

N_BANC DAD II6 - Texto de apoio

Site: [EAD Mackenzie](#)

Tema: BANCO DE DADOS {TURMAS 03B} 2023/1

Livro: N_BANC DAD II6 - Texto de apoio

Impresso por: FELIPE BALDIM GUERRA .

Data: sábado, 6 mai 2023, 23:57

Índice

1. SELECT – CONSULTAS COMPLEXAS

1.1. Junção na cláusula WHERE

1.2. Junção com INNER JOIN

1.3. Junção com LEFT JOIN

1.4. Junção com RIGHT JOIN

2. FULL JOIN

2.1. SELECT com consultas aninhadas

3. REFERÊNCIAS

1. SELECT – CONSULTAS COMPLEXAS

Introdução

Na Aula 5, você aprendeu o comando SELECT com uma única tabela, logo após a cláusula FROM. Mas o conceito de banco de dados reúne várias tabelas relacionadas. Então, em vários momentos, você precisará acessar dados de diferentes tabelas simultaneamente. Para isso, pode-se utilizar o conceito de *join*, ou seja, junção de tabelas que se relacionam.

Uma das formas de trabalhar com junção de duas ou mais tabelas é colocando na cláusula WHERE a condição de junção entre as tabelas que estão logo após o FROM. Uma outra forma é utilizando o INNER JOIN. Você aprenderá, a seguir, estas duas formas e, também: LEFT JOIN, RIGHT JOIN, FULL JOIN e consultas aninhadas.

Importante: você observará que tudo que aprendeu na Aula 5 será utilizado nesta aula também. Lembrando, novamente, que você deve usar algum banco de dados relacional para aprender e praticar o comando SELECT a ser estudado nesta aula. Uma sugestão é utilizar o Oracle na nuvem (assista à videoaula “Como utilizar o Oracle Live SQL” da Aula 5).

Só para você relembrar a sintaxe do comando SELECT:

```
SELECT [DISTINCT] nome_atributo1,... nome_atributoN
FROM nome_tabela1, ... nome_tabelaN
[WHERE (condições)]
[GROUP BY nome_atributo1,... nome_atributoN]
[HAVING (condições)]
[ORDER BY nome_atributo1 {ASC | DESC}, ...
               nome_atributoN {ASC | DESC}} ;
```

Os exemplos utilizados para explicar o comando SELECT desta aula utilizarão o seguinte banco de dados (é o mesmo banco de dados do Texto de Apoio sobre Álgebra Relacional – Aula 4):

- na Figura 1, temos o Modelo Relacional (foi utilizada a ferramenta DBDesigner);
- a Figura 2 tem as tabelas populadas deste banco de dados.

Atenção: o script de criação das tabelas e inserção de dados encontra-se no final desta aula.

Figura 1 – Representação do Modelo Relacional utilizando a ferramenta DBDesigner

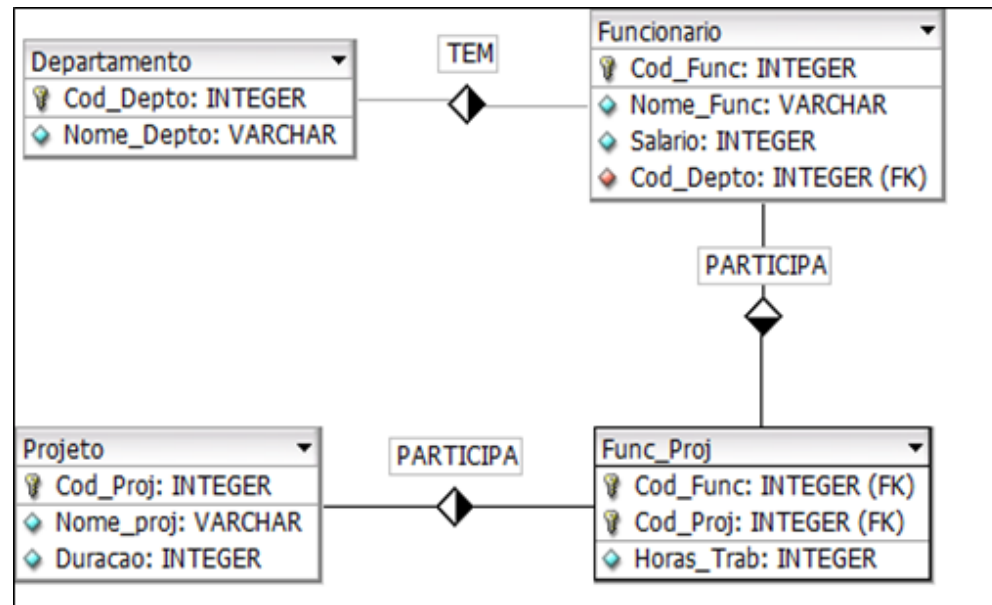


Figura 2 – Tabelas Departamento, Funcionario, Projeto e Func_Proj exemplificadas com dados

| Departamento | | Funcionario | | | |
|--------------|------------|-------------|----------------------|---------|-----------|
| Cod_Depto | Nome_Depto | Cod_Func | Nome_Func | Salario | Cod_Depto |
| 1 | Marketing | 101 | Joao da Silva Santos | 2000 | 2 |
| 2 | Vendas | 102 | Mario Souza | 1500 | 1 |
| 3 | Dados | 103 | Sergio Silva Santos | 2400 | 2 |
| 4 | Pesquisa | 104 | Maria Castro | 1200 | 1 |
| | | 105 | Marcio Silva Santana | 1400 | 4 |

| Projeto | | | Func_Proj | | |
|----------|-----------|---------|-----------|----------|------------|
| Cod_Proj | Nome_Proj | Duracao | Cod_Func | Cod_Proj | Horas_Trab |
| 1001 | SistemaA | 2 | 101 | 1001 | 24 |
| 1002 | SistemaB | 6 | 101 | 1002 | 160 |
| 1003 | SistemaX | 4 | 102 | 1001 | 56 |
| | | | 102 | 1003 | 45 |
| | | | 103 | 1001 | 86 |
| | | | 103 | 1003 | 64 |
| | | | 104 | 1001 | 46 |

1.1. Junção na cláusula WHERE

Para você entender a junção, será exemplificado, no Exemplo 1, o comando SELECT e os dados retornados, com a junção completa das tabelas Funcionario e Departamento, ou seja, todos os atributos e todas as linhas combinadas.

Exemplo 1: junção completa das tabelas Funcionario e Departamento.

```
SELECT *  
FROM Funcionario F, Departamento D  
WHERE (F.Cod_depto = D.Cod_depto);
```

Resultado da consulta (**Exemplo 1**):



| Cod_Func | Nome_Func | Salario | Cod_Depto | Cod_Depto | Nome_Depto |
|----------|----------------------|---------|-----------|-----------|------------|
| 101 | Joao da Silva Santos | 2000 | 2 | 2 | Vendas |
| 102 | Mario Souza | 1500 | 1 | 1 | Marketing |
| 103 | Sergio Silva Santos | 2400 | 2 | 2 | Vendas |
| 104 | Maria Castro | 1200 | 1 | 1 | Marketing |
| 105 | Marcio Silva Santana | 1400 | 4 | 4 | Pesquisa |

Considerações importantes sobre a junção de Funcionario com Departamento:

- a junção respeita o relacionamento entre Funcionario e Departamento, pois combina cada linha de Funcionario com a única linha do Departamento a que ela realmente pertence;
- observe que o departamento de código 3 (cujo nome é Dados) não apareceu neste resultado, pois não teve nenhuma linha combinada com Funcionario;

- foi inserida na cláusula WHERE a condição de junção entre elas – **(F.Cod_Depto = D.Cod_Depto)**, que combina duas linhas, uma de Funcionario e outra de Departamento, sempre que o valor do atributo Cod_Depto da tabela Funcionario for igual ao valor do atributo Cod_Depto da tabela Departamento;
- um mesmo nome de atributo pode ser utilizado em tabelas diferentes. Portanto, quando uma consulta envolver duas ou mais tabelas e fizer referência a atributo(s) com o mesmo nome de tabelas diferentes, é preciso dizer de qual tabela ele pertence. Isso é feito prefixando o nome da tabela ao nome do atributo e separando os dois por um ponto – nome_tabela.nome_atributo. Para não ter que ficar escrevendo o nome da tabela completa, podemos dar um apelido à ela, como foi feito no Exemplo 1 (é só colocar o apelido logo após o nome da tabela – Funcionario F).

Importante: a condição de junção entre tabelas combina chave primária de uma tabela com chave estrangeira da outra tabela. A junção de n tabelas em um único SELECT obriga a colocação de, pelo menos, $n-1$ condições de junção.

Se mais de uma tabela for especificada na cláusula FROM e não for inserida a condição de junção entre elas, gera-se o produto cartesiano, que você já viu na Aula 4 sobre Álgebra Relacional. O produto cartesiano gera todas as combinações possíveis entre as linhas das tabelas que estão na cláusula FROM e, neste caso, não serão respeitados os relacionamentos existentes entre as tabelas. O comando para gerar o produto cartesiano entre as tabelas Funcionario e Departamento é:

```
SELECT *  
FROM Funcionario F, Departamento D;
```

O Exemplo 1 mostrou a junção completa entre as tabelas Funcionario e Departamento, mas você pode inserir restrições de linhas e colunas, aplicar funções agregadas, fazer agrupamentos etc. Você verá isso nos próximos exemplos.

Exemplo 2: obtenha o **nome de cada funcionário** e o **nome do departamento de cada um**, mas somente para os funcionários que **ganham mais de 1600**.

Observe que, para fazer esta consulta, precisamos utilizar as tabelas Funcionario e Departamento, então, fazemos a junção entre elas (e você já sabe que elas se relacionam) para obter o resultado.

```
SELECT F.Nome_Func, D.Nome_Depto
FROM Funcionario F, Departamento D
WHERE (F.Salario > 1600)
AND (F.Cod_depto = D.Cod_depto);
```

Resultado da consulta (**Exemplo 2**) e alguns comentários:

| NOME_FUNC | NOME_DEPTO |
|----------------------|------------|
| Joao da Silva Santos | Vendas |
| Sergio Silva Santos | Vendas |

- a condição da cláusula WHERE – (**F.Salario > 1600**) – é uma condição de seleção que busca na tabela Funcionario somente as linhas em que os valores para o atributo Salario sejam maior que 1600;
- quando temos condições de seleção, a condição de junção deve, obrigatoriamente, ser precedida pelo operador AND, uma vez que essa condição de junção sempre deve ser verdadeira.

Exemplo 3 (junção com função agregada): obtenha a **quantidade de funcionários** do departamento de 'Vendas'.

```
SELECT COUNT(F.Cod_Depto) AS Total_Func
FROM Funcionario F, Departamento D
WHERE (D.Nome_Depto = 'Vendas')
AND (F.Cod_depto = D.Cod_depto);
```

Resultado da consulta (**Exemplo 3**) e um comentário:

| TOTAL_FUNC |
|------------|
| 2 |

- observe que foi feita a junção das tabelas Funcionario e Departamento e, também, foi utilizada a função agregada COUNT.

Exemplo 4 (junção com agrupamento): obtenha o **nome de cada departamento** (que tenha funcionário) e a **quantidade de funcionários em cada um deles** (em ordem crescente do nome do departamento).

```
SELECT D.Nome_Depto, COUNT(F.Cod_Depto) AS Total_Func
FROM Funcionario F, Departamento D
WHERE (F.Cod_depto = D.Cod_depto)
GROUP BY D.Nome_Depto
ORDER BY D.Nome_Depto ASC;
```

Resultado da consulta (**Exemplo 4**) e alguns comentários:

| NOME_DEPTO | TOTAL_FUNC |
|------------|------------|
| Marketing | 2 |
| Pesquisa | 1 |
| Vendas | 2 |

- observe que foi feita a junção das tabelas Funcionario e Departamento e, também, o agrupamento dos departamentos (por nome);
- o departamento de 'Dados' não apareceu no resultado, pois ele não tem nenhum funcionário relacionado a ele.

Exemplo 5 (junção de três tabelas): obtenha o **nome de cada funcionário** (que já tenha participado de algum projeto) e o **nome do(s) projeto(s)** de que cada um participou.

Observe que, para fazer esta consulta, precisamos utilizar as tabelas Funcionario, Func_Proj e Projeto, então, fazemos a junção dessas três tabelas para obter o resultado.

```
SELECT F.Nome_Func, P.Nome_Proj
FROM Funcionario F, Func_Proj FP, Projeto P
WHERE (F.Cod_Func = FP.Cod_Func)
AND (FP.Cod_Proj = P.Cod_Proj);
```

Resultado da consulta (**Exemplo 5**) e alguns comentários:

| NOME_FUNC | NOME_PROJ |
|----------------------|-----------|
| Joao da Silva Santos | SistemaA |
| Joao da Silva Santos | SistemaB |
| Mario Souza | SistemaA |
| Mario Souza | SistemaX |
| Sergio Silva Santos | SistemaA |
| Sergio Silva Santos | SistemaX |
| Maria Castro | SistemaA |

- observe que foi necessária a junção de três tabelas, portanto, temos duas condições de junção: **(F.Cod_Func = FP.Cod_Func)** e **(FP.Cod_Proj = P.Cod_Proj)**;
- se pelo menos uma das condições de junção acima não aparecer na cláusula WHERE, o produto cartesiano ocorrerá;
- observe, também, que o funcionário de nome 'Marcio Silva Santana' não apareceu no resultado desta consulta, pois ele não participou de nenhum projeto ainda.

Atenção: não deixe de assistir à videoaula “SELECT com junção de tabelas na cláusula WHERE”, com a professora Elisângela Botelho Gracias

1.2. Junção com INNER JOIN

Você aprendeu que a junção de várias tabelas pode ser feita colocando-se os nomes de todas as tabelas envolvidas na cláusula FROM e a(s) condição(ões) de junção na cláusula WHERE.

Agora, você verá que também podemos utilizar o INNER JOIN para fazer a junção de tabelas. Para isso, utilize as palavras-chave **INNER JOIN** e a(s) condição(ões) de junção é(são) indicada(s) pela palavra-chave **ON**, dentro da cláusula FROM (observe que a condição de junção não aparece mais na cláusula WHERE).

Exemplo 6: junção completa das tabelas Funcionario e Departamento utilizando o INNER JOIN.

```
SELECT *  
FROM Departamento D INNER JOIN Funcionario F  
ON (D.Cod_depto = F.Cod_depto);
```

Resultado da consulta (**Exemplo 6**) e alguns comentários:



| Cod_Func | Nome_Func | Salario | Cod_Depto | Cod_Depto | Nome_Depto |
|----------|----------------------|---------|-----------|-----------|------------|
| 101 | Joao da Silva Santos | 2000 | 2 | 2 | Vendas |
| 102 | Mario Souza | 1500 | 1 | 1 | Marketing |
| 103 | Sergio Silva Santos | 2400 | 2 | 2 | Vendas |
| 104 | Maria Castro | 1200 | 1 | 1 | Marketing |
| 105 | Marcio Silva Santana | 1400 | 4 | 4 | Pesquisa |

- observe que o resultado desta consulta é o mesmo do Exemplo 1;
- comparando esta consulta com a do Exemplo 1, no lugar da vírgula entre as tabelas, coloca-se o INNER JOIN e a condição de junção agora fica na cláusula ON;
- o INNER JOIN traz somente as linhas combinadas, de acordo com a condição de junção (F.Cod_Depto = D.Cod_Depto), ou seja, ele respeita o relacionamento entre Funcionario e Departamento, pois combina cada linha de Funcionario com a única linha do Departamento a que ele realmente pertence;
- observe, novamente, que o departamento de código 3 (cujo nome é Dados) não apareceu neste resultado, pois não teve nenhuma linha combinada com Funcionario.

Atenção: a seguir, você verá como ficam os Exemplos 2, 3, 4 e 5 com a utilização do INNER JOIN. Lembrando que o resultado é o mesmo da junção feita na cláusula WHERE.

Exemplo 7 (mesmo do Exemplo 2): obtenha o **nome de cada funcionário** e o **nome do departamento de cada um**, mas somente para os funcionários que **ganham mais de 1600**.

```
SELECT F.Nome_Func, D.Nome_Depto
FROM Funcionario F INNER JOIN Departamento D
ON (F.Cod_depto = D.Cod_depto)
WHERE (F.Salario > 1600);
```

Resultado da consulta (**Exemplo 7**) e um comentário:

| NOME_FUNC | NOME_DEPTO |
|----------------------|------------|
| Joao da Silva Santos | Vendas |
| Sergio Silva Santos | Vendas |

- na cláusula WHERE, temos apenas a condição (**F.Salario > 1600**), uma vez que a condição de junção (F.Cod_Depto = D.Cod_Depto) fica na cláusula ON.

Exemplo 8 (mesmo do Exemplo 3): obtenha a **quantidade de funcionários** do departamento de 'Vendas'.

```
SELECT COUNT(F.Cod_Depto) AS Total_Func  
FROM Funcionario F INNER JOIN Departamento D  
ON (F.Cod_depto = D.Cod_depto)  
WHERE (D.Nome_Depto = 'Vendas');
```

Resultado da consulta (**Exemplo 8**):

| TOTAL_FUNC |
|------------|
| 2 |

Exemplo 9 (mesmo do Exemplo 4): obtenha o **nome de cada departamento** (que tenha funcionário) e a **quantidade de funcionários em cada um deles** (em ordem crescente do nome do departamento).

```
SELECT D.Nome_Depto, COUNT(F.Cod_Depto) AS Total_Func  
FROM Funcionario F INNER JOIN Departamento D  
ON (F.Cod_depto = D.Cod_depto)  
GROUP BY D.Nome_Depto  
ORDER BY D.Nome_Depto ASC;
```

Resultado da consulta (**Exemplo 9**):

| NOME_DEPTO | TOTAL_FUNC |
|------------|------------|
| Marketing | 2 |
| Pesquisa | 1 |
| Vendas | 2 |

Exemplo 10 (mesmo do Exemplo 5): obtenha o **nome de cada funcionário** (que já tenha participado de algum projeto) e **o nome do(s) projeto(s)** de que cada um participou.

```
SELECT F.Nome_Func, P.Nome_Proj
FROM Funcionario F INNER JOIN Func_Proj FP
ON (F.Cod_Func = FP.Cod_Func)
INNER JOIN Projeto P ON (FP.Cod_Proj = P.Cod_Proj);
```

Resultado da consulta (**Exemplo 10**) e alguns comentários:

| NOME_FUNC | NOME_PROJ |
|----------------------|-----------|
| Joao da Silva Santos | SistemaA |
| Joao da Silva Santos | SistemaB |
| Mario Souza | SistemaA |
| Mario Souza | SistemaX |
| Sergio Silva Santos | SistemaA |
| Sergio Silva Santos | SistemaX |
| Maria Castro | SistemaA |

- observe que, primeiro, foi feita a junção – INNER JOIN – das tabelas Funcionario e Func_Proj, cuja condição de junção é **(F.Cod_Func = FP.Cod_Func)**. Depois, com o resultado da junção anterior, foi feito o INNER JOIN com Projeto, cuja condição de junção é **(FP.Cod_Proj = P.Cod_Proj)**;
- observe, novamente, que o funcionário de nome 'Marcio Silva Santana' não apareceu no resultado dessa consulta, pois ele não participou de nenhum projeto ainda.

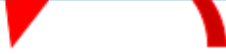
1.3. Junção com LEFT JOIN

Você já aprendeu que a junção entre duas tabelas gera um resultado no qual temos linhas que se combinam de acordo com a condição de junção.

Mas existem, também, as junções externas que geram o resultado da junção (as linhas combinadas) mais as linhas não combinadas.

A palavra-chave LEFT JOIN gera um resultado contendo as linhas combinadas e as linhas não combinadas da tabela que estiver do lado esquerdo.

Exemplo 11: junção externa completa das tabelas Funcionario e Departamento utilizando o **LEFT JOIN**.



```
SELECT *  
FROM Departamento D LEFT JOIN Funcionario F  
ON (D.Cod_depto = F.Cod_depto)  
ORDER BY D.Nome_Depto ASC;
```

Resultado da consulta (**Exemplo 11**) e alguns comentários:



| Cod_Depto | Nome_Depto | Cod_Func | Nome_Func | Salario | Cod_Depto |
|-----------|------------|----------|----------------------|---------|-----------|
| 3 | Dados | NULL | NULL | NULL | NULL |
| 1 | Marketing | 102 | Mario Souza | 1500 | 1 |
| 1 | Marketing | 104 | Maria Castro | 1200 | 1 |
| 4 | Pesquisa | 105 | Marcio Silva Santana | 1400 | 4 |
| 2 | Vendas | 101 | Joao da Silva Santos | 2000 | 2 |
| 2 | Vendas | 103 | Sergio Silva Santos | 2400 | 2 |

- o LEFT JOIN retorna as linhas combinadas entre as tabelas Departamento e Funcionario e, também, as linhas de Departamento (do lado esquerdo) que não estão ligadas a nenhum funcionário;
- observe, na primeira linha do resultado, que o departamento de 'Dados' apareceu no resultado, mas não está ligado a nenhum funcionário (por isso, os dados referentes a um funcionário aparecem como nulos);
- analise, com atenção, que, no LEFT JOIN, se trocarmos a posição das tabelas Departamento e Funcionario, o resultado será alterado.

Exemplo 12: obtenha os **nomes de todos os departamentos da empresa** (em ordem crescente), com os **nomes dos funcionários que trabalham em cada um deles**.

```
SELECT D.Nome_Depto, F.Nome_func  
FROM Departamento D LEFT JOIN Funcionario F  
ON (D.Cod_depto = F.Cod_depto)  
ORDER BY D.Nome_Depto ASC;
```

Resultado da consulta (**Exemplo 12**) e um comentário:

| NOME_DEPTO | NOME_FUNC |
|------------|----------------------|
| Dados | - |
| Marketing | Mario Souza |
| Marketing | Maria Castro |
| Pesquisa | Marcio Silva Santana |
| Vendas | Joao da Silva Santos |
| Vendas | Sergio Silva Santos |

- observe que o departamento de 'Dados' não tem funcionário, mas apareceu no resultado.

Exemplo 13: obtenha os **nomes de todos os departamentos da empresa** e a **quantidade de funcionários pertencentes a cada um deles** (retorne mesmo aqueles departamentos em que não tem funcionários).

Este exemplo é “parecido” com o Exemplo 9, sendo que a única diferença é que este trará um resultado com todos os departamentos (e não somente com os que têm funcionários).

```
SELECT D.Nome_Depto, COUNT(F.Cod_Func) AS Total_Func
FROM Departamento D LEFT JOIN Funcionario F
ON (D.Cod_depto = F.Cod_depto)
GROUP BY D.Nome_Depto
ORDER BY D.Nome_Depto ASC;
```

Resultado da consulta (**Exemplo 13**) e alguns comentários:

| NOME_DEPTO | TOTAL_FUNC |
|------------|------------|
| Dados | 0 |
| Marketing | 2 |
| Pesquisa | 1 |
| Vendas | 2 |

- observe que o departamento de 'Dados' apareceu no resultado, mas sem funcionários, ou seja, zero;
- atente-se ao atributo que você colocará no COUNT, pois, se ele for nulo, não contará com uma linha (que é o que aconteceu com o departamento de 'Dados', que tem zero funcionários).

1.4. Junção com RIGHT JOIN

Você aprendeu a junção externa LEFT JOIN anteriormente. Temos também a junção externa RIGHT JOIN.

A palavra-chave RIGHT JOIN gera um resultado contendo as linhas combinadas e as linhas não combinadas da tabela que estiver do lado direito.

Assim, o resultado de uma junção externa de um lado depende da direção (DIREITA ou ESQUERDA) e da posição dos nomes das tabelas.

Exemplo 14: obtenha os **nomes de todos os funcionários da empresa** e o **nome dos projetos de que cada um já participou** (retorne mesmo aqueles funcionários que ainda não participaram de nenhum projeto).

Este exemplo é “parecido” com o exemplo 10 (com INNER JOIN), sendo que a única diferença é que este trará um resultado com todos os funcionários (e não somente os funcionários que já participaram de algum projeto).

Analise, cuidadosamente, as três consultas, a seguir, que trazem o mesmo resultado:

Utilizando somente LEFT JOIN

```
SELECT F.Nome_Func, P.Nome_Proj
FROM Funcionario F LEFT JOIN Func_Proj FP
ON (F.Cod_Func = FP.Cod_Func)
LEFT JOIN Projeto P
ON (FP.Cod_Proj = P.Cod_Proj);
```

Utilizando RIGHT JOIN e LEFT JOIN

```
SELECT F.Nome_Func, P.Nome_Proj
FROM Func_Proj FP RIGHT JOIN Funcionario F
ON (FP.Cod_Func = F.Cod_Func)
LEFT JOIN Projeto P
ON (FP.Cod_Proj = P.Cod_Proj);
```

Utilizando INNER JOIN e RIGHT JOIN

```
SELECT F.Nome_Func, P.Nome_Proj
FROM Func_Proj FP INNER JOIN Projeto P
ON (FP.Cod_Proj = P.Cod_Proj)
RIGHT JOIN Funcionario F
ON (FP.Cod_Func = F.Cod_Func);
```

Resultado da consulta (**Exemplo 14**) e um comentário:

| NOME_FUNC | NOME_PROJ |
|----------------------|-----------|
| Joao da Silva Santos | SistemaA |
| Joao da Silva Santos | SistemaB |
| Mario Souza | SistemaA |
| Mario Souza | SistemaX |
| Sergio Silva Santos | SistemaA |
| Sergio Silva Santos | SistemaX |
| Maria Castro | SistemaA |
| Marcio Silva Santana | - |

- observe que o funcionário 'Marcio Silva Santana' apareceu no resultado, mas não tem nenhum projeto vinculado a ele.

Atenção: não deixe de assistir à videoaula “SELECT com INNER, LEFT e RIGHT JOIN”, com a professora Elisângela Botelho Gracias.

2. FULL JOIN

A palavra-chave FULL JOIN gera um resultado contendo as linhas combinadas e, também, todas as linhas não combinadas das tabelas envolvidas neste FULL JOIN.

Exemplo 15: obtenha **todas as informações de funcionários com as respectivas informações do departamento a que cada um pertence e**, também, as informações daqueles **departamentos que não têm funcionários** e dos **funcionários que não estão ligados a nenhum departamento**.

```
SELECT *  
FROM Departamento D FULL JOIN Funcionario F  
ON (D.Cod_depto = F.Cod_depto);
```

Um comentário (**Exemplo 15**):

- esta consulta retorna todas as linhas combinadas de Departamento e Funcionario, todos os departamentos que não têm funcionários e todos os funcionários que não estão ligados a um departamento.

2.1. SELECT com consultas aninhadas

Uma **consulta aninhada** é aquela que tem outra consulta embutida dentro dela , sendo que a consulta embutida é chamada de subconsulta.

Uma subconsulta aparece tipicamente dentro da cláusula WHERE de uma consulta, mas pode aparecer também na cláusula SELECT, FROM ou HAVING. E uma subconsulta pode conter outras subconsultas aninhadas.

Importante: uma subconsulta sempre retornará um valor ou um conjunto de valores para uma consulta aninhada. Então, o que está sendo retornado por uma subconsulta tem de ser compatível com o que se está comparando. Observe, atentamente, os exemplos que veremos a seguir.

Exemplo 16: obtenha os **nome e salário dos funcionários que ganham mais do que a média salarial paga na empresa.**

```
SELECT Nome_Func, Salario
FROM Funcionario
WHERE Salario > (SELECT AVG(Salario)
                 FROM Funcionario
                 );
```

Resultado da consulta (**Exemplo 16**) e alguns comentários:

| NOME_FUNC | SALARIO |
|----------------------|---------|
| Joao da Silva Santos | 2000 |
| Sergio Silva Santos | 2400 |

- observe que a **subconsulta retorna a média salarial** dos funcionários da empresa;
- esta média salarial, que está sendo retornada pela subconsulta, será utilizada para obter quais funcionários têm o salário maior que a média salarial.

Exemplo 17: obtenha, sem repetição e em ordem crescente, o **nome dos departamentos que têm algum funcionário**.

Observe que, nesta consulta, você poderá utilizar o INNER JOIN (que é a melhor opção) e, também, é uma opção a consulta aninhada.

Resolução com INNER JOIN

```
SELECT DISTINCT D.Nome_Depto  
FROM Funcionario F INNER JOIN Departamento D  
ON (F.Cod_depto = D.Cod_depto)  
ORDER BY D.Nome_Depto ASC;
```

Resolução com consulta aninhada

```
SELECT DISTINCT Nome_Depto  
FROM Departamento  
WHERE Cod_depto IN (SELECT Cod_depto  
                     FROM Funcionario  
                     )  
ORDER BY Nome_Depto ASC;
```

Resultado da consulta (**Exemplo 17**) e alguns comentários:

| NOME_DEPTO |
|------------|
| Marketing |
| Pesquisa |
| Vendas |

- observe, na segunda resolução, que a **subconsulta retornará todos os valores de código de departamento da tabela Funcionario**, ou seja, o código dos departamentos que têm funcionários;
- os valores dos códigos de departamentos, que estão sendo retornados pela subconsulta, serão utilizados para obter o nome dos departamentos em que temos algum funcionário;
- você deve se lembrar do operador IN, que é igualdade para um conjunto de valores. Utilizamos esse operador IN, pois uma subconsulta pode retornar vários valores;
- se você desejasse saber o nome dos departamentos que não têm funcionários, bastava trocar o IN por NOT IN, na segunda resolução dessa consulta.

Exemplo 18: obtenha o **nome de cada departamento** (que tenha funcionário) e a **quantidade de funcionários em cada um deles**, mas **somente para os departamentos que têm mais funcionários do que o departamento de 'Pesquisa'**.

```
SELECT D.Nome_Depto, COUNT(F.Cod_depto) AS Total_Func
FROM Funcionario F INNER JOIN Departamento D
ON (F.Cod_depto = D.Cod_depto)
GROUP BY D.Nome_Depto
HAVING COUNT(F.Cod_depto) > (SELECT COUNT(F.Cod_depto)
                             FROM Funcionario F INNER JOIN Departamento D
                             ON (F.Cod_depto = D.Cod_depto)
                             WHERE (D.Nome_Depto = 'Pesquisa')
                             );
```

Resultado da consulta (**Exemplo 18**) e alguns comentários:

| NOME_DEPTO | TOTAL_FUNC |
|------------|------------|
| Marketing | 2 |
| Vendas | 2 |

- observe que a **subconsulta** retornará somente a quantidade de funcionários do departamento de 'Pesquisa' e, para obter este valor, foi feito um INNER JOIN entre as tabelas Funcionario e Departamento;
- esta quantidade de funcionários de 'Pesquisa', a ser retornada pela subconsulta, será utilizada na cláusula HAVING.

Atenção: não deixe de assistir à videoaula “SELECT com subconsulta”, com a professora Elisângela Botelho Gracias.

Uma **subconsulta** pode ser utilizada nos comandos **UPDATE** e **DELETE**, e isso será necessário quando:

- a atualização dos dados for feita em uma determinada tabela, mas os critérios para essa atualização envolvem dados de outras tabelas;
- a eliminação de dados for feita em uma determinada tabela, mas os critérios para essa eliminação envolvem dados de outras tabelas.

Exemplo 19: aumente em 10% o salário dos funcionários do departamento de 'Marketing'.

```
UPDATE Funcionario
SET Salario = Salario * 1.1
WHERE Cod_depto IN (SELECT Cod_depto
                    FROM Departamento
                    WHERE (Nome_Depto = 'Marketing'))
);
```

Alguns comentários (**Exemplo 19**):

- observe que a atualização está sendo feita na tabela Funcionario, mas o critério para atualização envolve o nome do departamento, que se encontra na tabela Departamento;
- logo, a subconsulta retorna o código do departamento cujo nome seja 'Marketing';
- este código de departamento, que será retornado pela subconsulta, será utilizado no UPDATE.

-- Script do Banco de Dados utilizado nesta Aula 6

```
CREATE TABLE Departamento
(
  Cod_Depto INTEGER,
  Nome_Depto VARCHAR(20) NOT NULL,
  PRIMARY KEY(Cod_Depto)
);

CREATE TABLE Funcionario
(
  Cod_Func INTEGER,
  Nome_Func VARCHAR(20) NOT NULL,
  Salario INTEGER NOT NULL,
  Cod_Depto INTEGER NOT NULL,
  PRIMARY KEY(Cod_Func),
  FOREIGN KEY (Cod_Depto) REFERENCES Departamento (Cod_Depto)
);
```

```
CREATE TABLE Projeto
(Cod_Proj INTEGER,
 Nome_Proj VARCHAR(20) NOT NULL,
 Duracao INTEGER NOT NULL,
 PRIMARY KEY(Cod_Proj)
);
```

```
CREATE TABLE Func_Proj
(Cod_Func INTEGER,
 Cod_Proj INTEGER,
 Horas_Trab INTEGER,
 PRIMARY KEY(Cod_Func, Cod_Proj),
 FOREIGN KEY (Cod_Func) REFERENCES Funcionario(Cod_Func),
 FOREIGN KEY (Cod_Proj) REFERENCES Projeto(Cod_Proj)
);
```

```
INSERT INTO Departamento (Cod_Depto, Nome_Depto) VALUES (1, 'Marketing');
INSERT INTO Departamento (Cod_Depto, Nome_Depto) VALUES (2, 'Vendas');
INSERT INTO Departamento (Cod_Depto, Nome_Depto) VALUES (3, 'Dados');
INSERT INTO Departamento (Cod_Depto, Nome_Depto) VALUES (4, 'Pesquisa');
```

```
INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (101, 'Joao da
Silva Santos', 2000, 2);
```

```
INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (102, 'Mario
Souza', 1500, 1);
```

```
INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (103, 'Sergio
Silva Santos', 2400, 2);
```

```
INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (104, 'Maria
Castro', 1200, 1);
```

```
INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (105, 'Marcio
Silva Santana', 1400, 4);
```

```
INSERT INTO Projeto (Cod_Proj, Nome_Proj, Duracao) VALUES (1001, 'SistemaA', 2);  
INSERT INTO Projeto (Cod_Proj, Nome_Proj, Duracao) VALUES (1002, 'SistemaB', 6);  
INSERT INTO Projeto (Cod_Proj, Nome_Proj, Duracao) VALUES (1003, 'SistemaX', 4);
```

```
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (101, 1001, 24);  
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (101, 1002, 160);  
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (102, 1001, 56);  
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (102, 1003, 45);  
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (103, 1001, 86);  
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (103, 1003, 64);  
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (104, 1001, 46);
```

3. REFERÊNCIAS

- DAMAS, L. *SQL – Structured Query Language*. 6. ed. Rio de Janeiro: LTC, 2007.
- ELMASRI, R.; NAVATHE, S. *Sistemas de banco de dados*. 7. ed. São Paulo: Pearson, 2018.
- RAMAKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de banco de dados*. 3. ed. Porto Alegre: AMGH, 2011.