**Para saber mais: conhecendo o EXCEPT e INTERSECT**

Além do UNION e UNION ALL, em alguns SGBDs podemos ter também os operadores EXCEPT e INTERSECT, que são dois operadores relacionais que são usados em consultas SQL para realizar operações de conjunto em tabelas ou resultados de consultas. Eles permitem que você compare conjuntos de dados entre duas consultas e extraia informações específicas com base nas diferenças ou semelhanças entre esses conjuntos.

**1 - EXCEPT:**

O operador EXCEPT é usado para retornar todas as linhas que estão presentes na primeira consulta (conjunto A) e não estão presentes na segunda consulta (conjunto B). Em outras palavras, ele subtrai o conjunto B do conjunto A. A sintaxe geral é a seguinte:

**SELECT** colunas **FROM** tabelaA

**EXCEPT**

**SELECT** colunas **FROM** tabelaB;

**Exemplo:**

Suponha que você tenha duas tabelas, **TabelaA** e **TabelaB**, e deseje encontrar todas as linhas de **TabelaA** que não estão em **TabelaB**. Você pode usar o EXCEPT da seguinte forma:

**SELECT** \* **FROM** TabelaA

**EXCEPT**

**SELECT** \* **FROM** TabelaB;

**2 - INTERSECT:**

O operador INTERSECT é usado para retornar todas as linhas que estão presentes tanto na primeira consulta (conjunto A) quanto na segunda consulta (conjunto B). Em outras palavras, ele retorna a interseção dos dois conjuntos. A sintaxe geral é a seguinte:

**SELECT** colunas **FROM** tabelaA

**INTERSECT**

**SELECT** colunas **FROM** tabelaB;

**Exemplo:**

Suponha que você tenha duas tabelas, **TabelaA** e **TabelaB**, e deseje encontrar todas as linhas que estão presentes em ambas as tabelas. Você pode usar o INTERSECT da seguinte forma:

**SELECT** \* **FROM** TabelaA

**INTERSECT**

**SELECT** \* **FROM** TabelaB;

**Observações importantes:**

* As duas consultas dentro de EXCEPT ou INTERSECT devem retornar o mesmo número de colunas e essas colunas devem ter tipos de dados compatíveis.
* EXCEPT e INTERSECT geralmente são usados em consultas SELECT, mas também podem ser usados em subconsultas ou em combinação com outros operadores SQL, como UNION.
* Esses operadores são úteis para encontrar diferenças ou semelhanças entre conjuntos de dados, o que pode ser valioso em situações como validação de dados, comparação de listas e muito mais. Eles são especialmente úteis quando você precisa realizar operações complexas de conjunto em dados de tabelas relacionadas.

**Para saber mais: diferenças entre WHERE e HAVING em SQL**

Quando você está aprendendo a linguagem SQL, que é usada para comunicar e gerenciar dados em bancos de dados, você encontrará dois comandos muito importantes: WHERE e HAVING. À primeira vista, eles podem parecer fazer a mesma coisa, pois ambos filtram os dados que você obtém de uma consulta. No entanto, há diferenças cruciais entre eles, que afetam como você vai usar cada um em suas consultas SQL.

## O comando WHERE

O WHERE é usado para filtrar registros (linhas) com base em condições específicas que você define. Ele é aplicado antes que os dados sejam agrupados pela cláusula GROUP BY. Isso significa que você pode usar WHERE para filtrar as linhas individuais de dados antes de qualquer outra operação.

### Exemplo com WHERE:

**SELECT** nome, idade

**FROM** pessoas

**WHERE** idade >= 18;

Neste exemplo, estamos selecionando nomes e idades de uma tabela chamada pessoas, mas só queremos os registros onde a idade é maior ou igual a 18. O WHERE está sendo usado para filtrar esses registros antes que sejam exibidos na resposta à consulta.

## O comando HAVING

Em contrapartida, o HAVING é usado para filtrar dados depois que eles foram agrupados com a cláusula GROUP BY. Isso é útil quando você quer aplicar uma condição de filtro não nas linhas individuais, mas nos grupos resultantes. HAVING é frequentemente usado com funções de agregação, como COUNT(), SUM(), AVG(), etc.

### Exemplo com HAVING:

**SELECT** cidade, COUNT(\*)

**FROM** pessoas

**GROUP** **BY** cidade

**HAVING** COUNT(\*) > 5;

Neste exemplo, estamos contando o número de pessoas em cada cidade na tabela pessoas. Queremos apenas exibir as cidades que têm mais de 5 pessoas. A cláusula HAVING COUNT(\*) > 5 é usada após a criação dos grupos para filtrar apenas aqueles grupos (cidades) que têm mais de 5 pessoas.

## Motivação

Quando você está trabalhando com grandes conjuntos de dados, muitas vezes é necessário organizar e filtrar esses dados para fazer análises significativas. Por exemplo, se você está gerenciando um negócio, pode querer saber quais produtos são mais populares ou quais cidades têm o maior número de clientes. Usar WHERE e HAVING corretamente permite que você obtenha essas informações de maneira eficiente.

* Use WHERE quando precisar de uma condição para filtrar linhas antes de qualquer agrupamento.
* Use HAVING quando precisar aplicar um filtro depois de agrupar as linhas, especialmente quando estiver usando funções de agregação.

## Conclusão

Entender a diferença entre WHERE e HAVING é fundamental para fazer consultas SQL eficientes e precisas. Lembre-se de que WHERE é para filtrar antes do agrupamento e HAVING é para filtrar depois do agrupamento. Ao dominar essas duas cláusulas, você será capaz de manipular e analisar dados de bancos de dados com muito mais confiança e habilidade.