Apuntes accesibles UT8

Sitio: FRANCISCO DE GOYA
Curso: Bases de Datos Pendientes
Libro: Apuntes accesibles UT8

Imprimido por: Lucía Hernández Montero
Día: lunes, 24 de octubre de 2022, 09:44

Tabla de contenidos

1. Seguridad en Bases de Datos

La seguridad que se puede dotar a una base de datos abarca desde la seguridad del equipo donde se aloja el SGBD y su lugar físico, así como de la propia seguridad del SGBD.

Dentro de la seguridad propia de un SGBD hay muchos elementos a tener en cuenta: puertos utilizados, permisos y usuario del sistema que ejecuta el SGBD, hasta los propios usuarios internos SGBD.

Y las pérdidas o perjuicios que pueden tenerse son: pérdida de usuarios, accesos indebidos, pérdida de los datos.

En esta unidad de trabajo veremos algunos de los aspectos básicos de la seguridad.

- Transacciones.
- Bloqueos
- Usuarios

1.1. Usuarios

Otra forma de dar seguridad a un SGBD es mediante dos elementos básicos de la seguridad:

- Autenticación
- Autorización.

Autenticación

La autenticación sería la creación de usuarios diferentes, cada uno de ellos con contraseña. La contraseña, al instalarse el SGBD, puede tener restricciones de seguridad. En MySQL hay tres niveles: 0, 1 y 2. El nivel 0 admite cualquier contraseña y el nivel 2 requiere contraseñas largas con mayúsculas, minúsculas, números y caracteres no alfanuméricos. Además se puede indicar desde qué direcciones puede conectarse.

```
create user <nombre>@<direccion> identified by <contraseña>;
```

Para iniciar sesión se han de cumplir las tres partes: usuario, dirección y contraseña.

La dirección admite:

- Una dirección concreta, sea por URL o IP.
- Una dirección de red.
- Cualquier dirección, mediante %.

Un ejemplo de cada una es:

```
create user "user1"@"10.230.98.181" identified by "1234";

create user "user2"@"10.230.254.0" identified by "1234";

create user "user3"@"%" identified by "1234";
```

Como nota importante, se ha de decir que un usuario realmente es la conjunción de usuario-dirección, por lo que las siguientes sentencias crean dos usuarios diferentes:

```
create user "user4"@"10.230.98.171" identified by "1234";

create user "user4"@"10.230.98.172" identified by "1234";
```

Autorización

La autorización es, una vez sabido qué usuario es el que ha accedido correctamente, ver qué acciones puede realizar. Para ello, a cada usuario se le asignará permisos a diferentes niveles:

- Todo el sistema.
- Una base de datos concreta.
- Una tabla concreta.
- Una columna concreta.

Se abordarán los dos primeros.

```
grant <privilegios> on <destino> to <usuario>@<dirección> [opcion];
```

Así, algunos privilegios básicos son: insert, select, delete update. Pero hay más, como alter, create, drop, execute, trigger, event...

Se pueden dar todos los privilegios mediante **all privileges**.

Ejemplos:

```
grant select, insert on *.* to "user1"@"10.230.98.181";

grant select, insert, delete on prueba.* to "user1"@"10.230.98.181";
```

El usuario user1@10.230.98.181 podrá seleccionar e insertar en cualquier tabla de cualquier base de datos. Sin embargo solo podrá borrar en las tablas de la base de datos prueba.

Un usuario así no podrá a su vez dar privilegios a otros usuarios. Para ello, a la hora de darle los privilegios, ha de añadirse **with grant option**.

```
grant select, insert on *.* to "user2"@"10.230.254.0" with grant option;

grant select, insert, delete on prueba.* to "user2"@"10.230.254.0" with grant option;
```

Así, "user2"@"10.230.254.0" podrá a su vez dar los permisos que tiene (seleccionar e insertar en cualquier tabla, y borrar en las de la base de datos prueba) a cualquier otro usuario.

Para eliminar permisos, ha de seguirse la siguiente sintaxis

```
revoke <privilegios> [opcion] on <destino> from <usuario>@<dirección>;
```

Así, si realizamos lo siguiente:

```
revoke select, insert on *.* from "user1"@"10.230.98.181";

revoke grant option on prueba.* from "user2"@"10.230.254.0";
```

El usuario "user1"@"10.230.98.181" ya no podrá seleccionar e insertar en cualquier tabla. Sí seguirá pudiendo insertar y seleccionar en prueba, ya que esos permisos se le dieron a nivel base de datos.

Por otro lado "user2"@"10.230.254.0" podrá realizar las mismas acciones pero ya no podrá otorgar privilegios a otros usuarios.

1.2. Bloqueos

En otras ocasiones, es posible que sea necesario realizar unos cambios críticos en una base de datos, de forma que otros accesos no deban realizarse mientras se llevan a cabo.

Hay que recordar que un SGBD como MySQL, Oracle o cualquier otro de los más utilizados, son multitarea. Es decir, varias conexiones pueden realizarse simultáneamente. Por ello dos hilos, dos conexiones diferentes, pueden intentar realizar cambios sobre los mismos datos: en la misma tabla, la misma fila e incluso la misma columna. El SGBD está preparado para ello, de forma que se realizarán sin que produzcan errores.

Esta gestión de múltiples hilos o conexiones puede ralentizar la ejecución, pese a que asegura que no se produzcan errores.

Si un cambio que se va a realizar es crítico, esto es, que no responde al uso habitual de una base de datos, es posible que se requiera bloquear cualquier otro uso a parte de estos cambios críticos. Para ello están los bloqueos.

Hay dos modos:

```
lock table <tabla> read
```

- Impide que otras conexiones puedan actualizar, borrar o insertar filas en una tabla.

```
lock table <tabla> write
```

- Impide que otras conexiones puedan, además de lo anterior, seleccionar.

Una vez se ha terminado, todos los bloqueos se eliminan mediante:

```
unlock tables;
```

1.3. Transacciones

Una de las posibles pérdidas de información, o que los datos queden de forma incoherente, puede darse por algún error en mitad de las operaciones. Si en mitad de una operación crítica se produce un corte de corriente, de conexión, o un error crítico del sistema, puede alterarse los datos tal y como deberían quedar.

Por ejemplo, si se realiza una transacción importante entre dos cuentas bancarias:

```
update cuenta set saldo=saldo-1000000 where cuenta = 1;

update cuenta set saldo=saldo+1000000 where cuenta = 2;
```

Si ocurre un error crítico en el sistema y no llega a ejecutarse la segunda operación, habrá un problema: la transacción de dinero quedó a medias: se ha perdido 1000000 de €.

Para asegurarse de que una serie de operaciones se van a realizar todas o ninguna, se puede realizar mediante una transacción (de base de datos, no bancaria).

Esto requiere de las siguientes condiciones:

- La base de datos es transaccional: InnoDB o DBD. MyIsam y otras no son compatibles.
- La variable de sistema autocommit está a 0. Si está a 1, el SGBD para dicha conexión realizará un commit automáticamente tras cada sentencia sql. Por ello se ha de ejecutar:

```
set autocommit = 0;
```

Una vez se cumplan estas condiciones, la transacción se realizará de la siguiente forma:

```
start transaction;

update cuenta set saldo=saldo-1000000 where cuenta = 1;

update cuenta set saldo=saldo+1000000 where cuenta = 2;

commit;
```

En caso de que haya algún tipo de error, si no se ha llegado a ejecutar commit, se pueden desechar los últimos cambios:

```
start transaction;

update cuenta set saldo=saldo-1000000 where cuenta = 1;

update cuenta set saldo=saldo+1000000 where cuenta = 2;

rollback;
```

Por último, si la sesión en el servidor SQL se bloquea en mitad de la transacción, el SGBD automáticamente realizará un rollback.

2. Acceso remoto al servidor

Cuando desde un cliente mysql accedemos al propio servidor mysql que está en la misma máquina, por lo general usamos:

```
sudo mysql
```

O bien:

```
mysql -u usuario -p
```

Pero es posible conectarse a otro servidor mysql que esté ubicado en cualquier otro sitio, como ya se ha hecho para conectarse al servidor del ordenador del profesor, o a la nube (Azure).

En este caso se usaba:

```
mysql -u usuario -p -h url_o_direccion_ip
```

Pero para hacer esto, será necesario haber cambiado la configuración que venía por defecto en el servidor mysql. Se deberá acceder y editar un fichero de configuración.

Ubuntu

Ejecutamos los siguientes comandos:

```
cd /etc/mysql/mysql.conf.d/

sudo nano mysqld.cnf
```

Dentro, hay que cambiar la línea siguiente:

bind-address = 127.0.0.1

Por el valor siguiente:

bind-address = 0.0.0.0

Esto permitirá al servidor escuchar peticiones que vengan de cualquier IP, desde todo equipo a través de internet que tenga acceso al servidor.

Después de esto se deberá reiniciar el servidor mysql mediante:

```
sudo systemctl restart mysql.service
```

En Windows

El fichero de configuración está en:

C:\ProgramData\MySql\MySql Server <version>\bin\my.ini

Se abrirá como en Ubuntu. Dado que no hay sudo, desde el interfaz gráfico se abrirá mediante "Abrir como Administrador". De no aparecer se puede añadir dicha línea bajo la sección [mysqld].

Para reiniciar un servicio en Windows, se ha de abrir el Administrador de Tareas, ir a la pestaña "Servicios", buscar mysql y reiniciarlo.