TEMA 5
EDICCIÓN DE DATOS

BASE DE DATOS

Contenidos del Tema

1. Inserción de Datos
2. Actualización de Datos
3. Borrado de Datos
4. Actualización y Borrado con Subconsultas
5. Transacciones
6. Acceso Concurrente a los Datos

1. Inserción de Datos

TEMA 5: EDICIÓN DE DATOS

1. Inserción de Datos

INSERT [INTO] nombre_tabla [(nombre_colum,...)]

VALUES ({expr | DEFAULT},...)

ESPECIFICACIONES:

nombre_tabla: nombre de la tabla en la que se desea insertar el registro.

nombre_columna: opcionalmente puede especificarse la lista de columnas donde se insertará la información, en cuyo caso los valores de la clausula VALUES se insertarán correlativamente en dichas columnas.

Si no se especifica especifica una lista de columnas, los valores se insertarán en el orden en que estén definidas las columnas en la especificación de la tabla.

Puede utilizarse **DEFAULT** para establecer el valor por defecto.

Ejemplos:

1. Indicando las columnas: (BASE DE DATOS CICLISMO)

INSERT INTO ciclista (dorsal, nombre, nomeq)

VALUES (36, ‘Oscar Sevilla’, ‘Kelme’);

¿Qué sucede en el campo que no hemos introducido ningún valor?

¿Podemos alterar el orden en el que indicamos los campos respecto al orden establecido en la definición de la tabla?

2. Sin indicar columnas:

INSERT INTO ciclista

VALUES (37, ‘Bjarne Riis’, 25, ‘Telecom’);

3. Insertar varios registros:

INSERT INTO ciclista

VALUES (38, ‘Angel Arroyo’, 25, ‘Telecom’), (39, ‘Abraham Olano’, 27, ‘Banesto’);

Existe la posibilidad de insertar los datos obtenidos en una consulta combinando INSERT y SELECT.

INSERT [INTO] nombre_tabla [(nombre_colum,…)]

SELECT … FROM …

Ejemplo:

INSERT INTO BackupCiclista

*SELECT * FROM Ciclista;*

* La sentencia SELECT debe devolver una estructura de campos idéntica a la de la tabla en la que se almacenará la información .

2. Actualización de Datos

TEMA 5: EDICIÓN DE DATOS

2. Actualización de Datos

UPDATE nombre_tabla

SET nombre_col1=expr1 [, nombre_col2=expr2] …

[WHERE filtro] ;

Ejemplos:

1. **Pau Gasol ficha por los Knicks y pesa 210 lb:**

```
UPDATE Jugadores
SET Nombre_equipo='Knicks', Peso=210
WHERE Nombre = 'Pau Gasol';
```

2. **Convertimos el peso a kg para todos los jugadores:**

```
UPDATE Jugadores
SET Peso=Peso*0.4535;
```

3. Borrado de Datos

TEMA 5: EDICIÓN DE DATOS

3. Borrado de Datos

DELETE FROM nombre_tabla

[WHERE filtro] ;

Ejemplos:

1. **Jorge Garbajosa abandona la NBA:**

```
DELETE FROM  estadisticas
WHERE jugador=88;
```

```
DELETE FROM Jugadores
WHERE Nombre like ‘Jorge%’;
```

2. **Eliminamos a todos los jugadores:**

```
DELETE FROM Estadisticas;
```

4. Actualización y Borrado con Subconsultas

TEMA 5: EDICIÓN DE DATOS

4. Actualización y Borrado con Subconsultas

Es posible actualizar o borrar registros de una tabla filtrando por medio de una subconsulta.

La única limitación es la que imponen algunos SGBD que no permiten modificar la tabla que se está utilizando en la subconsulta.

Ejemplo: (BDD Jardinería)

1. Eliminar a los representantes de ventas que no tengan clientes:

```
DELETE FROM Empleados
WHERE Puesto='Representante Ventas' AND CodigoEmpleado NOT IN
(SELECT CodigoEmpleadoRepVentas FROM Clientes);
```

5. Transacciones

TEMA 5: EDICIÓN DE DATOS

5. Transacciones

DEFINICIÓN: conjunto de sentencias SQL que se tratan como si fueran una sola instrucción (atómica).

La transacción puede ser:

Confirmada (commit) si todas las operaciones individuales se realizan correctamente.

Abortada (rollback) si se produce algún problema durante la ejecución de cualquiera de las operaciones.

El uso de transacciones es fundamental para garantizar **la integridad de los datos** durante su manipulación.

Ejemplo: para realizar una transferencia bancaria será necesario reducir el saldo de una cuenta y aumentar en la misma medida otra. Mediante una transacción podemos establecer que si no se realizan correctamente ambas operaciones, operaciones, no se modifique el estado de ninguna cuenta

Generalmente cuando nos conectamos a un SGBD se activa el modo **AUTOCOMMIT ON,** es decir, cada operación será considerada una transacción independiente.

Para poder realizar transacciones de múltiples sentencias hemos de establecer el modo **AUTOCOMMIT OFF.**

A partir de este momento todos los comandos SQL deberán terminarse mediante una orden **COMMIT** para aceptar los cambios o **ROLLBACK** para rechazarlos.

Las motores no transaccionales como MyISAM funcionan en modo AUTOCOMMIT ON, por lo que la integridad de los datos se consigue mediante chequeo de operaciones y bloqueo de tablas.

1. **Inicializar variable:**

```
SET AUTOCOMMIT= 0;
```

2. **Comenzar una transacción en MySQL:**

```
START TRANSACTION;   ó
BEGIN WORK;
```

3. **Para comprobar el estado de la transacción:**

```
SHOW VARIABLES  LIKE 'AUTOCOMMIT';
```

4. **Finalizar una transacción tanto en MySQL :**

```
COMMIT [WORK];        #Acepta los cambios
ROLLBACK [WORK];   #Cancela los cambios.
```

* Cualquier conjunto de sentencias se considera cancelado si finaliza de forma abrupta la sesión de usuario.

Ejemplo:

Realizar un pedido mediante una transacción

```
SET AUTOCOMMIT= 0;
```

1. Actualizamos el stock disminuyendo el nº de unidades pedidas:

```
UPDATE Productos SET CantidadEnStock = CantidadEnStock-2
WHERE CodigoProducto='FR-10';
```

2. Creamos la cabecera del pedido

```
INSERT INTO Pedidos(CodigoPedido, FechaPedido, FechaEsperada, Estado, CodigoCliente)
VALUES (130, now(), now(),'Pendiente', 3);
```

3. Creamos el detalle del pedido

```
INSERT INTO DetallePedidos
VALUES (130,'FR-10', 2,10.3,1);
```

4. Aceptamos transacción:

```
COMMIT WORK;
```

6. Acceso Concurrente a los Datos

TEMA 5: EDICIÓN DE DATOS

6. Acceso Concurrente a los Datos

- **Concurrencia:** un conjunto de datos es accedido por varias transacciones de forma simultánea.

- **La concurrencia puede originar varios problemas:**
  - **Lectura Sucia (Dirty Read):** una transacción lee datos escritos por otra transacción que no ha hecho COMMIT.
  - **Lectura no Repetible (Nonrepeatable Read):** una transacción trata de leer datos que leyó previamente y encuentra que han sido modificados por otra transacción.
  - **Lectura Fantasma (Phantom Read):** una transacción lee datos que no existían cuando se inició la transacción.

Para solucionar los problemas de la concurrencia, el SGBD puede bloquear conjuntos de datos, existiendo 4 niveles de aislamiento:

- **Lectura No Acometida (Read Uncommitted):** todos los cambios hechos por una transacción pueden ser vistos por otras. Permite que sucedan los tres problemas.

- **Lectura Acometida (Read Committed):** los datos leídos por una transacción pueden ser modificados por otras. Pueden darse los problemas de lectura fantasma y no repetible.

- **Lectura Repetible (Repeatable Read):** ningún registro leído se puede modificar en otra transacción. Permite sólo el problema de lectura fantasma.

- **Serializable:** las transacciones ocurren de manera totalmente aislada. Para ello las transacciones se bloquean de manera que se realicen una detrás de otra y de esta manera se asegura que no haya ningún conflicto.

- En MySQL, las tablas innodb tienen por defecto el nivel de aislamiento REPETEABLE READ, pero se puede cambiar ejecutando:

```
SET TRANSACTION ISOLATION

LEVEL { READ UNCOMMITTED | READ COMMITTED |

        REPEATABLE READ | SERIALIZABLE}
```

- Si deseamos conocer el nivel de aislamiento con el que actualmente está trabajando nuestro servidor no tenemos más que lanzar la sentencia: *SHOW VARIABLES LIKE 'tx_isolation'*