Tutorial SQL básico

El *Structured Query Language*, (Lenguaje Estructurado de búsqueda) el un lenguaje estándar utilizado por los sistemas de gestión de bases de datos para realizar operaciones sobre los datos. Está formado por órdenes, cláusulas, operadores y funciones de agregados que se combinan para crear, extraer, manipular y actualizar las bases de datos. El lenguaje tiene dos partes bien diferenciadas el DDL (*Data Definition Language*), enfocado a la creación de las bases de datos, tablas, etc., y el DML (*Data Manipulation Language*), dedicado al trabajo con los datos ya creados. Este tutorial se dedicará sólo al DML.

1. Selección de registros

La recuperación de datos se hace con la sentencia `SELECT`. Su formato básico es el siguiente:

```
SELECT { * | listaColumnas}
        FROM tabla
       WHERE condiciónSelección
       ORDER BY listaColumnas [{ASC | DESC}]
```

En los ejemplos del tutorial se utilizarán las siguientes tablas de ejemplo[1]:

Clientes					
IdCliente	**Nombre**	**Apellidos**	**Ciudad**	**Provincia**	**CP**
10101	Juan	Esteban Monterías	Alicante	Alicante	03005
10298	Ana	Jiménez de la Calle	Madrid	Madrid	28034
10299	Benito	Noriega Pérez	Alicante	Alicante	03008
10315	María	Ramírez Salle	Carchelejo	Jaen	23192
10325	Miriam	Ruíz Bermudez	Piedrahita	Avila	05500
10329	Luis	Sanjosé de María	Salamanca	Salamanca	37008
10330	Juan José	Pons Gómez	Guadalajara	Guadalajara	19004
10338	María José	Herman Villa	Medina del Campo	Burgos	47400
10339	Dolores	Fuertes Abril	Mataró	Barcelona	08302
10408	Vicente	Clavo González	Barcelona	Barcelona	08007
10410	Jaime	Roman Pino	Toledo	Toledo	45001
10413	Esteban	Robles Montes	Madrid	Madrid	28027
10419	Gonzalo	Riofrío Sepúlveda	Móstoles	Madrid	28931
10429	Jimena	Silla Valencia	Illescas	Toledo	45200
10438	José	Casas de Juán	Sevilla	Sevilla	41004
10439	Bernardina	Lamadrid Cobos	Dos Hermanas	Sevilla	41701
10449	Bartolo	Tomé Sánchez	Leganés	Madrid	28912

Pedidos						
IdPedido	**IdCliente**	**Fecha**	**Producto**	**Cantidad**	**Precio**	**Pagado**
1	10101	30/12/2002	7	3	14,75	Sí
2	10101	02/01/2003	8	1	16	No
3	10101	01/07/2002	19	4	125	Sí
4	10101	30/06/2002	3	1	58	Sí
5	10101	18/08/2002	5	1	18,3	No
6	10101	08/03/2003	18	2	88,7	Sí
7	10298	01/04/2003	11	1	12,5	Sí
8	10298	01/12/2002	4	1	22	No
9	10298	19/09/2002	8	2	29	No
10	10298	18/03/2003	10	1	22,38	Sí

[1] Un ejemplo de la base de datos en formato Access se puede encontrar en el archivo ejemplo.mdb del campus virtual o de www.colimbo.net.

Pedidos						
IdPedido	**IdCliente**	**Fecha**	**Producto**	**Cantidad**	**Precio**	**Pagado**
11	10298	01/07/2002	15	1	33	Sí
12	10299	18/01/2003	6	1	38	Sí
13	10299	06/07/2002	13	1	1250	No
14	10315	02/02/2003	3	1	8	Sí
15	10330	01/01/2003	8	4	28	Sí
16	10330	30/06/2002	10	1	28	No
17	10330	19/04/2003	12	1	16,75	Sí
18	10339	27/07/2002	14	1	4,5	Sí
19	10410	28/10/2002	18	1	89,22	Sí
20	10410	30/01/2003	9	1	192,5	Sí

Productos	
IdProducto	**Producto**
1	Almohada
2	Bicicleta
3	Canoa
4	Casco
5	Chubasquero
6	Colchón inflable
7	Hoola Hoop
8	Linterna
9	Monociclo
10	Navaja
11	Orejeras
12	Pala
13	Paracaidas
14	Paraguas
15	Patines
16	Piolet
17	Raquetas de nieve
18	Saco de dormir
19	Salvavidas
20	Tienda

1.1. SELECT básico

La sentencia de recuperación básica es

```
SELECT {*| listaColumnas} FROM tabla
```

tabla es alguna de las tablas de la base de datos. La lista columnas será una o más columnas de la *tabla* separadas por comas El carácter * se utilizará para recuperar todos los campos.

```
SELECT * FROM Clientes
```

Recupera todas las columnas de la tabla clientes

IdCliente	Nombre	Apellidos	Ciudad	Provincia	CP
10101	Juan	Esteban Monterías	Alicante	Alicante	03005
10298	Ana	Jiménez de la Calle	Madrid	Madrid	28034
10299	Benito	Noriega Pérez	Alicante	Alicante	03008
10315	María	Ramírez Salle	Carchelejo	Jaen	23192
10325	Miriam	Ruíz Bermudez	Piedrahita	Avila	05500
10329	Luis	Sanjosé de María	Salamanca	Salamanca	37008
10330	Juan José	Pons Gómez	Guadalajara	Guadalajara	19004
10338	María José	Herman Villa	Medina del Campo	Burgos	47400
10339	Dolores	Fuertes Abril	Mataró	Barcelona	08302
10408	Vicente	Clavo González	Barcelona	Barcelona	08007

IdCliente	Nombre	Apellidos	Ciudad	Provincia	CP
10410	Jaime	Roman Pino	Toledo	Toledo	45001
10413	Esteban	Robles Montes	Madrid	Madrid	28027
10419	Gonzalo	Riofrío Sepúlveda	Móstoles	Madrid	28931
10429	Jimena	Silla Valencia	Illescas	Toledo	45200
10438	José	Casas de Juán	Sevilla	Sevilla	41004
10439	Bernardina	Lamadrid Cobos	Dos Hermanas	Sevilla	41701
10449	Bartolo	Tomé Sánchez	Leganés	Madrid	28912

```
SELECT Nombre, Apellidos, Ciudad FROM Clientes
```
Recupera las columnas Nombre, Apellidos y Ciudad de la tabla Clientes

Nombre	Apellidos	Ciudad
Juan	Esteban Monterías	Alicante
Ana	Jiménez de la Calle	Madrid
Benito	Noriega Pérez	Alicante
María	Ramiréz Salle	Carchelejo
Miriam	Ruíz Bermudez	Piedrahita
Luis	Sanjosé de María	Salamanca
Juan José	Pons Gómez	Guadalajara
María José	Herman Villa	Medina del Campo
Dolores	Fuertes Abril	Mataró
Vicente	Clavo González	Barcelona
Jaime	Roman Pino	Toledo
Esteban	Robles Montes	Madrid
Gonzalo	Riofrío Sepúlveda	Móstoles
Jimena	Silla Valencia	Illescas
José	Casas de Juán	Sevilla
Bernardina	Lamadrid Cobos	Dos Hermanas
Bartolo	Tomé Sánchez	Leganés

Filtrado de filas

La cláusula `WHERE` se utiliza para filtrar las columnas seleccionadas. Su formato básico es `WHERE` *`condiciónSelección`*, donde *`condiciónSeleccion`* está formado por uno o mas test unidos mediante los operadores lógicos `OR`, `AND` o `NOT`.

Test de comparación

Utiliza los operadores de relación (=, <>, <, >, <=, >=) para comparar dos expresiones.

```
SELECT Apellidos, nombre, Ciudad  FROM Clientes Where Ciudad='Madrid'
```
Recupera las columnas Apellido, Fecha y Ciudad de la tabla Clientes para aquellos clientes cuya ciudad sea Madrid

Apellidos	Nombre	Ciudad
Jiménez de la Calle	Ana	Madrid
Robles Montes	Esteban	Madrid

```
SELECT Producto, Fecha, Precio FROM Pedidos WHERE Precio > 100
```
Recupera las columnas Producto, Fecha y Precio de la tabla Pedidos para los productos cuyo precio sea mayor que 100 euros.

Producto	Fecha	Precio
19	01/07/2002 0:00:00	125
13	06/07/2002 0:00:00	1250
9	30/01/2003 0:00:00	192,5

```
SELECT IdPedido, Producto, Fecha, Precio FROM Pedidos WHERE Fecha >=
#19/03/2003#
```

Recupera las columnas IdPedido, Producto, Fecha y Precio de la tabla Pedidos para los pedidos cuya fecha sea posterior o igual al 19 de marzo de 2003 (Nota: el formato de fechas puede variar dependiendo del gestor de bases de datos utilizado, en Access , los literales de fecha se separan por el carácter # mientras que en SQL Express se utilizan las comillas simples como separador).

IdPedido	Producto	Fecha	Precio
7	11	01/04/2003 0:00:00	12,5
17	12	19/04/2003 0:00:00	16,75

Test de rango
Examina si el valor de una expresión se encuentra dentro de un rango determinado:

```
expresión [NOT] BETWEEN expr1 AND expr2
```

```
SELECT IdCliente, Apellidos,Nombre FROM Clientes
        WHERE IdCliente BETWEEN 10200 AND 10300
```
Recupera los clientes cuyo identificador sea mayor o igual que 10200 y menor o igual que 10300

IdCliente	Apellidos	Nombre
10298	Jiménez de la Calle	Ana
10299	Noriega Pérez	Benito

Test de pertenencia a un conjunto
Examina si la expresión es alguno de los valores incluidos en la lista de valores.

```
expresión IN (listaValores)
```

dónde `listaValores` está formada por uno o más valores separados por comas.

```
SELECT  Apellidos,Nombre, Ciudad FROM Clientes
                WHERE Ciudad IN ('Madrid','Barcelona')
```
Recupera los clientes de Madrid o Barcelona

Apellidos	Nombre	Ciudad
Jiménez de la Calle	Ana	Madrid
Clavo González	Vicente	Barcelona
Robles Montes	Esteban	Madrid

La lista de valores también puede recoger una consulta `SELECT` SQL para realizar subconsultas.

```
SELECT  Apellidos,Nombre, Ciudad FROM Clientes
          WHERE IdCliente IN
                (SELECT IdCliente FROM Pedidos  WHERE Precio > 100)
```
Recupera el apellido, nombre y ciudad de los clientes que tuvieran pedidos con un precio superior a 100 euros.

Apellidos	Nombre	Ciudad
Esteban Monterías	Juan	Alicante
Noriega Pérez	Benito	Alicante
Roman Pino	Jaime	Toledo

Test de valor nulo
Examina las filas para averiguar si algún campo tiene valor nulo.

```
columna IS [NOT] NULL
```

```
SELECT * FROM Pedidos WHERE Precio IS NULL
```
Recupera todos los campos de la tabla Pedidos de aquellas filas que no tengan valor en el campo Precio

IdPedido	IdCliente	Fecha	Producto	Cantidad	Precio

Test de correspondencia con patrón (comodines)
Examina el valor de una columna y lo compara con un patrón. Un patrón es una expresión de cadena que puede contener comodines. Los comodines utilizados en ANSI SQL son el símbolo de porcentaje (`%`) que,

situado al comienzo o final de una cadena, sustituye a cualquier número de caracteres, el guión bajo (_) que sustituye a un único carácter (Nota: el SQL de Access utiliza como comodines el asterisco (*) para sustituir a cualquier carácter y la interrogación (?) para sustituir a un único carácter. El formato del test de correspondencia es:

columna [NOT] LIKE *patrón*

```
SELECT Apellidos, Nombre, Ciudad FROM Clientes WHERE Ciudad LIKE 'M%'
```

Recupera las filas de la tabla Clientes cuyo campo ciudad comience por la letra M

Apellidos	Nombre	Ciudad
Jiménez de la Calle	Ana	Madrid
Herman Villa	María José	Medina del Campo
Fuertes Abril	Dolores	Mataró
Robles Montes	Esteban	Madrid
Riofrío Sepúlveda	Gonzalo	Móstoles

```
SELECT Apellidos, Nombre, Ciudad FROM Clientes WHERE Ciudad LIKE '%a'
```

Recupera las filas de la tabla Clientes cuyo campo ciudad termine por la letra a

Apellidos	Nombre	Ciudad
Ruíz Bermudez	Miriam	Piedrahita
Sanjosé de María	Luis	Salamanca
Pons Gómez	Juan José	Guadalajara
Clavo González	Vicente	Barcelona
Casas de Juán	José	Sevilla

```
SELECT Apellidos, Nombre, Ciudad FROM Clientes WHERE Ciudad LIKE 'Ma_ _ _ _'
```

Recupera las filas de la tabla Clientes cuyo campo ciudad comience por la cadena Ma y tenga cuatro letras más.

Apellidos	Nombre	Ciudad
Jiménez de la Calle	Ana	Madrid
Fuertes Abril	Dolores	Mataró
Robles Montes	Esteban	Madrid

Ordenación de filas

Para ordenar los resultados de la consulta según algún criterio determinado se utiliza la cláusula ORDER BY.

ORDER BY *columna* [{ASC | DESC}]…

De forma predeterminada el criterio de ordenación es ascendente, aunque es posible cambiarlo mediante la palabra DESC.

Es posible ordenar por varias columnas separándolas por comas, indicando por cada una si la ordenación se hará de forma ascendente o descendente.

```
SELECT * FROM Clientes WHERE Ciudad LIKE 'M%' ORDER BY Apellidos
```

Recupera las filas de la tabla Clientes cuyo campo ciudad empiece por la letra M y ordenando el resultado de forma ascendente por la columna Apellidos

IdCliente	Nombre	Apellido	Ciudad	Provincia
10339	Dolores	Fuertes Abril	Mataró	Barcelona
10338	María José	Herman Villa	Medina del Campo	Burgos
10298	Ana	Jiménez de la Calle	Madrid	Madrid
10419	Gonzalo	Riofrío Sepúlveda	Móstoles	Madrid
10413	Esteban	Robles Montes	Madrid	Madrid

```
SELECT * FROM Clientes WHERE Ciudad LIKE 'M%' ORDER BY Ciudad, Apellidos
```

Recupera las filas de la tabla Clientes cuyo campo ciudad empiece por la letra M y ordenando el resultado de forma ascendente por las columnas Ciudad y Apellido

IdCliente	Nombre	Apellido	Ciudad	Provincia
10298	Ana	Jiménez de la Calle	Madrid	Madrid

IdCliente	Nombre	Apellido	Ciudad	Provincia
10413	Esteban	Robles Montes	Madrid	Madrid
10339	Dolores	Fuertes Abril	Mataró	Barcelona
10338	María José	Herman Villa	Medina del Campo	Burgos
10419	Gonzalo	Riofrío Sepúlveda	Móstoles	Madrid

```
SELECT * FROM Pedidos
    WHERE IdCliente < 10300
    ORDER BY IdCliente ASC, Precio DESC
```

Recupera las filas de la tabla Pedidos cuyo campo IdCliente sea menor que 10300 y ordenando los resultados de forma ascendente por el identificador de cliente y descendente por el campo Precio

IdPedido	IdCliente	Fecha	Producto	Cantidad	Precio	Pagado
3	10101	01/07/2002 0:00:00	19	4	125	True
6	10101	08/03/2003 0:00:00	18	2	88,7	True
4	10101	30/06/2002 0:00:00	3	1	58	True
5	10101	18/08/2002 0:00:00	5	1	18,3	False
2	10101	02/01/2003 0:00:00	8	1	16	False
1	10101	30/12/2002 0:00:00	7	3	14,75	True
11	10298	01/07/2002 0:00:00	15	1	33	True
9	10298	19/09/2002 0:00:00	8	2	29	False
10	10298	18/03/2003 0:00:00	10	1	22,38	True
8	10298	01/12/2002 0:00:00	4	1	22	False
7	10298	01/04/2003 0:00:00	11	1	12,5	True
13	10299	06/07/2002 0:00:00	13	1	1250	False
12	10299	18/01/2003 0:00:00	6	1	38	True

1.2. Uso de expresiones en las columnas

En la lista de columnas también se pueden incluir expresiones. En estos casos el resultado de la columna aparecerá el resultado de la expresión.

```
SELECT IdCliente, Fecha, Producto, Cantidad*Precio FROM Pedidos
    WHERE IdCliente = 10101
```

Recupera los campos IdCliente, Fecha, Producto y el valor de multiplicar la columna Cantidad por la columna Precio de la tabla de Pedidos del cliente 10101

IdCliente	Fecha	Producto	Expr1
10101	30/12/2002 0:00:00	7	44,25
10101	02/01/2003 0:00:00	8	16
10101	01/07/2002 0:00:00	19	500
10101	30/06/2002 0:00:00	3	58
10101	18/08/2002 0:00:00	5	18,3
10101	08/03/2003 0:00:00	18	177,4

La cabecera de la columna resultante lo genera el gestor de bases de datos, aunque es posible dar un nombre mediante la cláusula `AS`.

```
nombreColumna AS cabeceraColumna
```

```
SELECT IdCliente, Fecha, Producto, Cantidad*Precio AS Total FROM Pedidos
            WHERE IdCliente = 10101
```

IdCliente	Fecha	Producto	Total
10101	30/12/2002 0:00:00	7	44,25
10101	02/01/2003 0:00:00	8	16
10101	01/07/2002 0:00:00	19	500
10101	30/06/2002 0:00:00	3	58
10101	18/08/2002 0:00:00	5	18,3
10101	08/03/2003 0:00:00	18	177,4

Esta cláusula también se puede utilizar para cambiar la cabecera de columna de cualquier campo.

```
SELECT IdCliente AS [Identificador cliente], Fecha, Producto,
         Cantidad*Precio AS Total FROM Pedidos
         WHERE IdCliente = 10101
```

Identificador cliente	Fecha	Producto	Total
10101	30/12/2002 0:00:00	7	44,25
10101	02/01/2003 0:00:00	8	16
10101	01/07/2002 0:00:00	19	500
10101	30/06/2002 0:00:00	3	58
10101	18/08/2002 0:00:00	5	18,3
10101	08/03/2003 0:00:00	18	177,4

1.3. Funciones de agregado

En los nombres de columnas también se pueden utilizar funciones de agregado que realizan un cálculo sobre el total de las filas seleccionadas. Las funciones que se pueden utilizar son:

Función	Descripción
`COUNT(`*nombreColumna*`)`	Cuenta el número de valores distintos de nulo que hay en la columna
`COUNT(*)`	Cuenta el numero de filas
`SUM(`*expresión*`)`	Realiza la suma de todas los valores devueltos por la expresión
`AVG(`*expresión*`)`	Realiza la media de todos los valores devueltos por la expresión
`MAX(`*expresión*`)`	Devuelve el valor máximo de la expresión
`MIN(`*expresión*`)`	Devuelve el valor mínimo de la expresión

```
SELECT COUNT(Precio) FROM Pedidos
```
Devuelve el número de filas cuyo campo precio es distinto de nulo

Expr1
20

```
SELECT COUNT(*) FROM Pedidos
```
Devuelve el número de filas total de la tabla Pedidos

Expr1
20

```
SELECT SUM(Precio*Cantidad) FROM Pedidos  WHERE IdCliente = 10101
```
*Devuelve la suma del Precio * Unidad* para el cliente 10101

Expr1
813,95

```
SELECT AVG(Precio) FROM Pedidos
```
Devuelve el precio medio de la tabla Pedidos

Expr1
104,73

```
SELECT MAX(Fecha) FROM Pedidos
```
Devuelve la fecha del último pedido

Expr1
19/04/2003 0:00:00

1.4. Unión de tablas

Las bases de datos relacionales se caracterizan, precisamente, por su capacidad para relacionar sus tablas a partir de campos comunes (en las tablas de ejemplo el campo común sería la columna IdCliente). Partiendo de esto, sería posible obtener información a partir el contenido de diversas tablas y relacionándolas por su campo común. Para relacionar varias tablas sería necesario especificar en la cláusula `FROM` las tablas que se van a relacionar y especificar la relación en la cláusula `WHERE`.

```
SELECT listaColumnas
   FROM tabla1,tabla2,…
   WHERE tabla1.campoRelación = tabla2.campoRelación…
```

Obsérvese que en la relación se han especificado las columnas con el cualificador *tabla1.campoRelación*. Esto es necesario si el nombre del campo que relaciona las dos tablas tiene el mismo nombre en ambas. El cualificador también será necesario si en la *listaColumnas* aparece algún campo con un nombre común.

```
SELECT Clientes.IdCliente, Apellidos, Nombre, Ciudad,
           IdPedido, Fecha, Productos.Producto
    FROM Clientes, Pedidos, Productos
      WHERE Clientes.IdCliente = Pedidos.IdCliente
      AND Pedidos.Producto = Productos.IdProducto
    ORDER BY Fecha, Pedidos.IdCliente
```

Devuelve las columnas IdCliente, Apellidos, Nombre, y Ciudad de la tabla Clientes; las columnas IdPedido, Fecha de la tabla Pedidos, y la columna Producto de la tabla productos sacando la información de las filas en las que coincida el campo IdCliente y el producto del pedido coincida con el identificador de producto de la tabla pedidos, ordenando el resultado por las columnas Fecha e IdCliente de la tabla Pedidos.

IdCliente	Apellido	Nombre	Ciudad	IdPedido	Fecha	Producto
10101	Esteban Monterías	Juan	Alicante	4	30/06/2002 0:00:00	Canoa
10330	Pons Gómez	Juan José	Guadalajara	16	30/06/2002 0:00:00	Navaja
10101	Esteban Monterías	Juan	Alicante	3	01/07/2002 0:00:00	Salvavidas
10298	Jiménez de la Calle	Ana	Madrid	11	01/07/2002 0:00:00	Patines
10299	Noriega Pérez	Benito	Alicante	13	06/07/2002 0:00:00	Paracaidas
10339	Fuertes Abril	Dolores	Mataró	18	27/07/2002 0:00:00	Paraguas
10101	Esteban Monterías	Juan	Alicante	5	18/08/2002 0:00:00	Chubasquero
10298	Jiménez de la Calle	Ana	Madrid	9	19/09/2002 0:00:00	Linterna
10410	Roman Pino	Jaime	Toledo	19	28/10/2002 0:00:00	Saco de dormir
10298	Jiménez de la Calle	Ana	Madrid	8	01/12/2002 0:00:00	Casco
10101	Esteban Monterías	Juan	Alicante	1	30/12/2002 0:00:00	Hoola Hoop
10330	Pons Gómez	Juan José	Guadalajara	15	01/01/2003 0:00:00	Linterna
10101	Esteban Monterías	Juan	Alicante	2	02/01/2003 0:00:00	Linterna
10299	Noriega Pérez	Benito	Alicante	12	18/01/2003 0:00:00	Colchón inflable
10410	Roman Pino	Jaime	Toledo	20	30/01/2003 0:00:00	Monociclo
10315	Ramiréz Salle	María	Carchelejo	14	02/02/2003 0:00:00	Canoa
10101	Esteban Monterías	Juan	Alicante	6	08/03/2003 0:00:00	Saco de dormir
10298	Jiménez de la Calle	Ana	Madrid	10	18/03/2003 0:00:00	Navaja
10298	Jiménez de la Calle	Ana	Madrid	7	01/04/2003 0:00:00	Orejeras
10330	Pons Gómez	Juan José	Guadalajara	17	19/04/2003 0:00:00	Pala

2. Actualización de datos

2.1. Modificar filas

La sentencia `UPDATE` modifica los registros de una tabla, asignando a una o más columnas un valor distinto. La sentencia actuará en todas las filas a no ser que se utilice una cláusula `WHERE`.

```
UPDATE nombreTabla SET (nombreColumna = expresión,…) [WHERE condiciónSelección]
```

El tipo de dato de la expresión debe coincidir con el tipo de dato de la columna.

```
UPDATE Pedidos SET Precio = Precio * 1.01
```

Actualiza todas las filas de la tabla Pedidos, aumentando el Precio en un 10%.

```
UPDATE Clientes SET Provincia='Madrid',Ciudad='Alcobendas'
       WHERE IdCliente = 10101
```

Actualiza los datos del cliente 10101, modificando el valor de sus columnas Provincia y Ciudad

2.2. Añadir filas

La sentencia `INSERT` se utiliza para añadir registros en una tabla. Para ello, habrá que proporcionar el nombre de la tabla mediante la cláusula `INTO`, la lista de campos entre paréntesis y la lista de valores en la cláusula `VALUES`.

```
INSERT INTO nombreTabla (listaColumnas) VALUES (listaExpresiones)
```

La *listaColumnas* puede contener todos o sólo algunos de los nombre de las columnas de la tabla. Si la tabla tiene definidos campos obligatorios (cómo por ejemplo, una clave primaria), será necesario incluirlos. La lista de expresiones contendrá una serie de expresiones separados por comas con los valores que tendrán las columnas de la nueva fila. La asignación entre la columna y los valores se hace por posición y el tipo de dato deberá ser el mismo.

```
INSERT INTO Clientes (IdCliente, Nombre, Apellidos) VALUES
(20202,'John','Smith')
```

Inserta una nueva fila en la tabla Clientes con el IdCliente 20202, el Nombre 'John' y el Apellidos 'Smith'. El resto de campos los dejará vacíos.

2.3. Eliminar filas

La instrucción `DELETE` se utiliza para eliminar filas de una tabla. La cláusula `WHERE` es opcional, aunque normalmente se utilizará ya que, en caso contrario, se eliminarían todas las filas.

```
DELETE FROM nombreTabla [WHERE condiciónSelección]
```

```
DELETE FROM Pedidos WHERE IdCliente = 10101
```

Elimina todos los pedidos del cliente 10101