# TEMA 9: EDICIÓN DE DATOS

## 1. Inserciones.

La introducción de datos en una tabla se realiza empleando la sentencia INSERT, que admite varios formatos.

### 1.1. Inserción con *values*.

El primero de los formatos que vamos a estudiar es el siguiente:

```
INSERT [INTO] Nombre_Tabla [(atributo1,…, atributon)]
VALUES ([DEFAULT|valor11],…, [DEFAULT|valor1n]),
       ([DEFAULT|valor21],…, [DEFAULT|valor2n]),… ;
```

Como se puede observar, se debe indicar obligatoriamente el nombre de la tabla en la que se desean insertar los datos y se pueden especificar a continuación entre paréntesis los nombres de los atributos a los que se va a dar valor. Los valores se especifican después de la palabra VALUES entre paréntesis y cada $valor_{ji}$ se asigna al correspondiente $atributo_i$, es decir, $valor_{11}$ se asigna al $atributo_1$, $valor_{12}$ se asigna al $atributo_2$ y así sucesivamente en la primera fila, $valor_{21}$ se asigna al $atributo_1$, $valor_{22}$ se asigna al $atributo_2$ en la segunda fila, y así sucesivamente. Debe haber obviamente una concordancia de tipos entre los atributos y sus correspondientes valores. Se puede en la misma instrucción añadir más de una fila a la tabla. En vez de indicar un valor concreto para un atributo se puede escribir DEFAULT para indicar que se le asigne el valor por defecto especificado para el mismo en la instrucción de creación de la tabla. En los valores de los atributos se puede hacer referencia a atributos indicados previamente en la sentencia INSERT.

Si no se indican los atributos a los que se va a dar valor después del nombre de la tabla, se sobreentiende que se va a dar valor a todos los atributos de la tabla en el mismo orden en el que aparecen en la definición de la tabla.

Por ejemplo, vamos a añadir dos nuevos artículos a la tabla *Articulo*: uno con código *A0022*, descripción *Cuaderno grande de espiral* y precio 2,80 € y otro con código *A0023*, descripción *Paquete de 500 folios DIN A-4* y precio 4,10 €. Para ello deberemos emplear la siguiente instrucción:

```
mysql> insert into Articulo (CodArt, DesArt, PVPArt)
    -> values ('A0022', 'Cuaderno grande de espiral', 2.80),
    -> ('A0023', 'Paquete de 500 folios DIN A-4', 4.10);
Query OK, 2 rows affected (0.10 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Como se puede observar, el sistema nos informa de que se han añadido dos nuevas filas a la tabla y que no se han producido duplicados ni hay advertencias. Nos habría valido también la siguiente instrucción:

```
mysql> insert into Articulo
    -> values ('A0022', 'Cuaderno grande de espiral', 2.80),
    -> ('A0023', 'Paquete de 500 folios DIN A-4', 4.10);
```

porque vamos a dar valor a todos los atributos y hemos especificado los valores en el mismo orden en el que están definidos en la tabla *Articulo*.

Ahora queremos añadir un nuevo empleado a la tabla *Emple* con los siguientes datos: número *2244*, apellido *PIÑEIRO*, oficio *ANALISTA*, fecha de alta *15/12/2019*, salario *2200 €* y número de departamento *10*. Como en este caso no vamos a dar valor a todos los atributos, podemos especificar tras el nombre de la tabla, los nombres de los atributos a los que vamos a dar valor:

```
mysql> insert into emple (emp_no, apellido, oficio, fecha_alt, salario, dept_no)
     -> values (2244, 'PIÑEIRO', 'ANALISTA', '2019/12/15', 2200, 10);
Query OK, 1 row affected (0.08 sec)
```

No obstante, si no queremos indicar los nombres de los atributos a los que se va a dar valor, podemos dar valor a todos los atributos en el orden en el que aparecen en la tabla asignando valores nulos a los atributos con valor desconocido:

```
mysql> insert into emple
     -> values (2244, 'PIÑEIRO', 'ANALISTA', null, '2019/12/15', 2200, null, 10);
```

### 1.2. Inserción con *set*.

Existe otra modalidad de sentencia INSERT, que no está presente en el estándar de SQL, que solo permite añadir una fila a la tabla y cuyo formato se muestra a continuación:

```
INSERT [INTO] Nombre_Tabla
SET atributo1 = {valor1|DEFAULT}, atributo2 = {valor2|DEFAULT}, …
```

Como se puede observar, se deben indicar por cada uno de los atributos a los que se desea dar valor, después de la palabra SET, su nombre y a continuación el valor que se le desea asignar o la palabra DEFAULT. Así, con la siguiente orden se añade un nuevo empleado a la tabla *Emple* con número *2245*, apellido *GÓMEZ*, oficio *PROGRAMADOR*, fecha de alta el *07/12/2019*, salario *1500 €*, comisión el 10% del salario y departamento el establecido por defecto:

```
mysql> insert into emple
    -> set emp_no=2245, apellido='GÓMEZ', oficio='PROGRAMADOR',
    -> fecha_alt='2019/12/07', salario=1500, comision = salario*10/100,
    -> dept_no= default;
Query OK, 1 row affected (0.09 sec)
```

### 1.3. Inserción con *select*.

Existe una tercera modalidad de sentencia INSERT que permite añadir varias filas a una tabla a partir de los datos de otra tabla obtenidos mediante una sentencia SELECT. Su formato es el siguiente:

```
INSERT [INTO] Nombre_Tabla [(atributo1, atributo2, …, atributon)]
SELECT...
```

Igual que con el otro formato de sentencia INSERT, si no se indican los atributos opcionales, se entiende que se va a dar valor a la totalidad de los atributos de la tabla y en el orden en el que aparecen en su definición.

Para explicar esta modalidad de sentencia INSERT primero vamos a crear una nueva tabla llamada *Directores*, con la misma estructura que la tabla *Emple* con la excepción del campo *oficio*. La creamos con la siguiente sentencia CREATE TABLE:

```
create table directores
(emp_no int primary key,
apellido varchar(40) not null,
dir int,
fecha_alt date not null,
salario float not null,
comision float,
dept_no int default 10 not null,
constraint FK_Dir_Directores  foreign key(dir) references emple(emp_no),
constraint   FK_Dept_no_Directores     foreign   key(dept_no)     references
                                       depart(dept_no) on update cascade);
```

Ahora vamos a añadir a esta tabla todos los datos de la tabla *Emple* correspondientes a los directores. Como vamos a dar valor a todos los atributos de la tabla *Directores*, no los indicamos:

```
mysql> insert into directores
    -> select emp_no, apellido, dir, fecha_alt, salario, comision, dept_no
    -> from emple
    -> where oficio= 'DIRECTOR';
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Tras la ejecución de esta inserción, el contenido de la tabla *Directores* es el siguiente:

```
mysql> select * from directores;
+--------+----------+------+------------+---------+----------+---------+
| emp_no | apellido | dir  | fecha_alt  | salario | comision | dept_no |
+--------+----------+------+------------+---------+----------+---------+
|   7566 | JIMÉNEZ  | 7839 | 2014-02-04 |    2300 |        0 |      20 |
|   7698 | NEGRO    | 7839 | 2014-05-01 |    2200 |        0 |      30 |
|   7738 | CEREZO   | 7839 | 2014-09-06 |    2160 |        0 |      10 |
+--------+----------+------+------------+---------+----------+---------+
3 rows in set (0.00 sec)
```

Vamos ahora a insertar un nuevo empleado en la tabla *Emple* con número *1111*, apellido *AGUIRRE*, oficio *VENDEDOR*, director el empleado número *7698*, fecha de alta el día de hoy (valor que nos proporciona la función *current_date*), salario 1500 €, comisión 0 € y número de departamento el correspondiente al departamento ubicado en Barcelona.

Pues bien, en este caso, los valores de todos los atributos son conocidos excepto el correspondiente al atributo *dept_no*. Para obtener el valor correspondiente a este atributo, debemos crear una consulta sobre la misma tabla sobre la que se realiza la inserción (*Emple*) con el fin de obtener el número de departamento (*dept_no*) correspondiente al departamento ubicado en Barcelona:

```
Select distinct dept_no from emple
where dept_no = (select dept_no from Depart where loc = 'BARCELONA');
```

Para que esta consulta nos devuelva un solo valor es necesario incorporar la palabra *distinct.*

En esta consulta deberemos señalar como valores constantes los indicados en el primer párrafo, por lo que la sentencia insert nos quedará como sigue:

```
mysql> Insert into Emple
    -> Select distinct 1111, 'AGUIRRE', 'VENDEDOR', 7698, current_date, 1500, 0, dept_no
    -> from emple
    -> where dept_no = (select dept_no from Depart where loc = 'BARCELONA');
Query OK, 1 row affected (0.15 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

En caso de no incluir la palabra distinct, la consulta nos devolvería el número de departamento 30 varias veces e intentaría insertar tantas filas en la tabla *Emple* como número de filas devuelva la consulta, lo que originaría un error debido a que intentaría insertar varios empleados con el número 1111.

Ahora vamos a insertar otro nuevo empleado, en este caso con número *1122*, apellido *FANJUL*, fecha de alta el día de hoy, y el resto de los datos serán iguales que los del empleado *AGUIRRE* que se acaba de insertar. Para realizar esta inserción tendremos que hacer una consulta que nos devuelva los valores de los atributos desconocidos (*oficio*, *dir*, *salario*, *comision* y *dept_no*) a partir de los datos del empleado apellidado AGUIRRE.

```
mysql> Insert into Emple
    -> Select 1122, 'FANJUL', oficio, dir, current_date, salario, comision, dept_no
    -> from emple
    -> where apellido = 'AGUIRRE';
Query OK, 1 row affected (0.09 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

Podemos observar como ahora tenemos los datos presentes de estos dos empleados en la tabla *Emple*:

```
mysql> select * from emple;
+--------+----------+-------------+------+------------+---------+----------+---------+
| emp_no | apellido | oficio      | dir  | fecha_alt  | salario | comision | dept_no |
+--------+----------+-------------+------+------------+---------+----------+---------+
|   1111 | AGUIRRE  | VENDEDOR    | 7698 | 2020-01-12 |    1500 |        0 |      30 |
|   1122 | FANJUL   | VENDEDOR    | 7698 | 2020-01-12 |    1500 |        0 |      30 |
. . .
|   7900 | JIMENO   | EMPLEADO    | 7698 | 2014-12-03 |     725 |        0 |      30 |
+--------+----------+-------------+------+------------+---------+----------+---------+
16 rows in set (0.00 sec)
```

### 1.4. Inserción con *REPLACE*.

REPLACE sirve, al igual que INSERT, para añadir filas a una tabla. La diferencia con respecto a INSERT es que, si al introducir una fila, se pretende asignar un valor repetido para un atributo que es clave primaria o único, no se produce un error, sino que el antiguo registro se borrará antes de insertar el nuevo. Su formato es el mismo que el de la orden INSERT, pero sustituyendo la palabra INSERT por REPLACE.

Para probar esta orden veamos, en primer lugar, cuál es el contenido de la tabla *Artículo*:

```
mysql> select * from Articulo;
+--------+-----------------------------+--------+
| CodArt | DesArt                      | PVPArt |
+--------+-----------------------------+--------+
| A0012  | Goma de borrar              |   0.16 |
| A0022  | Cuaderno grande de espiral |    2.8 |
| A0043  | Bolígrafo azul              |   0.82 |
| A0075  | Lápiz 2B                    |   0.58 |
| A0078  | Bolígrafo rojo normal       |   1.05 |
| A0089  | Sacapuntas                  |   0.26 |
+--------+-----------------------------+--------+
6 rows in set (0.00 sec)
```

Pues bien, en esta situación, si pretendemos añadir un nuevo artículo con código A0022 con la sentencia INSERT se producirá un error, como se puede ver a continuación:

```
mysql> insert into Articulo values ('A0022', 'Paquete de 100 folios', 1.45);
ERROR 1062 (23000): Duplicate entry 'A0022' for key 'PRIMARY'
```

Sin embargo, si ejecutamos esta instrucción con REPLACE, se modifica la fila correspondiente al artículo con código A0022:

```
mysql> replace into Articulo values ('A0022', 'Paquete de 100 folios', 1.45);
Query OK, 2 rows affected (0.04 sec)

mysql> select * from Articulo;
+--------+-----------------------+--------+
| CodArt | DesArt                | PVPArt |
+--------+-----------------------+--------+
| A0012  | Goma de borrar        |   0.16 |
| A0022  | Paquete de 100 folios |   1.45 |
| A0043  | Bolígrafo azul        |   0.82 |
| A0075  | Lápiz 2B              |   0.58 |
| A0078  | Bolígrafo rojo normal |   1.05 |
| A0089  | Sacapuntas            |   0.26 |
+--------+-----------------------+--------+
6 rows in set (0.00 sec)
```

## 2. Modificaciones.

La modificación de datos de una tabla se puede realizar con la sentencia UPDATE, cuyo formato es el siguiente:

```
UPDATE NomTabla
SET atributo1 = valor1, atributo2 = valor2,…, atributon = valorn
[WHERE condición]
[ORDER BY criterio]
```

Se debe especificar, como se puede observar, el nombre de la tabla en la que se desean modificar los datos tras la palabra UPDATE y tras la palabra SET, por cada uno de los atributos cuyo valor se desea modificar, el nombre del atributo y el nuevo valor que se le desea asignar. Tras la cláusula WHERE se indicará la condición que selecciona las filas que se desean modificar. Si se omite esta cláusula, se actualizarán todas las filas de la tabla. Se puede indicar opcionalmente el orden en el que se desean actualizar los registros mediante la cláusula ORDER BY.

Por ejemplo, supongamos que deseamos incrementar en un 5% los precios de los artículos con precio inferior a 1 €. Para ello, deberemos escribir la siguiente sentencia:

```
mysql> update Articulo
    -> set PVPArt = PVPArt + PVPArt*5/100
    -> where PVPArt < 1;
Query OK, 4 rows affected (0.08 sec)
Rows matched: 4  Changed: 4  Warnings: 0
```

Como se puede observar, el sistema nos informa de que en este caso se han modificado 4 filas de la tabla *Articulo*.

La sentencia UPDATE puede incluir una o varias subconsultas en la cláusula WHERE, siguiendo el formato:

```
UPDATE NomTabla
SET atributo1 = valor1, atributo2 = valor2,…, atributon = valorn
WHERE atributox operador (select …)
[ORDER BY criterio];
```

donde *operador* puede ser cualquiera de los que se pueden colocar antes de una subconsulta: =, !=, <>, <, <=, >, >=, etc.

Mediante la siguiente orden se asigna la fecha de hoy a los pedidos para los que no se ha solicitado ningún artículo, es decir, para aquellos que no aparecen en la tabla *LineaPedido*:

```
mysql> update Pedido
    -> set FecPed = current_date
    -> where RefPed not in (select RefPed from LineaPedido);
Query OK, 1 row affected (0.14 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

También se pueden incluir una o varias subconsultas en la cláusula SET, para obtener el valor que se desea asignar a cada atributo cuyo valor se desea modificar, siguiente uno de los siguientes formatos:

```
UPDATE NomTabla
SET atributo1 = (select … ), atributo2 = (select … ), …
[WHERE condición]
[ORDER BY criterio];

UPDATE NomTabla
SET (atributo1, atributo2, …) = (select atributox, atributoy , …)
[WHERE condición]
[ORDER BY criterio];
```

Se ha de tener en cuenta que esta subconsulta debe seleccionar una única fila y un atributo, si se sigue el primero de los formatos, o bien, el mismo número de atributos que los que hay entre paréntesis al lado de SET, en el caso del segundo formato.

Mediante la siguiente orden se modifica el departamento en el que trabaja el empleado apellidado AGUIRRE, asignándole el departamento ubicado en MADRID:

```
mysql> update Emple
    -> set dept_no = (select dept_no from Depart where loc = 'MADRID')
    -> where apellido = 'AGUIRRE';
Query OK, 1 row affected (0.12 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

## 3. Borrados.

La eliminación de datos de una tabla se puede realizar empleando la sentencia DELETE, cuyo formato es el siguiente:

```
DELETE FROM Nombre_Tabla
[WHERE condición]
[ORDER BY criterio]
```

Se debe especificar, como es obvio, el nombre de la tabla de la que se desean borrar los datos. Si no se incluye cláusula WHERE, se borrarán todas las filas de la tabla. En caso contrario, en la cláusula WHERE se indicará la condición que deben cumplir las filas que se desean eliminar. Se puede incluir en la cláusula ORDER BY el orden en el que se desean borrar las filas de la tabla.

Si deseamos eliminar, por ejemplo, de la tabla *Articulo* los productos con precio inferior a 0,30 €, pondremos:

```
mysql> delete from Articulo
    -> where PVPArt < 0.3;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint
fails (`pedidos`.`lineapedido`, CONSTRAINT `fk_CodArt_LineaPedido` FOREIGN KEY
(`CodArt`) REFERENCES `articulo` (`CodArt`) ON UPDATE CASCADE)
```

Como se puede observar, nos sale un mensaje de error porque falla una restricción de clave ajena. Y es que ocurre que hay alguna línea de pedido para algún artículo con precio inferior a 0,30 €, por lo que no se puede llevar a cabo el borrado. Sin embargo, sí que podremos borrar algún artículo con precio superior a 2 €:

```
mysql> delete from Articulo
    -> where PVPArt > 2;
Query OK, 2 rows affected (0.06 sec)
```

En este caso, como se puede observar, se eliminan dos artículos.

Si deseamos eliminar los pedidos para los cuales no se ha creado ninguna línea de pedido, precisaremos de una subconsulta en la cláusula WHERE, como se indica a continuación:

```
mysql> delete from Pedido
    -> where RefPed not in (select RefPed from LineaPedido);
Query OK, 2 rows affected (0.08 sec)
```

Si quisiésemos eliminar todos los pedidos de la tabla *Pedido*, deberíamos emplear la siguiente orden:

```
delete from Pedido;
```

## 4. Control de transacciones.

Una transacción está formada por un conjunto de instrucciones escritas en un lenguaje de manipulación de datos o en un lenguaje de programación y está delimitada por instrucciones de la forma “inicio de transacción” y “fin de transacción”. El lenguaje SQL incorpora instrucciones destinadas a indicar el comienzo y el fin de cada transacción.

En SQL una transacción comienza cuando un usuario se conecta al sistema o tras la finalización de otra transacción, mientras que una transacción finaliza con una de las instrucciones SQL siguientes:

- *Commit*, que hace que todas las modificaciones efectuadas sobre la base de datos desde el inicio de la transacción sean parte permanente de la base de datos y libera los recursos ocupados por la transacción.

- *Rollback*, que provoca que la transacción actual aborte, es decir, revierte la transacción desde el inicio.

Cuando realizamos operaciones de actualización sobre la base de datos, es decir, cuando llevamos a cabo inserciones, borrados o actualizaciones, las operaciones se aplicarán automáticamente sobre la base de datos o no dependiendo de si el sistema está o no en modo *autocommit*. Si el modo *autocommit* está activado (si *autocommit* toma el valor 1), toda operación de actualización sobre la base de datos se confirma automáticamente y, por tanto, no habrá opción de abortarlas. Para conocer si el modo *autocommit* está activado, podemos emplear la siguiente orden:

```
mysql> select @@autocommit;
+--------------+
| @@autocommit |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)
```

Por defecto el modo *autocommit* está activado, como podemos observar. Se puede desactivar este modo con la siguiente instrucción:

```
mysql> set autocommit = 0;
Query OK, 0 rows affected (0.00 sec)
```

Para validar los cambios que hayamos efectuado sobre la base de datos si no está activado el modo *autocommit*, es necesario escribir una orden *commit*:

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)
```

Por otro lado, la orden *rollback* aborta la transacción actual, volviendo la base de datos al estado en el que se encontraba tras el último *commit*.

```
mysql> rollback;
Query OK, 0 rows affected (0.00 sec)
```

Para probar el funcionamiento de las órdenes *rollback* y *commit*, se pueden realizar las siguientes operaciones:

Abrir dos sesiones con MySQL. Si no está hecho, dentro de la base de datos *Empresa*, copiamos desde cualquiera de las dos sesiones la tabla *Depart* en una tabla *Depart2* mediante el comando:

```
mysql> create table depart2 as select * from depart;
Query OK, 4 rows affected (0.09 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

A continuación añadimos una nueva fila a la tabla *Depart2* desde la primera sesión:

```
mysql> insert into depart2 values (100, 'Prueba', 'Ciudad de prueba');
Query OK, 1 row affected (0.02 sec)
```

Si consultamos desde la segunda sesión el contenido de la tabla *Depart2*, aparecerá esta nueva fila debido a que por defecto en MySQL está activado el modo *autocommit* y no lo hemos desactivado.

Ahora desactivemos el modo *autocommit* en la primera sesión escribiendo:

```
mysql> set autocommit = 0;
Query OK, 0 rows affected (0.00 sec)
```

Ahora añadamos desde esta primera sesión una nueva fila a la tabla *Depart2*.

```
mysql> insert into depart2 values (110, 'Prueba2', 'Ciudad de prueba 2');
Query OK, 1 row affected (0.00 sec)
```

Si consultamos el contenido de la tabla *Depart2* desde la segunda sesión, no aparecerá el departamento con número 110 debido a que el modo *autocommit* está desactivado y no hemos confirmado la inserción sobre la tabla *Depart2*.

Si ahora ejecutamos la orden *commit* desde la primera sesión y a continuación consultamos el contenido de la tabla *Depart2* desde la segunda sesión, veremos que la nueva fila aparece ya en la tabla *Depart2*.

Debe tenerse en cuenta que todos los comandos siguientes terminan una transacción implícitamente, como si se hubiera ejecutado un *commit* antes de ejecutar el comando: ALTER TABLE, BEGIN, CREATE INDEX, CREATE DATABASE, CREATE TABLE, DROP DATABASE, DROP TABLE, DROP INDEX, LOAD DATA MASTER, LOCK TABLES, RENAME TABLE, SET AUTOCOMMIT=1, START TRANSACTION y TRUNCATE TABLE.