SQL (Structured Query Languaje)

El lenguaje de consulta estructurado o SQL (por sus siglas en inglés (Structured Query Language) es un lenguaje declarativode acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ellas.

CREATE

Se usa para crear objetos de la base de datos. A continuación se muestra la sintaxis para crear una tabla:

```
CREATE TABLE nombre_de_la_tabla (
nombre_de_la_columna1 tipo_de_dato,
nombre_de_la_columna2 tipo_de_dato,
…
nombre_de_la_columnan tipo_de_dato
);
```

Por ejemplo, considere la tabla juguetes de la base de datos que se crea como sigue:

```
CREATE TABLE juguetes (
Id_comprador INTEGER NOT NULL,
producto VARCHAR (40) NOT NULL,
precio DOUBLE
);
```

ALTER

Se usa para modificar la estructura de objetos de la base de datos. Este comando se usa por ejemplo para agregar una nueva columna a una tabla.

```
ALTER TABLE juguetes
ADD COLUMN id_vendedor INTEGER;
```

DROP

Se usa para eliminar objetos de la base de datos. Por ejemplo, el siguiente comando SQL elimina la tabla juguetes por completo:

```
DROP TABLE juguetes;
```

Mientras que éste otro comando sólo elimina la columna id_vendedor de la tabla juguetes:

```
ALTER TABLE juguetes
DROP COLUMN id_vendedor;
```

INSERT

Se usa para agregar datos en la tabla. La sintaxis de este comando es la siguiente:

```
INSERT INTO nombre_de_la_tabla (col1, col2)
VALUES (val1, val2);
```

El siguiente comando inserta una fila en una tabla:

```
INSERT INTO juguetes (id_comprador, producto, precio)
VALUES (21, 'Barbie', 200);
```

Tal y como se muestra a continuación:

Id_comprador	producto	precio
21	Barbie	200

Otra forma de escribir la sentencia INSERT es la siguiente:

```
INSERT INTO juguetes VALUES (21, 'Barbie', 200);
```

Esta sentencia no incluye los nombres de las columnas, cuando esto sucede, la cláusula VALUES debe contener los valores para todas las columnas en el mismo orden en el que están listadas las columnas en la tabla.

DELETE

Se usa para eliminar datos de la tabla. La sintaxis del comando DELETE es el siguiente:

```
DELETE FROM nombre_de_la_tabla WHERE condición;
```

En el comando DELETE, la condición WHERE es opcional. Si la condición no es especificada, todas las filas son eliminadas. De otra forma, sólo las filas que satisfacen la condición serán eliminadas. Ejemplo:

```
DELETE FROM juguetes WHERE producto = 'Barbie';
```

UPDATE

Se usa para actualizar o cambiar los valores presentes en una tabla. La sintaxis del comando UPDATE es el siguiente:

```
UPDATE nombre_de_la_tabla SET col1 = valor1, col2 = valor2
WHERE condicion;
```

En el siguiente ejemplo, se actualiza el precio a los productos Barbie de la tabla juguetes.

```
UPDATE juguetes SET precio = 300
WHERE producto = 'Barbie';
```

SELECT

Se usa para recuperar datos de las tablas. Esta sentencia puede ser simple o condicional. La sintaxis básica de la sentencia SELECT es la siguiente:

```
SELECT*FROM nombre_de_la_tabla;
```

La siguiente es la sintaxis para obtener ciertas columnas específicas de la tabla:

```
SELECT nombre col1, nombrecol2, … nombrecoln
FROM nombre_de_la_tabla;
```

Para obtener las columnas producto y precio de la tabla anterior, la sentencia sería la siguiente:

```
SELECT producto, precio, FROM juguetes;
```

Mostrándose como resultado:

producto	precio
Barbie	300

WHERE

La siguiente es la sintanxis de la cláusa WHERE:

```
SELECT*FROM nombre_de_la_tabla WHERE nombrecol = val;
```

Esta cláusula se usa junto a los operadores relacionales tales como las siguientes:

=	Igual
<> o !=	Diferente
<	Menor que
>	Mayor que
<=	Menor o igual
>=	Mayor o igual

Considere la tabla estadisticaempleado:

estadisticaempleado

IdNoEmpleado	Salario	Beneficios	Posicion
10	75000	15000	Gerente
105	60000	15000	Gerente
152	60000	12500	Gerente
244	50000	12000	Personal
322	45000	10000	Principiante

Para ver los IdNoEmpleado de aquellos empleados que tienen un salario de 60000 o más, se puede ejecutar la siguiente sentencia SQL:

```

```

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE salario >= 60000;
```

Y el resultado es el siguiente:

IdNoEmpleado
10
105
152

SUM

Es una función que permite sumar. Por ejemplo:

```
SELECT SUM(salario) FROM estadisticaempleado;
```

La consulta anterior retorna el salario total de todos los empleados de la tabla. El resultado de la consulta es:

SUM(salario)
290000

AVG

Es una función que permite calcular el promedio de un conjunto de valores. Por ejemplo:

```
SELECT AVG(salario) FROM estadisticaempleado;
```

La consulta anterior retorna el salario promedio de los empleados listados en la tabla. El resultado de la consulta es:

AVG(salario)
58000

MAX

```
SELECT MAX(salario) FROM estadisticaempleado;
```

La función MAX identifica el valor máximo disponible en la columna salario. El resultado de la consulta es:

MAX(salario)
75000

Para el siguiente ejemplo, la cláusula WHERE limita a las filas a los empleados que son Gerente:

```
SELECT MAX( salario ) FROM estadisticaempleado
WHERE Posicion = 'Gerente';
```

MIN

La función MIN retorna el mínimo valor para una columna dada, entre el conjunto de filas seleccionadas. Para el siguiente ejemplo:

```
SELECT MIN( salario ) FROM estadisticaempleado
WHERE Posicion = 'Gerente';
```

El resultado arrojado es el siguiente:

MIN(salario)
60000

COUNT

```
SELECT COUNT(*) FROM estadisticaempleado
WHERE Posicion = 'Personal';
```

La consulta anterior cuenta el número de empleados que pertenecen a la posición Personal. Arrojando como resultado:

COUNT(*)
1

AND

El operador AND une dos o más condiciones y muestra todas las filas que satisfacen todas las condiciones en la cláusula WHERE. Por ejemplo, para mostrar todos los empleados cuya posición sea Gerente y cuyo Salario es mayor a 60000, se puede escribir la siguiente consulta:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Salario > 60000 AND Posicion = 'Gerente';
```

Mostrando como resultado:

IdNoEmpleado
10

OR

El operador OR une dos o más condiciones. Muestra todas las filas que satisfacen al menos una condición en la cláusula WHERE. Por ejemplo para mostrar los empleados que ganan un Salario menor que 75000 ó que obtienen Beneficios menores de 15000, se escribe la siguiente consulta:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Salario <75000 OR Beneficios <15000;
```

Obteniéndose como resultado:

IdNoEmpleado
105
152
244
322

AND y OR

Es posible combinar los operadores AND y OR en una sola sentencia. Por ejemplo, para listar todos los "Gerentes" que ganan un salario mayor que 60000 o que obtienen beneficios mayores que 12000, se puede escribir la siguiente consulta:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Posicion = 'Gerente'  AND Salario > 60000 OR Beneficios > 12000;
```

Obteniéndose como resultado

IdNoEmpleado
10
105
152

El orden de precedencia es importante en este caso, ya que el operador AND precede al operador OR, por lo que las condiciones con el operador AND se evalúan primero y luego se evalúan las condiciones con el operador OR.

La sentencia SQL anterior puede también ser escrita de la siguiente forma:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE (Posicion = 'Gerente'  AND Salario > 60000) OR Beneficios > 12000;
```

Donde las condiciones que están entre paréntesis se evalúan primero.

Pero si lo escribimos de la siguiente manera:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Posicion = 'Gerente'  AND (Salario > 60000 OR Beneficios > 12000);
```

Primero se evalúa la condición OR y posteriormente la condición AND, dado que las condiciones entre paréntesis se evalúa primero.

IN

Se usa para realizar comparaciones con una lista de valores. Por ejemplo:

```

```

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Posicion = 'Gerente'  OR  Posicion = 'Personal' ;
```

La consulta lista todos los empleados que son gerente o del personal. Dicha consulta se puede escribir usando un operador IN:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Posicion IN ('Gerente', 'Personal');
```

El operador IN verifica si la condición satisface alguno de los valores que están entre paréntesis, mostrándose:

IdNoEmpleado
10
105
152
244

BETWEEN

Se usa para comparar si cierto valor está dentro de un rango dado. Por ejemplo, si se está interesado en encontrar a todos los empleados que ganan salarios dentro del rango de 45000 y 60000, se podría escribir la consulta de la siguiente manera:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Salario >= 45000 AND Salario <= 60000;
```

O se podría escribir utilizando la sentencia BETWEEN:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Salario BETWEEN 45000 AND 60000;
```

Obteniéndose el mismo resultado:

IdNoEmpleado
105
152
244
322

NOT

Si estamos interesados en listar todos los empleados que no ganan un salario de 45000 a 60000, se escribe la siguiente consulta:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Salario NOT BETWEEN 45000 AND 60000;
```

IdNoEmpleado
10

Por otro lado, la siguiente consulta lista todos los empleados que no son gerente:

```
SELECT IdNoEmpleado FROM estadisticaempleado
WHERE Posicion NOT IN ('Gerente');
```

IdNoEmpleado
244
322

Cabe resaltar que el operador NOT tiene mayor precedencia que los operadores AND y OR.

LIKE

Se usa para verificar patrones dentro de cadenas de caracteres, compararlos y mostrar el resultado. Por ejemplo, si se quieren listar todas las posiciones que comienzan con “P”, se escribe la siguiente consulta:

```
SELECT Posicion FROM estadisticaempleado
WHERE Posicion LIKE  'P%' ;
```

Posicion
Personal
Principiante

El signo de % se usa para representar cero o más caracteres. En el ejemplo anterior, 'P%' especifica cualquier Posicion que comience con “P”, seguido de cero o más caracteres. Para listar aquellas posiciones que terminen en “P”, se usa '%P', que especifica cualquier posición que termine con “P”, precedido de cero o más caracteres.

ORDER BY

Se usa para dar formato a la salida basándose en un campo y un cierto orden, el cual puede ser descendente o ascendente. Por defecto, la cláusula ORDER BY lista los datos de una columna en orden ascendente.

```
SELECT * FROM estadisticaempleado ORDER BY salario;
```

IdNoEmpleado	Salario	Beneficios	Posicion
322	45000	10000	Principiante
244	50000	12000	Personal
105	60000	15000	Gerente
152	60000	12500	Gerente
10	75000	15000	Gerente

La consulta anterior lista los empleados en el orden ascendente de los valores de la columna salario, por el contrario, si queremos listar los empleados en el orden descendente de los valores de la columna salario, la consulta quedaría de la siguiente manera:

```
SELECT * FROM estadisticaempleado ORDER BY salario DESC;
```

IdNoEmpleado	Salario	Beneficios	Posicion
10	75000	15000	Gerente
105	60000	15000	Gerente
152	60000	12500	Gerente
244	50000	12000	Personal
322	45000	10000	Principiante

DISTINCT

Si se necesita encontrar una lista única de posiciones disponibles en la tabla estadisticaempleado, se puede ejecutar la siguiente consulta:

```
SELECT DISTINCT posicion FROM estadisticaempleado;
```

Arrojando COMO RESULTADO:

posicion
Gerente
Personal
Principiante

INNER JOIN

Cuando un usuario quiere obtener información completa consultando más de una tabla, debe JOIN (UNIR) los datos de las distintas tablas.

Un JOIN hace corresponder los registros de la primera tabla con los de la segunda tabla, basándose en la igualdad de los valores especificados en la condición JOIN. Por lo que sólo los registros que tengan una correspondencia exacta serán extraídos de ambas tablas.

Por ejemplo, una forma de aplicar la sentencia INNER JOIN sería la siguiente:

materia

NombreMateria	NoMateria	Creditos
Lengua	SS G211	1
Estadística	CS G311	4
Matemática	IS G411	4
Informática	EC G511	3
Comercio	MC G611	2

estudiante

NombreEstudiante	NoEstudiante	NoMateria
Pedro	1	SS G211
José	2	CS G311
María	3	MC G611
Carlos	4	IS G411
Carolina	5	EC G511

```
SELECT materia.NombreMateria, materia.Creditos, estudiante.NombreEstudiante
FROM materia, estudiante
WHERE materia.NoMateria=estudiante.NoMateria;
```

Obteniéndose como resultado:

NombreMateria	Creditos	NombreEstudiante
Lengua	1	Pedro
Estadística	4	José
Comercio	2	María
Informática	3	Daniela

La consulta anterior tiene una cláusula WHERE que verifica la igualdad de los valores de las columnas NoMateria de las dos tablas.

Otra manera de especificar la consulta sería;

```
SELECT materia.NombreMateria, materia.Creditos, estudiante.NombreEstudiante
FROM materia INNER JOIN estudiante
ON materia.NoMateria = estudiante.NoMateria;
```

Otro ejemplo es el siguiente:

```
SELECT materia.NombreMateria, materia.Creditos, estudiante.NombreEstudiante
FROM materia, estudiante
WHERE Creditos = 4
AND materia.NoMateria=estudiante.NoMateria;
```

NombreMateria	Creditos	NombreEstudiante
Estadística	4	José

GROUP BY

Un grupo puede ser especificado usando una cláusula GROUP BY. Más de una columna puede ser incluida en la cláusula GROUP BY. El agrupamiento lógico será hecho basado en las columnas dadas después de la cláusula GROUP BY. Por ejemplo si se quiere encontrar el número de artículos comprados por cada comprador en la tabla ventas, se debe hacer la siguiente consulta:

ventas

Id_articulo	Id_comprador	Nombre_articulo
01	50	Cama
02	15	Mesa
15	02	Silla
21	50	Espejo
50	01	Escritorio
01	21	Cama
02	21	Mesa
15	50	Silla
01	15	Cama
02	21	Mesa
21	02	Espejo
50	01	Escritorio

```
SELECT Id_comprador, COUNT(Nombre_articulo)
FROM ventas GROUP BY Id_comprador;
```

Obteniéndose como resultado:

Id_comprador	COUNT(Nombre_articulo)
1	2
2	2
15	2
21	3
50	3

En este ejemplo, se realiza una agrupación lógica para cada comprador. Hay cinco compradores distintos por lo que cinco conjuntos o agrupamientos de datos están disponibles. En cada agrupamiento o conjunto de datos, se aplica la función COUNT y se obtiene el número de elementos por cada grupo.

Por otro lado, si se utiliza la siguiente consulta:

```
SELECT COUNT(Nombre_articulo)
FROM ventas;
```

El resultado será totalmente diferente, ya que lo que se obtiene es el número de total de registros en la columna especificada:

COUNT(Nombre_articulo)
12