

Introdução a Banco de dados

Comandos SQL – Parte 6

SQL – Adicionar novos registros

```
INSERT INTO tbl_Editoras (Nome_Editora)
```

```
VALUES
```

```
('Apress'),('Bookman'),('Bookboon'),('Packtpub'),('Springer'),('Érica'),('For  
Dummies'),('Novatec'),('Cisco Press'),('Addison-Wesley'),('Companhia  
das Letras'),('Artech House'),('CRC Press'),('Manning'),('Penguin  
Books'),('Sage Publishing'),('Publishing');
```

SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

Funções de agregação são funções SQL que permitem executar uma operação aritmética nos valores de uma coluna em todos os registros de uma tabela.

SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

Retornam um valor simples baseado em um conjunto de valores de entrada.

SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

Sintaxe básica:

função(ALL | DISTINCT expressão)

- ◎ **ALL** – avalia todos os registros ao agregar o valor da função; é o comportamento padrão.
- ◎ **DISTINCT** – Usa apenas valores distintos (sem repetição) ao avaliar a função.

SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

As principais funções de agregação (mais comuns) em SQL são as seguintes:

- ① MIN = Valor Mínimo de um conjunto de valores
- ① MAX = Valor Máximo de um conjunto de valores
- ① AVG = Média Aritmética de um conjunto de valores
- ① SUM = Total (Soma) de um conjunto de valores
- ① COUNT = Contar quantidade total de itens

SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

Exemplo:

Retornar o número total de autores cadastrados na tabela de autores:

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

SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

Exemplo:

Contar o número de autores que possuem livros cadastrados na tabela de autores, sem repetições:

```
SELECT COUNT(DISTINCT id_autor) FROM  
tbl_Livro;
```


SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

Exemplo:

Descobrir o preço mais alto dos livros:

```
SELECT MAX(Preco_Livro) FROM tbl_Livro;
```

SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

Exemplo:

Descobrir a data de publicação do livro mais antigo:

```
SELECT MIN(Data_Pub) FROM tbl_Livro;
```

SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

Exemplo:

Retornar o preço médio dos livros cadastrados no banco:

```
SELECT AVG(Preco_Livro) FROM tbl_Livro;
```

SQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM)

Exemplo:

Descobrir o valor total dos livros presentes na tabela de livros:

```
SELECT SUM(Preco_Livro) FROM tbl_Livro;
```

Arredondar valores em consultas com função **ROUND()** no SQL

Uma tarefa comum ao realizar uma **consulta SQL** é efetuar o arredondamento de valores numéricos para um número de casas decimais pré-determinado. O SQL possui funções que permitem realizar esse arredondamento, e uma delas é a função **ROUND()**, que estudaremos neste tutorial.

Arredondar valores em consultas com função **ROUND()** no SQL

Função **ROUND()**

ROUND(valor, casas_decimais)

Parâmetros:

- ① **valor**: o número que se quer arredondar
- ② **casas_decimais**: número de casas decimais desejadas. Se omitido, retorna o valor sem nenhuma casa decimal (somente parte inteira).

Arredondar valores em consultas com função **ROUND()** no SQL

Arredondar o número 52.36956 para duas casas decimais:

SELECT ROUND(52.36956, 2) AS Arredondado;

Arredondar valores em consultas com função **ROUND()** no SQL

Arredondar os valores da coluna de preços dos livros para uma casa decimal:

```
SELECT Nome_Livro, ROUND(Preco_Livro, 1) AS  
Preço  
FROM tbl_Livro;
```

	NomeLivro	Preço
▶	Linux Command Line and Shell Scripting	182.6
	SSH, O Shell Seguro	324.5
	Using Samba	174.9
	A Arte da Eletrônica	194.7
	Windows Server 2012 Inside Out	198.0
	Murach's MySQL	271.8
	Practical Electronics for Inventors	132.0

Arredondar valores em consultas com função **ROUND()** no SQL

Arredondar o valor médio calculado sobre uma coluna de valores numéricos:

```
SELECT ROUND(AVG(Preco_Livro), 2) AS 'Preço  
Médio dos Livros'  
FROM tbl_Livro;
```

Arredondar valores para baixo com a função **FLOOR()** no **SQL**

A função **FLOOR()** (“piso”, em português) no **SQL** retorna o maior valor inteiro que é menor ou igual a um número passado como argumento. Em outras palavras, esta função arredonda um número decimal para baixo, mostrando apenas sua parte inteira.

Arredondar valores para baixo com a função **FLOOR()** no SQL

FLOOR(número)

Onde o parâmetro **número** é o valor que se deseja arredondar para baixo (inteiro sem parte decimal).

Arredondar valores para baixo com a função **FLOOR()** no SQL

Arredondar um número decimal passado como argumento:

```
SELECT FLOOR(63.75) AS Arredondado;
```

Arredondar valores para baixo com a função **FLOOR()** no SQL

Retornar um valor monetário armazenado em uma coluna de tabela, ignorando os centavos (casas decimais):

```
SELECT Preco_Livro AS 'Preco Real',  
FLOOR(Preco_Livro) AS 'Reais sem centavos'  
FROM tbl_Livro  
WHERE Preco_Livro > 60.00;
```

Preco Real	Reais sem centavos
68.35	68
61.45	61
62.24	62
66.80	66
67.80	67

Truncar valores em consultas com a função **TRUNCATE no SQL**

Todos os valores foram arredondados para baixo (número inteiro mais próximo inferior ao valor), independente da quantidade de dígitos presentes nas casas decimais.

Truncar valores em consultas com a função **TRUNCATE** no SQL

A função TRUNCATE trunca (“corta”) um valor numérico, mostrando um número especificado de casas decimais.

Note que esta função não arredonda valores, apenas oculta as casas decimais, de acordo com a quantidade que será exibida.

Truncar valores em consultas com a função **TRUNCATE** no SQL

Sintaxe:

TRUNCATE(valor, casas_decimais)

Parâmetros:

- ⦿ **valor:** o número a ser truncado
- ⦿ **casas_decimais:** números de casas decimais (depois da vírgula) a serem exibidas.

Truncar valores em consultas com a função **TRUNCATE** no SQL

Truncar um número para não exibir casas decimais (0 casas)

**SELECT CAST (52.69863 as DECIMAL (5,2)) AS
TRUNCADO;** Contraste o resultado retornado como
obtido com a função **ROUND()** aplicada no mesmo valor:
SELECT ROUND(52.69863, 0) AS Arredondado;

Arredondar valores para cima no SQL com função **CEILING()**

A função **CEILING()** (“teto”, em português) retorna o menor valor inteiro que é maior ou igual a um número. Em outras palavras, permite arredondar um número decimal qualquer para cima (para o inteiro mais próximo superior), mostrando apenas sua parte inteira.

Arredondar valores para cima no SQL com função CEILING()

Sintaxe:

CEILING(número)

Onde o parâmetro número indica o valor que se deseja arredondar para baixo.

Arredondar valores para cima no SQL com função CEILING()


Arredondar um número decimal para cima:

```
SELECT CEILING(72.25) AS 'Arredonda para cima';
```

Arredondar valores para cima no SQL com função CEILING()

Retornar a parte inteira de um valor armazenado em uma coluna de tabela:

```
SELECT Preço_Livro AS 'Preço Real',  
       CEILING(Preço_Livro) AS 'Arredondado para cima'  
FROM tbl_Livro  
WHERE Preço_Livro > 60.00
```


Result Grid		 Filter Rows:
	Preço Real	Arredondado para cima
	68.35	69
	61.45	62
	62.24	63
	66.80	67
	67.80	68

Arredondar valores para cima no SQL com função CEILING()

O mesmo que o anterior, mas usando a função **CEIL()**:

```
SELECT Preço_Livro AS 'Preço Real', CEILING(Preço_Livro) AS  
'Arredondado para cima'  
FROM tbl_Livro  
WHERE Preço_Livro > 60.00;
```

Note que a função **CEIL()** é idêntica à função de nome **CEILING()**, podendo ser usada no lugar de

Result Grid		 Filter Rows:	
€	Preço Real	Arredondado para cima	
▶	68.35	69	
	61.45	62	
	62.24	63	
	66.80	67	
	67.80	68	

SQL – UPDATE – Modificar Registros em Tabelas

Para alterar um registro e uma tabela usamos o comando UPDATE, segundo a sintaxe abaixo:

UPDATE tabela

SET coluna = novo_valor_armazenado

WHERE coluna = valor_filtro;

Observação importante: Caso não seja usada a cláusula WHERE para filtrar os registros, todos os dados da coluna serão alterados!

SQL – UPDATE – Modificar Registros em Tabelas

Iremos alterar o nome de um livro na tabela `tbl_livros`, cujo ID é igual a 2, para “SSH, o Shell Seguro”:

```
UPDATE tbl_Livro  
SET Nome_Livro = 'SSH, o Shell Seguro'  
WHERE ID_LIVRO = 2;
```


SQL – UPDATE – Modificar Registros em Tabelas

Vamos aumentar o preço do livro cujo ISBN13 é 9780735640610 para R\$ 47,20 (atualmente ele custa R\$ 45,30):

```
UPDATE tbl_livro
```

```
SET Preco_Livro = 47.20
```

```
WHERE ISBN13 = '9780735640610';
```

SQL – BETWEEN – Seleção de intervalos em consultas

Podemos usar a cláusula BETWEEN para, por exemplo, retornar registros cujos preços estejam entre dois valores distintos, ou registros contidos dentro de um intervalo de datas especificado.

SQL – BETWEEN – Seleção de intervalos em consultas

A sintaxe para uso da cláusula BETWEEN é a seguinte:

SELECT colunas **FROM** tabela

WHERE coluna **BETWEEN** valor1 **AND** valor2;

SQL – BETWEEN – Seleção de intervalos em consultas

Vamos retornar todos os livros da tabela `tbl_livro` cuja data de publicação esteja entre 17/05/2004 e 17/05/2011 (note como a data é fornecida no código: ano|mês|dia):

```
SELECT * FROM tbl_Livro  
WHERE Data_Pub BETWEEN '20040517' AND  
'20110517';
```

ID_Livro	Nome_Livro	ISBN13	ISBN10	ID_Categoria	ID_Autor	ID_Editora	Data_Pub
2	SSH, the Secure Shell	9780596008956	0596008953	1	1	2	2005-05-17
6	Microsoft Exchange Server 2010	9780735640610	0735640610	1	4	3	2010-12-01
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL – BETWEEN – Seleção de intervalos em consultas

Agora vamos retornar os nomes dos livros e seus respectivos preços, da tabela `tbl_livros`, porém somente os livros cujos preços estiverem entre R\$ 40,00 e 60,00:

```
SELECT Nome_Livro AS Livro, Preco_Livro AS  
Preço
```

```
FROM tbl_Livro
```

```
WHERE Preço_Livro BETWEEN 40.00 AND 60.00;
```

Livro	Preço
SSH, the Secure Shell	58.30
Microsoft Exchange Server 2010	45.30

SQL – LIKE e NOT LIKE – Padrões de caracteres em consultas

Quando realizamos uma consulta no SQL, utilizamos a cláusula WHERE para realizar um filtro dos registros a retornar. Porém, com o WHERE, só podemos aplicar filtros de correspondência exata de palavras. E se precisarmos aplicar um filtro que verifique palavras de forma parcial, como palavras que iniciem ou terminem com determinados caracteres, ou que possuam sequências de caracteres específicas? Neste caso, usamos a cláusula LIKE:

SQL – LIKE e NOT LIKE – Padrões de caracteres em consultas

- ◎ A cláusula LIKE determina se uma cadeia de caracteres (string) corresponde a um padrão especificado. Um padrão pode incluir caracteres normais e curingas.
- ◎ NOT LIKE inverte a comparação, verificando se a cadeia de caracteres NÃO corresponde ao padrão especificado.

SQL – LIKE e NOT LIKE – Padrões de caracteres em consultas

Padrões específicos (metacaracteres)

Usamos diversos conjuntos de caracteres para especificar os padrões a serem filtrados pelas cláusulas LIKE e NOT LIKE. Por exemplo:

- '%' — Qualquer cadeia de 0 ou mais caracteres
- '_' — Sublinhado: qualquer caracter único

SQL – LIKE e NOT LIKE – Padrões de caracteres em consultas

Selecionar todos os registros da tabela `tbl_livro` cujo nome comece com a letra F:

```
SELECT * FROM tbl_Livro  
WHERE Nome_Livro LIKE 'F%';
```

	Nome_Livro
▶	Fedora and Red Hat Linux

SQL – LIKE e NOT LIKE – Padrões de caracteres em consultas

Selecionar todos os registros da tabela `tbl_livro` cujo nome não começa com a letra S:

```
SELECT * FROM tbl_Livro  
WHERE Nome_Livro not LIKE 'S%';
```

	ID_Livro	Nome_Livro	ISBN13	ISBN10	ID_Categoria	ID_Autor	ID_Editora	Data_Pub	Preco_Livro
▶	1	Linux Command Line and Shell Scripting	9781118983843	111898384X	1	5	4	2015-01-09	68.35
	3	Using Samba	9780596002565	0596002564	1	2	2	2003-12-21	61.45
	4	Fedora and Red Hat Linux	9780133477436	0133477436	1	3	1	2014-01-10	62.24
	5	Windows Server 2012 Inside Out	9780735666313	0735666318	1	4	3	2013-01-25	66.80
	6	Microsoft Exchange Server 2010	9780735640610	0735640610	1	4	3	2010-12-01	47.20

SQL – LIKE e NOT LIKE – Padrões de caracteres em consultas

Selecionar os nomes de livros da tabela `tbl_livro` cujo nome se inicie com uma letra qualquer e a segunda letra seja a letra `i`:

```
SELECT Nome_Livro  
FROM tbl_Livro  
WHERE Nome_Livro LIKE '_i%'
```

	Nome_Livro
▶	Linux Command Line and Shell Scripting
	Windows Server 2012 Inside Out
	Microsoft Exchange Server 2010

SQL – LIKE e NOT LIKE – Padrões de caracteres em consultas

Selecionar os nomes dos livros e seus respectivos preços, na tabela de livros, cujo nome não comece com a letra F e que custem mais de R\$ 60,00:

```
SELECT Nome_Livro AS Livro, Preco_Livro AS  
Valor  
FROM tbl_livro  
WHERE Nome_Livro NOT L  
AND Preco_Livro > 60.00;
```

	Livro	Valor
▶	Linux Command Line and Shell Scripting	68.35
	Using Samba	61.45
	Windows Server 2012 Inside Out	66.80
	Practical Electronics for Inventors	67.80

SQL – GROUP BY – Agrupamento de Registros

Usamos a cláusula GROUP BY para agrupar registros em subgrupos baseados em colunas ou valores retornados por uma expressão.

Com o GROUP BY podemos agrupar os valores de uma coluna e também realizar cálculos sobre esses valores. Desta forma, ao realizarmos uma consulta, os valores encontrados nas linhas são agrupados e então uma função de agregação pode ser aplicada sobre esses grupos.

SQL – GROUP BY – Agrupamento de Registros

Sintaxe:

SELECT colunas, função_agregação()

FROM tabela

WHERE filtro

GROUP BY coluna

SQL – GROUP BY – Agrupamento de Registros

Criar uma tabela para testarmos o GROUP BY:

```
CREATE TABLE tbl_Vendas2 (  
ID Smallint Primary Key,  
Nome_Vendedor Varchar(30),  
Quantidade Int,  
Produto Varchar(20),  
Cidade Varchar(20)  
);
```

SQL – GROUP BY – Agrupamento de Registros

Inserir registros na tabela criada para teste de GROUP BY:

```
INSERT INTO tbl_Vendas2 (ID, Nome_Vendedor, Quantidade, Produto, Cidade)
```

```
VALUES
```

```
(10,'Jorge',1400,'Mouse','São Paulo'),
```

```
(12,'Tatiana',1220,'Teclado','São Paulo'),
```

```
(14,'Ana',1700,'Teclado','Rio de Janeiro'),
```

```
(15,'Rita',2120,'Webcam','Recife'),
```

```
(18,'Marcos',980,'Mouse','São Paulo'),
```

```
(19,'Carla',1120,'Webcam','Recife'),
```

```
(22,'Roberto',3145,'Mouse','São Paulo');
```


SQL – GROUP BY – Agrupamento de Registros

Consulta usando agregação para obter o total de vendas de Mouses (sem o GROUP BY):

```
SELECT SUM(Quantidade) As TotalMouses  
FROM tbl_Vendas2  
WHERE Produto = 'Mouse';
```

	TotalMouses
▶	5525

SQL – GROUP BY – Agrupamento de Registros

Consulta totalizando as vendas de todos os produtos por cidade:

```
SELECT Cidade, SUM(Quantidade) As Total  
FROM tbl_Vendas2  
GROUP BY Cidade;
```

	Cidade	Total
▶	Recife	3240
	Rio de Janeiro	1700
	São Paulo	6745

SQL – GROUP BY – Agrupamento de Registros

Consulta contando o número de registros de vendas (quantidade de vendas) por cidade:

```
SELECT Cidade, COUNT(*) As Total  
FROM tbl_Vendas2  
GROUP BY Cidade;
```

	Cidade	Total
▶	Recife	2
	Rio de Janeiro	1
	São Paulo	4

SQL – GROUP BY – Agrupamento de Registros

Consulta com o total (quantidades) nas vendas realizadas por cada vendedor:

```
SELECT Nome_Vendedor, SUM(Quantidade)  
FROM tbl_Vendas2  
GROUP BY Nome_Vendedor;
```

	Nome_Vendedor	SUM(Quantidade)
▶	Ana	1700
	Carla	1120
	Jorge	1400
	Marcos	980
	Rita	2120

SQL – HAVING – Filtrando os resultados do Agrupamento

A cláusula HAVING é usada para especificar condições de filtragem em grupos de registros ou agregações.

É frequentemente usada em conjunto com a **cláusula GROUP BY** para filtrar as colunas agrupadas.

SQL – HAVING – Filtrando os resultados do Agrupamento

Sintaxe:

```
SELECT colunas, função_agregação()  
FROM tabela  
WHERE filtro  
GROUP BY colunas  
HAVING filtro_agrupamento
```

SQL – HAVING – Filtrando os resultados do Agrupamento

Consulta retornando total de vendas das cidades com menos de 2500 produtos vendidos:

```
SELECT Cidade, SUM(Quantidade) As Total  
FROM tbl_Vendas2  
GROUP BY Cidade  
HAVING SUM(Quantidade) < 2500
```

	Cidade	Total
▶	Rio de Janeiro	1700

SQL – HAVING – Filtrando os resultados do Agrupamento

Consulta retornando total de vendas do produto 'Teclado' das cidades com menos de 1500 teclados vendidos:

```
SELECT Cidade, SUM(Quantidade) As  
TotalTeclados  
FROM tbl_Vendas2  
WHERE Produto = 'Teclado'  
GROUP BY Cidade  
HAVING SUM(Quantidade) < 1500;
```

	Cidade	TotalTeclados
▶	São Paulo	1220

Cláusula **SELECT TOP** no **SQL Server**

- ◎ A cláusula **SELECT TOP** é empregada para especificar o número de registros a retornar.
- ◎ Útil para tabelas com muitos registros.

Cláusula **SELECT TOP** no **SQL Server**

Sintaxe

```
SELECT TOP número|percentual colunas  
FROM tabela  
[ORDER BY coluna]
```

Cláusula **SELECT TOP** no **SQL Server**

Exemplo 1: Retornar os nomes do três primeiros livros da tabela de livros (por ordem de registro):

```
SELECT TOP (3) Nome_Livro, Preco_Livro  
FROM tbl_livro
```

Cláusula **SELECT TOP** no **SQL Server**

Exemplo 2: Retornar os nomes dos primeiros 10% de livros encontrados na tabela de livros (por ordem alfabética), ordenados por nome do livro:

```
SELECT TOP (10) PERCENT Nome_Livro  
FROM tbl_livro ORDER BY Nome_Livro
```

Cláusula **SELECT TOP** no **SQL Server**

Exemplo 3: Retornar os nomes dos três primeiros livros da tabela de livros (por ordem alfabética), ordenados por nome do livro:

```
SELECT TOP (3) Nome_Livro  
FROM tbl_livro ORDER BY Nome_Livro ASC;
```

Cláusula **SELECT TOP** no **SQL Server**

Exemplo 4: Retornar os nomes dos três últimos livros da tabela de livros (por ordem alfabética), ordenados por nome do livro:

```
SELECT TOP (3) Nome_Livro  
FROM tbl_livro ORDER BY Nome_Livro DESC;
```

Cláusula **SELECT TOP** no **SQL Server**

Exemplo 5: Retornar os nomes e os IDs dos três primeiros livros cadastrados na tabela:

```
SELECT TOP (3) Nome_Livro, ID_Livro  
FROM tbl_livro;
```