UNIVERSIDADE DE BRASÍLIA

Faculdade do Gama

Sistemas de Banco de Dados 1

Pesquisa (JOIN) com Exercício Extra 2 da Aula 10

Nome: Artur Rodrigues Sousa Alves

Matrícula: 211043638

Brasília, DF

2023

Em SQL, JOIN é uma cláusula utilizada para combinar registros de duas ou mais tabelas com base em uma coluna comum entre elas. Ela permite que você recupere dados relacionados de várias tabelas em uma única consulta. As vantagens e desvantagens da JOIN, assim como os seus tipos de junções, são:

Vantagens:

1. Recuperação de dados relacionados. JOIN permite obter informações de várias tabelas em uma única consulta, evitando a necessidade de consultas separadas e melhorando a eficiência das consultas.
2. Consistência de dados. JOIN garante a integridade referencial dos dados, ou seja, permite que você estabeleça relacionamentos entre tabelas usando tanto chaves primárias quanto estrangeiras.
3. Flexibilidade na consulta. Utilizando JOIN, você pode combinar dados de diferentes tabelas com base em condições específicas, permitindo crialçai de consultas complexas para atender às suas necessidades.
4. Simplificação do modelo de dados. O uso das junções JOIN permite que você normalize seu modelo de dados, dividindo as informações em várias tabelas relacionadas. Isso pode resultar em um esquema de banco de dados mais organizada e de fácil manutenção.

Desvantagens:

1. Complexidade. À medida que o número de tabelas envolvidas aumentam, juntamento com as condições de junção, a complexidade das consultas JOIN tendem a aumentar, tornando cada vez mais difícil de entender, manipular e manter as consultas JOIN.
2. Desempenho. Dependendo do volume de dados e da estrutura das tabelas, as consultas JOIN podem exigir recursos computacionais significativos e, em alguns casos, podem resultar em consultas lentas. Portanto, é importante otimizar as consultas e usar índices apropriados para melhorar e manter o desempenho das consultas.
3. Risco de resultados indesejados. Se as junções JOIN forem configuradas incorretamente ou as condições de junção não forem adequadas ao problema encontrado, pode haver um sério risco da junção retornar resultados incorretos ou indesejados. Portanto há a necessidade de uma atenção maior na hora de realizar as consultas.

- Para melhor entendimento e compreensão dos tipos de junção JOIN, vamos considerar as duas seguintes tabelas:

```
CREATE TABLE TabelaA(
  Nome varchar(50) NULL
)

GO

CREATE TABLE TabelaB(
  Nome varchar(50) NULL
)
```

Em seguida, vamos considerar as seguintes adições de valores nas respectivas tabelas para desmonstração das junções:

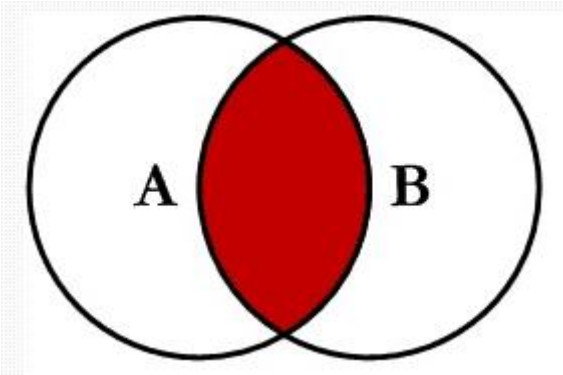
```
INSERT INTO TabelaA VALUES('Fernanda')
INSERT INTO TabelaA VALUES('Josefa')
INSERT INTO TabelaA VALUES('Luiz')
INSERT INTO TabelaA VALUES('Fernando')

INSERT INTO TabelaB VALUES('Carlos')
INSERT INTO TabelaB VALUES('Manoel')
INSERT INTO TabelaB VALUES('Luiz')
INSERT INTO TabelaB VALUES('Fernando')
```

Portanto agora, vamos para os tipos de junções:

Tipos de junção JOIN:

1. **INNER JOIN**: Ela retorna apenas os registros que possuem correspondências nas duas tabelas envolvidas na junção, ou seja, apenas os registros comuns entre as tabelas.
   - Representação em conjuntos da INNER JOIN:



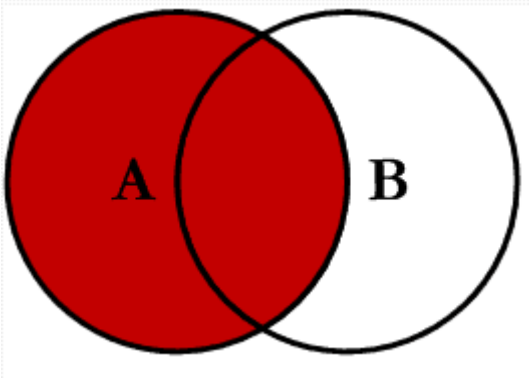
- Exemplo de consulta com INNER JOIN:

```sql
SELECT a.Nome, b.Nome
FROM TabelaA as A
INNER JOIN TabelaB as B
                on a.Nome = b.Nome
```

- Resultado da consulta:

2. **LEFT JOIN:** Retorna todos os registros da tabela à esquerda da primeira tabela mencionada e os registros correspondentes da tabela à direita (segunda tabela mencionada).

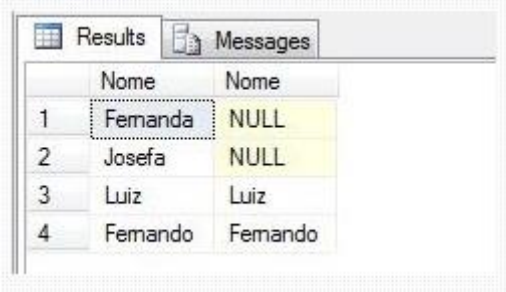
- Representação em conjuntos da LEFT JOIN:



- Exemplo de consulta com LEFT JOIN:

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

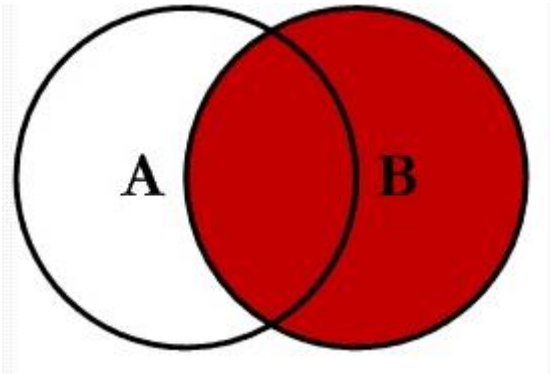
- Resultado da consulta com LEFT JOIN:



	Nome	Nome
1	Fernanda	NULL
2	Josefa	NULL
3	Luiz	Luiz
4	Fernando	Fernando

3. **<u>RIGHT JOIN:</u>** Retorna todos os registros da tabela à direirta (segunda tabela mencionada) e os registros correspondentes da tabela à esquerda (primeira tabela mencionada).

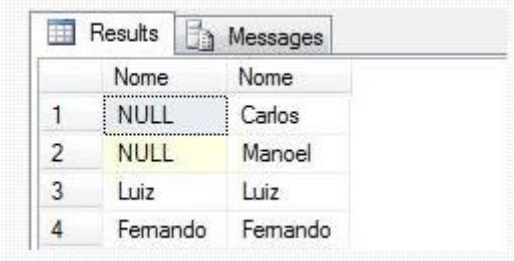
- Representação em conjuntos da RIGHT JOIN:



- Exemplo de consulta com RIGHT JOIN:

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

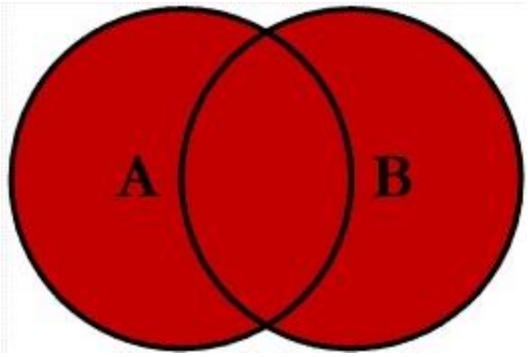
- Resultado da consulta com RIGHT JOIN:



	Nome	Nome
1	NULL	Carlos
2	NULL	Manoel
3	Luiz	Luiz
4	Fernando	Fernando

4. **<u>OUTER JOIN(FULL JOIN):</u>** Retorna todos os registros de ambas as tabelas, incluindo os registros comuns e os não comuns. Portanto, o FULL JOIN retornará todos os registros da tabela da direita quanto da esquerda.

   - Representação em conjuntos da OUTER JOIN:



   - Exemplo de consulta com OUTER JOIN:

```sql
SELECT a.Nome, b.Nome
FROM TabelaA as A
FULL OUTER JOIN TabelaB as B
    on a.Nome = b.Nome
```

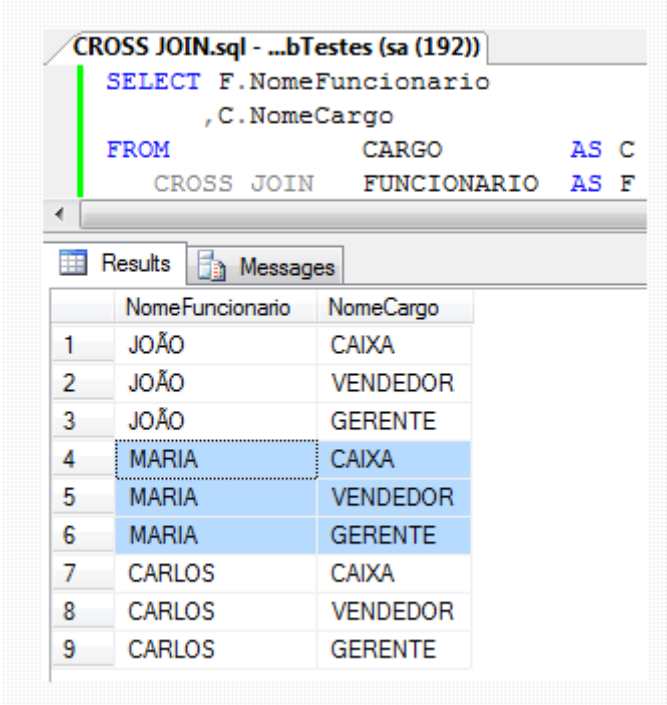
   - Resultado da consulta com OUTER JOIN:

Results | Messages

	Nome	Nome
1	Fernanda	NULL
2	Josefa	NULL
3	Luiz	Luiz
4	Fernando	Fernando
5	NULL	Carlos
6	NULL	Manoel

5. **<u>CROSS JOIN:</u>** Retorna o produto cartesiano entre as tabelas, ou seja, todas as combinações possíveis entre os registros das tabelas envolvidas.

   - Representação de um CROSS JOIN:



	NomeFuncionario	NomeCargo
1	JOÃO	CAIXA
2	JOÃO	VENDEDOR
3	JOÃO	GERENTE
4	MARIA	CAIXA
5	MARIA	VENDEDOR
6	MARIA	GERENTE
7	CARLOS	CAIXA
8	CARLOS	VENDEDOR
9	CARLOS	GERENTE