

Introdução a Banco de dados

Comandos SQL – Parte 5

Consultas Simples em SQL

Podemos dizer que a tarefa mais importante que um banco de dados SQL realiza, do ponto de vista do usuário, é permitir a realização de consultas aos dados armazenados. O comando básico que utilizamos para realizar as consultas é o comando **SELECT**.

Consultas Simples em SQL Server

Sintaxe:

SELECT coluna(s) **FROM** Tabela;

Consultas Simples em SQL

```
SELECT Nome_Autor FROM tbl_Autores;
```

```
SELECT * FROM tbl_Autores;
```

```
SELECT Nome_Livro FROM tbl_livro;
```

```
SELECT Nome_Livro, ID_Autor FROM tbl_livro;
```

```
SELECT Nome_Livro, ISBN10 FROM tbl_livro;
```

Consultas com ordenação

A palavra-chave ORDER BY é usada para ordenar o conjunto de resultado de registros em um consulta SQL, tanto de forma ascendente quanto descendente.

Consultas com ordenação

Sintaxe:

```
SELECT colunas FROM tabela  
ORDER BY coluna_a_ordenar;
```

Consultas com ordenação

Podemos configurar a ordenação como ascendente (crescente) ou descendente (decrescente) com o uso das palavras ASC ou DESC:

- 🎯 **ASC** – Ordem ascendente
- 🎯 **DESC** – Ordem descendente (inversa)

Ordem de Classificação por Tipo de Dados

A ordem padrão de classificação do ORDER BY (ASC) para os diversos tipos de dados é a seguinte:

- ① Valores numéricos são exibidos com os menores valores primeiro (do menor para o maior).
- ② Valores de data são mostrados com os valores menores primeiro, o que significa as datas mais antigas (do mais antigo para o mais recente).
- ③ Valores de caracteres são exibidos em ordem alfabética.
- ④ Quando houver valores nulos (NULL), eles serão mostrados por último (em sequências descendentes, com DESC, são

Ordem de Classificação por Tipo de Dados

Exemplos:

```
SELECT * FROM tbl_Livro  
ORDER BY Nome_Livro ASC;
```

Ordem de Classificação por Tipo de Dados

Exemplos:

```
SELECT * FROM tbl_Livro  
ORDER BY Nome_Livro ASC;
```

```
SELECT Nome_Livro, ID_Editora  
FROM tbl_Livro  
ORDER BY ID_Editora; -- (ordem crescente)
```

Ordem de Classificação por Tipo de Dados

Exemplos:

```
SELECT Nome_Livro, Preço_Livro  
FROM tbl_Livro  
ORDER BY Preço_Livro DESC;-- (ordem decrescente)
```

Tipo de dados **ENUM** no SQL

O tipo de dados enum é um objeto string cujo valor é escolhido a partir de uma lista de valores permitidos, enumerados de forma explícita durante a especificação de uma coluna, quando uma tabela é criada.

Tipo de dados ENUM no SQL Server

É uma forma de se economizar espaço em disco, quando se sabe de antemão que a coluna só poderá armazenar um conjunto limitado de valores. Isso ocorre porque os valores são codificados internamente automaticamente como números, independente dos dados armazenados serem números, strings, etc.; assim, independente do tamanho do dado inserido na coluna, cada registro ocupará apenas um byte de espaço neste campo.

Tipo de dados ENUM no SQL

Exemplo:

```
CREATE TABLE Camisas(idCamisa TINYINT PRIMARY KEY  
IDENTITY, Nome VARCHAR(25), tamanho VARCHAR(25) NOT  
NULL
```

```
CHECK (tamanho IN('pequena','média','grande','extra-grande'))))
```

Tipo de dados **ENUM** no SQL

Exemplo:

```
INSERT INTO Camisas (nome, tamanho)  
VALUES ('regata', 'grande');
```

```
SELECT * FROM camisas;
```

idCamisa	nome	tamanho
1	regata	grande

Tipo de dados ENUM no SQL Server

Vamos tentar inserir agora um registro de uma camisa social, porém escrevendo o tamanho “**large**” (que não consta na lista de enumeração):

```
INSERT INTO camisas (nome, tamanho)  
VALUES ('social', 'medium');
```

Agora ocorrerá um erro, pois o valor “medium” não consta na enumeração, e o registro não será inserido

Tipo de dados ENUM no SQL Server

Alterando o valor inserido para “média” resolve o problema.
Vamos aproveitar e inserir mais alguns registros:

```
INSERT INTO camisas (nome, tamanho)
```

```
VALUES
```

```
('social', 'média'),
```

```
('polo', 'pequena'),
```

```
('polo', 'grande'),
```

```
('camiseta', 'extra-grande ');
```

```
Select * from camisas
```

```
ORDER BY tamanho;
```

SQL – WHERE – Filtrar resultados de consultas

A cláusula **WHERE** permite filtrar registros nos resultados de uma consulta, de modo a trazer apenas as informações desejadas (e não o conteúdo completo das colunas).

SQL Server – WHERE – Filtrar resultados de consultas

Sintaxe:

SELECT colunas

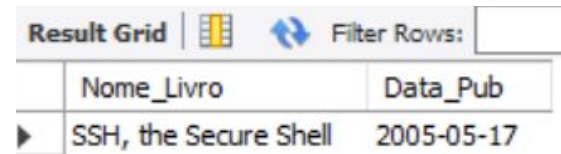
FROM tabela

WHERE coluna = valor;

SQL Server – WHERE – Filtrar resultados de consultas

Retornar o nome e a data de publicação do livro cujo ID do autor é 1:

```
SELECT Nome_Livro, Data_Pub  
FROM tbl_Livro  
WHERE ID_Autor = 1;
```



Result Grid   Filter Rows: <input type="text"/>	
Nome_Livro	Data_Pub
SSH, the Secure Shell	2005-05-17

SQL Server – WHERE – Filtrar resultados de consultas

Trazer o ID e Nome do autor do autor cujo sobrenome é Stanek:



```
SELECT ID_Autor, Nome_Autor  
FROM tbl_autores  
WHERE Sobrenome_Autor = 'Stanek';
```

	ID_Autor	Nome_Autor
▶	4	William
•	NULL	NULL

SQL Server – WHERE – Filtrar resultados de consultas

Mostrar os nomes e os preços dos livros publicados pela editora de ID igual a 3:

```
SELECT Nome_Livro, Preco_Livro  
FROM tbl_livro  
WHERE ID_Editora = 3;
```

Result Grid   Filter Rows: <input type="text"/>		
	Nome_Livro	Preco_Livro
▶	Windows Server 2012 Inside Out	66.80
	Microsoft Exchange Server 2010	45.30

SQL – AND, OR e NOT – Filtrar resultados de consultas

Operadores AND, OR e NOT no SQL

- Os operadores AND, OR e NOT são usados para filtrar registros baseados em mais de uma condição.
- O operador AND mostra um registro se ambas as condições forem verdadeiras.
- O operador OR mostra um registro se pelo menos uma das condições for verdadeira.
- O operador NOT é a negação de uma expressão (inverte seu estado lógico).

SQL Server – AND, OR e NOT – Filtrar resultados de consultas

Retornar todas as colunas da tabela de livros com os dados de livros de ID maior que 2 e ID de autor menor que 3, ao mesmo tempo:

```
SELECT * FROM tbl_Livro  
WHERE ID_Livro > 2 AND ID_Autor < 3;
```

	ID_Livro	Nome_Livro	ISBN13	ISBN10	ID_Categoria	ID_Autor	ID_Editora	Data_Pub	Preco_Livr
▶	3	Using Samba	9780596002565	0596002564	1	2	2	2003-12-21	61.45

SQL Server – AND, OR e NOT – Filtrar resultados de consultas

Trazer todos os dados da tabela de livros de livros cujo ID é maior que 2 **ou** cujo ID do autor seja menor do que 3:

```
SELECT * FROM tbl_Livro
```

```
WHERE ID_Livro > 2 OR ID_Autor < 3;
```

Result Grid								
Filter Rows:		Edit:		Export/Import:		Wrap Cell Content:		
ID_Livro	Nome_Livro	ISBN13	ISBN10	ID_Categoria	ID_Autor	ID_Editora	Data_P	
2	SSH, the Secure Shell	9780596008956	0596008953	1	1	2	2005-05	
3	Using Samba	9780596002565	0596002564	1	2	2	2003-12	
4	Fedora and Red Hat Linux	9780133477436	0133477436	1	3	1	2014-01	
5	Windows Server 2012 Inside Out	9780735666313	0735666318	1	4	3	2013-01	
6	Microsoft Exchange Server 2010	9780735640610	0735640610	1	4	3	2010-12	
7	Practical Electronics for Inventors	9781259587542	1259587541	1	13	5	2016-03	

Operadores **IN** e **NOT IN** no SQL

O operador SQL **IN** permite verificar se um valor específico corresponde a algum valor presente em uma lista ou subconsulta, passada como parâmetro a uma cláusula de filtragem **WHERE**.

Operadores IN e NOT IN no SQL Server

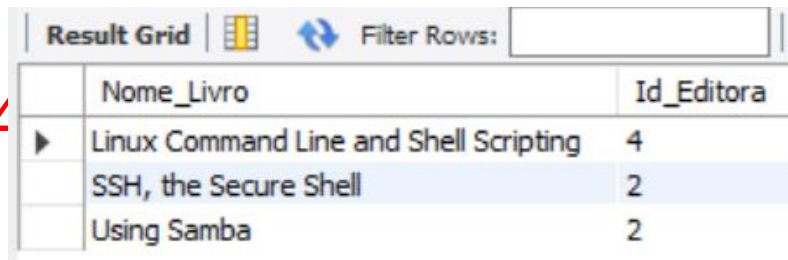
O operador IN retorna o valor 1 (true) se o valor da expressão ou da coluna na **cláusula**

WHERE corresponder a qualquer um dos valores presentes na lista passada, e retorna 0 (false) se não houver nenhuma correspondência.

Operadores IN e NOT IN no SQL Server

Retornar livros cujos Id_Editoras tem os códigos 2 ou 4:

```
SELECT Nome_Livro, Id_Editora  
FROM tbl_livro  
WHERE Id_Editora IN (2,4)
```



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. Below the header, three rows of data are displayed. The first row is 'Linux Command Line and Shell Scripting' with Id_Editora 4. The second row, 'SSH, the Secure Shell', is highlighted in blue and has Id_Editora 2. The third row, 'Using Samba', also has Id_Editora 2.

	Nome_Livro	Id_Editora
▶	Linux Command Line and Shell Scripting	4
	SSH, the Secure Shell	2
	Using Samba	2

Operadores IN e NOT IN no SQL

Retornar livros cuja edição não é da ID_Editora 1 e 2:

```
SELECT Nome_Livro, ID_Editora  
FROM tbl_livro  
WHERE ID_Editora NOT IN (1, 2);
```

Nome_Livro	ID_Editora
Linux Command Line and Shell Scripting	4
Windows Server 2012 Inside Out	3
Microsoft Exchange Server 2010	3
Practical Electronics for Inventors	5

Operadores IN e NOT IN no SQL

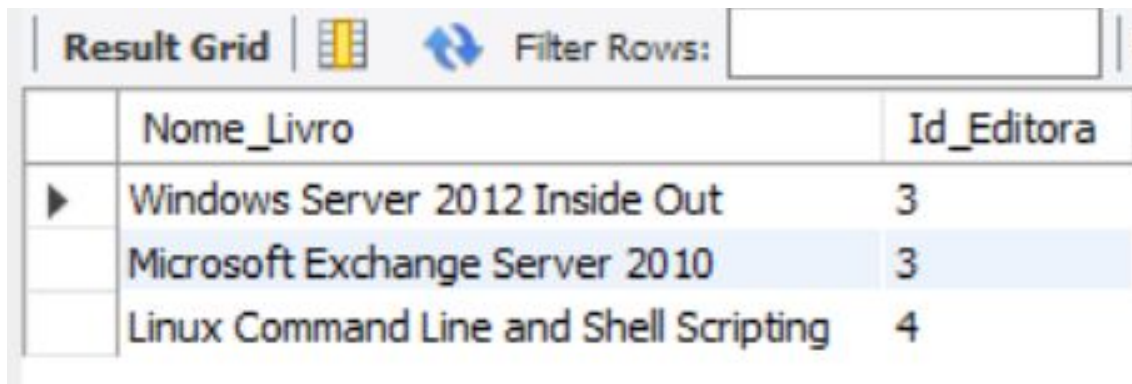
24/01

Retornar livros e IDs de editoras publicados pelas editoras Wiley ou Microsoft Press:

```
SELECT Nome_Livro, Id_Editora  
FROM tbl_livro  
WHERE Id_Editora IN (  
    SELECT Id_Editora  
    FROM tbl_editoras  
    WHERE Nome_Editora = 'Wiley' OR Nome_Editora =  
    'Microsoft Press');
```

Operadores IN e NOT IN no SQL Server

Retornar livros e IDs de editoras publicados pelas editoras Wiley ou Microsoft Press:



	Nome_Livro	Id_Editora
▶	Windows Server 2012 Inside Out	3
	Microsoft Exchange Server 2010	3
	Linux Command Line and Shell Scripting	4

Excluir Registros de uma Tabela no SQL com DELETE e TRUNCATE TABLE

Uma das tarefas mais comuns na manutenção de tabelas de bancos de dados é a exclusão de registros (linhas). É muito comum, por exemplo, excluir um produto de uma tabela que não é mais ofertado por uma loja, ou um cliente que se descadastrou de um sistema.

Excluir Registros de uma Tabela no SQL Server com DELETE e TRUNCATE TABLE

Além disso, em algumas situações muito especiais, pode ser necessário excluir todos os registros de uma tabela – ou seja, limpar a tabela.

Podemos excluir registros de uma tabela com SQL por meio de duas declarações: DELETE FROM e TRUNCATE TABLE.

Excluir um ou mais registros especificados – Cláusula DELETE

Sintaxe:

DELETE FROM tabela WHERE coluna = valor;

DELETE from tbl_pessoas
where codigo=5;

TRUNCATE TABLE

- ◎ O comando TRUNCATE TABLE permite remover todas as linhas de uma tabela em uma única operação, sem registrar as exclusões de linhas individuais.
- ◎ O TRUNCATE TABLE equivale a executar a instrução DELETE, porém sem usar a cláusula WHERE. Portanto, é usada para apagar completamente o conteúdo de uma tabela no SQL Server.

Entretanto, a cláusula TRUNCATE TABLE é mais

TRUNCATE TABLE

Vamos excluir todos os registros presentes na tabela *tbl_teste_incremento*:

```
TRUNCATE TABLE tbl_teste_incremento;
```

SQL – Alias com AS – Nomes alternativos para colunas e tabelas

Pode-se dar um nome diferente (e mais amigável) a uma coluna ou tabela ao retornar o resultado de uma consulta, de modo que seja mais fácil ou intuitivo entender os dados retornados. Para isso, usamos a cláusula **AS**.

SQL Server – Alias com AS – Nomes alternativos para colunas e tabelas

Sintaxe :

SELECT coluna

AS alias_coluna

FROM tabela AS alias_tabela;

SQL – Alias com AS – Nomes alternativos para colunas e tabelas

Sintaxe :

```
SELECT Nome_Livro  
AS 'Nome do Livro'  
FROM tbl_Livro;
```

SQL Server – Alias com AS – Nomes alternativos para colunas e tabelas

Podemos também aplicar um alias a uma coluna sem a necessidade de usar a palavra AS, bastando para isso inserir o alias desejado logo após o nome da coluna, sem separação por vírgulas. Veja o exemplo a seguir:

SQL – Alias com AS – Nomes alternativos para colunas e tabelas

```
SELECT Nome_Livro Livro  
FROM tbl_Livro;
```

SQL – Alias com AS – Nomes alternativos para colunas e tabelas

- ◎ Para aplicar alias em mais de uma coluna, basta acrescentá-las normalmente, separando-as por vírgulas, e incluindo os alias logo após o nome de cada coluna.
- ◎ Além disso, podemos criar alias usando palavras compostas, incluindo espaços, bastando para isso envolver o alias entre aspas. O exemplo a seguir mostra as duas possibilidades juntas:

SQL – Alias com AS – Nomes alternativos para colunas e tabelas

```
SELECT Nome_Livro Livro, Preço_Livro 'Preço do  
Livro'  
FROM tbl_Livro;
```