TEMA 10: PUESTA AL DÍA DE LOS DATOS.

2ª Parte.- TRANSACCIONES

OBJETIVOS:

- Utilizar asistentes y herramientas gráficas
- Identificar las herramientas y sentencias para modificar el contenido de la base de datos.
- Insertar, borrar y actualizar datos en las tablas.
- Incluir en una tabla la información resultante de la ejecución de una consulta.
- Adoptar medidas para mantener la integridad y consistencia de la información.
- Diseñar guiones de sentencias para llevar a cabo tareas complejas.
- Reconocer el funcionamiento de las transacciones.
- Anular parcial o totalmente los cambios producidos por una transacción.
  Identificar los efectos de las distintas políticas de bloqueo de registros.
- Utilizar el lenguaje de definición de datos.

TRANSACCIONES

Un SGBD actualiza múltiples datos a través de una transacción. Una transacción es un conjunto de sentencias SQL que se tratan como una sola instrucción (atómica). Una transacción puede ser confirmada (**commit**), si todas las operaciones individuales se ejecutaron correctamente, o, abortada (**rollback**) a la mitad de su ejecución si hubo algún problema (por ejemplo, el producto pedido no está en stock, por tanto no se puede generar el envío). Trabajar con transacciones puede ser esencial para mantener la integridad de los datos. Por ejemplo, se puede dar el *caso* de que se descuenta el stock de un producto antes de proceder a su envío, pero cuando se va a generar la cabecera del pedido, la aplicación cliente sufre un corte en las comunicaciones y no da tiempo a generarlo. Esto supone una pérdida de stock. La transacción garantiza la atomicidad de la operación: **O se hacen todas las operaciones, o no se hace ninguna.**

Generalmente, cuando se conecta con un cliente a un SGBD, por defecto está activado el modo *AUTOCOMMIT=ON*, es decir, cada comando SQL que se ejecute, será considerado como una transacción independiente. Para activar las transacciones de múltiples sentencias hay que establecer el modo *AUTOCOMMIT= OFF*. A partir de ese momento, todos los comandos SQL enviados al SGBD tendrán que terminarse con una orden COMMIT o una orden ROLLBACK. De este modo, se asegura la integridad de los datos a un nivel más alto. Muchos SGBD requieren de una orden *START TRANSACTION* o *START WORK* para comenzar una transacción y otros lo hacen de forma implícita al establecer el modo autocommit=off.

```
#MySQL, 3 formas para comenzar una transacci6n:
SET AUTOCOMMIT=0; #ó
START TRANSACTION; #ó
BEGIN WORK;

--Oracle:
SET AUTOCOMMIT OFF
```

Para terminar una transacción, tanto en MySQL como en ORACLE, hay que aceptar, o rechazar los cambios mediante:

```
#La palabra clave WORK es opcional
COMMIT WORK; #Acepta los cambios.
ROLLBACK WORK; #Cancela los cambios
```

Cualquier conjunto de sentencias SQL se considera cancelado si termina abruptamente la sesión de un usuario sin hacer COMMIT. Un ejemplo de transacción en MySQL sería la siguiente:

```
SET AUTOCOMMIT 0;
#se actualiza el stock
UPDATE Productos SET Stock=Stock-2 WHERE CodigoProducto='AAAF102';
#se inserta la cabecera del pedido
```

```
INSERT INTO Pedidos VALUES
    (25,now(),'Francisco Garcia','Pendiente de Entrega');
#se inserta el detalle del pedido
INSERT INTO DetallePedidos
(CodigoPedido,CodigoProducto,Unidades) VALUES
        (25,'AAAF102',2);
#aceptar transacción
COMMIT WORK;
```

Acceso concurrente a los datos

Cuando se utilizan transacciones, pueden suceder problemas de concurrencia en el acceso a los datos, es decir, problemas ocasionados por el acceso al mismo dato de dos transacciones distintas. Estos problemas están descritos por SQL estándar y son los siguientes:

Dirty Read (Lectura Sucia). Una transacción lee datos escritos por una transacción concurrente que no ha hecho COMMIT.

Nonrepeateable Read (Lectura No Repetible). Una transacción vuelve a leer datos que leyó previamente y encuentra que han sido modificados por otra transacción, es decir que esa transacción sido confirmada desde la lectura inicial

Phantom Read (Lectura Fantasma). Una transacción lee unos datos que no existían cuando se inició la transacción, es decir una transacción re-ejecuta una consulta retornando un conjunto de filas que satisfacen una condición de búsqueda y encuentran que el conjunto de filas satisfacen la condición que cambió gracias a otra transacción recientemente confirmada.

Cuando se trabaja con transacciones, el SGBD puede bloquear conjuntos de datos para evitar o permitir que sucedan estos problemas. Según el nivel de concurrencia que se desee, es posible solicitar al SGBD cuatro niveles de aislamiento. Un nivel de aislamiento define cómo los cambios hechos por una transacción son visibles a otras transacciones:

NIVEL DE AISLAMIENTO DE TRANSACCIONES

Read Uncommited (Lectura no acometida). Permite que sucedan los tres problemas. Las sentencias SELECT son efectuadas sin realizar bloqueos, por tanto, todos los cambios hechos por una transacción pueden verlos las otras transacciones.

Read Commited (Lectura acometida). Los datos leídos por una transacción pueden ser modificados por otras transacciones. Se pueden dar los problemas de Phantom Read y Non Repeteable Read.

Repeateable Read (Lectura Repetible). Tan sólo se permite el problema del Phantom Read. Consiste en que ningún registro leído con un SELECT se puede cambiar en otra transacción.

Serializable. Las transacciones ocurren de forma totalmente aislada a otras transacciones. Se bloquean las transacciones de tal manera que ocurren unas detrás de otras, sin capacidad de concurrencia. El SGBD las ejecuta concurrentemente si puede asegurar que no hay conflicto con el acceso a los datos.

Nivel aislamiento	*Lectura sucia*	*lectura no-repetitiva*	*Lectura fantasma*
Read uncommitted	Posible	Posible	Posible
Read committed	No posible	Posible	Posible
Repeatable read	No posible	No posible	Posible
Serializable	No posible	No posible	No posible

En MySQL, las tablas innodb tienen el nivel de aislamiento por defecto establecido en REPETEABLE READ, y se puede establecer el nivel predeterminado de aislamiento por todas las conexiones mediante el uso de la opción --transaction-isolation en la línea de comandos o en ficheros de opciones. Por ejemplo, se puede establecer la opción en la sección [mysqld] de my.cnf ó en my.ini de este modo:

```
[mysqld]
transaction-isolation = {READ-UNCOMMITTED | READ-COMMITTED
                         | REPEATABLE-READ | SERIALIZABLE}
```

Un usuario puede cambiar el nivel de aislamiento de una sesión individual o de todas las nuevas conexiones con la sentencia SET TRANSACTION. Su sintaxis es la siguiente:

```
SET TRANSACTION ISOLATION
LEVEL {READ UNCOMMITTED |READ COMMITTED
                            | REPEATABLE READ | SERIALIZABLE}
```

En Oracle, el nivel por defecto es *READ COMMITED* y, además de este, sólo permite SERIALIZABLE. Se puede cambiar ejecutando el comando:

```
SET TRANSACTION ISOLATION LEVEL {READ COMMITTEDISERIALIZABLE};
```

¿Sabías que ... ? Algunas transacciones pueden quedar interbloqueadas dependiendo del nivel de aislamiento seleccionado. Esta situación de *interbloqueo*, o *deadlock* entre dos transacciones, consiste en que ninguna de ellas puede seguir ejecutándose porque la primera intenta acceder a un bloque de datos que tiene bloqueada la segunda, y la segunda, intenta acceder a un bloque de datos que tiene bloqueada la primera. En esta situación de interbloqueo, el SGBD elimina a una de las dos transacciones, siendo la *víctima* del interbloqueo la que hace rollback de sus operaciones.