Larissa Siqueira Sales - 160130883

JOIN

1. O que é JOIN em SQL?

A operação JOIN tem como finalidade combinar dados de duas relações, baseado no relacionamento entre colunas destas tabelas. É usada quando se quer recuperar dados em mais de uma tabela através da igualdade de suas chaves estrangeiras, agregando as tabelas de acordo com um campo que faça sentido a ambas.

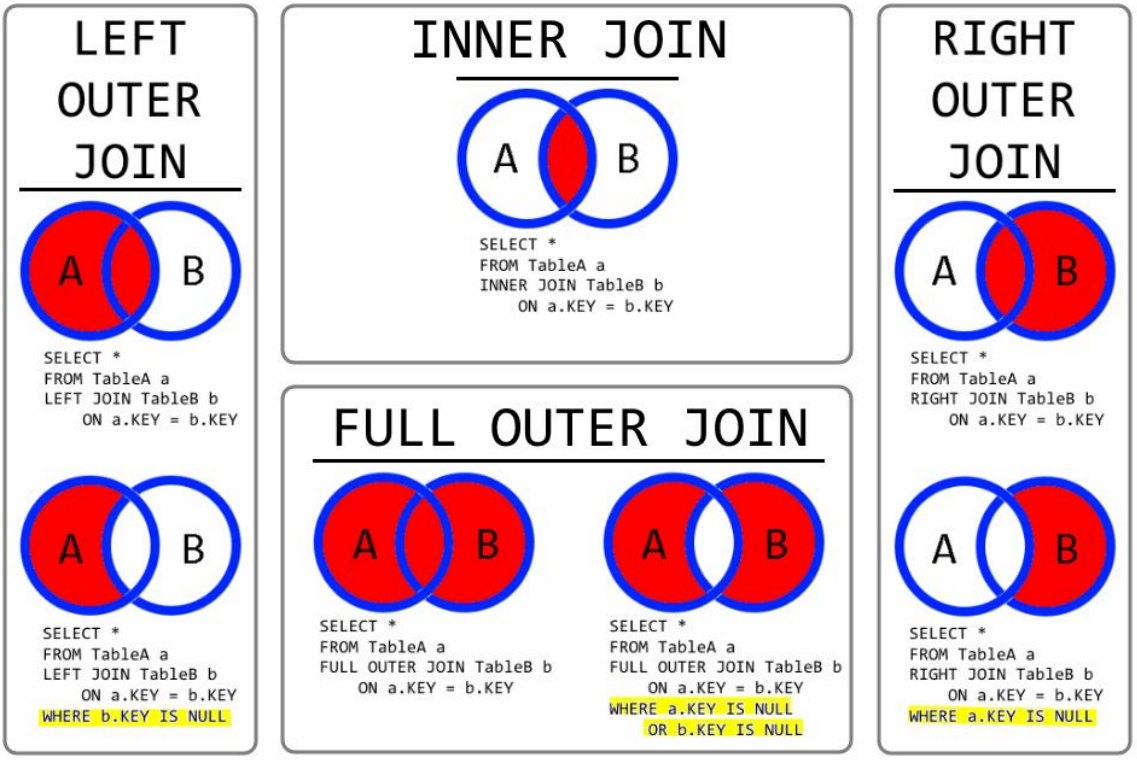
2. Vantagens

Algumas das vantagens do uso de JOIN é permitir que o SQL fique mais compreensível, diminuindo a complexidade da consulta, criar uma visão mais lógica para facilitar a modelagem, retornar a melhor forma de juntar os valores relacionados nas tabelas e melhorar o acesso aos dados inseridos.

3. Desvantagens

Algumas das desvantagens é a possibilidade de aumentar a complexidade de acesso a alguma informação, ser mais difícil de administrar o banco de dados, limitar o acesso do usuário (que em alguns casos pode ser ruim)

4. Tipos



a. INNER JOIN

Tem como objetivo juntar duas ou mais tabelas de acordo com atributos comuns e é utilizado as chaves primárias e estrangeiras.

Exemplo:

SELECT C.NOMECARGO [CARGO], F.NOMEFUNCIONARIO AS [FUNCIONÁRIO], F.SALARIOFUNCIONARIO AS [SALÁRIO]
FROM CARGO AS C
INNER JOIN FUNCIONARIO AS F ON C.IDCARGO = F.IDCARGO

b. LEFT JOIN

Tem como objetivo juntar duas tabelas de acordo com atributos comuns entre elas e os dados da tabela da esquerda que não são relacionados com os dados da tabela da direita. Caso não existam dados associados, serão retornados valores nulos.

Exemplo:

```
SELECT C.NOMECARGO [CARGO], F.NOMEFUNCIONARIO AS [FUNCIONÁRIO],
F.SALARIOFUNCIONARIO AS [SALÁRIO]
FROM CARGO AS C
LEFT JOIN FUNCIONARIO AS F ON C.IDCARGO = F.IDCARGO
```

c. LEFT EXCLUDING JOIN

Tem como objetivo juntar duas tabelas de acordo os dados da tabela da esquerda que não estão na tabela da direita.

Exemplo:

```
SELECT a.Nome, b.Nome
FROM TabelaA as A
LEFT JOIN TabelaB as B
        on a.Nome = b.Nome
WHERE b.Nome is null
```

d. RIGHT JOIN

Tem como objetivo juntar duas tabelas de acordo com atributos comuns entre elas e os dados da tabela da direita que não são relacionados com os dados da tabela da esquerda. Caso não existam dados associados, serão retornados valores nulos.

Exemplo:

```
SELECT C.NOMECARGO [CARGO], F.NOMEFUNCIONARIO AS [FUNCIONÁRIO],
F.SALARIOFUNCIONARIO AS [SALÁRIO]
FROM FUNCIONARIO AS F
```

```
RIGHT JOIN CARGO AS C ON F.IDCARGO = C.IDCARGO
```

e. RIGHT EXCLUDING JOIN

Tem como objetivo juntar duas tabelas de acordo os dados da tabela da direita que não estão na tabela da esquerda.

Exemplo:

```
SELECT a.Nome, b.Nome
FROM TabelaA as A
RIGHT JOIN TabelaB as B
   on a.Nome = b.Nome
WHERE a.Nome is null
```

f. FULL JOIN (FULL OUTER JOIN ou OUTER JOIN)

Tem como objetivo retornar todas as linhas de dados da tabela da esquerda e da tabela da direita. Caso uma linha de uma tabela não esteja associada a uma linha da outra tabela, os valores das colunas retornarão nulos.

Exemplo:

```
SELECT C.NOMECARGO [CARGO], F.NOMEFUNCIONARIO AS [FUNCIONÁRIO],
F.SALARIOFUNCIONARIO AS [SALÁRIO]
FROM FUNCIONARIO AS F
FULL JOIN CARGO AS C ON F.IDCARGO = C.IDCARGO
```

g. OUTER EXCLUDING JOIN

Tem como objetivo retornar todos os registros que estão na tabela da esquerda, mas não estão na tabela da direita e todos os registros que estão na tabela da direita, mas que não estão na tabela da esquerda.

Exemplo:

```
SELECT a.Nome, b.Nome
FROM TabelaA as A
FULL OUTER JOIN TabelaB as B
    on a.Nome = b.Nome
WHERE a.Nome is null or b.Nome is null
```