

SQL, o *Structured Query Language* (lenguaje de consulta estructurada), es un lenguaje para comunicarse con bases de datos. Se utiliza para seleccionar datos específicos y crear informes complejos. Hoy en día, SQL es un lenguaje de datos universal y se utiliza en prácticamente todas las tecnologías que procesan datos.

ESTRUCTURA GENERAL DE LA QUERY

- se construye secuencialmente con palabras reservadas que indican la acción a realizar y las condiciones o modificaciones aplicables. La sintaxis sigue este patrón básico:
- Palabra reservada principal: Define la acción (ej. **SELECT**, **CREATE**, **UPDATE**).
 - Cláusulas secundarias: Especifican detalles como tablas, condiciones, orden, límites, etc. (ej. **FROM**, **WHERE**, **ORDER BY**).
 - Condiciones o parámetros: Valores, columnas o reglas específicas (ej. **WHERE columna = valor**).

CATEGORÍAS Y ESTRUCTURA DE LA QUERY

1. Lenguaje de Definición de Datos (DDL)
- Propósito:** Define o modifica la estructura de objetos en la base de datos (tablas, índices, etc.).
 - Palabras reservadas principales:**
 - CREATE:** Crea un nuevo objeto (base de datos, tabla, índice).
 - ALTER:** Modifica un objeto existente.
 - DROP:** Elimina un objeto.
 - Estructura típica:**
 - CREATE** [OBJETO] [NOMBRE] [DEFINICIÓN];
 - ALTER** [OBJETO] [NOMBRE] [ACCIÓN];
 - DROP** [OBJETO] [NOMBRE];
2. Lenguaje de Manipulación de Datos (DML)
- Propósito:** Manipula los datos dentro de las tablas (insertar, actualizar, eliminar, consultar).
 - Palabras reservadas principales:**
 - SELECT:** Recupera datos.
 - INSERT:** Inserta datos.
 - UPDATE:** Actualiza datos.
 - DELETE:** Elimina datos.
 - Estructura típica:**
 - SELECT** [COLUMNAS] **FROM** [TABLA] [CLÁUSULAS];
 - INSERT INTO** [TABLA] ([COLUMNAS]) **VALUES** ([VALORES]);
 - UPDATE** [TABLA] **SET** [COLUMNAS] = [VALORES] **WHERE** [CONDICIÓN];
 - DELETE FROM** [TABLA] **WHERE** [CONDICIÓN];

COMANDOS / CLÁUSULAS

- SELECT:** Selecciona datos de una base de datos.
- FROM:** Especifica de qué tabla se extraen los datos.
- WHERE:** Filtra las filas que cumplen con una condición.
- JOIN:** Combina filas de dos o más tablas.
- AND / OR:** Combina condiciones. Todas deben cumplirse (AND) o al menos una (OR).
- LIMIT:** Limita la cantidad de filas devueltas. También usa **FETCH** o **TOP**.
- CASE:** Devuelve un valor según una condición específica.
- IS NULL:** Busca filas donde un valor es NULL.
- UPDATE:** Actualiza datos en una tabla.
- INSERT:** Crea datos; **INSERT INTO** agrega filas a una tabla.
- DELETE:** Elimina filas de una tabla.
- DROP:** Elimina una tabla, base de datos, o índice.
- GROUP BY:** Agrupa los resultados. Usa **DESC** para invertir el orden.
- HAVING:** Filtra grupos según condiciones.
- COUNT:** Devuelve el número de filas.
- SUM:** Devuelve la suma de una columna.
- AVG:** Devuelve el promedio de una columna.
- MIN:** Devuelve el valor mínimo de una columna.
- MAX:** Devuelve el valor máximo de una columna.

CREAR BASE DE DATOS

Crea una nueva base de datos llamada **mundo**:
CREATE DATABASE mundo;

CREAR TABLAS

Crear tabla con la **clave primaria**:
CREATE TABLE pais (
 id **SERIAL PRIMARY KEY**,
 nombre **VARCHAR(100)**,
 poblacion **BIGINT**,
 superficie **INT**
);

Crear tabla con la **clave foránea**:
CREATE TABLE ciudad (
 id **SERIAL PRIMARY KEY**,
 nombre **VARCHAR(100)**,
 id_pais **INT**,
 poblacion **INT**,
 clasificacion **INT**,
 FOREIGN KEY (**id_pais**) **REFERENCES** pais(**id**)
);

INSERTAR REGISTROS

Insertar registros en la tabla **pais**:
INSERT INTO pais (nombre, poblacion, superficie) VALUES
('Francia', 66600000, 640680),
('Alemania', 80700000, 357000),
('Islandia', 360000, 103000),
('Polonia', 38400000, 312696);

CONSULTAR A UNA SOLA TABLA

Recuperar todas las columnas de la tabla país:
SELECT *
FROM pais;

Recuperar las columnas id y nombre de la tabla ciudad:
SELECT id, nombre
FROM ciudad;

Recuperar los nombres de las ciudades ordenados por la columna clasificación en orden ascendente (ASC):
SELECT nombre
FROM ciudad
ORDER BY clasificación [ASC];

Recuperar los nombres de las ciudades ordenados por la columna clasificación en orden descendente (DESC):
SELECT nombre
FROM ciudad
ORDER BY clasificación DESC;

ALIAS

COLUMNAS
SELECT nombre AS nombre_ciudad
FROM ciudad;

TABLAS

SELECT pa.nombre, ci.nombre
FROM ciudad AS ci
JOIN pais AS pa
 ON ci.id_pais = pa.id;

FILTRAR RESULTADOS

OPERADORES DE COMPARACIÓN

Recuperar los nombres de las ciudades cuya clasificación sea superior a 3:
SELECT nombre
FROM ciudad
WHERE clasificacion > 3;

Recuperar los nombres de ciudades que no sean ni Berlín ni Madrid:
SELECT nombre
FROM ciudad
WHERE nombre != 'Berlín'
 AND nombre != 'Madrid';

OPERADORES DE TEXTO

Recuperar los nombres de las ciudades que empiecen por "P" o terminen por "s":
SELECT nombre
FROM ciudad
WHERE nombre LIKE 'P%'
OR nombre LIKE '%s';

Recuperar los nombres de las ciudades que empiecen por cualquier letra seguida de "adrid" (como Madrid, de España):
SELECT nombre
FROM ciudad
WHERE nombre LIKE '_adrid';

OTROS OPERADORES

Recuperar los nombres de las ciudades con poblaciones comprendidas entre 500.000 y 5 millones de habitantes:
SELECT nombre
FROM ciudad
WHERE poblacion BETWEEN 500000 AND 5000000;

Recuperar los nombres de las ciudades que tienen un valor en la clasificación:
SELECT nombre
FROM ciudad
WHERE clasificacion IS NOT NULL;

Recuperar los nombres de las ciudades situadas en países cuyo ID es 1, 4, 7 u 8:
SELECT nombre
FROM ciudad
WHERE id_pais IN (1, 4, 7, 8);

EJEMPLOS DE DATOS

PAIS				
id	nombre	poblacion	superficie	
1	Francia	66600000	640680	
2	Alemania	80700000	357000	
...	

CIUDAD				
id	nombre	id_pais	poblacion	clasificacion
1	París	1	2243000	5
2	Berlín	2	3460000	3
...

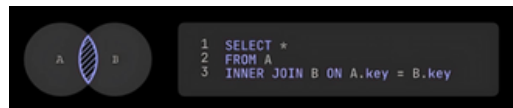
Cheatsheet SQL

CONSULTAR A VARIAS TABLAS

INNER JOIN

JOIN (o explícitamente **INNER JOIN**) devuelve las filas en las que coinciden los valores en ambas tablas.

```
SELECT ciudad.nombre, pais.nombre
FROM ciudad
[INNER] JOIN pais
ON ciudad.id_pais = pais.id;
```



CIUDAD			PAIS	
id	nombre	id_pais	id	nombre
1	París	1	1	Francia
2	Berlín	2	2	Alemania
3	Varsovia	4	3	Islandia

LEFT JOIN

LEFT JOIN devuelve todas las filas de la tabla izquierda con las filas correspondientes de la tabla derecha. Si no coincide ninguna fila de la tabla derecha, devuelve **NULL** como valores de la tabla derecha.

```
SELECT ciudad.nombre, pais.nombre
FROM ciudad
LEFT JOIN pais
ON ciudad.id_pais = pais.id;
```



CIUDAD			PAIS	
id	nombre	id_pais	id	nombre
1	París	1	1	Francia
2	Berlín	2	2	Alemania
3	Varsovia	4	NULL	NULL

RIGHT JOIN

RIGHT JOIN devuelve todas las filas de la tabla derecha con las filas correspondientes de la tabla izquierda. Si no coincide ninguna fila de la tabla izquierda, devuelve **NULL** como valores de la tabla izquierda.

```
SELECT ciudad.nombre, pais.nombre
FROM ciudad
RIGHT JOIN pais
ON ciudad.id_pais = pais.id;
```



CIUDAD			PAIS	
id	nombre	id_pais	id	nombre
1	París	1	1	Francia
2	Berlín	2	2	Alemania
NULL	NULL	NULL	3	Islandia

FULL JOIN

FULL JOIN (o explícitamente **FULL OUTER JOIN**) devuelve todas las filas de ambas tablas. Si no coincide ninguna fila de la otra tabla, devuelve valores **NULL**.

```
SELECT ciudad.nombre, pais.nombre
FROM ciudad
FULL [OUTER] JOIN pais
ON ciudad.id_pais = pais.id;
```




CIUDAD			PAIS	
id	nombre	id_pais	id	nombre
1	París	1	1	Francia
2	Berlín	2	2	Alemania
3	Varsovia	4	NULL	NULL
NULL	NULL	NULL	3	Islandia

AGREGAR Y AGRUPAR

GROUP BY **agrupa** las filas que tienen los mismos valores en columnas especificadas. Genera resúmenes (agregados) para cada combinación única de valores.

CIUDAD		
id	nombre	id_pais
1	París	1
101	Marsella	1
102	Lyon	1
2	Berlín	2
103	Hamburgo	2
104	Múnich	2
3	Varsovia	4
105	Cracovia	4



CIUDAD	
id_pais	cantidad
1	3
2	3
4	2

FUNCIONES DE AGREGADO

- avg(expr) - valor medio de las filas del grupo
- count(expr) - número de valores de las filas del grupo
- max(expr) - valor máximo del grupo
- min(expr) - valor mínimo del grupo
- sum(expr) - suma de los valores del grupo

EJEMPLOS DE CONSULTAS

Conocer el número de ciudades:

```
SELECT COUNT(*)
FROM ciudad;
```

Conocer el número de ciudades con una valoración que no sea nula:

```
SELECT COUNT(clasificacion)
FROM ciudad;
```

Conocer el número de valores distintos de los países:

```
SELECT COUNT(DISTINCT id_pais)
FROM ciudad;
```

Conocer los países con menor y mayor población:

```
SELECT MIN(poblacion), MAX(poblacion)
FROM pais;
```

Conocer la población total de las ciudades en sus respectivos países:

```
SELECT id_pais, SUM(poblacion)
FROM ciudad
GROUP BY id_pais;
```

SUBCONSULTAS

Una subconsulta es una consulta anidada en otra consulta o en otra subconsulta. Existen diferentes tipos de subconsultas.

VALOR ÚNICO

Es la subconsulta más sencilla, que devuelve exactamente una columna y una fila. Puede utilizarse con los operadores de comparación =, <, <=, > o >=. La siguiente consulta se utiliza para buscar ciudades con la misma clasificación que París:

```
SELECT nombre
FROM ciudad
WHERE clasificacion = (
    SELECT clasificacion
    FROM ciudad
    WHERE nombre = 'París'
);
```

COMANDOS PSQL ÚTILES

psql es el cliente de línea de comandos oficial de **PostgreSQL** (Motor de base de datos).

Bash

psql -U postgres -d nombre_base

- Ver bases de datos:
 - \l --(En psql "listar")
- Conectarse a una base de datos:
 - \c nombre_base
- Ver tablas
 - \dt
- Ver estructura de una tabla:
 - \d nombre_tabla

CONSEJOS PRÁCTICOS

- Evitá **SELECT *** en producción.
- Siempre usá **WHERE** en **UPDATE** o **DELETE**.
- Usá **LIMIT** cuando estés explorando datos.
- Empezá consultas complejas con **SELECT** simple y luego agregá condiciones.
- Usá alias (**AS**) para que tus resultados sean más legibles.
- Comentá tu código **SQL**.
- Usá **IS NULL** y **IS NOT NULL** en vez de = **NULL**
- No mezcles tipos de datos incompatibles.
- No hagas **DELETE** sin backup en entornos reales.