

IT Bootcamp

Introducción a SQL

DigitalHouse>

Índice

1. **SQL (Structured Query Language)**
2. **Sentencias DML**
3. **Consultas y cláusulas**
4. **Funciones de Agregación**

1 | SQL

¿Qué es SQL?

SQL (Structured Query Language - Lenguaje Estructurado de Consulta)

es un lenguaje estándar utilizado para crear, modificar, mantener y consultar una base de datos relacional.

Es un estándar ANSI, esto implica que conociendo este lenguaje se puede manipular cualquier base de datos de cualquier fabricante (claramente, cada fabricante tiene sus particularidades y extensiones, pero la sintaxis es similar en todos los casos).

DDL y DML

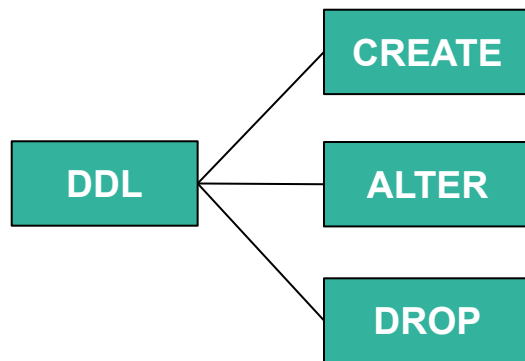
SQL se puede dividir en dos lenguajes: DDL y DML. Cada parte tiene sentencias/comandos para realizar operaciones de distinta naturaleza. Estos dos grupos son:

- **DDL (Data Definition Language)**
- **DML (Data Manipulation Language)**

DDL (Data Definition Language)

DDL (Data Definition Language) nos permitirá crear nuevas bases de datos, modificarlas y eliminarlas. Estos comandos afectan a tablas, campos e índices.

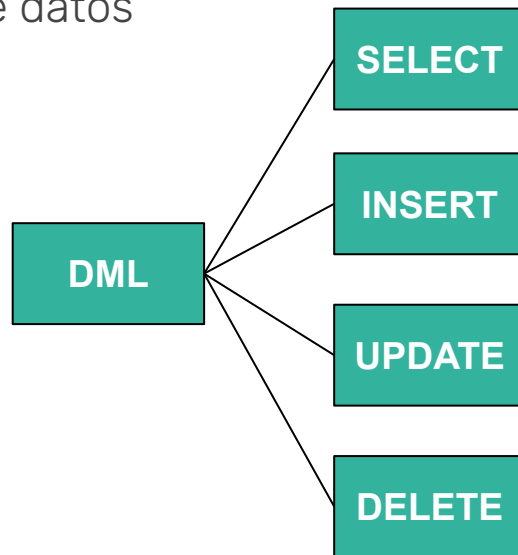
- **CREATE** Crea nuevas tablas, campos e índices.
- **ALTER** Modifica las tablas
- **DROP** Elimina tablas e índices.



DDL (Data Definition Language)

DML (Data Manipulation Language): Nos permite manipular. consultar, ordenar, filtrar, etc. los datos existentes en una base de datos.

- **SELECT** Consulta registros en la base de datos
- **INSERT** Inserta un nuevo registro
- **UPDATE** Modifica valores de un registro
- **DELETE** Elimina registros



2 | Sentencias DML

SELECT

Se utiliza esta sentencia para recuperar/mostrar datos de una tabla. El formato general al usar este comando es:

```
SQL  SELECT <columnas separadas por comas> FROM <tabla>;
```

Ejemplos SELECT

Teniendo en cuenta que se tiene una tabla Cliente:

Cliente (*id_cliente, dni, nombre, apellido, domicilio*);

SQL

```
SELECT * FROM Cliente;
```

//Recupera todas las columnas de la tabla; sin establecer cláusula WHERE, se obtienen todos los registros de la tabla Cliente.

```
SELECT nombre FROM Cliente;
```

//Recupera todos los nombres de la tabla.

```
SELECT apellido, domicilio FROM Cliente;
```

//Recupera todos los apellidos con su domicilio de la tabla.

SELECT y WHERE

WHERE: Se utiliza esta cláusula para proponer la condición específica que debe cumplir el dato que se pretende traer.

SQL

```
SELECT apellido, domicilio FROM Cliente WHERE nombre='Paolo';  
//De esta forma, estaría mostrando el apellido y domicilio de todos los  
clientes que tengan el nombre Paolo.
```

INSERT

Se utiliza esta sentencia para incorporar datos a una tabla. El formato general al usar este comando es:

```
SQL INSERT INTO <tabla> VALUES (dato 1, dato 2, dato n);
```

Ejemplos INSERT

Teniendo en cuenta que la tabla Cliente tiene la siguiente estructura:

Cliente (*id_cliente* INTEGER, *dni* VARCHAR, *nombre* VARCHAR, *apellido* VARCHAR, *domicilio* VARCHAR);

	//Para insertar filas en la tabla:
	INSERT INTO Cliente VALUES (0001, '00111000', 'Andriy', 'Shevchenko', 'Ucrania 1613');
SQL	INSERT INTO Cliente VALUES (0002, '00999111', 'Andrea', 'Pirlo', 'Italia 2006');
	INSERT INTO Cliente VALUES (0003, '00111222', 'Paolo', 'Maldini', 'Milan 2007');

UPDATE

Se utiliza esta sentencia para modificar datos en una tabla. El formato general al usar este comando es:

SQL

```
UPDATE <tabla> SET datoaModificar='NuevoValor'  
WHERE dato='Valor';
```

Ejemplo UPDATE

Continuando con los ejemplos anteriores, se estaría modificando el domicilio de Andrea Pirlo.

```
SQL  UPDATE Cliente SET domicilio='Turín 2012' WHERE dni='00999111';
```



Si no colocamos la cláusula WHERE, se modificarán todos los domicilios de la tabla Cliente.

DELETE

Se utiliza esta sentencia para eliminar registros en una tabla. El formato general al usar este comando es:

```
SQL DELETE FROM <tabla> WHERE dato='Valor';
```


Ejemplo DELETE

Continuando con los ejemplos anteriores, se estaría eliminando el registro completo del cliente Andriy Shevchenko.

```
SQL DELETE FROM Cliente WHERE id_cliente=0001;
```




Si no colocamos la cláusula WHERE, se eliminarán todos los registros de la tabla Cliente.

3 | Consultas y Cláusulas

ORDER BY

Esta cláusula permite ordenar los resultados a partir de uno o más campos.

```
SQL  SELECT campo1, campo2, campo3  
      FROM Tabla  
      ORDER BY campo2;
```



Por ejemplo, se puede ordenar por el campo2 (por default el ordenamiento es ascendente).

ORDER BY: Ascendente o Descendente

Esta cláusula permite ordenar de forma ascendente algunos campos y otros campos en forma descendente. Esto dependerá de cómo se requiere mostrar u obtener los datos.

SQL

```
SELECT campo1, campo2, campo3  
FROM Tabla  
ORDER BY campo1 DESC;
```




Por ejemplo, se indica explícitamente de qué forma ordenar.

LIMIT

La cláusula LIMIT se usa para restringir los registros que se retornan en una consulta SELECT. Se puede establecer un número "N" para limitar a esa cantidad de registros.

```
SQL  SELECT campo1, campo2, campo3
      FROM Tabla
      ORDER BY campo1 DESC
      LIMIT 5;
```



Por ejemplo, se indica que en esta consulta se mostrarán solamente 5 registros.

DISTINCT

Se utiliza **DISTINCT** cuando se desea omitir registros que contienen datos duplicados en los campos seleccionados.

Se usa en la instrucción **SELECT** para recuperar valores únicos de una tabla de base de datos.

SQL

```
SELECT DISTINCT campo2  
FROM Tabla;
```

OPERADORES



Recuerda que se puede hacer uso de operadores aritméticos, lógicos y de comparación.

SQL

```
SELECT campo1, campo2 FROM Tabla  
WHERE campo1 => 0 AND campo3 < 10 ;
```

OPERADORES AND y OR

→ El operador **AND** mostrará los resultados cuando se cumplan **las 2 condiciones**.

Ejemplo: condición1 **AND** condición2

→ El operador **OR** mostrará los resultados cuando se cumpla **alguna de las 2 condiciones**.

Ejemplo: condición1 **OR** condición2

SQL

```
SELECT campo1, campo2, campo3 FROM Tabla  
WHERE campo1 = 1 AND campo2 = 'TEST';
```


LIKE y NOT LIKE

El operador **LIKE** se utiliza en una cláusula **WHERE** para buscar un patrón específico en una columna. Para negar se puede usar el **NOT LIKE**.

SQL

```
SELECT campo1, campo2, campo3  
FROM Tabla  
WHERE campo1 = 1 AND campo2 LIKE 'T%';
```

BETWEEN

Este operador permite seleccionar un **rango**. Es utilizado para establecer un **criterio** de selección dentro de la cláusula **WHERE** y permite establecer los valores que deben tener los extremos (inclusivos) de ese **rango** deseado.

SQL

```
SELECT campo1, campo2, campo3  
FROM Tabla  
WHERE campo1 = 1 AND campo2 BETWEEN '20200101' AND  
'20200131';
```

4 | Funciones de Agregación

FUNCIONES DE AGREGACIÓN

Se utilizan para visualizar cierta información agrupada y resumida, para esto utilizaremos las funciones de agregación.

- **COUNT**: devuelve el número total de filas seleccionadas por la consulta.
- **MIN**: devuelve el valor mínimo del campo que se especifique.
- **MAX**: devuelve el valor máximo del campo que se especifique.
- **SUM**: suma los valores del campo que se especifique.
- **AVG**: devuelve el valor promedio del campo que se especifique.

FUNCIONES DE AGREGACIÓN

Estas funciones se utilizan en conjunto con la sentencia SELECT.

Ejemplos de sintaxis:

```
SQL SELECT MAX(campo1) FROM Tabla;
```

```
SQL SELECT COUNT(*) FROM Tabla;
```

FUNCIONES DE AGREGACIÓN

A modo de ejemplo, teniendo una base de datos de películas se quiere obtener el título de la película que más premios ha obtenido. Podríamos ejecutar:

SQL

```
SELECT MAX(awards), title  
FROM movies;
```

ALGUNAS BUENAS PRÁCTICAS

- **Evitar** el uso del **SELECT *** (Seleccionar solo los campos que se necesiten).
- **Evitar** realizar consultas **sin filtros** o **limite** de **registros**.
- No se recomienda usar el **ORDER BY** para consultas con **grandes volúmenes de datos**.
- Usar las cláusulas adecuadas para acotar el set de datos.
- Si se utilizan varias tablas en una consulta, especificar a qué tabla pertenece cada campo.

DigitalHouse>