

# SQL

## Junção horizontal e vertical de dados

Você sabia que seu material didático é interativo e multimídia? Isso significa que você pode interagir com o conteúdo de diversas formas, a qualquer hora e lugar. Na versão impressa, porém, alguns conteúdos interativos ficam desabilitados. Por essa razão, fique atento: sempre que possível, opte pela versão digital. Bons estudos!

Em grande parte dos casos nos bancos de dados, as tabelas são relacionadas, permitindo dessa forma, que possamos efetuar maior número de consultas, consequentemente podendo abstrair um maior número de informações. Para efetuar consultas em duas ou mais tabelas relacionadas, há a necessidade de técnicas que proporcionem junções, essas técnicas estão presentes na linguagem de programação de banco de dados SQL, por meio de três sintaxes: INNER JOIN, LEFT JOIN, RIGHT JOIN.

## Junções

Já vimos as formas de seleções de tuplas em uma única tabela, porém essas técnicas utilizadas até o momento não permitem realizar consultas em múltiplas tabelas. Portanto, vamos ver como utilizar o comando JOIN por meio do SELECT para unir duas ou mais tabelas.



Fonte: Shutterstock.

A seguir, veja o exemplo das tabelas Categoria e Produto:

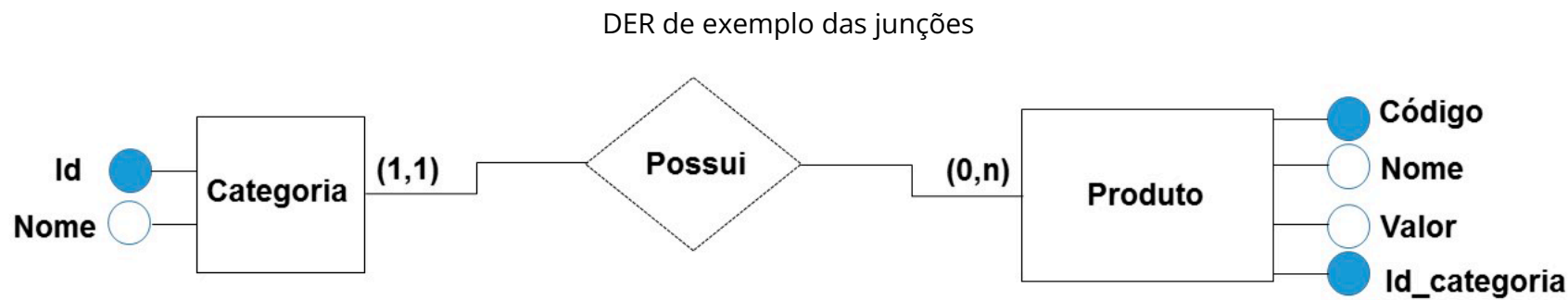
Tabelas e chaves

▼

Na tabela “Categoria” a coluna “Id” é a chave primária.

Já na tabela “Produto” a chave primária é a coluna “Codigo”.

A chave estrangeira que relaciona as duas tabelas é o campo “Id\_Categoria”.



## Criação das tabelas



*Script* em SQL para a representação do DER de exemplo das junções.

```

1  CREATE DATABASE Loja;
2  USE Loja;
3
4  CREATE TABLE Categoria (
5  Id INT(3) PRIMARY KEY AUTO_INCREMENT,
6  Nome VARCHAR(50) NOT NULL
7  );
8  CREATE TABLE Produto (
9  Codigo INT(3) PRIMARY KEY AUTO_INCREMENT,
10 Nome VARCHAR(50) NOT NULL,
11 Valor DECIMAL(6,2) NOT NULL,
12 Id_Categoria INT(3) NOT NULL,
13 FOREIGN KEY (Id_Categoria) REFERENCES Categoria (Id)
14 );

```

## Inserção de registros



Para que possamos efetuar as consultas relacionais adiante, é necessário inserir alguns registros em ambas as tabelas.

```

1  INSERT Categoria VALUES (0, "DVD"),
2  (0, "Livro"),
3  (0, "Informática");
4
5  INSERT Produto VALUES (0, "Código da Vinci", "39.99", 2),
6  (0, "Hancock", "89.99", 1),
7  (0, "Dario de um mago", "19.99", 2);
8  (0, "Eu sou a lenda", "39.99", 1);

```

Dessa forma, ao fazer uma simples seleção nas duas tabelas, teremos os resultados representados nas imagens a seguir.

```

mysql> SELECT * FROM Categoria;
+----+-----+
| Id | Nome |
+----+-----+
| 1  | DVD  |
| 2  | Livro |
| 3  | Informática |
+----+-----+
3 rows in set (0.00 sec)

```

```

mysql> SELECT * FROM Produto;
+----+-----+-----+-----+
| Codigo | Nome | Valor | Id_Categoria |
+----+-----+-----+-----+
| 1 | Código da Vinci | 39.99 | 2 |
| 2 | Hancock | 89.99 | 1 |
| 3 | Dario de um Mago | 19.99 | 2 |
| 4 | Eu sou a lenda | 39.99 | 1 |
+----+-----+-----+-----+
4 rows in set (0.08 sec)

```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

As condições para efetuar uma junção, depende diretamente do **tipo de junção** e uma **condição de junção**, dessa forma com o SQL será possível retornar relações como resultados.



## TIPO DE JUNÇÃO

Define/trata as tuplas em cada uma das relações que não correspondam a alguma das tuplas da outra relação. Sendo dividido em relação interna, com o comando INNER JOIN, e relações externas: LEFT JOIN, RIGHT JOIN e FULL JOIN.



## CONDIÇÃO DE JUNÇÃO

Define as tuplas nas duas relações que são correspondentes, garantindo que os atributos utilizados em ambas as tabelas estejam presentes tanto na sintaxe SQL, quanto nos seus resultados.

## Parâmetro JOIN

Para realizar as junções nas consultas as tabelas, faz-se necessário a utilização de um dos comandos mais importantes na estrutura SQL, o JOIN e suas variações. Com a utilização do comando **JOIN** (junção) é possível por meio do SELECT, unir duas ou mais tabelas, ao se apontar os campos correspondentes entre elas.

A sintaxe utilizada para se efetuar junções nas consultas em SQL é definida como:

```
SELECT[campo] FROM [tabela_1 JOIN tabela_2]
ON [tabela_1].[chave_primária] = [tabela_2].[chave_estrangeira]
WHERE [condição];
```

## INNER JOIN

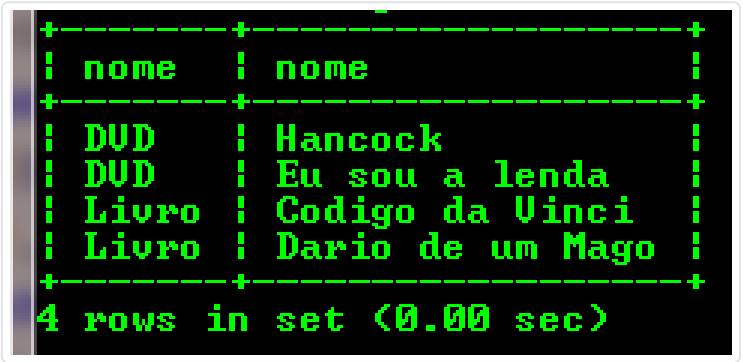
No nosso exemplo, temos a relação entre categorias e produtos, para efetuar uma consulta que nos retorne o nome da categoria e seus respectivos nomes dos produtos.

Sintaxe:

```
SELECT categoria.nome, produto.nome
FROM Categoria INNER JOIN Produto
ON Categoria.Id = Produto.Id_Categoria
```

[Saiba Mais](#)

### Resultado do exemplo



Fonte: elaborada pelo autor, captura de tela do software MySQL.

## Select JOIN com condição

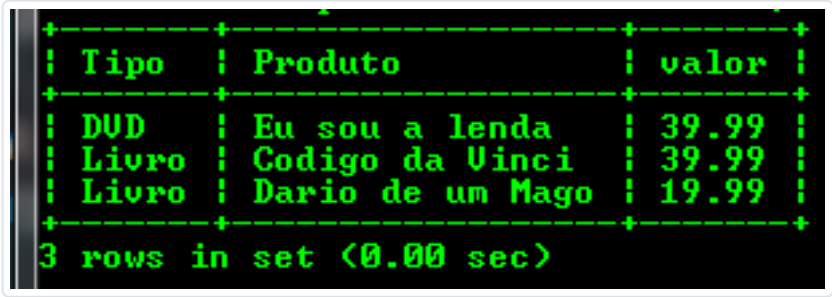
É possível utilizar condições nas consultas, quando necessário fazer as junções entre as tabelas.

**Exemplo:** exibir o nome da categoria como “Tipo”, o nome do produto como “Produto”, e o valor dos produtos, em uma condição que o valor dos produtos seja menor que R\$ 50,00.

Sintaxe:

```
SELECT categoria.nome as "Tipo", produto.nome as "Produto", produto.valor
FROM Categoria INNER JOIN Produto
ON Categoria.Id = Produto.Id_Categoria
WHERE produto.valor
```

### Resultado do exemplo



Tipo	Produto	valor
DVD	Eu sou a lenda	39.99
Livro	Codigo da Vinci	39.99
Livro	Dario de um Mago	19.99

3 rows in set (0.00 sec)

Fonte: elaborada pelo autor, captura de tela do software MySQL.

## Junção externa

Quando o operador de junção externa é utilizado no SQL, é gerado o resultado da junção mais as linhas não combinadas. Sendo possível efetuar junções externas em ambos os lados, ou seja, da esquerda para a direita, e da direita para a esquerda.

Dessa forma, a junção externa independente do lado escolhido, gera uma nova tabela que é a junção das linhas combinadas e não combinadas.

### LEFT JOIN

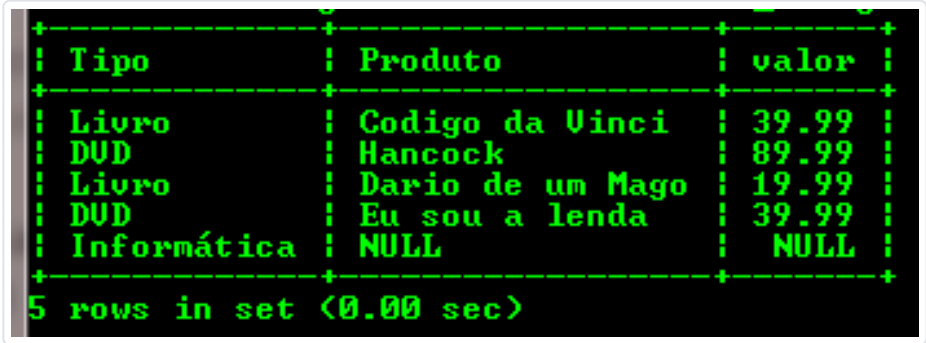
No comando LEFT JOIN, as linhas da tabela da esquerda são projetadas na seleção, juntamente com as linhas não combinadas da tabela da direita. Ou seja, como resultado dessa seleção, algumas linhas em que não se tenha relacionamento entre as tabelas da esquerda para a direita, retornarão o valor nulo (NULL).

Sintaxe:

```
SELECT categoria.nome as "Tipo", produto.nome as "Produto", produto.valor
FROM Categoria LEFT JOIN Produto
ON Categoria.Id = Produto.Id_Categoria;
```

#### Resultado do exemplo

São projetados todos os registros da tabela da esquerda, e na última seleção a linha não combinada. Pois não existe nenhum produto associado à categoria “Informática”, conseqüentemente as colunas “Produto” e “Valor” receberam nulo (NULL) como entrada.



Tipo	Produto	valor
Livro	Codigo da Vinci	39.99
DVD	Hancock	89.99
Livro	Dario de um Mago	19.99
DVD	Eu sou a lenda	39.99
Informática	NULL	NULL

5 rows in set (0.00 sec)

Fonte: elaborada pelo autor, captura de tela do software MySQL.

### RIGHT JOIN

Similar ao comando LEFT JOIN, no comando RIGHT JOIN, as linhas da tabela da direita são projetadas na seleção juntamente com as linhas não combinadas da tabela da esquerda.

Sintaxe:

```
SELECT categoria.nome as "Tipo", produto.nome as "Produto", produto.valor
FROM Categoria RIGHT JOIN Produto
ON Categoria.Id = Produto.Id_Categoria;
```

#### Resultado do exemplo

São projetados todos os registros da tabela da direita. Como não existe nenhum produto associado à categoria “Informática”, conseqüentemente nenhum registro do tipo nulo (NULL) é demonstrado na seleção dos registros.

Tipo	Produto	valor
DVD	Hancock	89.99
DVD	Eu sou a lenda	39.99
Livro	Codigo da Vinci	39.99
Livro	Dario de um Mago	19.99

4 rows in set (0.00 sec)

Fonte: elaborada pelo autor, captura de tela do software MySQL.

O comando JOIN é utilizado com maior frequência do que você possa imaginar. Alguns exemplos de aplicações web são

os sites de hospedagem, sites de compra e buscadores Web, onde são necessárias relações entre tabelas diferentes para gerar o resultado da pesquisa, na qual são efetuadas buscas ou filtros por opções ou palavras chave.

Portanto, vimos a estrutura de junções horizontais e verticais, e os comandos aplicados para estas tarefas, para que você possa se tornar cada vez melhor na programação de banco de dados e desenvolver as suas habilidades.



Fonte: Shutterstock.

# SQL

## Funções de agregação em banco de dados

Você sabia que seu material didático é interativo e multimídia? Isso significa que você pode interagir com o conteúdo de diversas formas, a qualquer hora e lugar. Na versão impressa, porém, alguns conteúdos interativos ficam desabilitados. Por essa razão, fique atento: sempre que possível, opte pela versão digital. Bons estudos!

---

É possível que você já tenha utilizado algum site de compras online. Quando efetuamos uma busca de um produto específico ou de uma classe de produtos, utilizando uma palavra chave, obtemos certa quantidade de produtos encontrados.

Nessas mesmas buscas é possível utilizar filtros para descobrir o menor valor ofertado, ou qual o produto mais comprado. Para que estes facilitadores de compra funcionem, os desenvolvedores dos sites comumente utilizam um recurso conhecido por funções de agregação. É este o assunto que trataremos nesta webaula.



Fonte: Shutterstock.

## Funções de agregação

Os bancos de dados possibilitam agregar diversas funcionalidades aos desenvolvimentos de sistemas. Tais funções devem ser exploradas a fim de permitir que as aplicações possam oferecer recursos de consultas avançadas à base de dados. Elas permitem que algumas informações quantitativas possam ser extraídas do banco de dados, como:

**Contar** o número de registros em uma tabela.

Saber o valor **máximo** ou **mínimo** em uma coluna.

Fazer operações matemáticas como **somatória** e **média** em uma coluna.

## Funções agregadas

As funções agregadas são aquelas que utilizam um multiconjunto de valores como entrada, porém o seu retorno é um único valor. O SQL oferece cinco funções de agregação nativamente.

COUNT	▼
Retorna a contagem de uma determinada consulta.	
MINIMUM	▼
Retorna o menor valor de uma consulta.	
MAXIMUM	▼
Retorna o maior valor de uma consulta.	
TOTAL	▼
Retorna à somatória de uma determinada consulta.	
AVERAGE	▼
Retorna a média de uma consulta.	

A seguir, veja o exemplo de banco de dados que será utilizado:

- DESCRIBE na tabela Veículos

```
mysql> describe Veiculos;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| Id     | int(3)        | NO   | PRI | NULL    | auto_increment |
| Marca  | varchar(30)   | NO   |     | NULL    |                |
| Modelo | varchar(30)   | NO   |     | NULL    |                |
| Valor  | decimal(10,2) | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

- SELECT na tabela Veículos

```
mysql> select * from Veiculos;
+----+-----+-----+-----+
| Id | Marca          | Modelo      | Valor      |
+----+-----+-----+-----+
| 1  | BMW            | 320i        | 160000.00  |
| 2  | Mercedes-Benz  | C180        | 140000.00  |
| 3  | Hyundai        | Azera       | 120000.00  |
| 4  | Mercedes-Benz  | CLA 200     | 140000.00  |
| 5  | BMW            | 328i        | 210000.00  |
| 6  | Volkswagen     | Passat      | 140000.00  |
| 7  | BMW            | 316i        | 115000.00  |
| 8  | Mercedes-Benz  | Classe E    | 248000.00  |
| 9  | Mercedes-Benz  | C 250       | 180000.00  |
| 10 | Jaguar         | XF          | 220000.00  |
| 11 | BMW            | 535i        | 500000.00  |
| 12 | Jaguar         | VZ          | NULL       |
+----+-----+-----+-----+
12 rows in set (0.00 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

## COUNT

Conta o número de registros de uma relação.

Sintaxe:

```
SELECT COUNT(*) FROM <tabela>
```

No nosso exemplo, temos :

```
SELECT COUNT(*) FROM Veiculos;
```

Resultado do exemplo

```
mysql> select count(*) from Veiculos;
+-----+
| count(*) |
+-----+
|       12 |
+-----+
1 row in set (0.00 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

Com a função COUNT o SQL, efetuou a contagem de todas as colunas, ao se utilizar a opção “(\*)”. Ao invés de utilizarmos “\*”, que propicia a contagem de todas as colunas, podemos fazer o apontamento para uma coluna em específico.

Sintaxe:

```
SELECT COUNT(Valor) FROM Veiculos;
```

Resultado do exemplo

```
mysql> select count(Valor) from Veiculos;
+-----+
| count(Valor) |
+-----+
|           11 |
+-----+
1 row in set (0.00 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

## ATENÇÃO

A função COUNT não efetua a contagem de tuplas com valor nulo. Pois o contador ignora os registros em que se tenha valor nulo (NULL). Portanto, é necessário cuidado ao escolher a coluna, pois ao tomar a coluna “Valor” para determinar a quantidade de veículos registrados no banco de dados, seríamos induzidos ao erro.

## \* DISTINCT

Para evitar resultados com vários registros repetidos, devemos agregar a função DISTINCT. Assim, o SQL vai selecionar os registros distintamente, evitando informações redundantes.

Sintaxe:

```
SELECT COUNT(DISTINCT MARCA) FROM Veiculos;
```

Resultado do exemplo



```
mysql> SELECT COUNT(DISTINCT MARCA) FROM Veiculos;
+-----+
| COUNT(DISTINCT MARCA) |
+-----+
| 5 |
+-----+
1 row in set (0.00 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

## MINIMUM (MIN)

Determina o menor valor de registro em uma coluna.

Sintaxe:

```
SELECT MIN(<coluna>) FROM <tabela>;
```

No nosso exemplo, vamos selecionar a marca, modelo e o veículo de menor valor registrado na tabela:

```
SELECT Marca, Modelo, MIN(Valor) as “Menor Valor” FROM Veiculos;
```

Resultado do exemplo ▾

```
mysql> SELECT Marca, Modelo, MIN(Valor) as "Menor Valor" FROM Veiculos;
+-----+-----+-----+
| Marca | Modelo | Menor Valor |
+-----+-----+-----+
| BMW   | 320i   | 115000.00   |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

## MAXIMUM (MAX)

Determina o maior valor de registro em uma coluna.

Sintaxe:

```
SELECT MAX(<coluna>) FROM <tabela>;
```

No nosso exemplo, vamos selecionar o modelo do veículo de maior valor registrado na tabela:

```
SELECT Marca, Modelo, MAX(Valor) as “Maior Valor” FROM Veiculos;
```

```
mysql> SELECT Modelo, MAX(Valor) as "Maior Valor" FROM Veiculos;
+-----+-----+
| Modelo | Maior Valor |
+-----+-----+
| 320i   | 500000.00   |
+-----+-----+
1 row in set (0.00 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

## AVERAGE (AVG)

A função AVG (abreviação do termo inglês *average*, que quer dizer média), retorna a média dos valores em uma determinada coluna. Para isso o SQL faz a somatória dos valores (obrigatoriamente numéricos) e divide o resultado pelo número de registros diferentes de nulo (NULL).

Sintaxe:

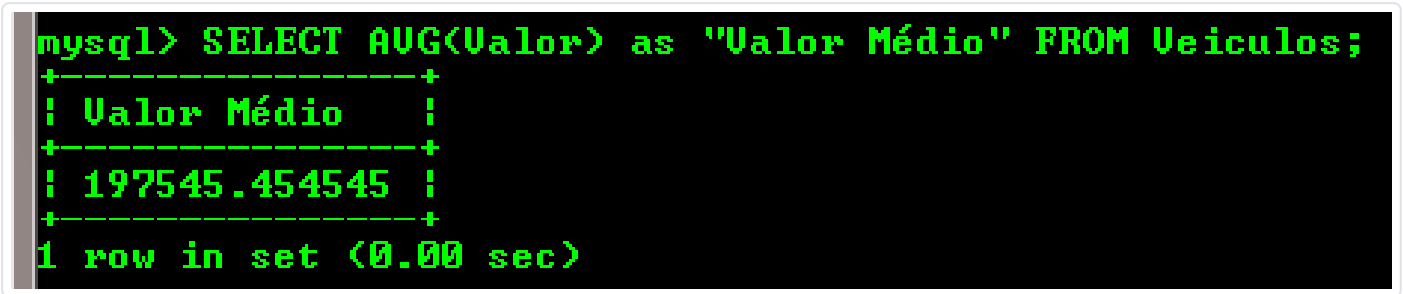
```
SELECT AVG(<coluna>) FROM <tabela>;
```

No nosso exemplo, vamos selecionar o valor médio dos veículos registrados na tabela:

```
SELECT AVG(Valor) as “Valor Médio” FROM Veiculos;
```

Resultado do exemplo ▾

Com esta sintaxe sendo aplicada, embora haja 12 veículos registrados (sendo um com o valor nulo), o SQL efetuou a somatória somente dos veículos com valor diferente de nulo, e fez a divisão pelo número de registros com valor também diferente de nulo.



Fonte: elaborada pelo autor, captura de tela do software MySQL.

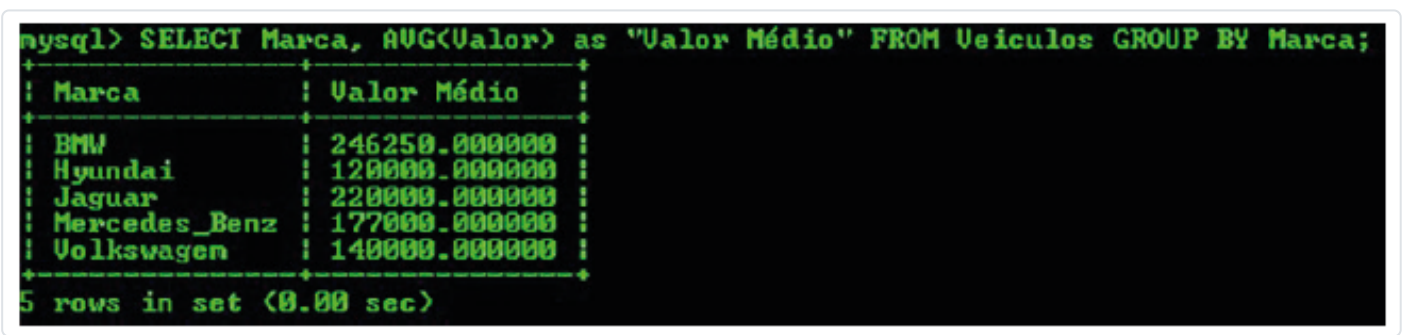
\* GROUP BY

A função de agregação AVG permite que o qualificador GROUP BY seja utilizado em conjunto. No nosso exemplo, deseja-se selecionar o valor médio dos veículos conforme as marcas, agrupando as informações.

Sintaxe:

```
SELECT Marca, AVG(Valor) as “Valor Médio”  
FROM Veiculos  
GROUP BY Marca;
```

Resultado do exemplo ▾



Fonte: elaborada pelo autor, captura de tela do software MySQL.

TOTAL (SUM)

Retorna o somatório dos valores em uma determinada coluna. Para isso o SQL faz o somatório dos valores (obrigatoriamente numéricos).

Sintaxe:

```
SELECT SUM(<coluna>) FROM <tabela>;
```

No nosso exemplo, vamos somatória do valor dos veículos registrados na tabela:

```
SELECT SUM(Valor) as "Total" FROM Veiculos;
```

Resultado do exemplo ▼

```
mysql> SELECT SUM(Valor) as "Total" FROM Veiculos;
+-----+
| Total |
+-----+
| 2173000.00 |
+-----+
1 row in set (0.00 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

Nesta webaula vimos como utilizar as funções de agregação em banco de dados, por meio da linguagem de programação de banco de dados SQL. E como essas funções permitem desenvolver consultas interessantes nas bases de dados.



Fonte: Shutterstock.

# SQL

## Subconsultas em banco de dados

Você sabia que seu material didático é interativo e multimídia? Isso significa que você pode interagir com o conteúdo de diversas formas, a qualquer hora e lugar. Na versão impressa, porém, alguns conteúdos interativos ficam desabilitados. Por essa razão, fique atento: sempre que possível, opte pela versão digital. Bons estudos!

Nesta webaula vamos ver como desenvolver subconsultas e como comparar grupos de seleções de dados. Com isso, será possível efetuar consultas avançadas nos bancos de dados, com aplicação de filtros.

## Subconsultas

Uma subconsulta é uma expressão em SQL, composta por SELECT-FROM-WHERE, que é aninhada dentro de outra consulta, permitindo fazer comparações entre os conjuntos de dados.

A sintaxe no SQL permite fazer consultas, em que podem ser utilizadas diversas relações entre inúmeras tabelas presentes nos bancos de dados, bastando a utilização dos conectivos. Para isso, temos o conectivo IN (que efetua o teste no conjunto de dados, em que esse conjunto é fruto de uma coleção de valores produzidos por meio de um SELECT) e o conectivo NOT IN (permite efetuar a ausência em um conjunto de valores).

## Exemplos

Nos exemplos que serão apresentados a seguir, vamos compreender como a subconsulta pode ser encadeada à consulta, ou seja, um SELECT dentro de outro SELECT, para testar a relação de um atributo no relacionamento entre as tabelas no banco de dados.

Representação do esquema do banco de dados

- aluno (RA, nome, telefone)
- funcionario (matricula, nome, cargo)
- livro (isbn, nome, secao)
- emprestimo (numero, retirada, devolução, aluno\_RA, funcionário\_matricula, livro\_isbn)
- restricao (Id, aluno\_RA, livro\_isbn)

SELECT – tabela aluno

```
mysql> select * from aluno;
+----+-----+-----+
| RA  | nome      | telefone |
+----+-----+-----+
| 11223 | Serj Tankian | 987658899 |
| 12345 | Joye Ramone  | 991213344 |
| 54321 | Lars Ulrich  | 977889966 |
| 56789 | Corey Taylor | 901238525 |
| 98765 | Vicky Psarakis | 922556688 |
+----+-----+-----+
5 rows in set (0.08 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

SELECT – tabela funcionario

```
mysql> select * from funcionario;
+-----+-----+-----+
| matricula | nome           | cargo           |
+-----+-----+-----+
| 1         | Melvil Dewey   | Bibliotecario 1 |
| 2         | Manuel Bastos Tigre | Bibliotecario 2 |
+-----+-----+-----+
2 rows in set (0.07 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

SELECT – tabela livro

```
mysql> select * from livro;
+-----+-----+-----+
| isbn | nome           | secao           |
+-----+-----+-----+
| 11111 | Uida Punk      | musica          |
| 22222 | Mestres da Bateria | musica          |
| 33333 | Sexta-feira 13  | terror          |
| 44444 | Mulheres do Rock | musica          |
| 55555 | O exorcista     | terror          |
| 66666 | O chamado      | terror          |
| 77777 | Mascaras       | musica          |
| 88888 | ToxiCity       | musica          |
| 99999 | Diario de un Mago | esoterismo      |
+-----+-----+-----+
9 rows in set (0.00 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

SELECT – tabela emprestimo

```
mysql> select * from emprestimo;
+-----+-----+-----+-----+-----+-----+
| numero | retirada | devolucao | aluno_RA | funcionario_matricula | livro_isbn |
+-----+-----+-----+-----+-----+-----+
| 8      | 2018-01-02 | 2018-01-17 | 12345    | 1                     | 11111      |
| 9      | 2018-01-15 | 2018-02-01 | 11223    | 2                     | 88888      |
| 10     | 2018-04-05 | 2018-04-20 | 56789    | 2                     | 77777      |
| 11     | 2018-03-15 | 2018-03-30 | 98765    | 1                     | 44444      |
| 12     | 2018-06-06 | 2018-06-21 | 56789    | 1                     | 55555      |
| 13     | 2018-08-01 | 2018-08-16 | 12345    | 2                     | 22222      |
| 14     | 2018-10-10 | 2018-10-25 | 11223    | 1                     | 66666      |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.03 sec)
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

SELECT – tabela restricao

```
mysql> select * from restricao;
+-----+-----+-----+
| Id | aluno_RA | livro_isbn |
+-----+-----+-----+
| 1  | 12345    | 22222      |
+-----+-----+-----+
```

Fonte: elaborada pelo autor, captura de tela do software MySQL.

Conectivo IN

Sintaxe:

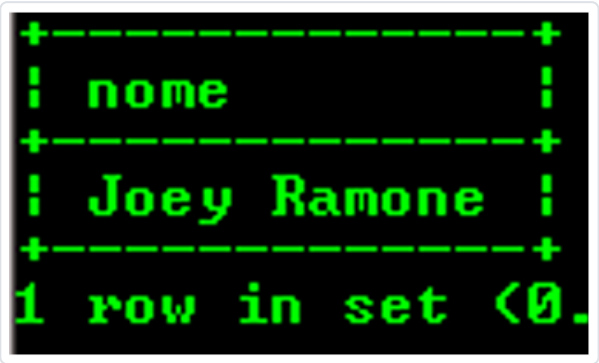
```
SELECT [campo]
FROM [tabela]
WHERE [campo] IN (SELECT [campo] FROM [tabela]);
```

**Exemplo:** gerar uma consulta que encontre o nome de todos os alunos que efetuaram empréstimos, porém estão com restrição para locar novos livros.

```
SELECT aluno.nome
FROM aluno
WHERE aluno.RA IN (SELECT aluno_RA FROM restricao);
```

Resultado do exemplo

Esta sintaxe diz que deve **SELECIONAR** o nome do aluno, **NA** tabela aluno, **QUANDO** o RA do aluno estiver na (seleção do RA do aluno na tabela restricao).



Fonte: elaborada pelo autor, captura de tela do software MySQL.

**Outro exemplo:** consultar o nome do livro que o aluno “Joey Ramone” não entregou, deixando-o com restrição para efetuar a locação de um novo livro.

Sintaxe:

```
SELECT aluno.nome as “ALUNO”, livro.nome as “LIVRO”
FROM aluno, livro
WHERE aluno.RA IN (SELECT aluno_RA FROM restricao) AND
Livro.isbn IN (SELECT livro_isbn FROM restricao);
```

Resultado do exemplo



Fonte: elaborada pelo autor, captura de tela do software MySQL.

Conectivo NOT IN

Embora os conectivos possuam sintaxes idênticas, a sua aplicação é diferente, pois o NOT IN permite que a seleção seja negada.

Sintaxe:

```
SELECT [campo]
FROM [tabela]
WHERE [campo] NOT IN (SELECT [campo] FROM [tabela]);
```

**Exemplo:** selecionar o nome dos alunos que nunca tomaram um livro emprestado.

```
SELECT aluno.nome as "ALUNO"
FROM aluno
WHERE aluno.RA NOT IN (SELECT aluno_RA FROM emprestimo);
```

Resultado do exemplo

Esta sintaxe irá **SELECIONAR** o nome do aluno, **NA** tabela aluno, **QUANDO** o RA do aluno **NÃO** estiver nela (seleção do RA do aluno na tabela emprestimo).



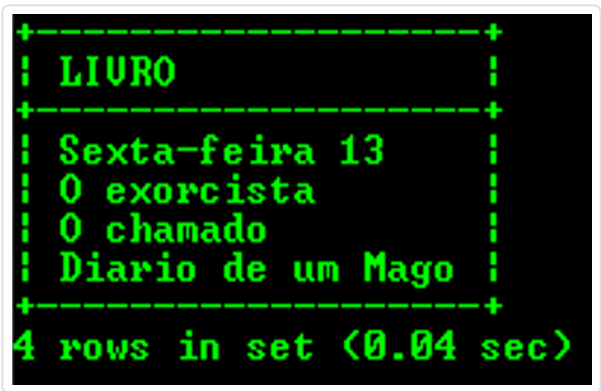
Fonte: elaborada pelo autor, captura de tela do software MySQL.

Não necessariamente as subconsultas necessitam serem feitas em duas tabelas relacionadas, sendo possível utilizar esse recurso em apenas uma tabela.

**Exemplo:** fazer uma seleção do nome dos livros, em que são ignorados os livros da seção “música”:

```
SELECT nome as "LIVRO"
FROM livro
WHERE secao NOT IN (SELECT secao FROM emprestimo where secao = "música");
```

Resultado do exemplo



Fonte: elaborada pelo autor, captura de tela do software MySQL.

# Comparação de Conjuntos

A sintaxe SQL permite o desenvolvimento de subconsultas aninhadas em que é possível fazer a comparação entre conjuntos de dados, utilizando-se condições (WHERE). Porém para que seja efetuado as comparações deve ser inserido a palavra some na sintaxe, nos operadores comparativos.

Operador Matemático	SELECT com WHERE SQL	Subconsulta SQL
=	WHERE campo = condição	WHERE campo = some (SELECT)
≠	WHERE campo <> condição	WHERE campo <> some (SELECT)
>	WHERE campo > condição	WHERE campo > some (SELECT)
≥	WHERE campo >= condição	WHERE campo >= some (SELECT)
<	WHERE campo < condição	WHERE campo < some (SELECT)

Operador Matemático	SELECT com WHERE SQL	Subconsulta SQL
≤	WHERE campo <= condição	WHERE campo <= some (SELECT)

Fonte: adaptada de Silberschatz (2010)

Exemplo de comparação de conjuntos


Realizar uma consulta a nomes dos livros e a seção, se a quantidade de livros da seção “esoterismo” for menor do que a quantidade de livros.

Sintaxe:

```
SELECT nome as “Livro”, secao as “Seção”
FROM livro
WHERE NOME > some (SELECT nome FROM livro WHERE secao = “esoterismo”;
```

Consulta na comparação de conjuntos

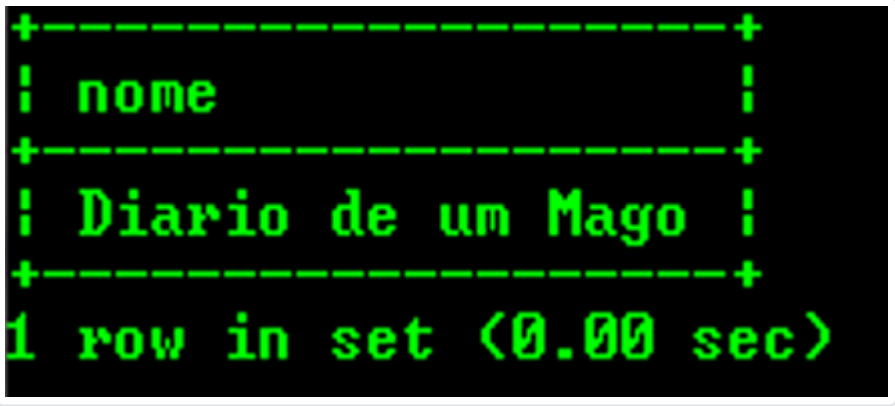
A consulta “SELECT nome as "Livro", secao as "Seção" FROM livro” gera a saída com nove registros.



Fonte: elaborada pelo autor, captura de tela do software MySQL.

Subconsulta na comparação de conjuntos 1

A subconsulta SELECT nome FROM livro WHERE secao = "esoterismo" gera apenas uma saída.



Fonte: elaborada pelo autor, captura de tela do software MySQL.

Subconsulta verdadeira

O trecho da sintaxe SQL “WHERE nome > some”, faz a comparação nome de livros 9 é maior que 1 registro de livro da seção esoterismo. Como essa afirmativa é verdadeira, pois 9 é maior do que 1, tem se a subconsulta verdadeira.



Livro	Seção
Uida Punk	musica
Mestres da Bateria	musica
Sexta-feira 13	terror
Mulheres do Rock	musica
O exorcista	terror
O chamado	terror
Mascaras	musica
ToxiCity	musica

8 rows in set (0.00 sec)

Fonte: elaborada pelo autor, captura de tela do software MySQL.

Subconsulta falsa

Se for invertida a condição, onde é trocado o sinal de maior (>) por menor (<), nenhum livro será exibido. Isso ocorre porque a condição “WHERE nome > some”, é falsa. Analisando matematicamente, temos que 9 não é maior do que 1.

Empty set (0.00 sec)
----------------------

Fonte: elaborada pelo autor, captura de tela do software MySQL.

Vimos nesta webaula que as técnicas de subconsultas permitem que possamos realizar consultas, dentro de outras consultas. Portanto, as subconsultas são recursos de seleção de dados que pode proporcionar diversas aplicações práticas muito interessantes.



Fonte: Shutterstock.

Para visualizar o vídeo, acesse seu material digital.