Consultas de Acción: INSERT, DELETE y UPDATE

Las consultas de acción,son aquellas que no devuelven ningún registro, sino que modifican los datos de una o varias tablas de una base de datos, de forma que permiten añadir, borrar o modificar registros.

INSERT INTO

Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos. Esta consulta puede ser de dos tipos: Insertar un único registro ó Insertar en una tabla los registros contenidos en otra tabla.

Para insertar un único registro:

En este caso la sintaxis es la siguiente:

```
INSERT INTO Tabla [(campo1, campo2, .., campoN)]
 VALUES (valor1, valor2, ..., valorN);
```

Ejemplo: Añadir un nuevo suministrador: “S7”, “Londres”, “Luis”.

```
INSERT INTO Suministrador (Snum, Ciudad, Snombre)
VALUES (‘S7’, ‘Londres’, ‘Luis’);
```

Los nombres de los campos son opcionales, y si no se ponen, se asignarán los valores por posición del campo en el diseño de la tabla.

Esta es la opción más habitual. Hay que prestar especial atención a acotar entre comillas simples (') los valores literales (cadenas de caracteres) y las fechas indicarlas en formato del sistema gestor de bases de datos (en MySQL, el formato es aaaa-mm-dd). Los nombres de los campos en la tabla, sino se ponen, se asignaran por la posición que ocupan en el diseño de la tabla. También se pueden poner como valores de un campo, el valor devuelto por una función (por ejemplo, Current_date() para un campo de tipo fecha, donde se insertaría como valor la fecha del sistema).

Para insertar varios registros en la tabla

```
INSERT INTO  Suministrador (Snum, Ciudad,Snombre)
VALUES          ('S6','Roma','pepe'),
                ('S7','Italia','Juan');
```

Para insertar Registros de otra Tabla:

En este caso la sintaxis es:

```
INSERT INTO Tabla [(campo1, campo2, ..., campoN) ]
 SELECT TablaOrigen.campo1, TablaOrigen.campo2, ..., TablaOrigen.campoN
FROM TablaOrigen
```

Ejemplo: Insertar en la tabla suministradores los artículos cuyo peso sea mayor o igual a 17. Este ejemplo solo sirve para probar esta posibilidad de inserción de registros de otra tabla.

```
INSERT INTO Suministrador
SELECT Pnum, Pnombre, Peso, Ciudad
FROM Producto
WHERE Peso>=17;
```

DELETE

Crea una consulta de eliminación que elimina los registros de una o más de las tablas listadas en la cláusula FROM que satisfagan la cláusula WHERE. Esta consulta elimina los registros completos, no es posible eliminar el contenido de algún campo en concreto. Su sintaxis es:

```
DELETE FROM Tabla
WHERE criterio ;
```

Ejemplo: Eliminar el suministrador número 'S7'.

```
DELETE FROM Suministrador
WHERE Snum='S7';
```

La cláusula DELETE sin una condición eliminaría todos los registros de la tabla, lo cual puede ser muy peligroso. Es equivalente a la cláusula TRUNCATE TABLE.

```
DELETE FROM Suministrador;
TRUNCATE TABLE Suministrador;
```

Una vez que se han eliminado los registros utilizando una consulta de borrado, no puede deshacer la operación. Si desea saber qué registros se eliminarán, primero examine los resultados de una consulta de selección que utilice el mismo criterio y después ejecute la consulta de borrado. Mantenga copias de seguridad de sus datos en todo momento. Si elimina los registros equivocados podrá recuperarlos desde las copias de seguridad.

Ejemplo: Eliminar los suministradores insertados anteriormente de la tabla de productos.

```
DELETE FROM Suministrador
WHERE Snum LIKE 'P_';
```

ejemplo: se desea borrar los campos de una tabla que tengan un valor 0 en el campo limiteCredito=0

```
delete from clientes
where codigocliente in(
select codigocliente
    from clientes
    where limitecredito=0)
```

Esto daría error pues estamos borrando datos de la misma tabla que se quiere actualizará

UPDATE

Crea una consulta de actualización que cambia los valores de los campos de una tabla especificada basándose en un criterio específico. Su sintaxis es:

```
UPDATE Tabla
SET Campo1=Valor1, Campo2=Valor2, ... CampoN=ValorN
WHERE Criterio;
```

UPDATE es especialmente útil cuando se desea cambiar un gran número de registros o cuando éstos se encuentran en múltiples tablas. Puede cambiar varios campos a la vez.

Ejemplo: Incrementar en 10 unidades el peso de todos los artículos de color rojo.

```
UPDATE Producto
 SET Peso=Peso + 10
WHERE Color=’Rojo’;
```

Si en la actualización no especificamos ningún criterio, serán actualizados todos los registros de la tabla.

OTRAS CLÁUSULAS DE SQL:

UNION [ALL]

Se usa para combinar los resultados de varias sentencias SELECT en un único conjunto de resultados. Si no se usa la palabra clave ALL para la UNION, todas las filas devueltas serán únicas, igual que si se hubiese especificado DISTINCT para el conjunto de resultados total. Si se especifica ALL, se obtendrán todas las filas coincidentes de todas las sentencias SELECT.

Ejemplo: Devolver todas las ciudades distintas de los suministradores y los productos.

```
SELECT Ciudad FROM Suministrador
UNION
SELECT Ciudad FROM Producto;
```

Ejemplo: Devolver todas las ciudades de los suministradores y los productos, ordenando la salida en orden descendente.

```
SELECT Ciudad FROM Suministrador
UNION ALL
SELECT Ciudad FROM Producto ORDER BY Ciudad DESC;
```

BORRADO Y MODIFICACIÓN DE REGISTROS CON RELACIONES

Hay que tener en cuenta que no siempre se pueden borrar o modificar los datos, hay que tener en cuenta las restricciones, las cláusulas REFERENCES de la sentencia al crear una tabla para crear las relaciones de clave foránea-clave primaria de alguna columna de una tabla:

definición de referencias:

REFERENCES nombre_tabla[(nombre_columna,...)]

[ON DELETE opción_referencia]

[ON UPDATE opción_referencia]

opción referencia:

CASCADE|SET NULL|NO ACTION

Si se intenta eliminar un registro con otros registros relacionados y se ha seleccionado la opción ON DELETE NO ACTION Y ON UPDATE NO ACTION no se podrá realizar la acción de borrado o actualización. Sin embargo si la creación de la relación estuviese personalizada con las opciones ON UPDATE CASCADE y ON DELETE CASCADE si se podría

INSTRUCCIONES VARIADAS

Añadir columnas a una tabla:

ALTER TABLE *table_name* ADD *column_name datatype*;

Añadir una columna en una tabla indicando la posición

ALTER TABLE table_name ADD new_column_name column_definition

[FIRST | AFTER column_name];

```
ALTER TABLE contacts
 ADD last_name varchar(40) NOT NULL
  AFTER contact_id;
```

Añadir múltiples columnas

```
ALTER TABLE table_name
 ADD new_column_name column_definition
  [ FIRST | AFTER column_name ],
 ADD new_column_name column_definition
  [ FIRST | AFTER column_name ],
 ...
;
```

Borrar una columna en una tabla:

ALTER TABLE *table_name* DROP COLUMN *column_name*;

Modificar una columna:

ALTER TABLE table_name MODIFY column_name datatype;

```
ALTER TABLE contacts
 MODIFY last_name varchar(50) NULL;
```

Modificar múltiples columnas:

```
ALTER TABLE table_name
 MODIFY column_name column_definition
  [ FIRST | AFTER column_name ],
 MODIFY column_name column_definition
  [ FIRST | AFTER column_name ],
 ...
;
```

Borrar una columna de una tabla:

```
ALTER TABLE table_name
 DROP COLUMN column_name;
```

Renombrar una columna en una tabla.-

```
ALTER TABLE table_name
 CHANGE COLUMN old_name new_name
  column_definition
  [ FIRST | AFTER column_name ]
```

```
ALTER TABLE contacts
 CHANGE COLUMN contact_type ctype
  varchar(20) NOT NULL;
```

Renombrar una tabla

```
ALTER TABLE table_name
 RENAME TO new_table_name;
```

El acceso a la información.-

Cuando se administra la seguridad en el acceso a información de una base de datos, es común utilizar dos tipos de seguridad, la integrada en el propio sistema operativo y la que proporciona el SGBD. En la seguridad integrada, se suele contar con los usuarios de un sistema de dominio o un servicio de directorio (LDAP) para proporcionar el acceso a determinados recursos del gestor de base de datos. En la seguridad nativa del SGBD es el propio software servidor el que proporciona los mecanismos mediante los cuales se autoriza a un usuario a utilizar distintos elementos de la base de datos.

LAS VISTAS

Una vista es una tabla sin contenidos, totalmente virtual que devuelve las filas resultado de ejecutar una consulta SQL La diferencia con una consulta ejecutada directamente es que, mientras cada sentencia SQL enviada al SGBD tiene que pasar por un proceso de compilación, la vista es una consulta cuya definición ha sido almacenada previamente y que ya ha sido compilada, siendo por tanto el tiempo de ejecución bastante menor. También tiene una implicación importante en el hecho de que un usuario podría no tener acceso a la información de varias tablas y sin embargo, sí tener acceso a la vista que consulta esas tablas, proporcionando de esta manera un acceso controlado solo a determinadas filas y columnas de esas tablas.

Sintaxis para crear una vista:

```
CREATE [or replace] VIEW [esquema.]nombre_vista [(lista_columnas)] AS sentencia_select
```

Si la vista ya existe se puede reemplazar con replace. La ejecución del create view provoca que se compile la sentencia select y que se almacene con el nombre_vista. Los nombres de las columnas de la vista se pueden especificar mediante lista_columnas. Si se especifica la lista de columnas, cada columna tendrá el alias correspondiente, si no, se obtendrá el nombre devuelto por la consulta.

Ejemplo de CREATE VIEW:

```
CREATE VIEW nba.jugadores AS
select nombre, posicion from nba.jugadores where nombre_equipo='HEAT';
```

Para eliminar una vista se hace uso de DROP VIEW

```
DROP VIEW [esquema].nombre_vista;
```