SQL

COMANDOS BÁSICOS SQL

Algunos de los comandos más importantes de SQL.

- **SELECT**: Extrae datos de una base de datos
- **UPDATE**: Actualiza la base de datos
- **DELETE**: Elimina datos de una tabla
- **INSERT INTO**: Insertar nuevos datos a una tabla
- C**REATE DATABASE**: Crea una nueva base de datos
- **ALTER DATABASE**: Modifica una base de datos
- **CREATE TABLE**: Crea una tabla
- **ALTER TABLE**: Modifica una tabla
- **DROP TABLE**: Elimina una tabla
- **CREATE INDEX**: Crea un índex (llave de búsqueda)
- **DROP INDEX**: Elimina índex

Para seleccionar una tabla de la BBDD:

SELECT * FROM NOMBRE_TABLA

Para seleccionar columnas específicas se cambia el '*' por los nombres de las columnas que necesitamos visualizar.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Si en esta tabla necesitamos visualizar por ejemplo solo la columna 'CustomerName' de la tabla 'Customers' la query es la siguiente:
SELECT CustomerName, City FROM Customers;

DISTINCT

El SELECT DISTINCT es para seleccionar solamente los valores que son diferentes. Por ejemplo; si tenemos una columna de países, y se repiten varios de los mismos, el SELECT DISTINCT solo seleccionará uno de cada uno sin importar cuantas veces se repitan.
SELECT DISTINCT column1, column2, ...
FROM table_name;

WHERE

La cláusula WHERE es usada para filtrar búsquedas o palabras a la hora de visualizar datos.

WHERE SYNTAX:

SELECT column1, column2, ...
FROM table_name
WHERE condition;

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Si en esta tabla queremos seleccionar todos los clientes que sean del país 'México' debemos poner esta query:

SELECT * FROM Customers
WHERE Country='Mexico';

OPERADORES LÓGICOS EN LA CLÁUSULA WHERE

=	Igual
<	Menor que
>	Mayor que
<=	Menor igual que
>=	Mayor igual que
<>	Distinto que
BETWEEN	Entre
LIKE	Como
IN	Especificar posibles valores

OPERADORES SQL AND, OR, NOT

La cláusula WHERE puede ser combinada con los operadores AND, OR, y NOT.

Los operadores AND y OR son usados para filtrar registros basados en más de una condición.

- **AND**: Sirve para agregar más filtros de distintas columnas.
- **OR**: Sirve para agregar más filtros de la misma columna, si no está x filtro, filtra con y.

EJEMPLO AND

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin';
```

EJEMPLO OR

```
SELECT * FROM Customers
WHERE City='Berlin' OR City='München';
```

Los operadores NOT sirven para visualizar todos los registros menos el que agregaste.

EJEMPLO NOT

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

SQL ORDER BY

ORDER BY funciona para ordenar los registros de forma descendente o ascendente.

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

INSERT INTO

INSERT INTO funciona para agregar nuevos valores a la tabla. Se especifica el nombre de la tabla y las columnas entre paréntesis. Después se agregan los valores entre otros paréntesis. El número de valores y columnas debe ser el mismo.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

También es posible insertar datos sólo en columnas específicas y no en todas. Los campos que no se agregan datos serán autocompletados con valores nulos.

VARIABLES NULL

Una variable null es una variable que no tiene ningún tipo de dato.

NOTA: Un valor NULL es diferente de un valor cero o un campo que contiene espacios. Un campo con un valor NULL es aquel que se ha dejado en blanco durante la creación del registro!

ES NULL SYNTAX

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

NO ES NULL SYNTAX

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

UPDATE

UPDATE es usado para modificar un registro existente en la tabla.

UPDATE SYNTAX

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

CUIDADO: Tener cuidado con actualizar una tabla, si se omite la cláusula WHERE **TODOS** los datos serán actualizados.

DELETE

La declaración DELETE es usada para eliminar registros existentes en una tabla.

DELETE SYNTAX

```
DELETE FROM table_name WHERE condition;
```

CUIDADO: Tener cuidado con eliminar un registro, si se omite la cláusula WHERE **TODOS** los datos serán eliminados.

CLÁUSULAS SQL TOP, LIMIT, FETCH FIRST o ROWNUM

El SELECT TOP es utilizado para especificar el número exacto de registros retornados para limitar el número de registros. (top x).

NOTA: No todos los sistemas de bases de datos admiten la cláusula. MySQL admite **LIMIT** la cláusula para seleccionar un número limitado de registros, mientras que Oracle utiliza **FETCH FIRST** n **ROWS ONLY** y **ROWNUM**.

SQL Server / MS Access Syntax:

```
SELECT TOP number|percent column_name(s)
FROM table_name
WHERE condition;
```

MySQL Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

Oracle 12 Syntax:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s)
FETCH FIRST number ROWS ONLY;
```

Older Oracle Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE ROWNUM <= number;
```

SQL MIN() y MAX()

- La función MIN() retorna el valor mínimo de una columna seleccionada.
- La función MAX() retorna el valor máximo de una columna seleccionada.

MIN() Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

MAX() Syntax

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

FUNCIONES SQL COUNT(), AVG() y SUM()

- La función COUNT() retorna el número de filas que combina con el criterio especificado
- La función AVG() retorna el valor promedio de una columna numérica.
- La función SUM() retorna la suma de una columna numérica.

COUNT() Syntax

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

AVG() Syntax

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

SUM() Syntax

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

OPERADOR LIKE SQL

El operador LIKE es utilizado en la cláusula WHERE para buscar un patrón específico en una columna.

Hay dos operadores que son utilizados en el operador LIKE:

- Signo de porcentaje (%) representa cero, ninguno o múltiples caracteres.
- Signo guión bajo (_) representa uno o un solo carácter.

NOTA: MS Access utiliza un asterisco (*) en vez del signo de porcentaje (%), y un signo de pregunta (?) en vez de un guión bajo (_).

LIKE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

TIP: Se pueden combinar los operadores AND y OR.

OPERADOR IN SQL

El operador IN permite especificar varios valores en una cláusula WHERE.

IN Syntax

SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);

IN Operator Examples

Ejemplo para visualizar todos los clientes que están localizados en ‘Alemania’, ‘Francia’ o ‘UK’:

SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');

OPERADOR BETWEEN

El operador BETWEEN permite seleccionar entre dos valores numéricos, textos o datos.

BETWEEN Syntax

SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;

OPERADOR NOT BETWEEN

Lo mismo que el BETWEEN pero en vez de visualizar los datos entre medio visualiza todos **MENOS** los datos entre el rango.

NOT BETWEEN Syntax

SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;

ALIASES

Los alias SQL sirven para cambiar los nombres de tablas y/o columnas, para facilitar a la hora de escribir o diferenciar una o varias. Se escribe AS.

Alias Column Syntax

SELECT column_name AS alias_name
FROM table_name;

Alias Table Syntax

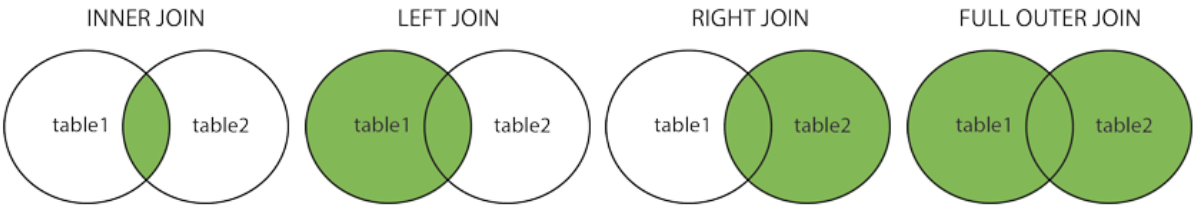
SELECT column_name(s)
FROM table_name AS alias_name;

SQL JOINs

La cláusula JOIN es utilizada para combinar columnas de una o más tablas.

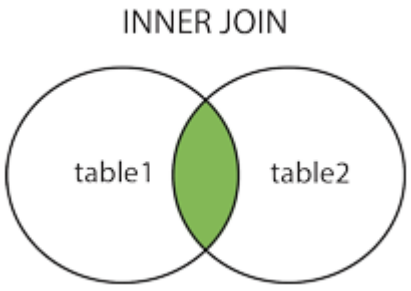
Diferentes tipos de JOINS:

- **INNER JOIN:** Retorna valores que matchean entre las dos tablas.
- **LEFT JOIN:** Retorna todos los valores de la tabla izquierda y las que matchea con la segunda.
- **RIGHT JOIN:** Retorna todos los valores de la tabla derecha y las que matchea con la segunda.
- **FULL JOIN**: Retorna todo.



INNER JOIN Syntax

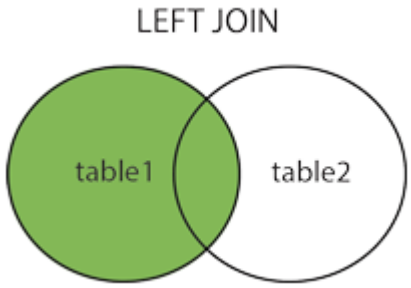
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;



NOTA: INNER JOIN selecciona todas las filas de ambas tablas siempre que haya una coincidencia entre las columnas. Si hay registros en la tabla "Pedidos" que no tienen coincidencias en "Clientes", estos pedidos no se mostrarán!

LEFT JOIN Syntax

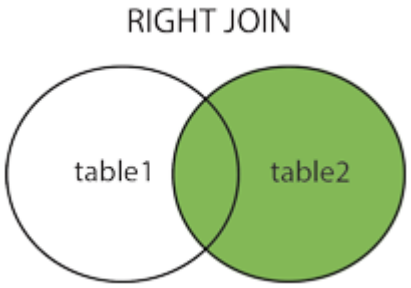
```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```



NOTA: En algunas bases de datos LEFT JOIN es LEFT OUTER JOIN.

RIGHT JOIN Syntax

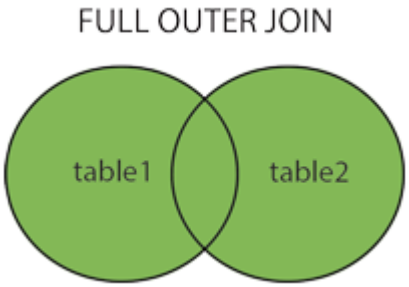
```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```



NOTA: En algunas bases de datos RIGHT JOIN es RIGHT OUTER JOIN.

FULL OUTER JOIN Syntax

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```



SQL Self Join

Self join es un join regular, se junta la tabla con sí misma.

Self Join Syntax

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

T1 and T2 are different table aliases for the same table.

OPERADOR UNION

El operador UNION es utilizado para combinar el resultado de dos o más SELECTs.

UNION Syntax

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

UNION ALL Syntax

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

GROUP BY

La instrucción GROUP BY busca filas que tienen los mismos valores, como "buscar el número de clientes en cada país".

La instrucción GROUP BY también es utilizada con las funciones de agregado (COUNT(), MAX(), MIN(), SUM(), AVG()).

GROUP BY Syntax

SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);

HAVING

La cláusula HAVING fué agregada a SQL porque WHERE no puede ser utilizada con funciones de agregación.

HAVING Syntax

SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);

SQL HAVING Ejemplos

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

La siguiente instrucción SQL enumera el número de clientes en cada país. Solo incluye países con más de 5 clientes:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

COUNT(CustomerID)	Country
9	Brazil
11	France
11	Germany
7	UK
13	USA