Practica 4. (SGBD: MySQL)

Instrucciones generales: Elabore un archivo PDF en donde reporte lo que se pide en cada una de las partes de la práctica utilizando capturas de pantalla y descripciones detalladas de lo que se hizo y de lo que sucedió.

Primera parte.

Reportar lo sucedido al establecer los límites de recursos por cuenta: Consultas por hora, actualizaciones por hora, Conexiones por hora y Máximo de conexiones por usuario.

Segunda Parte.

Para asignar o quitar privilegios y roles a los usuarios, los administradores del sistema cuentan con las sentencias GRANT y REVOKE.

Los privilegios pueden ser asignados por los administradores del sistema tanto a los usuarios como a los roles.

Supongamos ahora, que ya estamos conectados a nuestro sistema gestor. Procedemos a crear los usuarios necesarios para nuestra nueva aplicación, pero cada uno de ellos debe de tener privilegios diferentes sobre la o las Bases de Datos. Para esto utilizaremos las sentencias CREATE USER y GRANT, la sintaxis de cada una de ellas es la siguiente:

```
CREATE USER 'usuario'@'host' IDENTIFIED BY 'password';
```

En la siguiente tabla podemos observar las diferentes configuraciones que puede tomar los campos o valores de usuario y de host.

Valor del usuario	Valor del host	Conexiones permitidas
'fred'	'h1.example.net'	Usuario fred conectado desde el host h1.example.net
''	'h1.example.net'	Cualquier usuario conectado desde el host h1.example.net
'fred'	'%'	Usuario fred conectado desde cualquier host
''	'%'	Cualquier usuario desde cualquier host
'fred'	'%.example.net'	Usuario fred conectado desde cualquier host del dominio example.net
'fred'	'x.example.%'	Usuario fred conectado desde x.example.net, x.example.com, x.example.edu, y así sucesivamente.
'fred'	'198.51.100.177'	Usuario fred conectado desde el host con la IP 198.51.100.177
'fred'	'198.51.100.%'	Usuario fred conectado desde cualquier host que pertenezca a la subred 198.51.100 clase C.
'fred'	'198.51.100.0/255.255.255.0'	Lo mismo que el de arriba.

Existirán ocasiones en las que el nombre del host y del usuario de una conexión entrante coincidan en más de una de los registros en la tabla de usuarios. (Se puede ver como ejemplo la tabla anterior).

Cuando esto ocurre, el servidor debe determinar cuál de ellas utilizar. Por lo que esto se resuelve así:

- Por cada conexión que recibe el servidor, este reordena la tabla de usuarios en la memoria.
- El servidor siempre utiliza el primer registro que coinciden con el nombre del host (cliente) y el usuario.
- Las reglas de clasificación van desde las más específicas hasta las menos específicas. Una dirección IP y una dirección IP con su máscara de red son igualmente especificas por lo que estas se acomodan primero, después se ordenan a los que tiene el patrón ´%´ y después los de la cadena vacía (´ ´).

Cuando existen registros con el mismo host, pasa exactamente lo mismo que el punto anterior solo que ahora lo hace con los usuarios más específicos.

Entonces para crear los usuarios tenemos:

```
CREATE USER 'desarrollador'@'localhost' IDENTIFIED BY 'despass';
```

```
CREATE USER 'administrador'@'%' IDENTIFIED BY 'adminpass';

CREATE USER 'administrador'@'localhost' IDENTIFIED BY 'adminpass';
```

Se han creado 3 cuentas, una de desarrollador y 2 más de administrador. Ahora procederemos a signarle los privilegios necesarios a cada usuario. Para esto tenemos GRANT y su sintaxis es:

```
GRANT [privilegios o roles] base_de_datos.tabla_o_tablas TO 'usuario'@'host';
```

Por lo tanto, para asignar los privilegios hacemos:

```
GRANT ALL ON *.* TO 'administrador'@'%';

GRANT RELOAD, PROCESS *.* TO 'administrador'@'localhost';

GRANT ALL ON db_de_la_aplicacion.* TO 'desarrollador'@'localhost';
```

En la primera de las tres líneas anteriores se le otorgan todos los privilegios (crear, eliminar, modificar, etc.) al usuario administrador. En la segunda de manera local solo los privilegios de administración globales mysqladmin reload, mysqladmin refresh, and mysqladmin flush-xxx commands, así como mysqladmin processlist.

Y en la última línea al usuario desarrollador todos los privilegios en la base de datos solamente de la aplicación. Existe otra manera de otorgar privilegios a los usuarios y es por medio de los roles.

Los roles son conocimos también como ***colecciones de privilegios***. Los roles tienen una sintaxis y semántica parecida a la de las cuentas de usuarios. Por lo que un nombre de rol al igual que un nombre de usuario no puede estar en blanco. Por lo tanto, el ***rol anónimo*** no existe, como no existe el ***usuario anónimo***.

Si se usa en la parte de host '%' como en las cuentas de usuario no surte el mismo efecto de comodín (cualquier host), solo pasa como valor literal.

Para crear un rol, lo hacemos así:

```
CREATE ROLE 'nombre_del_rol';
```

Supongamos que necesitamos crear 3 roles diferentes, uno para los usuarios desarrolladores, otro para usuarios de solo lectura y el último para aquellos usuarios que puedan insertar, borrar o modificar los registros de una tabla o base de datos especifica.

```
CREATE ROLE 'desarrollador', 'lectura', 'escritura';
```

Para asignarle a cada uno los privilegios específicos hacemos:

```
GRANT ALL ON db_de_la_aplicacion.* TO 'desarrollador';

GRANT SELECT ON db_de_la_aplicacion.* TO 'lectura';

GRANT INSERT, UPDATE, DELETE ON db_de_la_aplicacion.* TO 'escritura';
```

Ahora solo nos falta asignar esto roles a los usuarios necesarios, por lo que por el momento contamos con un usuario desarrollador, dos usuarios que harán solo consultas de lectura y otro usuario más que tenga acceso de lectura/escritura.

```
CREATE USER 'desarrollador1'@'localhost' IDENTIFIED BY 'des1pass';

CREATE USER 'usuario_r1'@'localhost' IDENTIFIED BY 'read1pass';

CREATE USER 'usuario_r2'@'localhost' IDENTIFIED BY 'read2pass';

CREATE USER 'usuario_rw'@'localhost' IDENTIFIED BY 'rw1pass';
```

Entonces, ahora para asignar los privilegios correspondientes a cada uno de los usuarios hacemos uso nuevamente de la sentencia GRANT

```
GRANT 'desarrollador' TO 'desarrollador1'@'localhost';

GRANT 'lectura' TO 'usuario_r1'@'localhost', 'usuario_r2'@'localhost';

GRANT 'escritura', 'lectura' TO 'usuario_rw'@'localhost';
```

GRANT no puede otorgar privilegios y roles en la misma sentencia. Una declaración GRANT debe otorgar o privilegios o roles.

Se utiliza ON para distinguir si se están otorgando privilegios o roles:

- Con ON se otorgan privilegios
- Sin ON se otorgan roles

Se pueden asignar privilegios y roles a una cuenta, pero utilizando sentencias GRANT por separado, con su sintaxis adecuada.

Con la sentencia SHOW GRANTS, podemos obtener un listado de los privilegios y roles asignados a un usuario determinado.

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

Con esta sentencia anterior solo se muestran el o los roles asignados al usuario sin dar una lista detallada de los privilegios. Para que nos muestre un listado detallado de los privilegios que el usuario tiene utilizamos la cláusula USING acompañada del nombre del rol o roles.

```
SHOW GRANTS FOR 'usuario_rw'@'localhost' USING 'escritura';
```

Para revocar o eliminar roles o colecciones de privilegios, tenemos la sentencia REVOKE como ya lo habíamos mencionado. Su sintaxis es la siguiente:

```
REVOKE role FROM user;
```

Solo aquellos roles que sean nombrados en la variable del sistema `mandatory_roles`, no pueden ser eliminados o revocados.

Supongamos que temporalmente necesitamos que los usuarios con los privilegios de lectura y escritura, sean usuarios de solo lectura. Entonces hacemos lo siguiente:

```
REVOKE INSERT, UPDATE, DELETE ON db_de_la_aplicacion.* TO 'escritura';
```

Con esto no solo se afecta el rol, si no también todos los usuarios que tiene este rol asignado.