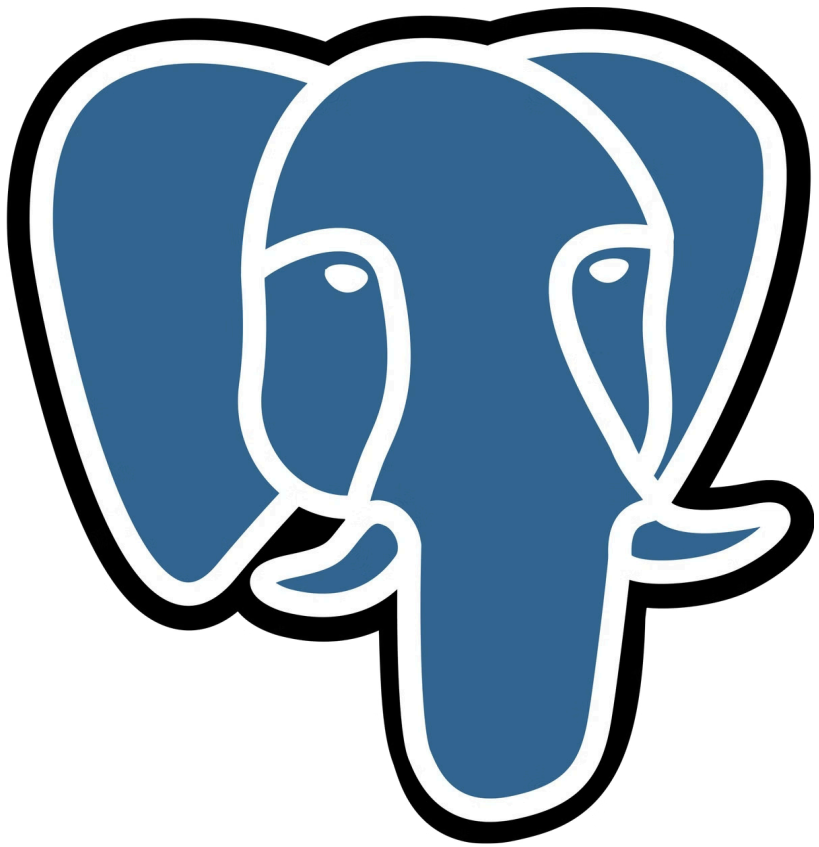
🎓 Clase 3: Joins, Subconsultas y Optimización de consultas



👋 **Bienvenidos al tercer módulo**
En esta sesión, se aprenderá a combinar datos utilizando diferentes tipos de JOINs y a integrar datos de múltiples tablas. Además, se explorarán las diferencias entre UNION y UNION ALL.

También se introducirá el concepto de subconsultas correlacionadas y no correlacionadas. 🚀

✍️ **Profesor:** Yoseph Ayala Valencia

🌟 ¡Prepárate para dominar los fundamentos de SQL y llevarlos al siguiente nivel con Python! 🐍

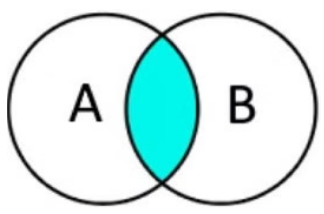
Joins en SQL

Tipos de Joins

Existen varios tipos de joins en SQL:

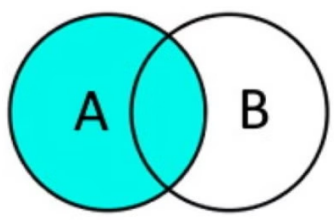
INNER JOIN

Devuelve solo las filas que tienen valores coincidentes en ambas tablas.



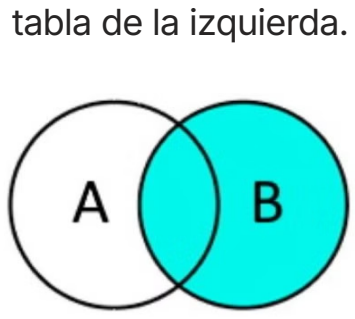
LEFT JOIN

Devuelve todas las filas de la tabla de la izquierda, junto con las filas coincidentes de la tabla de la derecha.



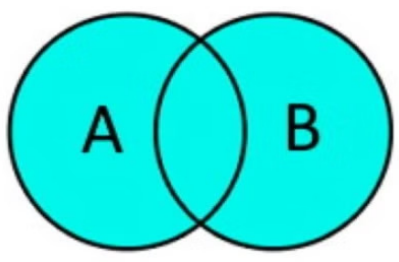
RIGHT JOIN

Devuelve todas las filas de la tabla de la derecha, junto con las filas coincidentes de la tabla de la izquierda.



FULL JOIN

Devuelve todas las filas de ambas tablas, independientemente de si hay valores coincidentes o no.



¿Qué son los Joins?

Los joins en SQL son una forma de combinar datos de dos o más tablas en una única consulta. Permiten unir filas de diferentes tablas en base a una columna común entre ellas.

1 2 3 4

¿Para qué sirven?

Los joins son muy útiles cuando necesitamos relacionar información que está distribuida en diferentes tablas de una base de datos. Nos permiten obtener datos más completos y enriquecidos para nuestras consultas.

Es importante revisar que cuando hagamos un join al menos una de nuestras tablas tenga valores únicos en su primary key/llave

Siempre se debe revisar cúantos observaciones tiene nuestra tabla antes y después de hacer el join

INNER JOIN: definición y uso

¿Qué es INNER JOIN?

INNER JOIN es el tipo de JOIN más común. Combina filas de dos tablas basándose en una condición específica, devolviendo solo los registros que tienen coincidencia en ambas tablas.

¿Cómo se usa?

La sintaxis básica es:

SELECT *

FROM tabla1

INNER JOIN tabla2

ON tabla1.campo = tabla2.campo.

Esto genera una nueva tabla con los campos de ambas tablas.

Ventajas

INNER JOIN es eficiente y preciso, ya que solo devuelve los datos relevantes. Es ideal para unir tablas relacionadas y obtener información consolidada.

LEFT JOIN: definición y uso

¿Qué es LEFT JOIN?

LEFT JOIN devuelve todos los registros de la tabla izquierda (tabla1) y los registros coincidentes de la tabla derecha (tabla2).

1

Uso

Es útil cuando se necesita mantener toda la información de una tabla principal, aun si no hay coincidencias en la otra tabla.

3

2

Sintaxis

SELECT *

FROM tabla1

LEFT JOIN tabla2

ON tabla1.campo = tabla2.campo

RIGHT JOIN: definición y uso

1

¿Qué es RIGHT JOIN?

RIGHT JOIN devuelve todos los registros de la tabla derecha (tabla2) y los registros coincidentes de la tabla izquierda (tabla1).

2

Sintaxis

SELECT *

FROM tabla1

RIGHT JOIN tabla2

ON tabla1.campo = tabla2.campo

3

Uso

Es útil cuando se quiere mantener toda la información de una tabla secundaria, aun si no hay coincidencias en la tabla principal.

Joins vs UNION y UNION ALL

Joins

- Los joins combinan filas de diferentes tablas en base a una columna común entre ellas.
- No es necesario que las tablas involucradas tengan la misma estructura. Puedes hacer un JOIN entre tablas con diferentes números de columnas, tipos de datos, etc.
- Lo que es importante es que las columnas que uses para hacer el JOIN deben ser compatibles, es decir, deben tener tipos de datos compatibles (por ejemplo, unir una columna `INT` con otra `INT` o una `VARCHAR` con otra `VARCHAR`).

Ejemplos: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN.

UNION y UNION ALL

- UNION y UNION ALL combinan los resultados de dos o más consultas SELECT en una sola salida.
- UNION elimina los registros duplicados, mientras que UNION ALL conserva todos los registros.
- En este caso, las estructuras de las tablas o subconsultas deben ser idénticas o al menos muy similares.
- Específicamente, las siguientes condiciones deben cumplirse:
  - El número de columnas debe ser el mismo en las consultas que estás combinando.
  - Los tipos de datos de las columnas correspondientes deben ser compatibles.
  - El orden de las columnas debe coincidir en ambas consulta

Ejemplos

- UNION
- UNION ALL

Subconsultas: ¿qué son y para qué sirven?

¿Qué son las subconsultas?

Las subconsultas son consultas SQL anidadas dentro de otras consultas, permitiendo obtener datos más complejos y específicos.

Ejemplo: Queremos encontrar el nombre del empleado con el salario más alto.

```
SELECT Name
FROM Employees
WHERE Salary = (
    SELECT MAX(Salary)
    FROM Employees
);
```

¿Para qué sirven?

Las subconsultas se utilizan para filtrar, agregar o comparar datos de manera más precisa, ampliando las capacidades de las consultas SQL.

Tipos de subconsultas

Pueden ser correlacionadas o no correlacionadas, y se pueden usar en cláusulas WHERE, SELECT, FROM o HAVING.

Ventajas

Las subconsultas permiten crear consultas más poderosas y flexibles, optimizando el procesamiento de datos.

Cómo utilizar subconsultas en SQL

Paso 1: Identificar la información necesaria

Supongamos que queremos obtener el nombre y el salario de los empleados que ganan más que el promedio de la empresa.

La subconsulta nos ayudará a calcular el salario promedio, y luego podremos usar ese dato en la consulta principal.

Paso 2: Construir la subconsulta

La subconsulta sería:

SELECT AVG(salary) AS avg_salary

FROM employees;

Esta consulta calcula el salario promedio de todos los empleados.

Paso 3: Integrar la subconsulta

Ahora podemos usar el resultado de la subconsulta en la consulta principal:

SELECT name, salary FROM employees WHERE salary > (SELECT AVG(salary) FROM employees);

Esta consulta devuelve el nombre y el salario de los empleados que ganan más que el promedio.

Subconsultas: Correlacionadas vs No Correlacionadas

Subconsultas No Correlacionadas

Las subconsultas no correlacionadas son independientes de la consulta principal. Se ejecutan una sola vez y devuelven un conjunto de resultados que se utiliza en la consulta principal.

Ejemplo:

Supongamos que tenemos dos tablas:

- `Employees`: contiene los empleados con sus `EmployeeID`, `Name`, y `DepartmentID`.
- `Departments`: contiene los departamentos con sus `DepartmentID` y `DepartmentName`.

Queremos encontrar los nombres de todos los empleados que trabajan en el departamento llamado 'Sales'.

```sql
SELECT Name
FROM Employees
WHERE DepartmentID = (
    SELECT DepartmentID
    FROM Departments
    WHERE DepartmentName = 'Sales'
);
```

- Primero, se ejecuta la subconsulta:

`SELECT DepartmentID FROM Departments WHERE DepartmentName = 'Sales';`

- Esta subconsulta busca el `DepartmentID` del departamento llamado 'Sales'. Supongamos que devuelve `3`.
- Luego, la consulta principal se ejecuta utilizando el resultado de la subconsulta:

`SELECT Name`

`FROM Employees WHERE DepartmentID = 3;`

- Aquí, la consulta principal busca todos los empleados cuyo `DepartmentID` es `3`.

Subconsultas Correlacionadas

Las subconsultas correlacionadas dependen de la consulta principal y se ejecutan una vez por cada fila de la consulta principal. Utilizan valores de la consulta principal en su propio procesamiento.

Ejemplo:

Siguiendo con las mismas tablas, queremos encontrar los nombres de todos los empleados cuyo salario es mayor que el salario promedio de su propio departamento.

Supongamos que también tenemos una tabla:

- `Salaries`: contiene los `EmployeeID` y sus `Salary`.

```sql
SELECT E.Name
FROM Employees E
WHERE E.EmployeeID IN (
    SELECT S.EmployeeID
    FROM Salaries S
    WHERE S.Salary > (
        SELECT AVG(S2.Salary)
        FROM Salaries S2
        WHERE S2.DepartmentID = E.DepartmentID
    )
);
```

Pasos:

Consulta Principal:

`SELECT E.Name FROM Employees E`

- La consulta principal selecciona los nombres de los empleados:
- Por cada empleado en `Employees`, se ejecuta la subconsulta.

Subconsulta Correlacionada:

`SELECT S.EmployeeID`

`FROM Salaries S`

`WHERE S.Salary >`

`( SELECT AVG(S2.Salary) FROM Salaries S2 WHERE S2.DepartmentID = E.DepartmentID )`

- La subconsulta busca los `EmployeeID` de los empleados cuyo salario es mayor que el salario promedio de su departamento:
- Aquí, la subconsulta interna calcula el salario promedio de todos los empleados en el mismo departamento (`E.DepartmentID`) del empleado actual en la consulta externa.
- Esta subconsulta interna se ejecuta para cada fila de `Employees`, comparando el salario de cada empleado con el promedio de su departamento.

Filtrando Resultados:

- Si la subconsulta devuelve el `EmployeeID` del empleado actual, significa que su salario es mayor que el promedio de su departamento, y su nombre será incluido en el resultado final.