# SQL (II)

## DML
## Lenguaje de Manipulación de Datos

"**Una manera de hacer Europa**". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

# Introducción

- Es una de las partes fundamentales del lenguaje SQL.
- El DML (Lenguaje de Manipulación de Datos) **está formado por las instrucciones capaces de modificar los datos de las tablas.**
- Se llama transacción al conjunto de instrucciones DML que se ejecutan consecutivamente.
- Una transacción se ejecutará completamente o no se ejecutará ninguna de sus instrucciones, ya que una instrucción DML no es realmente efectuada hasta que no se acepta (con la orden COMMIT).



UNION EUROPEA
Fondo Social Europeo
"Una manera de hacer Europa"

"**Una manera de hacer Europa**". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

# INSERTAR DATOS EN UNA TABLA

Instrucción **INSERT.**

**Su sintaxis fundamental es:**

**INSERT INTO tabla [(listaDeCampos)]**

**VALUES (valor1 [,valor2 *...]*)**

**Donde:**

- **tabla** representa la tabla a la que queremos añadir el registro.
- La lista de valores que siguen a VALUES son los valores que damos a los distintos campos del registro.
- Si no se especifica la lista de campos, la lista de valores debe seguir el orden de las columnas según fueron creadas.

# INSERTAR DATOS EN UNA TABLA

- Especificar una lista de campos a rellenar es obligatorio si no queremos rellenar todos los campos o queremos hacerlo en un orden distinto al que tienen en la tabla.
- Los campos no rellenados explícitamente con la orden INSERT, se rellenan con su valor por defecto (DEFAULT) o bien con NULL, si no se ha especificado un valor por defecto.
- Si algún campo tiene una restricción de obligatoriedad (NOT NULL), debemos asignarle un valor obligatoriamente.

# Ejemplo de INSERT

Supongamos que tenemos la tabla CLIENTES, donde la localidad tiene como valor por defecto Plasencia y la dirección permite valores nulos.

| Field | Type | Null | Key | Default |
|---|---|---|---|---|
| DNI | varchar(9) | NO | PRI | NULL |
| NOMBRE | varchar(25) | NO | | NULL |
| APELLIDO1 | varchar(25) | NO | | NULL |
| APELLIDO2 | varchar(25) | NO | | NULL |
| LOCALIDAD | varchar(25) | YES | | PLASENCIA |
| DIRECCION | varchar(25) | YES | | NULL |

Vamos a insertar una fila con los valores siguientes:

**DNI= 11111111**

**NOMBRE=Pedro**

**APELLIDO1=Perez**

**APELLIDO2=Lopez**

**Dejaremos el valor por defecto al campo LOCALIDAD y dejaremos en blanco el campo DIRECCIÓN**

# Ejemplo de INSERT

Observa que los valores de los campos de tipo CHAR y VARCHAR se escriben entre comillas simples.

**1ª OPCIÓN DE SINTAXIS, no especificamos la lista de campos y damos el valor DEFAULT y NULL a los campos LOCALIDAD Y DIRECCIÓN:**

```
INSERT INTO CLIENTES VALUES
('11111111','Pedro','Perez','Lopez',DEFAULT,NULL);
```

| DNI | NOMBRE | APELLIDO1 | APELLIDO2 | LOCALIDAD | DIRECCION |
|---|---|---|---|---|---|
| 11111111 | Pedro | Perez | Lopez | PLASENCIA | NULL |

**2ª OPCIÓN DE SINTAXIS: especificamos sólo los nombres de campos para los que le damos un valor concreto y los que tienen valor por defecto o no queremos rellenar no los ponemos (en LOCALIDAD se guardará PLASENCIA y en DIRECCIÓN el valor NULL**

```
INSERT INTO clientes (dni,nombre,apellido1,apellido2)
VALUES('11111111','Pedro','Gutiérrez', 'Crespo');
```

| DNI | NOMBRE | APELLIDO1 | APELLIDO2 | LOCALIDAD | DIRECCION |
|---|---|---|---|---|---|
| 11111111 | Pedro | Gutiérrez | Crespo | PLASENCIA | NULL |

# ACTUALIZACIÓN DE DATOS. SENTENCIA UPDATE

La modificación de los datos que están almacenados en los registros se hace con la instrucción **UPDATE.**
Su sintaxis es:

**UPDATE tabla**

**SET columna1=valor1 [,columna2=valor2*...*]**

**[WHERE condición]**

- Se modifican los valores para las columnas indicadas en el apartado SET con los valores indicados después del signo igual.
- La cláusula WHERE permite especificar qué registros serán modificados (si no incluimos la cláusula where, se modificarán el valor de la columna en todos los registros de la tabla)

# Ejemplo con UPDATE

En la misma tabla de antes, cambiamos el valor del campo provincia. En aquellas filas donde su valor sea Cáceres, lo cambiaremos por CC.

Preparamos la tabla:

En primer lugar, añado a la tabla el campo PROVINCIA:

```
ALTER TABLE CLIENTES
ADD COLUMN PROVINCIA VARCHAR(15);
```

Añado una nueva fila:

```
INSERT INTO CLIENTES (DNI,NOMBRE,APELLIDO1,APELLIDO2,
    PROVINCIA)
VALUES('11111112','María','Martínez', 'López','CACERES');
```

| DNI | NOMBRE | APELLIDO1 | APELLIDO2 | LOCALIDAD | DIRECCION | PROVINCIA |
|---|---|---|---|---|---|---|
| 11111111 | Pedro | Gutiérrez | Crespo | PLASENCIA | NULL | NULL |
| 11111112 | María | Martínez | López | PLASENCIA | NULL | CACERES |

```
UPDATE clientes SET provincia='CC'
```

# Ejemplo con UPDATE

Con la sentencia UPDATE, realizaré las modificaciones:

**UPDATE CLIENTES  SET PROVINCIA='CC'**

**WHERE PROVINCIA='Cáceres';**

actualiza la provincia de los clientes de Cáceres para que aparezca como CC.

| DNI | NOMBRE | APELLIDO1 | APELLIDO2 | LOCALIDAD | DIRECCION | PROVINCIA |
|---|---|---|---|---|---|---|
| 11111111 | Pedro | Gutiérrez | Crespo | PLASENCIA | NULL | NULL |
| 11111112 | María | Martínez | López | PLASENCIA | NULL | CC |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

En la condición (where) se pueden utilizar cualquiera de los siguientes operadores de comparación:

| Operador | Significado |
|---|---|
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |
| = | Igual |
| <> | Distinto |
| != | Distinto |

Y se pueden combinar varias condiciones utilizando los operadores lógicos siguientes

| Operador | Significado |
|---|---|
| **AND** | Devuelve verdadero si las expresiones a su izquierda y derecha son las dos verdaderas |
| **OR** | Devuelve verdadero si cualquiera de las dos expresiones a izquierda y derecha del OR, son verdaderas |
| **NOT** | Invierte la lógica de la expresión que está a su derecha. Si era verdadera, mediante NOT pasa a ser falso. |

# BORRAR REGISTROS (DELETE)

Para borrar registros se utiliza la instrucción DELETE:

**DELETE FROM tabla [WHERE condición]**

- Si no incluimos una cláusula WHERE se borrarán todos los registros de la tabla.
- El borrado de un registro no puede provocar fallos de integridad en la BD.
- La opción de integridad ON DELETE CASCADE hace que no sólo se borren los registros indicados en el SELECT, sino todos los relacionados.

# EJEMPLOS CON DELETE

Vamos a borrar los registros de los clientes que son de Plasencia.

```
DELETE FROM clientes
WHERE localidad='Plasencia';
```

| DNI | NOMBRE | APELLIDO1 | APELLIDO2 | LOCALIDAD | DIRECCION | PROVINCIA |
|---|---|---|---|---|---|---|
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |