CAPÍTULO 13°: ADMINISTRACIÓN DE BASES DE DATOS RELACIONALES

1 **Conceptos de seguridad en SQL** .-

La implementación de un sistema de seguridad y el reforzamiento de las restricciones de seguridad se realizan mediante el software del SGBD. El SQL define un panorama general para la seguridad de la base de datos y las sentencias SQL se utilizan para especificar restricciones de seguridad. El esquema de seguridad de SQL se basa en tres conceptos principales:

- Los *Usuarios* como parte activa de la Base de datos. Cada vez que el SGBD recupera, inserta, suprime o actualiza datos, lo hace a cuenta de algún usuario. El SGBD permite o prohibe la acción dependiendo de qué usuario esté efectuando la operación.

- Los *Objetos de la base de datos* o elementos a los cuales se puede aplicar la protección de seguridad SQL. Esta seguridad se aplica generalmente a tablas y a vistas, aunque otros objetos como formularios, aplicaciones o bases de datos enteras también pueden ser protegidos. Los usuarios tendrán permitido el acceso a ciertos objetos y prohibido el acceso a otros.

- Los *Privilegios* o las acciones que el usuario tiene permitido realizar sobre un determinado objeto de la base de datos

1.1 ***Identificadores de Usuario*** .-

Cada usuario de una base de datos tiene asignado un nombre breve (id-usuario) que le identifica dentro del software del SGBD. Toda sentencia SQL ejecutada por el SGBD se realiza a cuenta de un identificador de usuario específico, de forma que dicho identificador determina si la sentencia va a ser permitida o prohibida por el SGBD.

Los identificadores de usuario son asignados por el administrador. El estándar SQL permite identificadores de hasta 18 caracteres, pero algunos sistemas gestores, DB2 y SQL/DS sólo admiten 8 caracteres mientras que Sybase o SQL Server permiten hasta 30 caracteres. No obstante es conveniente limitar los identificadores de usuario a 8 caracteres.

1.2 ***Grupos de Usuarios*** .-

Son conjuntos de usuarios con necesidades similares de información para acceder a los datos y que deben tener, por tanto, privilegios idénticos.

Bajo el esquema de seguridad SQL de ANSI/ISO pueden manejarse loas grupos de usuarios con necesidades similares de dos formas distintas:

1.- Asignando el mismo identificador de usuario a todas las personas del grupo. Este método simplifica la administración de la seguridad pero, sin embargo, no puede distinguirse a las personas que comparten el identificador ni en las visualizaciones ni en los informes que genera el sistema.

2.- Asignando un identificador de usuario diferente a cada persona del grupo. Permite identificar a los usuarios en los diferentes informes para los usuarios individuales, pero deben establecerse individualmente los privilegios para cada usuario haciendo la administración de la seguridad complicada y propensa a errores.

2 **Seguridad en Objetos** .-

Las protecciones de seguridad SQL se aplican a los objetos específicos contenidos en la Base de Datos, en particular a las tablas y las vistas. Cada tabla y cada vista puede ser individualmente protegida permitiendo el acceso a ella para ciertos identificativos de usuario y prohibiendo el acceso a otros identificativos. SQL2 amplía esta seguridad a los dominios y los conjuntos de caracteres definidos por el usuario.

La mayoría de los productos comerciales soportan objetos de seguridad adicionales tanto en la propia base de datos como en el espacio físico donde ésta se almacena.

3 **Seguridad en Privilegios** .-

El conjunto de acciones que un usuario puede efectuar sobre un objeto de una base de datos se denomina *privilegios* para el objeto. SQL especifica cuatro privilegios para tablas o vistas:

- El privilegio SELECT permite recuperar datos de una vista o una tabla. Mediante dicho privilegio puede especificarse la tabla o vista en la cláusula FROM de una sentencia SELECT o subconsulta.

- El privilegio INSERT permite insertar nuevas filas en una tabla o vista. Con este privilegio se puede especificar la tabla o vista en la cláusula INTO de la sentencia.

- El privilegio DELETE permite eliminar filas de datos en una tabla o vista. Con este privilegio se puede especificar la tabla o vista en la cláusula FROM de la sentencia.

- El privilegio UPDATE permite modificar filas de datos en una tabla o vista. Este privilegio puede restringirse a columnas específicas de la tabla o vista, permitiendo actualizaciones de dichas columnas pero desautorizando actualizaciones a cualesquiera otras columnas.

3.1 ***Privilegios sobre tablas y vistas*** .-

Cuando se crea una tabla mediante CREATE TABLE quien lo hace se convierte automáticamente en propietario de la tabla y, consecuentemente, adquiere para dicha tabla

privilegios totales (SELECT, INSERT, DELETE, UPDATE y cualesquiera otros privilegios soportados por el SGBD).

Otros usuarios inicialmente no tienen privilegios sobre la tabla recién creada, por lo que es preciso concederles explícitamente privilegios utilizando la sentencia GRANT.

Con respecto a las vistas, el usuario que, mediante una sentencia CREATE VIEW crea una vista, se convierte en propietario de ella. Pero para poder crear la vista con éxito debe tener previamente el privilegio SELECT en cada una de las tablas fuente de la vista. Además el SGBD proporciona al creador el privilegio sobre la vista solamente si dispone de ese mismo privilegio sobre todas las tablas fuente de la vista.

3.1.1 ***Concesión de Privilegios*** (GRANT) .-

La sentencia GRANT se utiliza, por el propietario de un objeto de la Base de Datos, para proporcionar a otros usuarios el acceso a los datos. La sentencia GRANT incluye una lista específica de los privilegios a conceder, el nombre de la tabla sobre la que se conceden dichos privilegios y el identificador del usuario al que se le conceden los privilegios.

La sintaxis de la sentencia es la siguiente:

GRANT SELECT, INSERT, DELETE, UPDATE

ON Tabla

TO Id-Usuario

La sentencia GRANT permite delimitar qué privilegios se conceden al usuario. En el caso de concederle todos los privilegios la sentencia puede simplificarse:

GRANT ALL PRIVILEGES

ON Tabla

TO Id-Usuario

La sentencia GRANT también permite conceder privilegios a todos los usuarios de la Base de Datos:

GRANT SELECT

ON Tabla TO PUBLIC

Como se ve la sentencia concede el privilegio SELECT a todos los usuarios, presentes y futuros, no solamente los conocidos actualmente por el SGBD esto elimina la necesidad de conceder explícitamente privilegios a los nuevos usuarios conforme son autorizados.

3.1.2 ***Privilegios de columna*** .-

El estándar SQL1 permite conceder el privilegio UPDATE para columnas individuales de una tabla o vista y el estándar SQL2 permite una lista de columnas para los privilegios UPDATE, INSERT y REFERENCES (como utilización de clave ajena). Las columnas actualizables se listan tras las palabras clave UPDATE, INSERT y REFERENCES y se encierran entre paréntesis. La sintaxis es:

GRANT UPDATE (Columna1, Columna2)

ON Tabla

TO Id-Usuario

Pudiéndose hacer la autorización pública:

GRANT UPDATE (Columna1, Columna2)

ON Tabla

TO PUBLIC

3.2 ***Paso de Privilegios*** (GRANT OPTION) .-

Cuando se crea un objeto en una Base de Datos el creador se convierte en el propietario del objeto y, consecuentemente, es la única persona que puede conceder privilegios para utilizar dicho objeto.

Cuando se conceden privilegios a otros usuarios, éstos tienen permitido la utilización del objeto pero no pueden transmitir esos privilegios a otros usuarios. Es exclusivamente el propietario del objeto quien mantiene el control de los permisos para su utilización.

Ocasionalmente puede desearse permitir que otros usuarios concedan privilegios sobre objetos que no son de su propiedad, para ello se utiliza la sentencia GRANT OPTION, cuya sintaxis es:

GRANT SELECT, INSERT, DELETE, UPDATE

ON Tabla

TO Id-Usuario

WITH GRANT OPTION

O bien:

GRANT ALL PRIVILEGES

ON Tabla

TO Id-Usuario

WITH GRANT OPTION

3.3 ***Revocación de privilegios*** (REVOKE)

Los privilegios concedidos mediante una sentencia GRANT pueden retirarse con la sentencia REVOKE cuya sintaxis se asemeja bastante a la de la sentencia GRANT:

REVOKE SELECT, INSERT, DELETE, UPDATE

ON Tabla

FROM Id-Usuario

Debe considerarse que la sentencia REVOKE únicamente retira los privilegios concedidos previamente por el usuario que la ejecuta a otro usuario. Éste último puede tener además otros privilegios concedidos por otros usuarios que no se ven afectados por la sentencia REVOKE.

Un caso particular se produce cuando se retiran privilegios a un usuario al que se le había concedido la capacidad de "paso de privilegios". Al revocar dicha concesión, se revocarán automáticamente en cascada todos los privilegios derivados de dicha concesión. Por ello, la concesión y revocación de privilegios con la opción de paso de privilegios debe ser manejada muy cuidadosamente, para asegurar que los resultados obtenidos sean los que se pretenden

4 **Transacciones** .-

Una transacción es una secuencia de una o más sentencias SQL que juntas forman una unidad de trabajo. Dichas sentencias suelen estar estrechamente relacionadas y efectuar acciones independientes. Cada sentencia específica realiza una parte de una tarea pero todas ellas son necesarias para completar la tarea.

La agrupación de sentencias en una sola transacción indica al SGBD que la secuencia debe ser ejecutada atómicamente, esto es todas y cada una de las sentencias deben completarse para que la base de datos se encuentre en cualquier momento en estado consistente.

El concepto de transacción es crítico para los programas que actualizan una base de datos ya que garantizan la integridad de dicha base. Por tanto, las sentencias de una transacción se ejecutarán como una unidad atómica de la Base de Datos. O todas las sentencias son ejecutadas con éxito o ninguna de las sentencias es ejecutada.

Este concepto de transacción lo mantiene el SGBD incluso si el programa de la aplicación aborta o se produce un fallo de hardware en mitad de la transacción.

4.1 ***Inicio, fin y aborto de una transacción*** .-

SQL soporta transacciones mediante dos sentencias de procesamiento:

- La sentencia COMMIT señala el final correcto de una transacción e informa de ello al SGBD que la transacción está completa, que todas las sentencias han sido ejecutadas y que la base de datos se mantiene autoconsistente.

- La sentencia ROLLBACK determina el final sin éxito de una transacción. Informa al SGBD que debe deshacer los cambios efectuados en la base de datos por la transacción y que debe restaurar la base de datos al estado en que se encontraba cuando la transacción comenzó.

El estándar SQL de ANSI/ISO define su modelo de transacción SQL considerando que una transacción empieza automáticamente con la primera sentencia SQL ejecutada por un usuario o un programa. La transacción continúa con las sentencias SQL subsiguientes hasta que finaliza mediante una de las cuatro formas siguientes:

- Una sentencia COMMIT que indica que la transacción ha terminado con éxito. Los cambios realizados en la base de datos se convierten en permanentes y automáticamente se inicia una nueva transacción inmediatamente después de la sentencia COMMIT.

- Una sentencia ROLLBACK que aborta la transacción. La base de datos se restaura a su situación inicial deshaciendo los cambios efectuados en la base de datos y automáticamente se inicia una nueva transacción inmediatamente después de la sentencia ROLLBACK.

- La terminación de un programa con éxito (para SQL programado) finaliza la transacción igual que si se hubiera ejecutado una sentencia COMMIT. Como el programa ha finalizado no hay una nueva transacción que comenzar.

- La terminación anormal de un programa (para SQL programado) aborta la transacción igual que si se hubiera ejecutado una sentencia ROLLBACK. Como el programa ha finalizado no hay una nueva transacción que comenzar.

Algunos productos comerciales se apartan del estándar de ANSI/ISO para el modelo de transacción, así el SGBD Sybase, diseñado para aplicaciones de proceso de transacciones en línea, al igual que SQL Server (derivación de Sybase)incluyen en su dialecto Transact-SQL cuatro sentencias para el procesamiento de transacciones:

- La sentencia BEGIN TRANSACTION que inicia el comienzo de una transacción.

- La sentencia COMMIT TRANSACTION que señala el final con éxito de la transacción.

- La sentencia SAVE TRANSACTION que establece un "punto de guarda" en mitad de una transacción. El SGBD guarda el astado de la base de datos en el punto actual de la transacción y asigna al estado guardado un "nombre de punto de guarda" especificado en la sentencia.

- La sentencia ROLLBACK TRANSACTION deshace los cambios efectuados en la base de datos desde el inicio de la transacción desde el inicio de la transacción o, en el caso de que se hubiera realizado un punto de guarda, deshace los cambios realizados desde el punto de guarda.

5 **Seguridad en las transacciones** .-

Por definición, una base de datos se encuentra en un estado consistente antes de que se empiece a ejecutar una transacción y también deberá estarlo cuando la transacción termine.

Por ello, las principales propiedades que debe poseer toda transacción son las siguientes:

1 *Atomicidad* (Atomicity), en el sentido de que o bien se ejecutan todas las sentencias de una transacción o no se ejecuta ninguna.

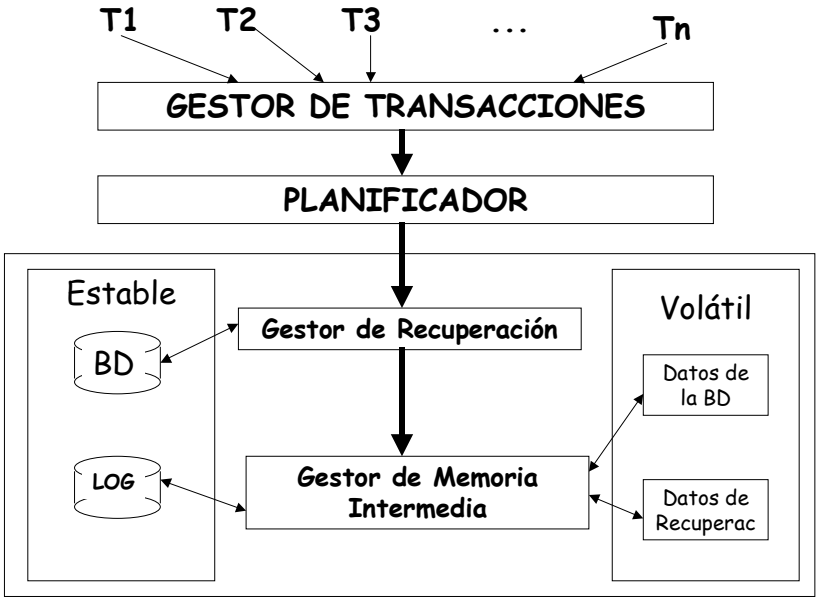
2 *Consistencia* (Consistency), la transacción debe dejar la base de datos en estado consistente.

3 *Aislamiento* (Isolability) ya que una transacción no muestra los cambios que produce hasta que finaliza.

4 *Persistencia* (Durability), una vez finalizada con éxito una transacción sus efectos perduran en la basa de datos.

5.1 ***Arquitectura de un SGBD en la gestión de las transacciones y su recuperación*** .-

Los principales componentes de un SGBD que se encargan de la gestión y la recuperación de transacciones son:



- *Gestor de transacciones*.- Que coordina las transacciones para los programas de aplicación.

- *Planificador*.- Que es el responsable de llevas a cabo el control de concurrencia.

- *Gestor de recuperación*.- Que se encarga de asegurar que la base de datos quede consistente después de algún fallo.

- *Gestor de memoria intermedia* (Caché).- Que se ocupa de la transferencia de los datos de memoria volátil (tanto de los datos de la base de datos como de los relativos a la recuperación) a disco y viceversa.

5.2 ***El fichero diario*** (LOG) .-

Para conseguir anular y recuperar transacciones el método más extendido suele ser la utilización de un fichero diario (log) en el que se va guardando toda la información necesaria para *deshacer*, en caso de fracasar, o *rehacer*, si hay que recuperar las transacciones. Un registro del fichero diario suele constar de:

- Identificador de la transacción

- Hora de la modificación
- Identificador del registro afectado
- Tipo de acción
- Valor anterior del registro
- Nuevo valor del registro
- Información adicional (por ejemplo, un puntero al registro previo del diario que concierne a la misma transacción).

Ante la posibilidad de que se realice un cambio en la Base de Datos que no quede registrado en el fichero diario por algún fallo en el equipo, normalmente se obliga a que los registros que se modifican y que se encuentran en áreas de memoria o memoria principal se escriban antes en el fichero diario que en la Base de Datos, para poder anular así, en caso de necesidad, las transacciones.

El fichero diario puede ser un fichero *circular*, esto es, un fichero tal que, una vez lleno va eliminando registros según van entrando otros nuevos, aunque lo normal es que conste de dos partes: la primera *en línea* (en disco), que almacena las actualizaciones que se llevan a cabo hasta que se llena, momento en el que pasa el contenido a la segunda parte (por ejemplo, en cinta).

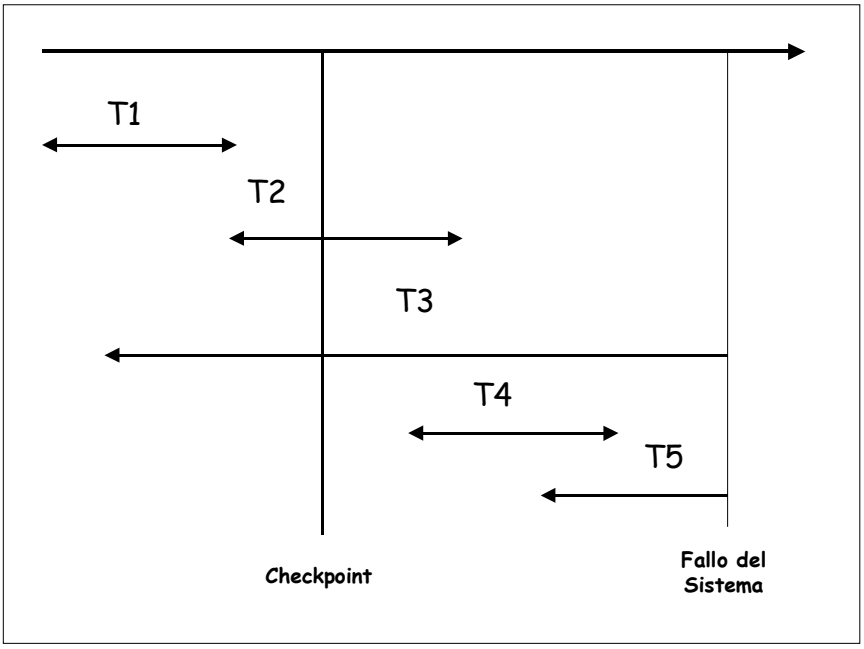
Para evitar tener que recorrer todo el fichero diario, cosa que consumiría mucho tiempo, se introduce el concepto de *punto de verificación* o *punto de recuperación* (*checkpoint*) que se ejecuta periódicamente e implica:

- Pasar el contenido de las memorias de área intermedia al fichero diario (al igual que para la base de datos, para el fichero diario existen unas áreas intermedias que guardan los registros de este fichero).
- Escribir un registro de punto de recuperación en el fichero diario.
- Pasar el contenido de las memorias de área intermedia de la base de datos a soporte secundario.
- Escribir la dirección del registro de recuperación en un fichero de rearranque.

En el registro del punto de recuperación se reflejan todas las transacciones que se encuentran activas en el momento de producirse el punto de recuperación.

5.3 ***Recuperación en caliente*** .-

Al ocurrir un fallo que dé lugar a pérdida de memoria volátil es preciso realizar la operación que suele denominarse *recuperación en caliente*, mediante la cual el sistema consulta el fichero diario para determinar las transacciones que hay que deshacer porque no han sido completadas y las que hay que rehacer porque, si bien se han completado, no han sido grabadas en la base de datos cuando se produjo el fallo.



Cuando se produce el fallo después de la caída del sistema se obtiene la dirección del registro de recuperación mas reciente y se recorre el fichero desde ese punto hasta el final.

Para una situación estándar como la representada en la figura, la transacción T1 no se verá afectada por el proceso de recuperación (ya que estaba completada antes del checkpoint), las transacciones T2 y T4 deberán rehacerse ya que, aunque han finalizado no fueron grabadas en la base de datos y las transacciones T3 y T5 deberán deshacerse puesto que no han concluido.

5.4 ***Recuperación en frío*** .-

En el caso de un fallo en la memoria secundaria que afecte a la base de datos, se lleva a cabo una *recuperación en frío* que consiste en utilizar una copia de seguridad de la base de datos, llamada *copia de respaldo (Backup)* que permitirá, junto con los ficheros diarios que se han construido desde que se realizó la copia de seguridad, reconstruir la base de datos, llevándola de forma consistente a la situación anterior a la que se produjera el fallo.