

UT_04.3_SQL_Lenguaje de consulta estructurado

¿Qué es SQL?

SQL, o lenguaje de consulta estructurado, es el lenguaje de fachada que se utiliza para comunicarse con todos los **RDMS**, sin embargo, la mayoría de las veces, como ingenieros de backend (no somos ingenieros de bases de datos), no utilizamos SQL directamente, sino que utilizamos mapeadores relacionales de objetos **ORM**, que proporcionan una interfaz mejor y más fiable sobre el SQL sin procesar para comunicarnos con la base de datos en nuestro lenguaje de programación preferido. No obstante, seguimos necesitando conocimientos básicos de SQL para manipular e interactuar con la base de datos de forma rápida y sencilla.

Consultas SQL básicas

Las consultas SQL básicas implican seleccionar datos específicos de una tabla de base de datos utilizando comandos como **SELECT**, **DISTINCT**, **WHERE**, **LIMIT** y **OFFSET**.

```
-- Seleccionar todas las columnas de una tabla

SELECT * FROM employees;

-- Seleccionar columnas específicas

SELECT first_name, last_name FROM employees;

-- Usar DISTINCT para obtener valores únicos

SELECT DISTINCT department_id FROM employees;

-- Usar WHERE para filtrar resultados

SELECT * FROM employees WHERE department_id = 10;

-- Usar LIMIT para limitar el número de resultados

SELECT * FROM employees LIMIT 5;

-- Uso de OFFSET para omitir determinadas filas

SELECT * FROM employees OFFSET 5;
```

Filtrado de datos

El filtrado de datos en SQL permite recuperar registros específicos de una tabla en función de determinadas condiciones, como el uso de operadores de comparación como **>**, **<**, **=** y operadores lógicos como **AND**, **OR** y **NOT**.

```
-- Uso de operadores de comparación

SELECT * FROM employees WHERE salary > 50000;
```

```
-- Uso de operadores lógicos

SELECT * FROM employees WHERE department_id = 10 AND salary > 50000;

-- Uso de IN y NOT IN

SELECT * FROM employees WHERE department_id IN (10, 20);

-- Uso de BETWEEN

SELECT * FROM employees WHERE salary BETWEEN 40000 AND 60000;

-- Uso de LIKE para la coincidencia de patrones

SELECT * FROM employees WHERE last_name LIKE 'S%';
```

Ordenación de datos

La ordenación de datos en SQL organiza los registros recuperados en orden ascendente o descendente según las columnas especificadas utilizando la cláusula ORDER BY.

```
-- Ordenación de datos en orden ascendente

SELECT * FROM employees ORDER BY salary;

-- Ordenar datos en orden descendente

SELECT * FROM employees ORDER BY salary DESC;

-- Ordenar por varias columnas

SELECT * FROM employees ORDER BY department_id, salary DESC;
```

Funciones agregadas

Las funciones agregadas en SQL realizan cálculos sobre un conjunto de valores y devuelven un único valor. Las funciones agregadas comunes incluyen COUNT, SUM, AVG, MIN y MAX.

```
-- Contar el número de filas

SELECT COUNT(*) FROM employees;

-- Calcular el salario total

SELECT SUM(salary) FROM employees;
```

```
-- Hallar el salario medio  
  
SELECT AVG(salary) FROM employees;  
  
  
-- Hallar el salario mínimo  
  
SELECT MIN(salary) FROM employees;  
  
  
-- Hallar el salario máximo  
  
SELECT MAX(salary) FROM employees;
```

Agrupación de datos

La agrupación de datos en SQL permite agrupar filas que tienen los mismos valores en columnas específicas utilizando la cláusula GROUP BY, que a menudo se utiliza junto con funciones agregadas.

```
-- Agrupación de datos por departamento  
  
SELECT department_id, COUNT(*) FROM employees GROUP BY department_id;  
  
-- Uso de HAVING para filtrar datos agrupados  
  
SELECT department_id, COUNT(*) FROM employees GROUP BY department_id HAVING  
COUNT(*) > 5;
```

Uniones

Las uniones en SQL combinan datos de varias tablas basándose en columnas relacionadas para recuperar datos que abarcan todas esas tablas.

```
-- Unión interna  
  
SELECT * FROM employees INNER JOIN departments ON employees.department_id =  
departments.department_id;  
  
  
-- Unión izquierda  
  
SELECT * FROM employees LEFT JOIN departments ON employees.department_id =  
departments.department_id;  
  
  
-- Unión derecha  
  
SELECT * FROM employees RIGHT JOIN departments ON employees.department_id =  
departments.department_id;
```

```
-- Unión externa completa

SELECT * FROM employees FULL OUTER JOIN departments ON employees.department_id =
departments.department_id;
```

Subconsultas

Las subconsultas en SQL son consultas anidadas dentro de otra consulta, que se utilizan para recuperar datos que dependen del resultado de otra consulta.

```
-- Ejemplo de subconsulta

SELECT * FROM employees WHERE department_id IN (SELECT department_id FROM departments
WHERE location_id = 1700);

-- Ejemplo de subconsulta correlacionada

SELECT * FROM employees e WHERE salary > (SELECT AVG(salary) FROM employees WHERE
department_id = e.department_id);

-- Devuelve todos los empleados cuyo salario es mayor que el promedio de salario de
su propio departamento. La subconsulta devuelve el salario medio del departamento de
cada empleado.
```

Vistas

Las vistas en SQL son tablas virtuales generadas a partir del resultado de una consulta, que proporcionan una forma de simplificar consultas complejas y restringir el acceso a determinados datos.

```
-- Creación de una vista

CREATE VIEW high_paid_employees AS SELECT * FROM employees WHERE salary > 80000;

-- Actualización de una vista

CREATE OR REPLACE VIEW high_paid_employees AS SELECT * FROM employees WHERE salary >
90000;

-- Eliminación de una vista

DROP VIEW IF EXISTS high_paid_employees;
```

Indexación

La indexación en SQL mejora el rendimiento de las consultas mediante la creación de índices en las columnas, lo que permite una recuperación más rápida de los datos.

```
-- Creación de un índice

CREATE INDEX idx_lastname ON employees(last_name);

-- Eliminación de un índice

DROP INDEX idx_lastname;
```

Transacciones

Las transacciones en SQL garantizan la integridad de los datos agrupando las sentencias SQL en unidades atómicas, lo que garantiza que todas las sentencias se ejecuten correctamente o que ninguna de ellas se ejecute.

```
-- Inicio de una transacción

BEGIN TRANSACTION;

-- Confirmar una transacción

COMMIT;

-- Revertir una transacción

ROLLBACK;
```

Procedimientos almacenados

Los procedimientos almacenados en SQL son código SQL precompilado almacenado en la base de datos que se puede ejecutar con un solo comando, a menudo utilizado para encapsular tareas que se ejecutan con frecuencia

```
-- Crear un procedimiento almacenado

CREATE PROCEDURE get_employee (IN employee_id INT)

BEGIN

    SELECT * FROM employees WHERE employee_id = employee_id;

END;

-- Ejecutar un procedimiento almacenado

CALL get_employee(100);

-- Modificar un procedimiento almacenado

ALTER PROCEDURE get_employee (IN employee_id INT)

BEGIN

    SELECT employee_id, first_name, last_name FROM employees WHERE employee_id =
employee_id;

END;

-- Eliminación de un procedimiento almacenado

DROP PROCEDURE IF EXISTS get_employee;
```

Copia de seguridad y recuperación

La copia de seguridad y la recuperación en SQL implican la creación de copias de seguridad de las bases de datos para protegerlas contra la pérdida de datos y restaurarlas en caso de fallo o corrupción de la base de datos.

```
-- Creación de una copia de seguridad completa  
  
BACKUP DATABASE dbname TO disk = "path_to_backup";  
  
-- Creación de una copia de seguridad diferencial  
  
BACKUP DATABASE dbname TO disk = "path_to_backup" WITH DIFFERENTIAL;  
  
-- Creación de una copia de seguridad del registro de transacciones  
  
BACKUP LOG dbname TO disk = "path_to_backup";  
  
-- Restaurar desde una copia de seguridad  
  
RESTORE DATABASE dbname FROM disk = "path_to_backup";
```

Dialectos SQL y extensiones específicas de proveedores

Aunque SQL es un lenguaje estandarizado, los diferentes proveedores de bases de datos han implementado sus propias extensiones y dialectos, lo que ha dado lugar a ligeras variaciones en la forma de escribir y ejecutar SQL en los diferentes RDBMS. Estas variaciones pueden afectar a la sintaxis, las funciones y las características disponibles para los desarrolladores que trabajan con sistemas de bases de datos específicos.

A continuación se muestran algunos ejemplos de dialectos y extensiones SQL específicos de proveedores:

Oracle SQL:

- Oracle SQL incluye extensiones propias, como consultas jerárquicas, funciones analíticas y la cláusula MODEL para el modelado de datos.
- Ejemplo: `SELECT CUBE (product, region) FROM sales;` (CUBE es una extensión específica de Oracle para generar subtotales)

SQL Server (Microsoft):

- SQL Server incluye T-SQL (Transact-SQL), que amplía el SQL estándar con características como construcciones de lenguaje de control de flujo, gestión de errores y construcciones de programación procedimental.
- Ejemplo: `SELECT ISNULL(column_name, "default_value") FROM table_name;` (ISNULL es una función T-SQL).

MySQL:

- MySQL incluye extensiones como tipos y funciones de datos espaciales, funciones de ventana y el operador REGEX para la coincidencia de expresiones regulares.
- Ejemplo: `SELECT nombre_columna REGEXP '^patrón' FROM nombre_tabla;` (REGEXP es un operador específico de MySQL)

PostgreSQL:

- PostgreSQL incluye extensiones como matrices, compatibilidad con JSON, búsqueda de texto completo y tipos de datos de rango.
- Ejemplo: `SELECT nombre_columna || "sufijo" FROM nombre_tabla;` (`||` es un operador de concatenación de cadenas específico de PostgreSQL)