# Linguagem SQL
# Junções

# JUNÇÕES (JOIN)

Associações (ou junções) de tabelas podem ser utilizadas para diversas finalidades, como converter em informação os dados encontrados em duas ou mais tabelas.

Além disso, as tabelas podem ser combinadas por meio de uma condição ou um grupo de condições de junção. Por exemplo, podemos usar as chaves estrangeiras como condição para relacionar as tabelas.

Esse tipo de operação pode ser feito por meio das cláusulas **WHERE** e **JOIN**.

É importante salientar que as tabelas devem ser associadas em pares, embora seja possível usar um único comando para juntar várias tabelas. Uma das formas mais usadas é a **associação da chave primária da primeira tabela com a chave estrangeira da segunda**.

Diferentes tipos de associação podem ser escritos com a ajuda das cláusulas JOIN e WHERE. Outro exemplo: podemos obter apenas os dados relacionados entre duas tabelas associadas. Também podemos combinar duas tabelas de forma que seus dados relacionados e os não-relacionados sejam obtidos.

# JUNÇÕES (JOIN)

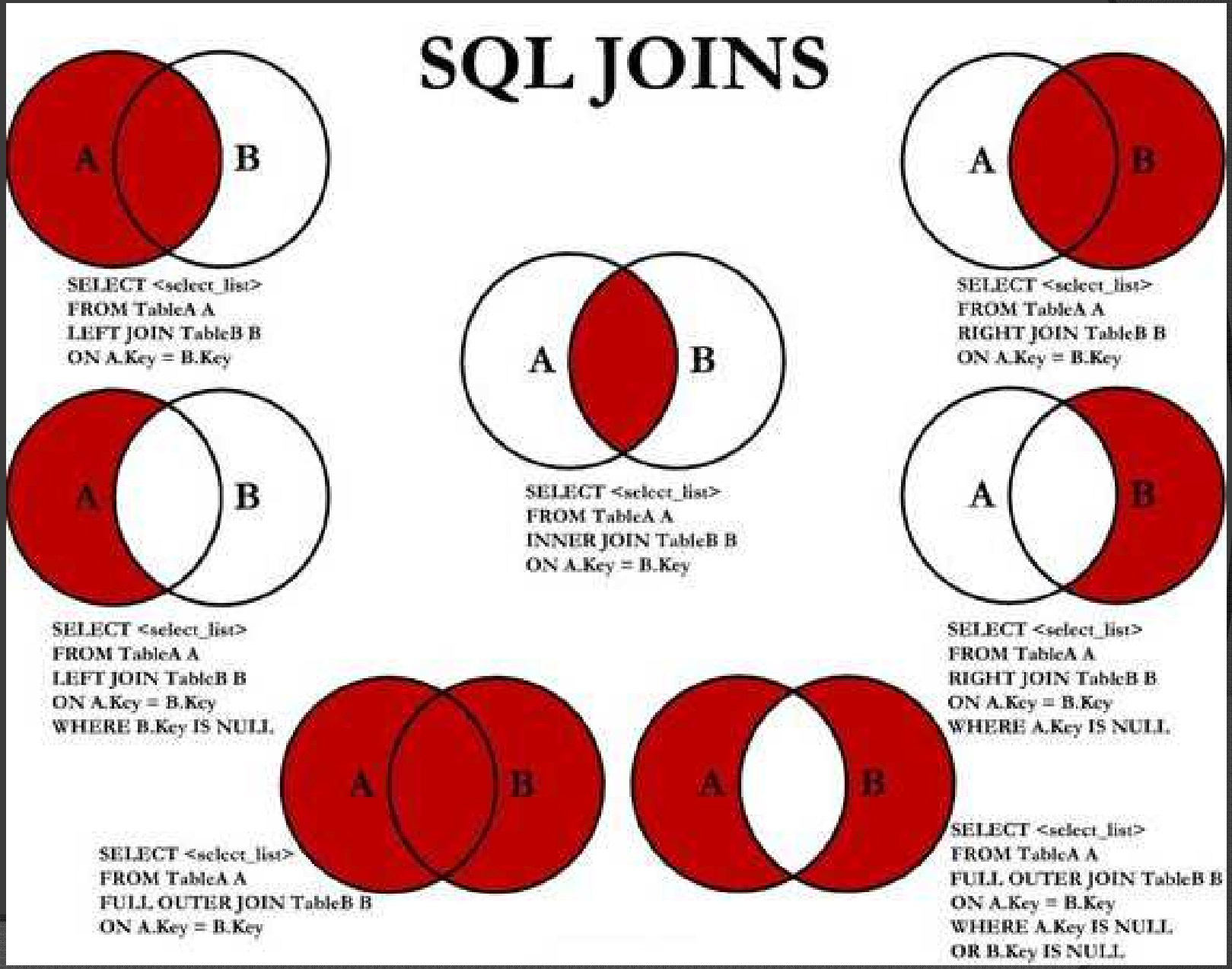
As tabelas podem, ainda, ser associadas de modo que sejam gerados não apenas dados relacionados entre elas, mas também dados não relacionados da tabela encontrada à esquerda ou à direita da cláusula JOIN.

Também podemos associar as tabelas por meio das cláusulas JOIN e WHERE para a obtenção do produto cartesiano, que nada mais é do que um conjunto de resultados com todas as linhas que são geradas a partir desta associação.

Muitos desenvolvedores encontram dificuldade de saber qual resultado é retornado por cada modelo de JOIN no SQL e, portanto, quando devem utilizar cada um. Para facilitar esse entendimento, a figura seguinte traz uma representação gráfica, baseada na ***Teoria dos Conjuntos***, muito conhecida na matemática.

Nessa imagem, temos a representação de duas tabelas (A e B) e o resultado esperado por cada tipo de join (a área em vermelho representa os registros retornados pela consulta).

# JUNÇÕES (JOIN)



# JUNÇÕES (JOIN)

A cláusula JOIN permite que os dados de várias tabelas sejam combinados com base na **relação existente entre elas**. Por meio dessa cláusula, os dados de uma tabela são usados para selecionar os dados pertencentes à outra tabela.

Com a cláusula JOIN, podemos especificar quais colunas das tabelas serão associadas. Para isso, será preciso definir uma chave estrangeira de uma tabela e a chave relacionada em outra tabela.

Os valores pertencentes às colunas das tabelas associadas podem ser comparados entre si por meio de um operador lógico definido pela cláusula JOIN e usada pelo operador ON, como o sinal de igual (=).

# JUNÇÕES (JOIN)

Veja o exemplo simplificado de uma sintaxe de cláusula de associação:

FROM nome_primeira_tabela **tipo_de_associação** nome_segunda_tabela [**ON** (**condição_de_associação**) ]

**Onde:**

• **tipo_de_associação:** Permite identificar uma das seguintes associações: inner join, outer join e cross join (que serão descritas em detalhes mais adiante);

• **condição_de_associação:** Define um critério para avaliar duas linhas de dados que já estão associadas.

A forma mais indicada para a especificação de associações é a cláusula FROM, que permite que as condições JOIN sejam identificadas em relação às condições de busca referenciadas na cláusula WHERE.

# JUNÇÕES (JOIN)

As explicações a seguir utilização as seguintes tabelas:

| CLIENTES |
| --- |
| IDCLIENTE<br>NOME<br>STATUS |

| PEDIDOS |
| --- |
| IDPEDIDO<br>IDCLIENTE<br>DESCRIÇÃO<br>VALOR |

Exemplo simples da associação de tabelas

```
SELECT *
FROM CLIENTES AS C
JOIN PEDIDOS AS P ON C.IDCLIENTE = P.ICLIENTE;
```

Utilização de “alias” para identificar tabelas de forma mais simples (explicação no slide seguinte)

Podemos fazer também o mesmo especificando quais tabelas serão retornadas:

```
SELECT C.NOME, C.STATUS, P.DESCRICAO, P.VALOR
FROM CLIENTES AS C
JOIN PEDIDOS AS P ON C.IDCLIENTE = P.IDCLIENTE;
```

Repare que é possível consultar colunas de outras tabelas **no mesmo SELECT**, desde que, é claro, se faça referência a esta tabela na cláusula JOIN.

# JUNÇÕES (JOIN)

Repare que é possível consultar colunas de outras tabelas **no mesmo SELECT**, desde que, é claro, se referencie esta tabela no JOIN em questão.

*É normal encontrarmos colunas com o mesmo nome, vindos de tabelas diferentes*. Para que não haja confusão e consigamos a identificação correta da procedência dos dados, *devemos qualificar o nome de uma coluna com o nome da tabela da qual ela é originada*.

É recomendado que se use esta qualificação principalmente quando usamos o SELECT, já que este comando não permite identificar a origem de cada coluna. **Os aliases (apelidos) da tabela também auxiliam nessa identificação**.

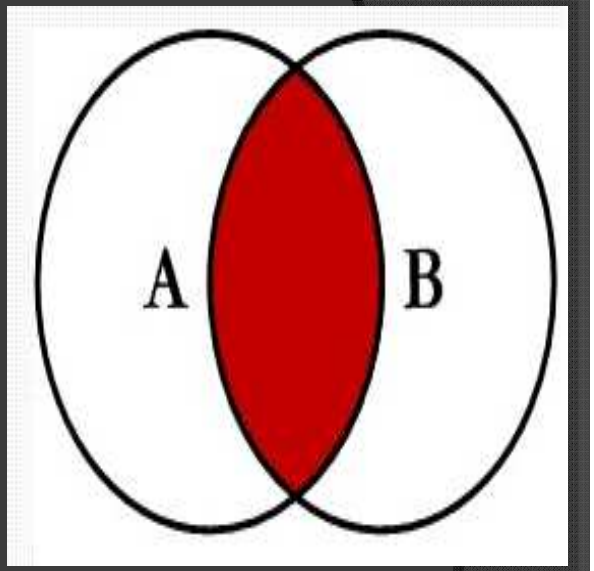
Veja um exemplo usando a cláusula WHERE para especificar uma condição de associação, já com as recomendações acima:

```
SELECT CLIENTES.IDCLIENTE, CLIENTES.NOME, PEDIDOS.DESCRICAO,
PEDIDOS.VALOR
FROM CLIENTES, PEDIDOS
WHERE CLIENTES.IDCLIENTE = PEDIDOS.IDCLIENTE AND PEDIDOS.VALOR > 50.00;
```

# JUNÇÕES (JOIN)

1. **INNER JOIN**

A cláusula INNER JOIN permite usar um operador de comparação para comparar os valores de colunas provenientes de tabelas associadas. Através desta cláusula, os registros de duas tabelas são usados para que sejam gerados os dados relacionados de ambas. As cláusulas WHERE e FROM são utilizadas para especificar esse tipo de associação. Este é o tipo mais simples, o mais fácil de compreender e o mais comum. Esta consulta retornará todos os registros da tabela esquerda (tabela A) que possuem correspondência com a tabela direita (tabela B).




No exemplo a seguir serão utilizadas as tabelas Cargo e Funcionário descritas ao lado.

Repare que a coluna IdCargo consta nas duas tabelas, porém, com finalidades distintas: enquanto na tabela Cargo, ela é chave primária, na tabela Funcionário ela é chave estrangeira.

| | IdCargo | NomeCargo |
|---|---|---|
| 1 | 1 | Programador Jr. |
| 2 | 2 | Web Designer Pl. |
| 3 | 3 | Programador Pl. |
| 4 | 4 | DBA Jr. |
| 5 | 5 | Programador Sr. |

| | IdFuncionario | IdCargo | NomeFuncionario | SalarioFuncionario |
|---|---|---|---|---|
| 1 | 1 | 2 | Tiririca | 2500.00 |
| 2 | 2 | 1 | Zé da Pizza | 2250.00 |
| 3 | 3 | 3 | Tiozão do Gás | 2750.00 |
| 4 | 4 | 4 | Adalberto do Sacolão | 2300.00 |
| 5 | 5 | 1 | Marisa da Horta | 2500.00 |

# JUNÇÕES (JOIN)

1. **INNER JOIN**

Assim, a associação entre as tabelas é feita pela coluna IdCargo e é possível identificar os cargos existentes e o nome dos funcionários que desempenham cada um deles.

É utilizada a cláusula INNER JOIN para se obter os dados relacionados das duas tabelas, para que sejam retornados todos os cargos ocupados pelos funcionários, bem como todos os funcionários que desempenham algum cargo.

Observe como isso é feito no script abaixo:

```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM CARGO AS C
INNER JOIN FUNCIONARIO AS F ON C.IDCARGO = F.IDCARGO;
```

Nossa consulta haverá o seguinte retorno:

| | CARGO | FUNCIONÁRIO | SALÁRIO |
|---|---|---|---|
| 1 | Web Designer Pl. | Tiririca | 2500.00 |
| 2 | Programador Jr. | Zé da Pizza | 2250.00 |
| 3 | Programador Pl. | Tiozão do Gás | 2750.00 |
| 4 | DBA Jr. | Adalberto do Sacolão | 2300.00 |
| 5 | Programador Jr. | Marisa da Horta | 2500.00 |

# JUNÇÕES (JOIN)

1. **INNER JOIN**

Podemos usar também a cláusula WHERE e termos o mesmo resultado. O script ficará assim:

```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM CARGO AS C, FUNCIONARIO AS F
WHERE C.IDCARGO = F.IDCARGO;
```

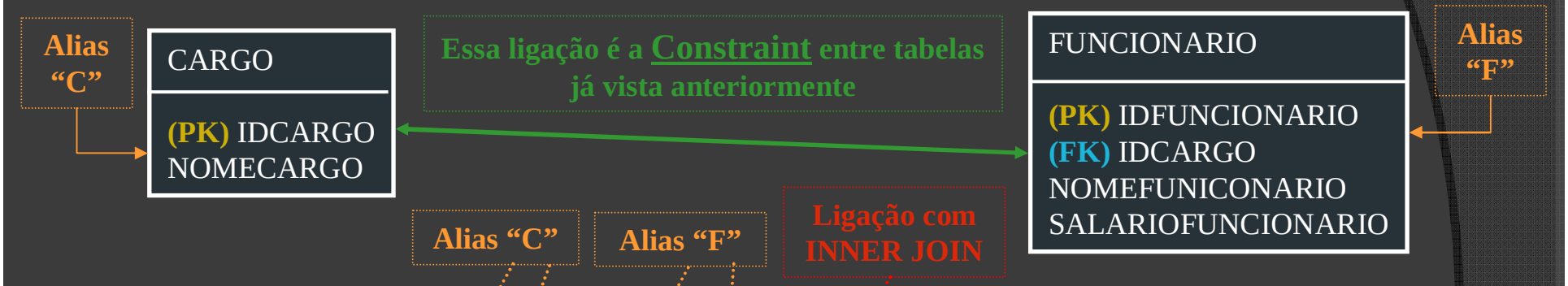
Independente da cláusula usada, o relacionamento entre as tabelas será feito, como dito anteriormente, por meio da chave primária da tabela Cargo e da chave estrangeira da tabela Funcionário.

Observe que, no retorno de nossas consultas, o único cargo que não é exibido pra nós é o "Programador Sr." já que não há nenhum funcionário relacionado a este cargo.

# JUNÇÕES (JOIN)

1. **INNER JOIN**

**Exemplo no formato visual:**




```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM CARGO AS C
INNER JOIN FUNCIONARIO AS F ON C.IDCARGO = F.IDCARGO;
```

***Primeira forma* de uso (mais profissional e performática)**

```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM CARGO AS C, FUNCIONARIO AS F
WHERE C.IDCARGO = F.IDCARGO;
```

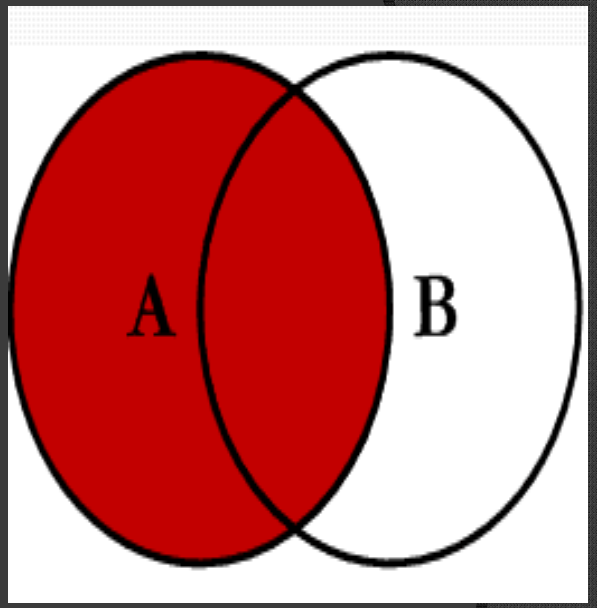
**Ligação através de WHERE**

***Segunda forma* de uso (menos performática)**

# JUNÇÕES (JOIN)

## 2. LEFT JOIN

A cláusula LEFT JOIN (ou LEFT OUTER JOIN) permite obter não apenas os **dados relacionados de duas tabelas** (Tabela A e Tabela B), mas também os dados não relacionados encontrados na tabela à esquerda da cláusula JOIN (Tabela A). Caso não existam dados relacionados entre as tabelas à esquerda (Tabela A) e a direita (Tabela B) do JOIN, os valores resultantes de todas as colunas da lista de seleção da tabela à direita serão nulos. Resumindo: O LEFT JOIN tem como resultado todos os registros que estão na tabela A (mesmo que não estejam na tabela B) e os registros da tabela B que são comuns à tabela A.




No exemplo a seguir serão utilizadas as tabelas Cargo e Funcionário descritas ao lado (as mesmas do exemplo de INNER JOIN).

| | IdCargo | NomeCargo |
|---|---|---|
| 1 | 1 | Programador Jr. |
| 2 | 2 | Web Designer Pl. |
| 3 | 3 | Programador Pl. |
| 4 | 4 | DBA Jr. |
| 5 | 5 | Programador Sr. |

| | IdFuncionario | IdCargo | NomeFuncionario | SalarioFuncionario |
|---|---|---|---|---|
| 1 | 1 | 2 | Tiririca | 2500.00 |
| 2 | 2 | 1 | Zé da Pizza | 2250.00 |
| 3 | 3 | 3 | Tiozão do Gás | 2750.00 |
| 4 | 4 | 4 | Adalberto do Sacolão | 2300.00 |
| 5 | 5 | 1 | Marisa da Horta | 2500.00 |

# JUNÇÕES (JOIN)

2. **LEFT JOIN**

Associação entre as tabelas é feita pela coluna IdCargo.

É utilizada a cláusula LEFT JOIN para se obter os dados relacionados das duas tabelas e também aqueles existentes na tabela à esquerda da cláusula LEFT JOIN (Tabela A) que não possuem correspondência na tabela à direita da mesma cláusula (Tabela B). Como dito anteriormente, o único cargo que não contém funcionário vinculado a ele é o Programador Sr. Para obtermos mesmo assim esse cargo, usamos a cláusula LEFT JOIN à esquerda do sinal de igual (=), como no script abaixo:

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

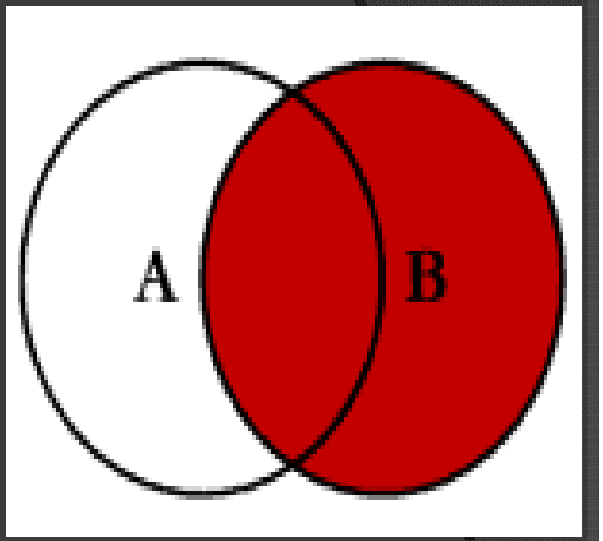
Nossa consulta haverá o seguinte retorno:

| CARGO | FUNCIONÁRIO | SALÁRIO |
|---|---|---|
| Programador Jr. | Zé da Pizza | 2250.00 |
| Programador Jr. | Marisa da Horta | 2500.00 |
| Web Designer Pl. | Tiririca | 2500.00 |
| Programador Pl. | Tiozão do Gás | 2750.00 |
| DBA Jr. | Adalberto do Sacolão | 2300.00 |
| Programador Sr. | NULL | NULL |

# JUNÇÕES (JOIN)

3. **RIGHT JOIN**

Ao contrário do LEFT JOIN, a cláusula RIGHT JOIN ou (RIGHT OUTER JOIN) retorna todos os dados encontrados na tabela à direita de JOIN (tabela B). Caso não existam dados associados entre as tabelas à esquerda (Tabela A) e à direita de JOIN (Tabela B), serão retornados valores nulos. Resumindo: o RIGHT JOIN possui como resultado todos os registros que estão na tabela B (mesmo que não estejam na tabela A) e os registros da tabela A que são comuns à tabela B.




No exemplo a seguir serão utilizadas as tabelas Cargo e Funcionário descritas ao lado (as mesmas do exemplo de INNER JOIN).

| | IdCargo | NomeCargo |
|---|---|---|
| 1 | 1 | Programador Jr. |
| 2 | 2 | Web Designer Pl. |
| 3 | 3 | Programador Pl. |
| 4 | 4 | DBA Jr. |
| 5 | 5 | Programador Sr. |

| | IdFuncionario | IdCargo | NomeFuncionario | SalarioFuncionario |
|---|---|---|---|---|
| 1 | 1 | 2 | Tiririca | 2500.00 |
| 2 | 2 | 1 | Zé da Pizza | 2250.00 |
| 3 | 3 | 3 | Tiozão do Gás | 2750.00 |
| 4 | 4 | 4 | Adalberto do Sacolão | 2300.00 |
| 5 | 5 | 1 | Marisa da Horta | 2500.00 |

# JUNÇÕES (JOIN)

3. **RIGHT JOIN**

Associação entre as tabelas é feita pela coluna IdCargo.

Suponhamos que a posição das tabelas usadas nos exemplos anteriores foi trocada. Se mesmo assim desejamos obter o mesmo resultado obtido anteriormente, podemos usar a cláusula RIGHT JOIN, assim iremos conseguir tanto os dados relacionados como os não relacionados disponíveis na tabela à direita da cláusula JOIN:

```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM FUNCIONARIO AS F
RIGHT JOIN CARGO AS C ON N F.IDCARGO = C.IDCARGO;
```

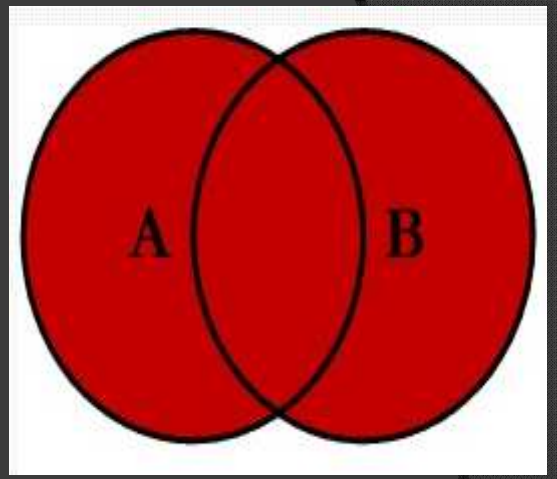
Nossa consulta haverá o seguinte retorno:

```
+-------------------+---------------+--------+
| cargo             | func          | salario |
+-------------------+---------------+--------+
| Web Designer PI| tiririca         | 2500   |
| Programador Jr   | ze da pizza    | 2250   |
| programador PI   | Tiozao do gas| 2750   |
| DBA Jr           | Adalberto      | 2300   |
| Programador Jr   | Marisa         | 2500   |
+------------------+----------------+--------+
```

**OBS.:** Observe que somente houve a inversão da ordem das tabelas nas cláusulas FROM e RIGHT JOIN em relação ao exemplo com LEFT JOIN

# JUNÇÕES (JOIN)

## 4. FULL OUTER JOIN




O FULL OUTER JOIN (também conhecido por OUTER JOIN ou FULL JOIN) tem como resultado todos os registros que estão na tabela A e todos os registros da tabela B. Caso uma linha de dados não esteja associada a qualquer linha da outra tabela, os valores das colunas a lista de seleção serão nulos. Caso contrário, os valores obtidos serão baseados nas tabelas usadas como referência.

**ATENÇÃO:** FULL OUTER JOIN não é suportado pelo MySQL. Em outros BDs (Oracle, SQL Server etc.) ele funciona normalmente.

No exemplo a seguir serão utilizadas as tabelas Cargo e Funcionário descritas ao lado (as mesmas do exemplo de INNER JOIN).

| | IdCargo | NomeCargo |
|---|---|---|
| 1 | 1 | Programador Jr. |
| 2 | 2 | Web Designer Pl. |
| 3 | 3 | Programador Pl. |
| 4 | 4 | DBA Jr. |
| 5 | 5 | Programador Sr. |

| | IdFuncionario | IdCargo | NomeFuncionario | SalarioFuncionario |
|---|---|---|---|---|
| 1 | 1 | 2 | Tiririca | 2500.00 |
| 2 | 2 | 1 | Zé da Pizza | 2250.00 |
| 3 | 3 | 3 | Tiozão do Gás | 2750.00 |
| 4 | 4 | 4 | Adalberto do Sacolão | 2300.00 |
| 5 | 5 | 1 | Marisa da Horta | 2500.00 |

# JUNÇÕES (JOIN)

## 4. FULL OUTER JOIN

Associação entre as tabelas é feita pela coluna IdCargo.

Nesse código, a palavra reservada OUTER é opcional. Portanto, se ela for removida, deixando apenas a expressão FULL JOIN, o resultado será o mesmo:

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

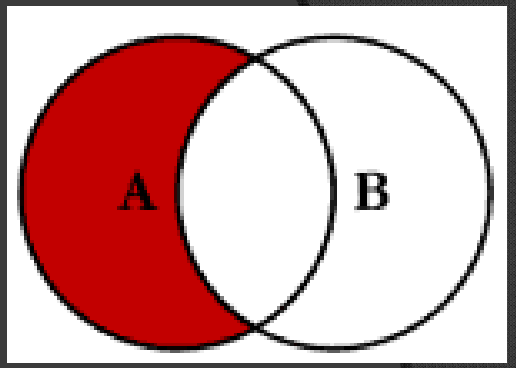
Nossa consulta haverá o seguinte retorno:

| CARGO | FUNCIONÁRIO | SALÁRIO |
|---|---|---|
| Web Designer Pl. | Tiririca | 2500.00 |
| Programador Jr. | Zé da Pizza | 2250.00 |
| Programador Pl. | Tiozão do Gás | 2750.00 |
| DBA Jr. | Adalberto do Sacolão | 2300.00 |
| Programador Jr. | Marisa da Horta | 2500.00 |
| Programador Sr. | NULL | NULL |

Caso uma linha de dados não esteja associada a qualquer linha da outra tabela, os valores das colunas a lista de seleção serão nulos

# JUNÇÕES (JOIN)

5. **LEFT Excluding JOIN**



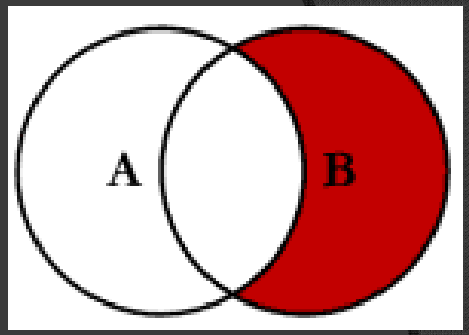

Esta consulta retornará apenas os registros da tabela à esquerda (Tabela A) que não têm correspondência com a tabela à direita do LEFT JOIN (Tabela B).

```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM CARGO AS C
LEFT JOIN  FUNCIONARIO AS F ON C.IDCARGO = F.IDCARGO;
WHERE F.IDCARGO IS NULL;
```

Essa cláusula WHERE exclui a correspondência existente entre as tabelas

# JUNÇÕES (JOIN)

6. **RIGHT Excluding JOIN**



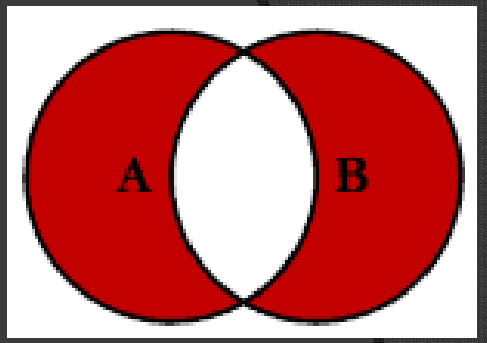

Esta consulta retornará apenas os registros da tabela à direita (Tabela B) que não têm correspondência com a tabela à esquerda do RIGHT JOIN (Tabela A).

```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM FUNCIONARIO AS F
RIGHT JOIN CARGO AS C ON N F.IDCARGO = C.IDCARGO
WHERE C.IDCARGO IS NULL;
```

Essa cláusula WHERE exclui a correspondência existente entre as tabelas

# JUNÇÕES (JOIN)

7. **FULL OUTER Excluding JOIN**




Esta consulta retornará todos os registros das duas tabelas (Tabela A e Tabela B) que não têm correspondência.

**ATENÇÃO:** FULL OUTER JOIN não é suportado pelo MySQL. Em outros BDs (Oracle, SQL Server etc.) ele funciona normalmente.

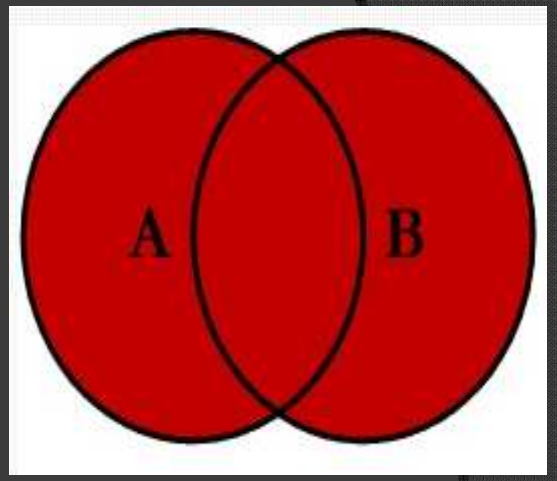
```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM FUNCIONARIO AS F
FULL JOIN CARGO AS C ON F.IDCARGO = C.IDCARGO
WHERE F.IDCARGO IS NULL OR C.IDCARGO IS NULL;
```

Essa cláusula WHERE exclui a correspondência existente entre as tabelas

# JUNÇÕES (JOIN)

8. **CROSS JOIN**

Todos os dados da tabela à esquerda de JOIN (Tabela A) são cruzados com os dados da tabela à direita de JOIN (tabela B) por meio do CROSS JOIN, também conhecido como *produto cartesiano*. É possível cruzarmos informações de duas ou mais tabelas.




Para facilitar a compreensão a respeito desse tipo de associação, serão utilizadas as tabelas ao lado como exemplo.

| | IdCargo | NomeCargo |
|---|---|---|
| 1 | 1 | Programador Jr. |
| 2 | 2 | Web Designer Pl. |
| 3 | 3 | Programador Pl. |
| 4 | 4 | DBA Jr. |
| 5 | 5 | Programador Sr. |

| | IdFuncionario | IdCargo | NomeFuncionario | SalarioFuncionario |
|---|---|---|---|---|
| 1 | 1 | 2 | Tiririca | 2500.00 |
| 2 | 2 | 1 | Zé da Pizza | 2250.00 |
| 3 | 3 | 3 | Tiozão do Gás | 2750.00 |
| 4 | 4 | 4 | Adalberto do Sacolão | 2300.00 |
| 5 | 5 | 1 | Marisa da Horta | 2500.00 |

# JUNÇÕES (JOIN)

## 8. CROSS JOIN

Associação entre as tabelas é feita pela coluna IdCargo.

Caso a intenção seja exibir os dados de modo que todos os funcionários tenham todos os cargos e vice-versa. Para isso, deve-se usar o CROSS JOIN, como no exemplo a seguir:

```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM CARGO AS C
CROSS JOIN FUNCIONARIO AS F
ORDER BY 1;
```

O resultado será o seguinte:

| | CARGO | FUNCIONÁRIO | SALÁRIO |
|---|---|---|---|
| 1 | DBA Jr. | Tiririca | 2500.00 |
| 2 | DBA Jr. | Zé da Pizza | 2250.00 |
| 3 | DBA Jr. | Tiozão do Gás | 2750.00 |
| 4 | DBA Jr. | Adalberto do Sacolão | 2300.00 |
| 5 | DBA Jr. | Marisa da Horta | 2500.00 |
| 6 | Programador Jr. | Tiririca | 2500.00 |
| 7 | Programador Jr. | Zé da Pizza | 2250.00 |
| 8 | Programador Jr. | Tiozão do Gás | 2750.00 |
| 9 | Programador Jr. | Adalberto do Sacolão | 2300.00 |
| 10 | Programador Jr. | Marisa da Horta | 2500.00 |
| 11 | Programador Pl. | Tiririca | 2500.00 |
| 12 | Programador Pl. | Zé da Pizza | 2250.00 |
| 13 | Programador Pl. | Tiozão do Gás | 2750.00 |
| 14 | Programador Pl. | Adalberto do Sacolão | 2300.00 |
| 15 | Programador Pl. | Marisa da Horta | 2500.00 |
| 16 | Programador Sr. | Tiririca | 2500.00 |
| 17 | Programador Sr. | Zé da Pizza | 2250.00 |
| 18 | Programador Sr. | Tiozão do Gás | 2750.00 |
| 19 | Programador Sr. | Adalberto do Sacolão | 2300.00 |
| 20 | Programador Sr. | Marisa da Horta | 2500.00 |
| 21 | Web Designer Pl. | Tiririca | 2500.00 |
| 22 | Web Designer Pl. | Zé da Pizza | 2250.00 |
| 23 | Web Designer Pl. | Tiozão do Gás | 2750.00 |
| 24 | Web Designer Pl. | Adalberto do Sacolão | 2300.00 |
| 25 | Web Designer Pl. | Marisa da Horta | 2500.00 |

# JUNÇÕES (JOIN)

## 8. CROSS JOIN

Conseguimos o mesmo resultado usando a seguinte sintaxe:

```
SELECT C.NOMECARGO AS 'CARGO',
       F.NOMEFUNCIONARIO AS 'FUNCIONÁRIO',
       F.SALARIOFUNCIONARIO AS 'SALÁRIO'
FROM CARGO AS C, FUNCIONARIO AS F
ORDER BY 1;
```

# Dúvidas ?