ILERNA Online

MP02-A: BASES DE DATOS

UNIDAD FORMATIVA 2

LENGUAJES SQL:

DML y DDL

Estrategias para el control de las transacciones y de la concurrencia

Concepto de integridad

La **integridad** es una pieza fundamental de las bases de datos y se encarga de que los datos que la componen sean **lo más correctos posibles**. Estos datos almacenados en la base de datos deben cumplir una serie de restricciones con el objetivo de facilitar el trabajo del usuario en cuanto a la manipulación de datos de las BDD.

Aunque es una pieza fundamental puede ocasionar una serie de problemas.

Si, por ejemplo, **borramos un registro** de nuestra tabla principal que, a su vez está relacionado con algunos registros de otra tabla secundaria, va a provocar un error al detectar un fallo de integridad.

Estrategias para el control de las transacciones y de la concurrencia

Concepto de integridad

Existen claves secundarias que se refieren a una cave principal que ya no está. por este motivo se pueden aplicar una serie de soluciones que se añaden detrás de la cláusula “**REFERENCES”** como:

- **ON DELETE SET NULL**: asigna valores nulos a aquellas claves secundarias que estén relacionadas con la que se ha borrado.

- **ON DELETE CASCADE**: elimina aquellos registros que tienen su clave secundaria idéntica a la del registro que se ha eliminado.

- **ON DELETE SET DEFAULT**: sitúa en el registro relacionado un valor asignado por defecto en la columna relacionada.

- **ON DELETE NOTHING**: no hace ningún cambio.

Estrategias para el control de las transacciones y de la concurrencia

Concepto de integridad

Podemos sustituir la palabra “DELETE” por “UPDATE”. Así, el funcionamiento, se va a mencionar cada vez que se produzca algún cambio en la tabla principal.

Existen, además, reglas de integridad que debemos controlar si nos encontramos con violaciones de la integridad.

Estas reglas se dividen en dos principales:

- Reglas de integridad de dominios: si se le asigna un valor a un atributo sin saber la relación que éste tiene con los demás que forman la BDD.

- Reglas de integridad de relaciones: cuando se admite una tupla dada para ser insertada o bien cuando se van a relacionar varias tuplas.

Estrategias para el control de las transacciones y de la concurrencia

Concepto de transacción. Control

Cuando hablamos de una **transacción** nos referimos a un conjunto de diferentes acciones capaces de realizar transformaciones sobre los estados de un sistema conservando su integridad.

Una transacción puede ser cualquier tipo **de operación atómica** que se realice con éxito.

Estas acciones que se van a realizar son independientes las unas de las otras pero relacionadas. De esta manera, inicialmente se comienza **abriendo la transición** y, seguidamente, si todas estas acciones se ejecutan de forma correcta, se confirma y **se cierra la transacción**.

Sin embargo, **si se observa cualquier tipo de error en ellas la transacción se deshace.**

De esta forma se tiene siempre en cuenta **la integridad de los datos.**

Un buen ejemplo sería una transferencia de dinero de una cuenta a otra: débito y crédito.

Cuando ejecutamos varias lineas de código de una vez se quedan realizadas hasta la linea que dé error. Con las transacciones o se ejecuta todo o nada.

Estrategias para el control de las transacciones y de la concurrencia

Propiedades de las transacciones: atomicidad, consistencia, aislamiento y permanencia

Atomicidad (*Atomicity*): actúa como un proceso atómico, es decir, o todo (modificación, agregación o borrado) se realiza con éxito, o nada. Basta con que falle una mínima parte para que la operación no sea satisfactoria.

- **Consistencia** (*Consistency*): cuando se ejecuta la transacción, el sistema debe pasar de en un estado consistente a otro que también lo sea pese a los cambios que se han realizado.

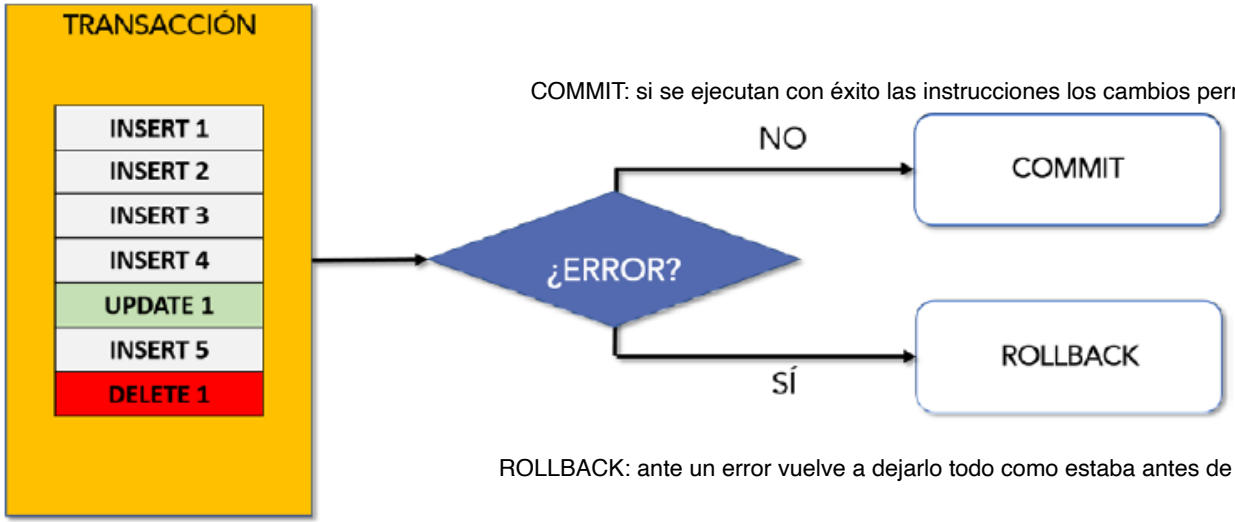
Cuando empieza esta todo correcto y cuando termina también debe estarlo.

- **Aislamiento** (*Isolation*): cada transacción debe actuar de forma secuencial.

- **Permanencia** (*Durability*): todos los cambios que se hayan producido cuando se realiza una transacción no se pierden, sino que permanecen

Estrategias para el control de las transacciones y de la concurrencia

Estados de una transacción: activa, parcialmente comprometida, fallida, abortada y comprometida



COMMIT: si se ejecutan con éxito las instrucciones los cambios permanecen.

ROLLBACK: ante un error vuelve a dejarlo todo como estaba antes de comenzar las instrucciones.

Estrategias para el control de las transacciones y de la concurrencia

Problemas derivados de la ejecución concurrente de transacciones

Anteriormente hemos visto el conjunto de normas que deben cumplir las transacciones.
Además, también es importante que nos detengamos un poco ante los problemas más frecuentes que pueden ocasionar.

Uno de los principales es el que se ocasiona cuando dos transacciones quieren acceder al mismo dato de manera simultánea, le llamamos problema de concurrencia. Cuando nos encontramos ante un caso así podemos diferenciar entre **tres tipos diferentes de problemas**:

- ***Dirty read*** (lectura sucia): cuando una transacción consulta datos escritos de otra que aún no ha sido confirmada.

- ***Nonrepeatable read*** (lectura irrepetible): cuando una transacción vuelve a hacer una lectura de unos datos que ya había leído y comprueba entonces que han sido modificados en alguna transacción.

- ***Phantom read*** (lectura fantasma): cuando una transacción realiza una consulta y encuentra datos que antes eran inexistentes. Alguna transición los ha insertado.

Estrategias para el control de las transacciones y de la concurrencia

Control de concurrencia: técnicas optimistas y pesimistas

Uno de los principales problemas que se ocasiona en las transacciones de los datos de una BDD es que se pida acceso a un mismo dato des de dos lugares distintos. Es en ese momento en el que se precisa un **control de concurrencia** para darle solución.

El encargado de este control de concurrencia es el **planificador**, este va a realizar diferentes esquemas para que las transacciones no se solapen entre ellas.

Siempre es conveniente que el planificador no realice ningún cambio en el sistema, tanto si las transacciones se ejecutan de forma concurrente como si lo hacen una detrás de otra.

Estrategias para el control de las transacciones y de la concurrencia

Control de concurrencia: técnicas optimistas y pesimistas

Veamos qué tipo de técnicas podemos utilizar para evitar este tipo de problemas:

• **Técnicas pesimistas**:

Técnicas de bloqueo (*locks*)

Su tarea principal es poder bloquear aquellos datos para que no se acceda a ellos desde diferentes transacciones (sincronizar el acceso). Hasta que no haya finalizado del todo la transacción.

Técnicas de marcas de tiempo (*time-stamping*)

Las marcas de tiempo se utilizan para que sólo exista un único identificador para cada transacción. Estas marcas deben ir en orden para poder el acceso a los diferentes datos sin que estos se solapen.

Estrategias para el control de las transacciones y de la concurrencia

Control de concurrencia: técnicas optimistas y pesimistas

- **Técnicas optimistas:**

También conocidas como técnicas de validación o de certificación.

Estas técnicas no llevan impuestas ninguna restricción específica ni ningún bloqueo.

Aunque, al final, hacen una comprobación de tres fases diferentes que se pueden dar: lectura, validación y escritura.

Son bastante adecuadas cuando existen pocas transacciones, así hay menos operaciones.

Lenguajes de las BBDD para la creación de su estructura

Vistas y otras extensiones del lenguaje

Cuando hablamos de vistas nos estamos refiriendo a almacenar el resultado de una consulta sobre una o varias tablas en una estructura. Esta vista se podrá tratar, en un futuro, como si fuera una tabla. Podemos diferenciar entre dos tipos de vistas:

• **Simples**: cuando está formada por una única tabla y no tiene ninguna función de agrupación. Permite operaciones DML.

• **Complejas**: cuando está formada por más de una tabla y sí utiliza funciones de agrupación. No permite operaciones DML

Lenguajes de las BBDD para la creación de su estructura

Vistas y otras extensiones del lenguaje

- **Creación**

Sintaxis para crear una tabla

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW nombre_vista [lista_columnas]
AS sentencia_select
[WITH CHECK OPTION [CONSTRAINT restricción]]
[WITH READ ONLY [CONSTRAINT restricción]]
```

- **OR REPLACE:** lo utilizamos si la vista ya existe, la cambia por la actual.
- **FORCE**: aunque no se disponga de los datos necesarios para realizar la consulta, crea la vista.
- **Lista_columnas**: listado de las columnas que devuelve la consulta.
- **WITH CHECK OPTION**: ofrece la posibilidad de añadir ("INSERT") o modificar ("UPDATE") las filas a visualizar.
- **WITH READ ONLY**: vista de solo lectura con posibilidad de asignarle un nombre.

Lenguajes de las BBDD para la creación de su estructura

Vistas y otras extensiones del lenguaje

Crear una vista con el nombre y apellidos de los empleados que trabajen en el departamento de ropa de unos grandes almacenes:

CREATE OR REPLACE VIEW vista_ejemplo_ropa
AS SELECT nombre, apellidos
FROM empleados
WHERE depot = ropa;

Cuando definimos una vista es habitual que usemos la sintaxis “CREATE OR REPLACE”. Por lo tanto se puede utilizar para crear o modificar una vista ya definida previamente.

Lenguajes de las BBDD para la creación de su estructura

Vistas y otras extensiones del lenguaje

13.1.32 DROP VIEW Statement

```
DROP VIEW [IF EXISTS]
    view_name [, view_name] ...
    [RESTRICT | CASCADE]
```

`DROP VIEW` removes one or more views. You must have the `DROP` privilege for each view.

Borrado

Sintaxis para borrar una tabla:

DROP VIEW *nombreDeLaVista*;

Lenguajes de las BBDD para la creación de su estructura