Vem Ser Tech - Dados (5) | #1104

Matheus Muniz
Id: 1104005
MM

Union

UNION e UNION ALL são operadores em SQL que permitem combinar os resultados de duas ou mais consultas SELECT em uma única tabela de resultados. Ambos os operadores são usados para combinar as linhas retornadas pelas consultas SELECT, mas há uma diferença importante entre eles.

O operador UNION remove automaticamente quaisquer linhas duplicadas do resultado, enquanto o operador UNION ALL não as remove e simplesmente une todas as linhas resultantes.

Aqui está um exemplo de como usar o UNION:

```
SELECT coluna1, coluna2 FROM tabela1
UNION
SELECT coluna1, coluna2 FROM tabela2;
```

Neste exemplo, o UNION é usado para combinar os resultados de duas consultas SELECT, que selecionam as mesmas colunas de duas tabelas diferentes, e o operador UNION é usado para remover automaticamente quaisquer linhas duplicadas.

E aqui está um exemplo de como usar o UNION ALL:

```
SELECT coluna1, coluna2 FROM tabela1
UNION ALL
SELECT coluna1, coluna2 FROM tabela2;
```

Neste exemplo, o UNION ALL é usado para combinar os resultados de duas consultas SELECT, que selecionam as mesmas colunas de duas tabelas diferentes, mas o operador UNION ALL não remove quaisquer linhas duplicadas.

O uso do UNION pode ser útil quando queremos combinar os resultados de duas ou mais consultas SELECT e remover automaticamente quaisquer linhas duplicadas. No entanto, o uso do UNION ALL pode ser útil quando desejamos combinar os resultados de duas ou mais consultas SELECT sem remover as linhas duplicadas. É importante notar que o uso do UNION ALL pode ser mais rápido do que o uso do UNION, já que não há o custo extra de se verificar quais linhas são duplicadas.

Abaixo, faremos um exemplo para deixar o conceito um pouco mais concreto.

Suponha que temos duas tabelas, `clientes` e `fornecedores`, que têm a mesma estrutura de colunas:

clientes | **id** | **primeiro** | **ultimo** | **email** | |:------:|:----------:|----------|----------| | 1 | Joao | Silva | joao@email.com | | 2 | Maria | Santos | maria@email.com | | 3 | Jose | Oliveira | jose.o@email.com | | 4 | Ana | Souza | ana@email.com |

fornecedores | **id** | **primeiro** | **ultimo** | **email** | |:------:|:----------:|----------|----------| | 4 | Ana | Souza | ana@email.com | | 5 | Carlos | Lima | carlima@email.com | | 6 | Renata | Pereira | renata@email.com |

Veja que temos nas duas tabelas, um registro idêntico, poderia ser uma pessoa que é cliente e fornecedor.

Podemos usar o operador UNION para combinar as duas tabelas e obter uma lista completa de contatos:

```
SELECT id, primeiro, ultimo, email FROM clientes
UNION
SELECT id, primeiro, ultimo, email FROM fornecedores;
```

Resultando em:

| **id** | **primeiro** | **ultimo** | **email** | |:------:|:----------:|----------|----------| | 1 | Joao | Silva | joao@email.com | | 2 | Maria | Santos | maria@email.com | | 3 | Jose | Oliveira | jose.o@email.com | | 4 | Ana | Souza | ana@email.com | | 5 | Carlos | Lima | carlima@email.com | | 6 | Renata | Pereira | renata@email.com |

No entanto, se quisermos obter uma lista completa de todos os contatos, incluindo aqueles que podem aparecer em ambas as tabelas, podemos usar o operador UNION ALL:

```
SELECT id, primeiro, ultimo, email FROM clientes
UNION ALL
SELECT id, primeiro, ultimo, email FROM fornecedores;
```

Resultando em:

| **id** | **primeiro** | **ultimo** | **email** | |:------:|:----------:|----------|----------| | 1 | Joao | Silva | joao@email.com | | 2 | Maria | Santos | maria@email.com | | 3 | Jose | Oliveira | jose.o@email.com | | 4 | Ana | Souza | ana@email.com | | 4 | Ana | Souza | ana@email.com | | 5 | Carlos | Lima | carlima@email.com | | 6 | Renata | Pereira | renata@email.com |

Observe que o operador UNION ALL simplesmente une todas as linhas resultantes, incluindo quaisquer linhas duplicadas.

> Pensando em performance, quando sabemos que nas duas tabelas não terão registros duplicados, prefira utilizar o UNION ALL.

Além do UNION, que une registros, ainda temos outras operações similares entre tabelas:

INTERSECT: Retorna apenas dados que são iguais nas duas tabelas;

EXCEPT: Retorna apenas dados da primeira tabela que não existem na segunda tabela.

Referências e materiais complementares

https://www.postgresql.org/docs/current/queries-union.html

Próximo Tópico