Certificación y controles de seguridad en Bases de Datos: Sistemas de gestión de bases de datos

Miguel Expósito Martín

Universidad de Cantabria

miguel.exposito@unican.es

26/11/2018

Visión general

Definición

Structured Query Language, o SQL, es el lenguaje utilizado tanto para la definición como para la manipulación de datos en los SGBD relacionales. Se trata de uno de los lenguajes de bases de datos más populares y en uso en la industria. Entre sus características se pueden destacar:

- Presente desde 1986, la versión más reciente del estándar ISO es SQL:2016.
- Cada proveedor implementa su dialecto particular, si bien el núcleo del estándar es soportado por todos.
- Es un lenguaje declarativo y orientado a conjuntos.
- Puede utilizarse directamente desde una herramienta o consola o bien a través de otro lenguaje de programación.

Comandos administrativos

- `USE [database name]`: establece la base de datos actual.
- `SHOW DATABASES`: muestra las bases de datos existentes.
- `SHOW TABLES`: muestra todas las tablas no temporales.
- `SHOW COLUMNS FROM [table name]`: proporciona información sobre las columnas de una determinada tabla.
- `SHOW INDEX FROM TABLENAME [table name]`: proporciona información sobre los índices de una determinada tabla.
- `SHOW TABLE STATUS LIKE [table name]\G`: proporciona más información sobre tablas no temporales utilizando el patrón después del LIKE.

DDL: databases y tablas

Creación de bases de datos y tablas

```
CREATE [OR REPLACE] {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name}
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
```

Ejemplos

```
CREATE DATABASE db1;
DROP DATABASE db1;
create table if not exists test (
id bigint auto_increment primary key,
name varchar(128) charset utf8,
key name (name(32))
) engine=InnoDB default charset latin1;
```

DDL: modificación y borrado de objetos

Modificación y borrado de tablas

```
ALTER [ONLINE] [IGNORE] TABLE tbl_name
[WAIT n | NOWAIT]
alter_specification [, alter_specification] ...

DROP [TEMPORARY] TABLE [IF EXISTS] [/*COMMENT TO SAVE*/]
tbl_name [, tbl_name] ...
[WAIT n|NOWAIT]
[RESTRICT | CASCADE]
```

Ejemplos

```
ALTER TABLE t1 ADD x INT;
ALTER TABLE t1 DROP x;
DROP TABLE Employees, Customers;
```

DML: consulta

Consulta de tablas

```
SELECT
[ALL | DISTINCT | DISTINCTROW]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]
[ FROM table_references
[WHERE where_condition]
[GROUP BY {col_name | expr | position} [ASC | DESC], ... ]
[HAVING where_condition]
[ORDER BY {col_name | expr | position} [ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name' [CHARACTER SET charset_name]
```

DML: inserción y actualización

Inserción de datos en tablas

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE
col=expr [, col=expr] ... ]
```

Actualización de datos en tablas

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
[PARTITION (partition_list)]
SET col1={expr1|DEFAULT} [,col2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

DML: borrado

Borrado de datos en tablas

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
FROM tbl_name [PARTITION (partition_list)]
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
[RETURNING select_expr
[, select_expr ...]]
```

Vaciado de tablas

```
TRUNCATE [TABLE] tbl_name
[WAIT n | NOWAIT]
```

DML: uniones

- `INNER JOIN`: el conjunto resultante contiene filas que tienen coincidencia en ambas tablas para la condición especificada.
- `CROSS JOIN`: producto cartesiano (todos con todos).
- `LEFT JOIN`: el conjunto resultante contiene todas las filas de la tabla en el lado izquierdo de la unión. Si no están en el derecho, aparecerá `NULL`.
- `RIGHT JOIN`: el conjunto resultante contiene todas las filas de la tabla en el lado derecho de la unión. Si no están en el izquierdo, aparecerá `NULL`.

Ejemplos

```
SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.b;
SELECT * FROM t1 LEFT JOIN t2 ON t1.a = t2.b;
```

Ejercicio 3.1

Cree una tabla provincia con los siguientes campos:

- id
- cautonoma_id
- literal

Asegúrese de que la combinación de columnas id y cautonoma_id es única. Utilice las referencias proporcionadas para insertar valores en la tabla: CCAA, provincias.

Ejercicio 3.2

En primer lugar, será necesario descargar la base de datos sakila en el directorio homevagrant, por ejemplo, y descomprimirla:

```
wget http://downloads.mysql.com/docs/sakila-db.zip
unzip http://downloads.mysql.com/docs/sakila-db.zip
```

Instalación de sakila

```
mysql -u root -p
mysql> SOURCE /home/vagrant/sakila-db/sakila-schema.sql;
mysql> SOURCE /home/vagrant/sakila-db/sakila-data.sql;
mysql> USE sakila;
mysql> SHOW TABLES;
```

Ejercicio 3.2

Realizar las siguientes consultas:

- Seleccionar la columna last_name de la tabla actor
- Seleccionar las columnas title, description, rating y length de la tabla film que duren 3 horas o más.
- Seleccionar todas las columnas de la tabla customer en las que el cliente está inactivo o cuyo apellido comienza con la letra M.
- Seleccionar la fecha de pago y la cantidad de la tabla payment para los primeros 20 pagos ordenados por cantidad descendentemente.

Vistas

Una vista se define por medio de una consulta SQL y su contenido se genera al ser invocada por una aplicación o por ora consulta. Se pueden considerar como **tablas virtuales** sin registros físicos. O como consultas SELECT guardadas para ser consultadas. Ventajas:

- Permiten ocultar y reutilizar consultas complejas, como uniones.
- Protegen los datos originales ocultando columnas o filas de usuarios no autorizados.
- Pueden utilizarse para proporcionar alias de los nombres de columnas para hacerlas más legibles.

Vistas

Sintaxis de creación de una vista en MariaDB

```
CREATE
[OR REPLACE]
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
[DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
[SQL SECURITY { DEFINER | INVOKER }]
VIEW [IF NOT EXISTS] view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

Ejercicio 3.3

Liste el "top five"de géneros en ingresos brutos en orden descendente. genres in gross revenue in descending order. (Pista: puede necesitar las tablas *category*, *film_category*, *inventory*, *payment y rental*).

En su nuevo puesto de ejecutivo, le gustaría disponer de una forma sencilla de consultar este "top five". Utilice la consulta anterior para crear una vista que satisfaga esta necesidad.

Privilegios SQL: definición y tipos

Un privilegio se corresponde con un derecho a hacer uso de ciertas sentencias SQL como SELECT o INSERT sobre uno o más objetos de base de datos. Los privilegios pueden concederse o revocarse. Tanto el DBA como el propietario de un esquema dado pueden conceder o revocar privilegios a cuentas de usuario, a nivel de cuenta, tabla, vista o columna.

- Privilegios administrativos: permiten a los usuarios gestionar la operación del SGBD. Son globales y no específicos de una base de datos en particular. Ej: CREATE USER, SHOW DATABASES.
- Privilegios a nivel de BD: se aplican a una base de datos y a todos los objetos dentro de la misma. Pueden aplicarse globalmente o para BDs específicas.
- Privilegios a nivel de objetos de BD: tablas, índices, vistas. Pueden aplicarse globalmente o a una BD específica.

Privilegios SQL: Cuentas de usuario

Las cuentas de usuario en MariaDB consisten en un usuario y un nombre de host. Esto permite crear cuentas de usuario con el mismo nombre que puedan conectarse desde diferentes hosts. Las reglas a utilizar son las siguientes:

- La sintaxis es: 'username'@'hostname'
- Una cuenta para un único usuario puede definirse así: 'username'@'%'.
- No es necesario entrecomillar el usuario y el host si no contienen caracteres especiales (como guiones).
- Para entrecomillar, pueden usarse backticks, comillas simples o comillas dobles.
- Si se entrecomilla usuario y host, deben hacerse de forma separada e independiente.

Privilegios SQL más comunes

Privilegio	**Columna**	**Contexto**
CREATE	Create_priv	databases, tables, or indexes
DROP	Drop_priv	databases, tables, or views
REFERENCES	References_priv	databases or tables
ALTER	Alter_priv	tables
DELETE	Delete_priv	tables
INSERT	Insert_priv	tables or columns
SELECT	Select_priv	tables or columns
UPDATE	Update_priv	tables or columns
CREATE VIEW	Create_view_priv	views
SHOW VIEW	Show_view_priv	views
CREATE USER	Create_user_priv	server administration
SHOW DATABASES	Show_db_priv	server administration
ALL	[PRIVILEGES]	server administration

Cuadro: Privilegios más comunes

Privilegios SQL: GRANT y REVOKE

Los privilegios se conceden con GRANT...

Sintaxis de GRANT en MariaDB

```
GRANT
priv_type [(column_list)]
[, priv_type [(column_list)]] ...
ON [object_type] priv_level
TO user [ authentication_option ] [ user_options... ]
authentication_option:
IDENTIFIED BY 'password'
```

Privilegios SQL: GRANT y REVOKE

...y se revocan con REVOKE.

Sintaxis de REVOKE en MariaDB

```
REVOKE
priv_type [(column_list)]
[, priv_type [(column_list)]] ...
ON [object_type] priv_level
FROM user [, user] ...

REVOKE ALL PRIVILEGES, GRANT OPTION
FROM user [, user] ...
```

Privilegios SQL: ejemplo

Lo habitual:

Ejemplo de concesión de privilegios

```
GRANT ALL PRIVILEGES ON *.* TO 'user'@%
identified by 'strongpassword';
FLUSH privileges;
```

`FLUSH PRIVILEGES`: recarga las tablas de concesión de privilegios. Se ejecuta después de modificarlos.

Privilegios SQL: tablas de sistema relevantes

La información sobre privilegios se almacena en tablas de sistema:

- *user:* cuentas de usuario, privilegios globales y otras columnas.
- *db:* privilegios a nivel de base de datos.
- *host:* en las últimas versiones de MariaDB ya no existe esta tabla.
- *tables_priv:* privilegios a nivel de tabla.
- *columns_priv:* privilegios a nivel de columna.
- *procs_priv:* privilegios a nivel de funciones y procedimientos almacenados.
- *proxies_priv:* privilegios a nivel de proxy.

Privilegios SQL: roles

Los roles agrupan una serie de privilegios. Son útiles en grandes organizaciones, donde muchos usuarios tienen los mismos privilegios. Antes de los roles, la única manera de cambiar los privilegios de un grupo de usuarios era hacerlo individualmente.

Los roles se crean con la sentencia `CREATE ROLE` y se eliminan con `DROP ROLE`. Se asignan y revocan con `GRANT` y `REVOKE`, respectivamente. Para que el rol se active tras la autenticación, será necesario asignárselo al usuario como rol por defecto.

Ejemplo de asignación de rol

```
CREATE ROLE journalist;
GRANT SHOW DATABASES ON *.* TO journalist;
GRANT journalist to mexposito;
SET default role journalist FOR mexposito;
```

Se pueden conceder roles a roles.

Ejercicio 3.4: privilegios SQL

1. Usando la BD de ejemplo sakila, otorgar, con el usuario root, control total (DELETE, INSERT, SELECT, UPDATE) a un usuario 'owner' a la tabla 'actor'. Indicar el comando utilizado.
2. Salir de la sesión de root y entrar como el usuaro owner. Comprobar las tablas a las que tiene acceso. Lanzar una consulta a esas tablas y mostrar el resultado con una captura de pantalla.
3. Cerrar sesión con el usuario owner y volver a entrar como root. Crear un usuario 'manager' con los siguientes privilegios sobre la tabla 'actor': DELETE, INSERT, SELECT, UPDATE. Indicar el comando utilizado.
4. Revocar los privilegios DELETE, INSERT y UPDATE para el usuario 'manager' en la tabla 'actor'. Indicar el comando utilizado.
5. Cerrar la sesión de root e iniciar una nueva con el usuario manager. Ejecutar una consulta sobre la tabla actor y mostrar el resultado.
6. Borrar las filas de la tabla actor cuyo campo actor_id sea igual a 1 y mostrar el resultado.

Ejercicio 3.4: roles

7. Crear un rol llamado 'r_sakila_read' y comprobar que se ha creado consultando la tabla user.
8. Otorgar el privilegio de lectura al rol creado en todas las tablas de la base de datos sakila.
9. Crear un usuario 'sakila_reader' en localhost con una contraseña no vacía.
10. Otorgar el rol 'r_sakila_read' al usuario 'sakila_reader' en localhost y hacer efectivos los privilegios. Asignárselo como rol por defecto.
11. Iniciar sesión con el usuario 'sakila_reader', conectarse a la base de datos sakila y lanzar una consulta a una de las tablas. Intentar borrar una de las tablas.

Incluir todos los comandos SQL utilizados y capturas de sus resultados.

Ejercicio 3.4: seguridad a nivel de columna y de fila

12. Conectarse como root a la BD sakila y añadir una columna 'salary' de tipo 'decimal (10,2)' a la tabla 'staff'. Añadir también una columna 'ss_number' de tipo 'varchar(9)' a la misma tabla.
13. Revocar cualquier permiso del usuario 'manager' en la tabla 'staff'.
14. Otorgar privilegios de consulta al usuario 'manager' en la tabla 'staff' para las siguientes columnas: 'staff_id', 'first_name', 'last_name', 'address_id', 'picture', 'email', 'store_id', 'active', 'username' y 'last_update'.
15. Conectarse con el usuario 'manager' a la BD 'sakila' y mostrar sus columnas con la sentencia 'EXPLAIN'. Lanzar una consulta a las columnas 'password', 'ss_number' y 'salary'.

Incluir todos los comandos SQL utilizados y capturas de sus resultados.

MariaDB no dispone de una funcionalidad específica para aplicar restricciones de seguridad a filas. ¿Cómo implementaría esta medida de seguridad con los mecanismos ya existentes vistos hasta la fecha?

Resumen

Referencias

Abraham Silberschatz et al. (2010)

Database System Concepts, 6th edition

Wilfried Lemahieu et al. (2018)

Principles of Database Management