# UT5 - Apuntes DML accesibles

Sitio: FRANCISCO DE GOYA
Curso: Bases de Datos Pendientes
Libro: UT5 - Apuntes DML accesibles

Imprimido por: Lucía Hernández Montero
Día: lunes, 24 de octubre de 2022, 09:34

# Tabla de contenidos

# 1. Modificación de los datos

Una vez se han creado las tablas de una base de datos se puede proceder a insertar datos, borrarlos y modificarlos. Todas estas operaciones no afectan a la estructura de la tabla: sus columnas, restricciones y tipos de datos.

# 2. Inserción de datos

La sentencia INSERT permite insertar una fila en una tabla, es decir, añadir un registro de información a una tabla.

La operación de inserción se expresa mediante el comando INSERT, cuya sintaxis es:

INSERT INTO <tabla>[(columna1, ... columnaN)] VALUES( valor[, valor...])

o bien

INSERT INTO <tabla>[(columna1, ... columnaN)] <sentencia_select>

Recordemos que la sintaxis se lee de la siguiente forma: las palabras entre < y > se sustituyen por un nombre u otra sentencia, en este caso una tabla. Los corchetes significan que el interior de lo que encierran es opcional, no hay que escribir los corchetes.

<Tabla> es el nombre de la tabla donde se quiere insertar la fila. Después del nombre de la tabla, de forma optativa, se puede indicar las columnas donde se va a insertar la información. Si se especifican las columnas, la lista de valores (VALUES) a insertar se asociará correlativamente con los valores de las columnas indicadas. Si no se especifican las columnas, la lista de valores se escribirá conforme al orden de las columnas en la definición de la tabla.

## 2.1. Insert especificando la lista de columnas

```
INSERT INTO mascotas (Nombre, Propietario, Especie, Sexo, Nacimiento, Fallecimiento) VALUES ('Fluffy', 'David', 'Gato', 'f', '1999-02-04', NULL);
```

Este tipo de insert, hace corresponder a la columna Nombre el valor 'Fluffy', a la columna especie el valor 'Gato', etc.

Si dejamos algún campo sin indicar, deben ser aquellos que no son obligatorios y que, por tanto, podrán tomar el valor NULL. Si la tabla Mascotas tuviera además un campo 'codigo', definido como clave y como AUTO INCREMENT, si no especificamos su valor, sería asignado automáticamente al valor del último código que se asignó + 1. En ese caso, si no queremos dar un código concreto por nosotros mismos, deberemos dejar a NULL la columna Codigo, para que el campo AUTO INCREMENT haga su función.

## 2.2. Insert sin especificar la lista de columnas

```
INSERT INTO mascotas VALUES ('Fluffy', 'David', 'Gato', 'f', '1999-02-04', NULL');
```

En este caso, al no especificarse la lista de columnas, hay que indicar todos los valores para todas las columnas en el orden en que están definidas las columnas en la tabla.

## 2.3. Inserción múltiple en la misma sentencia

La única diferencia es tras cerrar el paréntesis de los valores a insertar, se puede añadir una coma y abrir de nuevo otro paréntesis con los valores de una segunda fila, y así sucesivamente:

```
INSERT INTO mascotas VALUES ('Fluffy', 'David', 'Gato', 'f', '1999-02-04', NULL'), ('Kaiser', 'Diana', 'Perro', 'm', '1997-07-29', '2004-05-23');
```

## 2.4. Inserción con valores por defecto

```
INSERT INTO mascotas (‘Kaiser’, ‘Diana’, ‘Perro’, DEFAULT, ‘1997-07-29’, ‘2004-05-23’);
```

En este caso, hemos usado el valor DEFAULT para asignar el valor por defecto a la cuarta columna de la tabla Mascotas, en este caso, la columna sexo que tiene definida la asignación por defecto del valor 'm'.

## 2.5. Insert select

Una variante de la sentencia insert consiste en utilizar la sentencia select para obtener un conjunto de datos y posteriormente insertarlos en la tabla.

```
INSERT INTO BackupMascotas SELECT * FROM Mascotas;
```

La sentencia SELECT debe devolver tantas columnas como columnas tenga la tabla donde vamos a introducir la información. En el ejemplo anterior la tabla BackupMascotas tiene una estructura idéntica a la tabla Mascotas.

Observamos además, que la sentencia SELECT puede ser tan compleja como deseemos, usando filtros, agrupaciones, ordenaciones, etc.

## 2.6. Error típico en insert

Si construimos una sentencia INSERT con más campos en la lista de valores que el número de columnas especificadas (o número de columnas de la tabla) el gestor nos informará del error.

```
INSERT INTO mascotas (Nombre, Propietario, Especie) VALUES ('Piolín','Canario');
```

ERROR 1136 (21S01): Column count doesn't match value count at row 1

# 3. Actualización de datos

La sentencia UPDATE permite modificar el contenido de cualquier columna y de cualquier fila de una tabla.

La modificación se realiza mediante el comando UPDATE, cuya sintaxis es:

UPDATE <tabla> SET columna1=valor1, ... columnaN=valorN WHERE <condicion>

O bien,

UPDATE <tabla> SET columna1=valor1, ... columnaN=valorN WHERE columnas=(SELECT ...)

La actualización se realiza dando a las columnas columna1, columna2... los valores valor1, valor2,... Se actualizan todas las filas seleccionadas por el filtro indicado mediante la cláusula WHERE. Esta cláusula WHERE es idéntica a la que hemos utilizado hasta ahora en el uso de la sentencia SELECT.

Por ejemplo si quisiéramos cambiar al propietario de la mascota 'Piolín' habrá que ejecutar la siguiente sentencia:

```
UPDATE mascotas SET propietario='Lucía' WHERE Nombre='Piolín';
```

Query OK, 1 row affected (0.02 sec)

Rows matched: 1 Changed: 1 Warnings: 0

El gestor nos informa de que el filtro seleccionó una fila, y que, por tanto, cambió 1 fila, en este caso la columna propietario de esa fila seleccionada.

# 4. Borrado de datos

En SQL se utiliza la sentencia DELETE para eliminar filas de una tabla.

Cuya sintaxis es:

```
DELETE [FROM] <tabla> [WHERE condicion]
```

El comando DELETE borrará las filas seleccionados por el filtro WHERE, que al igual que UPDATE es idéntico al de la sentencia SELECT. Por ejemplo, si quisiéramos borrar a la mascota 'Piolín' de la base de datos, tendríamos que escribir la siguiente sentencia:

```
DELETE FROM mascotas WHERE Nombre = 'Piolín';
```

Query OK, 1 row affected (0.01 sec)

El gestor nos informa de que 1 fila fue borrada.

# 5. Condiciones

En cualquiera de las sentencias que se han mostrado anteriormente se pueden utilizar condiciones. Estas condiciones sirven para que la acción afecte únicamente a las filas que cumplan las condiciones, por ejemplo a la hora de borrar filas o de actualizarlas.

Los comparadores son similares a cualquier lenguaje de programación:

- Igual: = (un solo igual, no dos)
- No igual: !=
- Menor, mayor, etc: <, >, <=, >=
- SImilar: like

La condición ha de tener un comparador entre dos elementos, que habitualmente son un campo de una tabla (sobre la que se hace un update o delete) y un valor estático, aunque puede darse cualquier combinación, incluso entre campos de la misma o distintas tablas. También se puede comparar un campo con el resultado de un select.

De esta forma un ejemplo sería:

```
delete from tabla where edad > 22;
```

Eliminará todas las filas cuyo  campo edad sea superior a 22. En caso de querer borrar un único elemento, se deberá comparar con el identificador:

```
delete from persona where dni = "12345678A";
```

## 5.1. Comparador like

El comparador like es un comparador específico de mysql, aunque su funcionalidad se puede conseguir en otros lenguajes con funciones.

Lo que hace es comparar entre dos cadenas si estas son similares. Para ello se sirve de dos metacaracteres.

%: se sustituye por 0, 1 o cualquier número de caracteres cualesquiera.

_: se sustituye obligatoriamente  por un carácter cualquiera.

Así tendríamos:

nombre like "a%" : nombre empieza por a.

nombre like "a_" : nombre de dos caracteres siendo el primero una a.

nombre like "%a" : nombre acaba por a.

nombre like "%a%": nombre contiene al menos una a.

Las combinaciones son enormes por lo que es una herramienta muy potente.