".
- Para peticiones sobre bases de datos (INSERT, UPDATE, etc), el servidor comprueba primero los privilegios globales de usuario (en la tabla "user"). Si en esta tabla el privilegio está a "Y", se otorga el acceso. En caso contrario, se comprueban las tablas "db" y "host":
  - El servidor busca en la tabla "db" una fila que concuerde con los valores de "host", "db" y "user". Las columnas "host" y "user" se comparan con el host y usuario usados en la conexión y la columna "db" se compara con la base de datos sobre la que se está intentando hacer la operación.

  - Si no hay ningún registro en la tabla "db" que concuerde que estos 3 valores, no se permite realizar la operación (se deniega).
  - Si hay un registro que concuerda y su columna "host" no está vacía, este registro define los privilegios para la operación. *
  - Si hay un registro que concuerda y su columna "host" está vacía, se busca concordancia para el "host" y "bd" en la tabla "host". Si no existe, la operación se deniega y, si existe, el permiso se calcula como la intersección de los privilegios en los registro concordantes de las tablas "db" y "host". Es decir, el privilegio será "Y" si tablas tienen el valor del privilegio a "Y" en sus registros concordantes.
- Una vez determinados los privilegios específicos de la base de datos otorgados por las tablas "db" y "hosts", si el privilegio está a "Y", se permite realizar la operación. Pero si no es así, el servidor comprueba las tablas "tables_priv" y "columns_priv".

Si nos conectamos con un usuario y no tenemos privilegios suficientes para ejecutar alguna operación, ponemos comprobar:
a) El host/usuario de la tabla "user" con el que realmente nos hemos conectado:
mysql> SELECT CURRENT_USER();
b) Los privilegios del usuario de conexión:
mysql> SHOW PRIVILEGES

1.2. Gestión de recursos

Existen varias formas de crear cuentas de usuario y asignarles privilegios en MySQL:

- Con los comandos GRANT y CREATE USER. Es el método recomendado.
- Manipulando directamente las tablas de permisos de MySQL, es decir, insertando filas en las tablas "user", "db", "host", "table_privs" y "column_privs"
- Usando herramientas de administración de tipo gráfico, como "phpmyadmin".

1.2.1. Comandos CREATE USER y GRANT

Es el método más recomendado, ya que es el menos propenso a errores.

- ***Ver los usuarios disponibles***

  select user,host,password from mysql.user;

  Por ejemplo:

```
mysql> select user,host,password from mysql.user;
+------------------+-----------+-------------------------------------------+
| user             | host      | password                                  |
+------------------+-----------+-------------------------------------------+
| root             | localhost | *4ACFE3202A5FF5CF467898FC58AAB1D615029441 |
| root             | debian    | *4ACFE3202A5FF5CF467898FC58AAB1D615029441 |
| root             | 127.0.0.1 | *4ACFE3202A5FF5CF467898FC58AAB1D615029441 |
| debian-sys-maint | localhost | *A9FAC38B2551F61953C70BC0DE0F0D9A229DB4EB |
+------------------+-----------+-------------------------------------------+
4 rows in set (0.00 sec)
```

- ***Saber con qué usuario estamos conectados***

  select user();

- ***Crear un usuario***

Para crear un usuarios se usará:

create user 'usuario'@'host';

Por ejemplo:

create user 'lolo'@'192.168.56.101';

Crea el usuario 'lolo' que se puede conectar desde la IP 192.168.56.101.

Otro ejemplo:

create user 'lolo'@'%';

Importante: Si se pone el símbolo '%' en el host, se indicará que se puede conectar desde cualquier IP.

Se ha creado el usuario, pero no se le ha asignado contraseña, como se puede ver en la lista de usuarios:

select user,host,password from mysql.user;

```
| lolo             | 192.168.56.101 |
```

- ***Cambiar contraseña a un usuario***

Para cambiar la contraseña a un usuario se usará:

set password for usuario@servidor=password('contraseña');

** El comando "password()" permite encriptar la contraseña.

Por ejemplo, para cambiar la contraseña al usuario root que se conecta desde localhost, se usará:

set password for root@localhost=password('admin1');

Con lo que la contraseña para root en localhost cambiará:

```
mysql> select user,host,password from mysql.user;
+------------------+-----------+-------------------------------------------+
| user             | host      | password                                  |
+------------------+-----------+-------------------------------------------+
| root             | localhost | *6D45FD76D5E9C6A404E39C25106A7F032659ACB8 |
| root             | debian    | *4ACFE3202A5FF5CF467898FC58AAB1D615029441 |
| root             | 127.0.0.1 | *4ACFE3202A5FF5CF467898FC58AAB1D615029441 |
| debian-sys-maint | localhost | *A9FAC38B2551F61953C70BC0DE0F0D9A229DB4EB |
+------------------+-----------+-------------------------------------------+
4 rows in set (0.00 sec)
```

Para volver a cambiar la contraseña:

set password for root@localhost=*password('admin');*

Para cambiar la contraseña de nuestra cuenta (la cuenta con la que estamos conectados) omitimos el nombre de la cuenta:

set password =password('guara');

También se puede usar el programa mysqladmin:

mysqladmin -u usuario -h servidor password 'contraseña' -p

Por ejemplo:

mysqladmin -u root -h localhost password 'admin1' -p

- ***Eliminación de usuarios***

  Para eliminar un usuario, se usará:

  drop user 'usuario'@'host';

  Por ejemplo:

  drop user 'lolo'@'192.168.56.101';

- ***Asignación/revocación de privilegios a usuarios***

  Para administrar los privilegios se usarán las sentencias 'grant' y 'revoke'.
  Supongamos que se desean dar permisos al usuario 'lolo'@'%' para que pueda hacer consultas (permiso SELECT) dentro de la base de datos "ejemplo":

  grant SELECT on ejemplo. to 'lolo'@'%';*

  Con esta orden se dan permisos (grant), de lectura (SELECT), en todas las las tablas de la base de datos ejemplo (on ejemplo.*), al usuario 'lolo'@'%' (to 'lolo'@'%').

  En el caso de asignar varios permisos se pueden separar con comas:

  grant SELECT,INSERT on ejemplo. to 'lolo'@'%';*

  O si queremos asignar todos los permisos, podemos usar:

  grant ALL PRIVILEGES on ejemplo. to 'lolo'@'%';*

  De forma análoga se pueden eliminar permisos usando "revoke":

  revoke SELECT,INSERT on ejemplo. from 'lolo'@'%';*

  O quitar todos los privilegios disponibles:

  revoke ALL PRIVILEGES on ejemplo. from 'lolo'@'%';*

  Si no hay permisos que eliminar, MySQL puede generar mensajes de error. Por ejemplo:

```
[(none)]> revoke ALL PRIVILEGES on ejemplo.* from 'lolo'@'%';
ERROR 1141 (42000): There is no such grant defined for user 'lolo'
 on host '%'
[(none)]> grant SELECT,INSERT on ejemplo.* to 'lolo'@'%';
Query OK, 0 rows affected (0.00 sec)

[(none)]> revoke ALL PRIVILEGES on ejemplo.* from 'lolo'@'%';
Query OK, 0 rows affected (0.00 sec)
```

  Si se desean dar permisos para todas las bases de datos existentes, se usará "*.*" a la hora de seleccionar la base de datos:

  *grant ALL PRIVILEGES on *.* to 'lolo'@'%';*

  *** Esta sentencia crea un "superusuario" o administrador de todas las bd.*

  De igual manera para quitar permisos:

  *revoke ALL PRIVILEGES on *.* from 'lolo'@'%';*

  En el caso de querer dar permisos sólo a una columna de una tabla, se indicará entre paréntesis el nombre de la columna cuando se especifique el permiso. Por ejemplo, para permitir hacer búsquedas sobre la columna "id" de la tabla "ejemplo.nombres":

  grant select(id) on ejemplo.nombres to 'lolo'@'%';

La lista de permisos posibles es:

Permiso	Significado
`ALL [PRIVILEGES]`	Da todos los permisos simples excepto `GRANT OPTION`
`ALTER`	Permite el uso de `ALTER TABLE`
`ALTER ROUTINE`	Modifica o borra rutinas almacenadas
`CREATE`	Permite el uso de `CREATE TABLE`
`CREATE ROUTINE`	Crea rutinas almacenadas
`CREATE TEMPORARY TABLES`	Permite el uso de `CREATE TEMPORARY TABLE`
`CREATE USER`	Permite el uso de `CREATE USER`, `DROP USER`, `RENAME USER`, y `REVOKE ALL PRIVILEGES`.
`CREATE VIEW`	Permite el uso de `CREATE VIEW`
`DELETE`	Permite el uso de `DELETE`
`DROP`	Permite el uso de `DROP TABLE`
`EXECUTE`	Permite al usuario ejecutar rutinas almacenadas
`FILE`	Permite el uso de `SELECT ... INTO OUTFILE` y `LOAD DATA INFILE`
`INDEX`	Permite el uso de `CREATE INDEX` y `DROP INDEX`
`INSERT`	Permite el uso de `INSERT`
`LOCK TABLES`	Permite el uso de `LOCK TABLES` en tablas para las que tenga el permiso `SELECT`
`PROCESS`	Permite el uso de `SHOW FULL PROCESSLIST`
`REFERENCES`	No implementado
`RELOAD`	Permite el uso de `FLUSH`
`REPLICATION CLIENT`	Permite al usuario preguntar dónde están los servidores maestro o esclavo
`REPLICATION SLAVE`	Necesario para los esclavos de replicación (para leer eventos del log binario desde el maestro)
`SELECT`	Permite el uso de `SELECT`
`SHOW DATABASES`	`SHOW DATABASES` muestra todas las bases de datos
`SHOW VIEW`	Permite el uso de `SHOW CREATE VIEW`
`SHUTDOWN`	Permite el uso de **mysqladmin shutdown**
`SUPER`	Permite el uso de comandos `CHANGE MASTER`, `KILL`, `PURGE MASTER LOGS`, and `SET GLOBAL` , el comando **mysqladmin debug** le permite conectar (una vez) incluso si se llega a `max_connections`
`UPDATE`	Permite el uso de `UPDATE`
`USAGE`	Sinónimo de “no privileges”
`GRANT OPTION`	Permite dar permisos

- ***Crear usuarios con GRANT***

La sentencia GRANT asigna privilegios al usuario especificado pero, además, si el usuario no existe, lo crea. Es posible asignar contraseña en la misma sentencia:

*grant ALL PRIVILEGES on *.* to 'lolo'@'%' identified by 'sierra' WITH GRANT OPTION;*

- ***WITH GRANT OPTION***

La cláusula "WITH GRANT OPTION " sirve para permitir que un usuario transmita sus privilegios a otros usuarios.
Por ejemplo:

grant SELECT,INSERT on *.* to 'lolo'@'%' WITH GRANT OPTION;

Con lo cual el usuario 'lolo@%' podría dar los permisos SELECT o INSERT sobre cualquier Base de datos a otros usuarios.

1.2.2. Creación de usuarios y privilegios manipulando las tablas de permisos

Se pueden crear usuarios y asignarle privilegios insertando filas en las tablas de permisos (user, host, db, tabla_privs y column_privs). En este caso debemos recargar las tablas de permisos usando el comando "FLUSH PRIVILEGES", ya que dichas tablas se cargan un memoria cuando se inicia "mysqld" y no se vuelven a cargar hasta que se reinicia el servicio. Por tanto, si manipulamos a mano esas tablas, es necesario recargarlas con "FLUSH PRIVILEGES" o reiniciar el servicio.

Ejemplo 1:

Dada la sentencia:

*grant ALL PRIVILEGES on *.* to 'antonio'@'%' identified by 'sierra' WITH GRANT OPTION;*

Obtener el mismo resultado manipulando directamente las tablas de permisos

```
c:\mySQL--u root mysql
mysql> INSERT INTO user VALUES ('%','antonio',PASSWORD('sierra'),
        'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
        // Es necesario usar el comando PASSWORD() para encriptar la contraseña
mysql>FLUSH PRIVILEGES
```

Ejemplo 2:

Dadas la sentencias:

grant SELECT, INSERT, UPDATE, DELETE, CREATE, DROP on prueba.*
to 'user_mblog'@'localhost' identified by 'guara';

grant SELECT, INSERT, UPDATE, DELETE, CREATE, DROP on prueba.*
to 'user_mblog'@'guara.com';

Obtener el mismo resultado manipulando directamente las tablas de permisos

```
c:\mySQL--u root mysql
mysql> INSERT INTO user (user,host,password)
         VALUES ('user_mblog','localhost',PASSWORD('guara'));
        // Solo datos valor para user, host y contraseña. Los permisos por defecto tienen el valor 'N'
mysql> INSERT INTO user (user,host) VALUES ('user_mblog','guara.com');
        // Idem
mysql> INSERT INTO db (user,host,db,Select_priv, Insert_priv, Update_priv,
         Delete_priv, Create_priv, Drop_priv)
        VALUES('user_mblog','localhost','prueba','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db (user,host,db,Select_priv, Insert_priv, Update_priv,
          Delete_priv, Create_priv, Drop_priv)
          VALUES('user_mblog','guara.com','prueba','Y','Y','Y','Y','Y','Y');
```

```
mysql>FLUSH PRIVILEGES
```

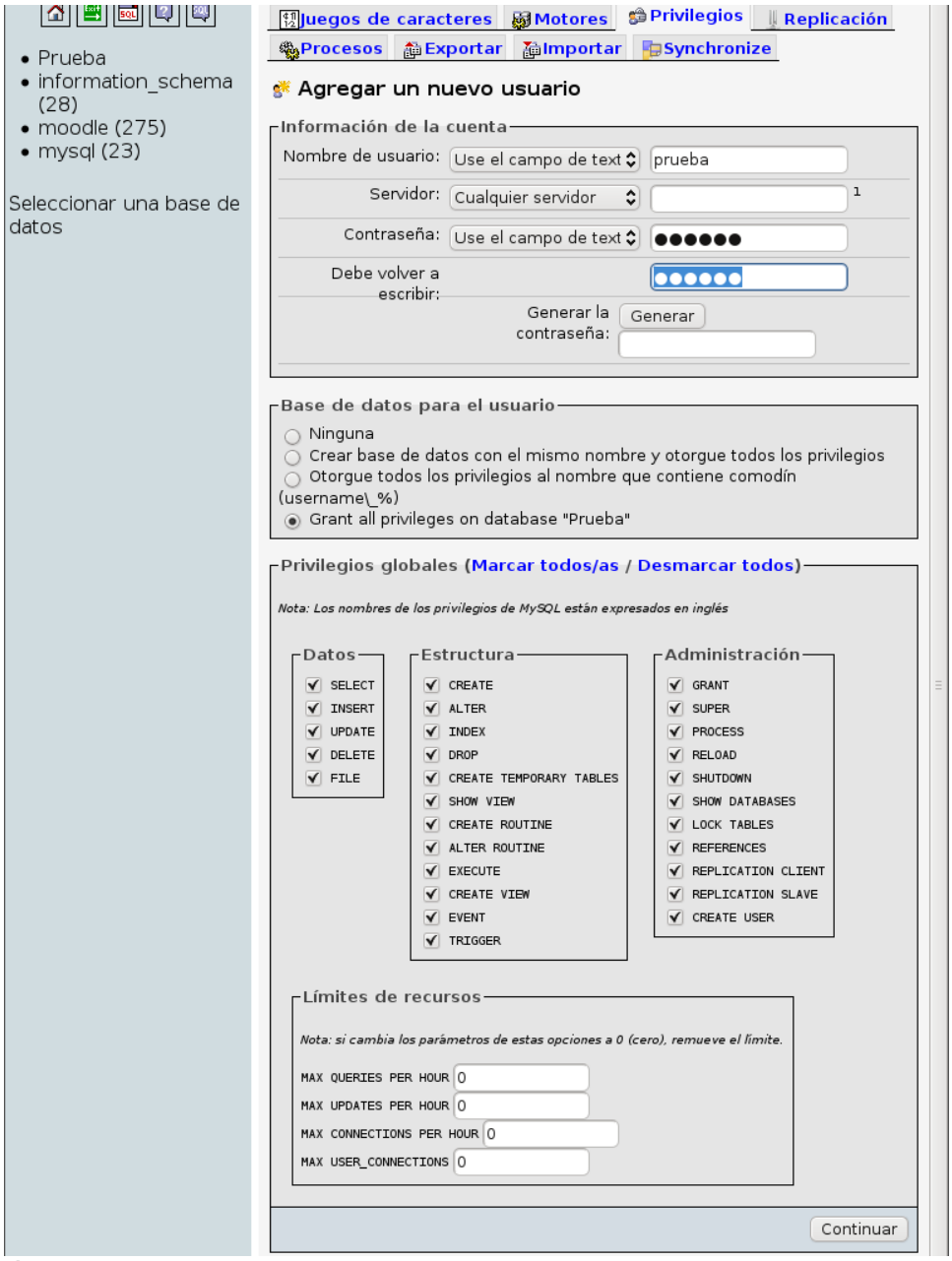
Las dos primeras sentencias INSERT añaden registros en la tabla "user" que permiten al usuario "user_mblog" conectar desde los equipos local y guara.com, pero no otorga privilegios globales (todos los privilegios se inicializan por defecto a 'N'). Si asignáramos permisos en esta tabla, se asignarían no solo para la base de datos "motorblog", si no para todas las BD del servidor.
A continuación, las dos siguientes INSERT añaden registros en la tabla "db" que otorgan privilegios a user_mblog para la base de datos "motorblog" desde los host especificados. Para que los registros insertados tengan efecto debemos recargar los permisos usando el comando "FLUSH PRIVILEGES".

1.2.3. Creación de usuarios y privilegios con phpMyAdmin

Para crear un usuario "prueba" en la base de datos "Prueba" con PhpMyAdmin, se siguen los siguientes pasos:

1. Se hace clic en la base de datos para poder gestionarla.
2. Después se hace clic en la pestaña "Privilegios" y se selecciona "Agregar un nuevo usuario":

3. En la ventana que aparecerá, se le pone nombre al usuario, contraseña y se marcan los privilegios que se desee que tenga el usuario en la base de datos. En este caso se desea que el usuario tenga todos los privilegios, por lo que se han marcado todos:



Se podrían limitar los privilegios del usuario para, por ejemplo, no pudiese crear otros usuarios ("CREATE USER"), o que sólo tenga permisos para ver las tablas (se dejaría activa la opción "SELECT" y el resto desactivadas).

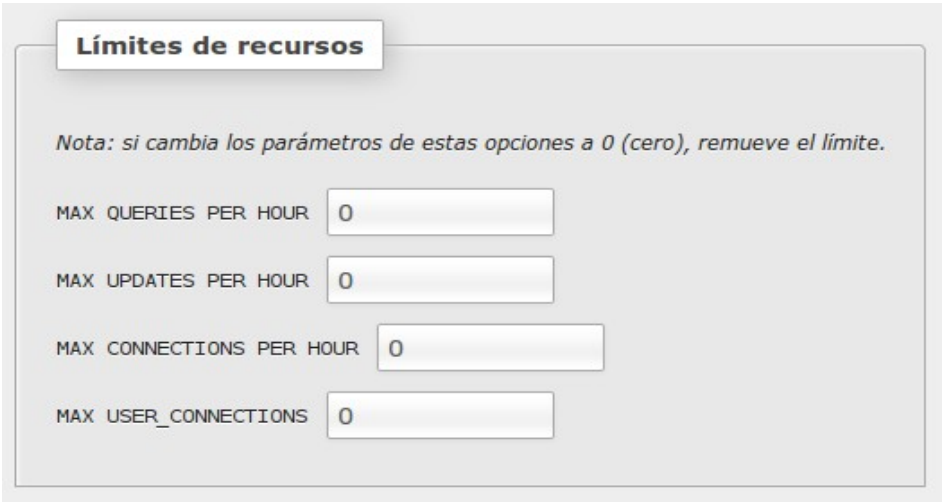
4. Por último se hace clic en el botón "Continuar".

Para cambiar las propiedades de un usuario, se debe hacer clic en la pestaña Privilegios y hacer clic en la casilla "Acción" del usuario que se desee modificar:

1.2.4. Limitar recursos de cuentas

En MySQL podemos limitar los siguientes recursos de servidor para cuentas individuales:

- Número de consultas que una cuenta puede realizar por hora (cualquier comando)
- Número de actualizaciones que una cuenta puede realizar por hora (comandos que modifiquen la BD)
- Número de veces que una cuenta se puede conectar con el servidor por hora o en general.



El valor de estos recursos se guardan en la tabla "user", concretamente en los campos "**max_questions", "max_updates", "max_connections" y "max_user_connections".**

- "**max_connections"** indica el número de conexiones que la cuenta puede hacer en una hora.
- "**max_user_connections"** limita el máximo número de conexiones simultáneas que la cuenta puede hacer. Si es 0 (por defecto), la variable del sistema con el mismo nombre (max_user_connections) determina el número de conexiones simultáneas para las cuentas.

Para cambiar el valor límite de estos recursos usamos la sentencia GRANT con las cláusula WITH, que nombre cada recursos y le asigna un valor. Por ejemplo:

```
GRANT ALL ON motorblog.* TO 'admin2_mblog'@'localhost'
IDENTIFIED BY 'sierra'
WITH MAX_QUERIES_PER_HOUR 20
MAX_UPDATES_PER_HOUR 10
MAX_CONNECTIONS_PER_HOUR 5
MAX_USER_CONNECTIONS 2;
```

No es necesario indicar todos los recursos. Los recursos que no se indiquen toman por defecto el valor 0 que significa "recurso ilimitado" (excepto MAX_USER_CONNECTIONS, como se ha especificado antes).
Para inicializar o cambiar los límites de una cuenta que ya existía debemos usar el comando GRANT USAGE a nivel global (ON *.*). Por ejemplo, si queremos modificar el límite de consultas para "user_mblog" a 100:

```
GRANT USAGE ON *.* to 'user_mblog'@'localhost' WITH MAX_QUERIES_PER_HOUR 100
```

* El comando deja los permisos de la cuenta inalterados

Para eliminar un límite existente ponemos su valor a cero. Por ejemplo, para eliminar el línite de cuántas veces por hora se puede conectar "user_mblog" haremos:

```
GRANT USAGE ON *.* to 'user_mblog'@'localhost' WITH MAX_CONNECTIONS_PER_HOUR 0
```

1.3. Conexiones seguras con SSH

Vamos a explicar como usar el servicio ssh para establecer comunicaciones seguras entre el cliente y el servidor MySQL.

SSH (Secure Shell) es un protocolo similar a telnet que permite abrir un shell en una máquina remota. La diferencia respecto a Telnet es que SSH encapsula toda la información que viaja por la red usando criptografía de clave pública.

Este protocolo se implementa con arquitectura cliente-servidor, por lo que necesita un servidor SSH en la máquina remota (por defecto, en el puerto 22 TCP) y un cliente SSH que nos permita conectarnos a ese servidor.

Con OpenSSH se puede crear un túnel encriptado entre dos máquinas y enviar a través de él los datos de otros protocolos (algo parecido a una VPN). Esto es útil para *entunelar* protocolos inseguros como Telnet, FTP, HTTP, MySQL, etc.

En nuestro caso, desde el cliente queremos hacer consultas al servidor de BD MySQL instalado en el servidor. Pero MySQL no soporta conexiones seguras entre los clientes y el servidor, de manera que la identificación de los usuarios remotos y los datos de las consultas serán visibles a cualquier máquina de la red. Por ello, utilizaremos SSH de la siguiente manera:

- En el cliente, el cliente MySql (mysql) hace una petición al puerto 3306 TCP de localhost, donde escucha el cliente de SSH (ssh).
- El cliente SSH (ssh) escucha el puerto 3306 y redirecciona las peticiones al servidor SSH (sshd) a través del túnel, de manera que todo el tráfico de datos entre el cliente y el servidor MySQL van cifrados por la red.
- En el servidor, el servidor SSH (sshd) redirecciona el tráfico procedente del túnel al puerto 5000 TCP de localhost, donde escucha el servidor MySQL (mysqld), de manera que éste ve las conexiones como procedentes de la propia máquina.

Procedimiento a seguir:

- En el servidor MySQL (máquina Ubuntu), cambiar en el fichero de configuración /etc/mysql/my.cnf lo siguiente (se trata de que escuche en el puerto 5000 en vez de en el 3306):

  [mysqld]
  #skip-networking
  bind-address=127.0.0.1
  port=5000

  Además, el servidor tiene que tener instalado un servidor SSH.

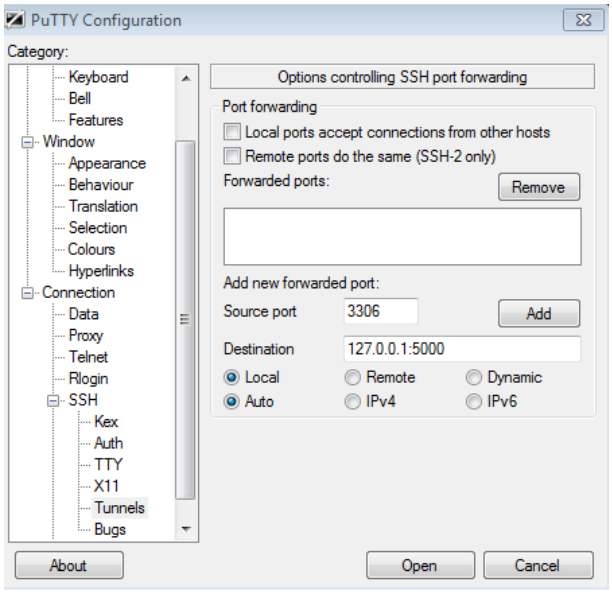
- En el cliente (máquina desde la que accederemos a la base de datos en remoto, por ejemplo W7standard), debemos tener instalado:
  - Un cliente "mysql" (puede valer el WAMP, no hace falta que esté arrancado el servicio, con tener disponible el programa "mysql" vale).
  - Un cliente "ssh". Podemos probar con "openSSH" o con putty.

- Desde el cliente debemos crear un túnel SSH que redirija las peticiones que el cliente hace a su puerto 3306 (cliente de "mysql") al puerto 5000 del servidor (que es el puerto en el que escucha el servidor "mysql" en la máquina de Ubuntu). Para hacer el puente:

  - Si tenemos instalado openSSH u otro software de SSH por línea de comandos:

    ssh -N -L 3306:127.0.0.1:5000 usuario@servidor
    **La opción N establece el túnel y L indica que este extremo es el cliente del túnel*

  - También podemos hacerlo por vía gráfica si tenemos "putty". En la opción "SSH/Tunnels" configuramos el puerto origen (3306) y el servidor/puerto destino (127.0.0.1:5000) (fijarse que estamos referenciando al servidor con la dirección 127.0.0.1).



  A continuación rellenamos la IP de servidor en "Session" y nos conectamos por "ssh".

- Una vez creado el túnel, hacemos una conexión mysql desde el cliente:

  mysql -u usuario -p

Si en el cliente tenemos instalado "mysql" (con WAMP) podríamos pensar que con esta sentencia se llama a la BD del cliente. Pero no es así, porque esta sentencia hace una petición al puerto 3306 del cliente y dicho puerto, gracias al túnel, se redirige al puerto 5000 de la máquina "ubuntu", donde escucha el servidor "mysql". Por tanto, nos conectamos al servidor, no al cliente. Y además, puesto que el túnel es SSH, lo hacemos de forma segura.

***IMPORTANTE**: No funciona utilizando el puerto "3306" desde el cliente, pero sí cambiándolo por el 3306 (por tanto, el túnel se debe hacer al puerto 3307). Puede que la razón sea que el cliente tiene WAMP y el puerto 3306 escucha tanto a peticiones cliente como servidor. PAra cambiar el puerto a 3307 solo se debe tocar la sección [client] del fichero "my.sql":*

```
[client]
#password       = your_password
port            = 3307
socket          = /tmp/mysql.sock
```

*** Al crear el puente desde putty, si pongo "192.168.50.4:5000" en vez de "127.0.0.1" no funciona.*