Introducción a SQL

Gestión y Arquitectura de Datos, Universidad de San Andrés

Si encuentran algún error en el documento o hay alguna duda, mandenmé un mail a rodriguezf@udesa.edu.ar y lo revisamos.

1. ¿Qué es SQL?

SQL (Structured Query Language) es el lenguaje estándar para interactuar con bases de datos relacionales. Es un lenguaje declarativo, lo que significa que especificamos qué queremos obtener en lugar de cómo obtenerlo.

Las principales operaciones que podemos realizar con SQL son:

- Consultar datos (SELECT)
- Insertar datos (INSERT)
- Actualizar datos (UPDATE)
- Eliminar datos (DELETE)
- Crear y modificar estructuras de datos (CREATE, ALTER, DROP)

2. Estructura Básica de una Consulta

La estructura más básica de una consulta SQL es:

```sql
SELECT columna1, columna2
FROM tabla
WHERE condicion;
```

Resultado ejemplo:

columna1	columna2
valor1	valor2
valor3	valor4

Donde:

- SELECT especifica qué columnas queremos ver
- FROM indica de qué tabla obtenemos los datos
- WHERE (opcional) filtra los resultados según una condición

3. Operadores y Funciones Básicas

3.1. Operadores de Comparación

Los operadores de comparación nos van a servir para poder comparar valores entre dos cosas. Estos son:

- = (igual)
- <(menor que)
- >(mayor que)
- <= (menor o igual que)
- >= (mayor o igual que)
- <>o != (distinto)

Ejemplos:

```
-- Buscar salarios mayores a 50000
SELECT nombre, apellido, salario
FROM empleados
WHERE salario > 50000;
```

nombre	apellido	salario
Juan	Pérez	55000

```
-- Buscar apellidos distintos a Perez
SELECT nombre, apellido
FROM empleados
WHERE apellido <> 'Perez';
```

nombre	apellido
Juan	García
María	López

3.2. Operadores Lógicos

Los operadores lógicos nos van a servir para poder combinar condiciones. Estos son:

- AND
- OR
- NOT

Ejemplos:

```sql
-- Buscar empleados con salario mayor a 50000 y que trabajen en el departamento 1
SELECT nombre, apellido, salario
FROM empleados
WHERE salario > 50000 AND nombre = 'Juan';
```

nombre	apellido	salario
Juan	García	55000

```sql
-- Buscar empleados con salario exactamente 42000 y que no sean de la ciudad de Buenos Aires
SELECT nombre, apellido, salario, ciudad
FROM empleados
WHERE salario = 42000 AND ciudad <> 'Buenos Aires';
```

nombre	apellido	salario	ciudad
Marcos	Gómez	42000	Córdoba

3.3. Operador IN

El operador IN se usa para buscar valores en una lista de valores que le damos.

Ejemplos:

```
-- Buscar empleados que tengan de apellido Garcia o Lopez
SELECT nombre, apellido
FROM empleados
WHERE apellido IN ('Garcia', 'Lopez');
```

nombre	apellido
Juan	García
María	López

3.4. Operador BETWEEN

El operador BETWEEN se usa para buscar valores entre dos valores que le damos.

Ejemplos:

```
-- Buscar empleados con salario entre 40000 y 50000
SELECT nombre, apellido, salario
FROM empleados
WHERE salario BETWEEN 40000 AND 50000;
```

nombre	apellido	salario
Marcos	Gómez	42000

3.5. Operador LIKE

El operador LIKE se usa para buscar patrones en texto. Tenemos dos opciones, el % y el _. El % representa cualquier cantidad de caracteres, mientras que el _ representa un único caracter.
Ejemplos:

```
-- Busca apellidos que terminan en G
SELECT nombre, apellido
FROM empleados
WHERE apellido LIKE '%G';
```

nombre	apellido
Juan	Rodriguez

```sql
-- Busca apellidos que tienen G como segundo caracter
SELECT nombre, apellido
FROM empleados
WHERE apellido LIKE '_arcia';
```

nombre	apellido
Maria	Garcia

```sql
-- Busca apellidos que empiezan con G
SELECT nombre, apellido
FROM empleados
WHERE apellido LIKE 'G%';
```

nombre	apellido
Maria	Garcia
Carlos	Gonzalez

```sql
-- Busca apellidos que contienen G en cualquier posicion
SELECT nombre, apellido
FROM empleados
WHERE apellido LIKE '%G%';
```

nombre	apellido
Juan	Rodriguez
Maria	Garcia
Carlos	Gonzalez

Nota: No en todos los motores de SQL el %G % devolvería Rodríguez dado que es una g minúscula y no una G mayúscula como estamos pidiendo. En la mayoría sí lo devolvería.

3.6. Valores NULL

NULL representa la ausencia de un valor. Para trabajar con NULL usamos:

- IS NULL
- IS NOT NULL

Ejemplo:

```
SELECT nombre, telefono
FROM empleados
WHERE telefono IS NULL;
```

nombre	telefono
Carlos	NULL

4. Ejemplos Prácticos

4.1. Seleccionar Todos los Registros

```
SELECT *
FROM empleados;
```

id	nombre	apellido	salario
1	Juan	Pérez	50000
2	María	García	60000
3	Carlos	López	45000

Nota: Evitar usar SELECT *, en cambio es recomendable especificar las columnas que necesitamos. Esto es para evitar problemas de performance en bases de datos gigantes.

4.2. Filtrar Registros

```
SELECT nombre, apellido, salario
FROM empleados
WHERE salario > 50000;
```

nombre	apellido	salario
María	García	60000

4.3. Ordenar Resultados

```sql
SELECT nombre, apellido, edad
FROM empleados
ORDER BY edad DESC;
```

nombre	apellido	edad
Carlos	López	45
María	García	35
Juan	Pérez	30

5. Alias

Los alias nos permiten dar nombres temporales a tablas o columnas para hacer las consultas más legibles.

5.1. Alias de Tablas

```sql
SELECT e.nombre, d.nombre_departamento
FROM empleados AS e
JOIN departamentos AS d ON e.id_departamento = d.id;
```

nombre	nombre_departamento
Juan	Ventas
María	IT
Carlos	Ventas

5.2. Alias de Columnas

```sql
SELECT
    nombre AS nombre_empleado,
    salario AS sueldo_mensual
FROM empleados;
```

nombre_empleado	sueldo_mensual
Juan	50000
María	60000

6. Funciones de Agregación

Las funciones de agregación nos permiten realizar cálculos sobre grupos de registros.

6.1. COUNT()

Cuenta el número de registros que cumplen con una condición.

```
SELECT COUNT(*) as total_empleados
FROM empleados;
```

total_empleados
3

6.2. SUM()

Suma los valores de una columna numérica.

```
SELECT SUM(salario) as total_salarios
FROM empleados;
```

total_salarios
155000

6.3. AVG()

Calcula el promedio de los valores en una columna.

```
SELECT AVG(salario) as salario_promedio
FROM empleados;
```

salario_promedio
51666.67

6.4. MAX()

Encuentra el valor máximo en una columna.

```
SELECT MAX(salario) as salario_maximo
FROM empleados;
```

salario_maximo
60000

6.5. MIN()

Encuentra el valor mínimo en una columna.

```
SELECT MIN(salario) as salario_minimo
FROM empleados;
```

salario_minimo
45000

6.6. Agrupación con GROUP BY

Podemos combinar estas funciones con GROUP BY para obtener estadísticas por grupo:

```
SELECT
    departamento,
    COUNT(*) as cantidad_empleados,
    AVG(salario) as salario_promedio
FROM empleados
GROUP BY departamento;
```

departamento	cantidad_empleados	salario_promedio
Ventas	2	55000
IT	1	45000

```
SELECT nombre, edad
FROM empleados
WHERE edad IS NOT NULL AND nombre LIKE '%er%'
```

nombre	edad
Fermín	26
Hernán	42

7. Joins

Los JOINs nos permiten combinar filas de dos o más tablas basándonos en una condición de relación entre ellas.

7.1. INNER JOIN

Devuelve solo los registros que coinciden en ambas tablas. Ejemplo:

```
SELECT e.nombre, d.nombre_departamento
FROM empleados e
INNER JOIN departamentos d ON e.id_departamento = d.id;
```

nombre	nombre_departamento
Juan	Ventas
María	IT
Carlos	Ventas

7.2. LEFT JOIN

Devuelve todos los registros de la tabla izquierda y los coincidentes de la derecha. Ejemplo:

```
SELECT e.nombre, d.nombre_departamento
FROM empleados e
LEFT JOIN departamentos d ON e.id_departamento = d.id;
```

nombre	nombre_departamento
Juan	Ventas
María	IT
Carlos	Ventas
Pedro	NULL

7.3. RIGHT JOIN

Devuelve todos los registros de la tabla derecha y los coincidentes de la izquierda. Ejemplo:

```
SELECT e.nombre, d.nombre_departamento
FROM empleados e
RIGHT JOIN departamentos d ON e.id_departamento = d.id;
```

nombre	nombre_departamento
Juan	Ventas
María	IT
Carlos	Ventas
NULL	Marketing

7.4. FULL JOIN

Devuelve todos los registros cuando hay coincidencia en cualquiera de las tablas. Ejemplo:

```
SELECT e.nombre, d.nombre_departamento
FROM empleados e
FULL JOIN departamentos d ON e.id_departamento = d.id;
```

nombre	nombre_departamento
Juan	Ventas
María	IT
Carlos	Ventas
Pedro	NULL
NULL	Marketing