

Implementação de Banco de Dados

Aula 3: Linguagem SQL – Select – Parte 1

Apresentação

Anteriormente, você aprendeu a criar tabelas e a inserir, alterar e deletar linhas. Chegou, inclusive, a dar o comando Select visando recuperar dados sem, entretanto, entender as suas nuances.

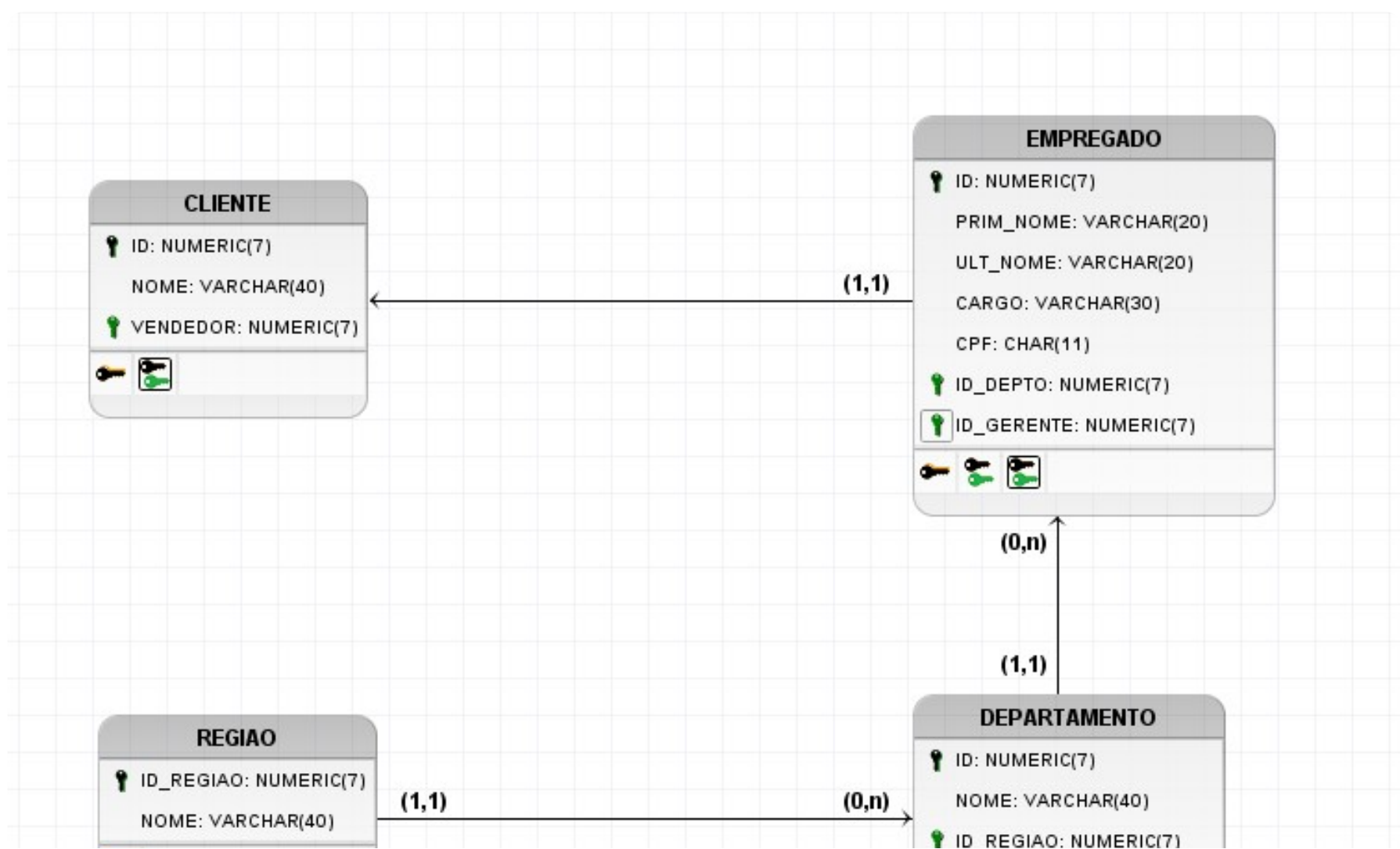
Nesta aula, você começará o estudo desse importante comando, sem dúvida o mais utilizado da linguagem SQL.

Objetivo

- Analisar o comando de Select;
- Consultar dados em tabelas.

Banco de Dados de exemplo

Um banco de dados denominado "Empresa" será utilizado para os exemplos desta aula e das próximas. O Banco da empresa possui o seguinte Modelo Lógico:



As tabelas possuem os seguintes dados:

	id_regiao	nome
	numeric (7)	character varying (40)
1	1	Norte
2	2	Sul

REGIÃO

	id	nome	id_regiao
	numeric (7)	character varying (40)	numeric (7)
1	10	Administrativo	1
2	20	Vendas	1
3	30	Compras	2

DEPARTAMENTO

	id	ult_nome	prim_nome	cargo	salario	dt_admissao	cpf	id_depto	id_gerente
	numeric (7)	character varying (20)	character varying (20)	character varying (30)	numeric (7,2)	date	character (11)	numeric (7)	numeric (7)
1	1	Velasques	Carmen	Presidente	29500.00	2009-05-05	34567890125	10	[null]
2	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1
3	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1
4	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2
5	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3
6	6	Ugarte	Marlene	Vendedor	3500.00	2009-03-03	87654345678	20	3

EMPREGADO

	id	nome	vendedor
	numeric (7)	character varying (40)	numeric (7)
1	110	Ponto Quente	5
2	120	Casa Supimpa	6
3	130	Coisas e Tralhas	5
4	140	Casa Desconto	[null]

CLIENTE

Agora vamos ver como criá-lo.

Criação de tabelas

Clique no botão acima.

Comandos para criação de tabelas:

```
CREATE TABLE REGIAO
(ID_REGIAO NUMERIC(7) PRIMARY KEY,
NOME VARCHAR(40));
```

```
CREATE TABLE DEPARTAMENTO
( ID NUMERIC(7) PRIMARY KEY,
NOME VARCHAR(40) NOT NULL,
ID_REGIAO NUMERIC(7 )REFERENCES REGIAO(ID_REGIAO));
```

```
CREATE TABLE EMPREGADO
( ID NUMERIC(7) PRIMARY KEY,
  ULT_NOME VARCHAR(20) NOT NULL,
  PRIM_NOME VARCHAR(20) NOT NULL,
  CARGO VARCHAR(30),
  SALARIO NUMERIC(7,2),
  DT_ADMISSAO DATE,
  CPF CHAR(11) UNIQUE,
  ID_DEPTO NUMERIC(7) REFERENCES DEPARTAMENTO(ID),
  ID_GERENTE NUMERIC(7) REFERENCES EMPREGADO(ID));
```

```
CREATE TABLE CLIENTE
( ID NUMERIC(7) PRIMARY KEY
NOME VARCHAR(40) NOT NULL,
VENDEDOR NUMERIC(7) REFERENCES EMPREGADO(ID));
```

INSERINDO LINHAS NA TABELAS

```
INSERT INTO REGIAO VALUES (1, 'Norte');
INSERT INTO REGIAO VALUES (2, 'Sul');
```

```
INSERT INTO DEPARTAMENTO VALUES (10, 'Administrativo',1);
INSERT INTO DEPARTAMENTO VALUES (20, 'Vendas',1);
INSERT INTO DEPARTAMENTO VALUES (30, 'Compras',2);
```

```
INSERT INTO EMPREGADO VALUES (1, 'Velasques', 'Carmen', 'Presidente',29500, '05/05/2009','34567890125',10,
null);
INSERT INTO EMPREGADO VALUES (2, 'Neves', 'Lauro', 'Diretor de Compras',19500,
'03/03/2009','23456789012',30,1);
INSERT INTO EMPREGADO VALUES (3, 'Nogueira', 'Ernane','Diretor de Vendas', 18000,
'07/04/2010','34567890123',20,1);
INSERT INTO EMPREGADO VALUES (4, 'Queiroz', 'Mark','Gerente de Compras',8000,
'11/11/2010','12345432123',30,2);
INSERT INTO EMPREGADO VALUES (5, 'Rodrigues', 'Alberto', 'Vendedor',4000, '10/10/2008', '87965432123', 20,3);
INSERT INTO EMPREGADO VALUES (6, 'Ugarte', 'Marlene', 'Vendedor', 3500,'03/03/2009', '87654345678',20,3);
```

```
INSERT INTO CLIENTE VALUES (110, 'Ponto Quente',5);
INSERT INTO CLIENTE VALUES (120, 'Casa Supimpa',6);
INSERT INTO CLIENTE VALUES (130, 'Coisas e Tralhas',5);
INSERT INTO CLIENTE VALUES (140, 'Casa Desconto',null);
```

Se preferir, pode obter os [comandos aqui](#).

Os comandos listados acima mostram como o script disponível funciona normalmente no postgresql e no SqlServer. No ORACLE, deve ser comandado COMMIT após o último insert.

Tendo esse Banco em mente, é altamente recomendável que você execute os comandos de exemplo no PostGreSql.

Foi escolhido como base o PostgreSql, por ser um SGBD mais leve e fácil de instalar, porém, se você puder usar o SqlServer ou o Oracle quando houver diferença entre os SGBD's, será avisado.


Consultando dados de uma tabela

O comando SQL que permite recuperar dados de uma ou mais tabelas é o SELECT. Esse comando nos permite escolher as colunas que retornarão, bem como filtrá-las da tabela.

O comando de Select é uma implementação prática da teoria dos conjuntos, mais especificamente da Álgebra Relacional. Dessa forma, um único Select pode retornar zero ou várias linhas, de acordo com as restrições colocadas no comando.

Os componentes básicos do comando são:

SELECT e FROM

 Clique no botão acima.

Cláusula **SELECT**

- Lista as colunas que serão recuperadas;
- Se utilizarmos o artifício do * (asterisco) na cláusula SELECT, estaremos definindo que todas as colunas serão recuperadas.

Cláusula **FROM**

- Define a tabela que será recuperada.

Veja a sintaxe abaixo:

```
SELECT nome-col1, nome_col2, nome coln
FROM  nome_da_tabela;
```

OU

```
SELECT *
FROM  nome_da_tabela ;
```

Em que:

Palavra-Chave	Descrição
nome_da_tabela	Nome da tabela que contém os dados a serem recuperados.
nome_coln	Nome de uma coluna a ser recuperada.
* (asterisco)	Recupera todas as colunas da tabela.

Retornando uma tabela inteira

Acesse o SGBD e digite o seguinte comando:

SELECT * FROM EMPREGADO.

No comando acima, selecionamos todas as colunas e todas as linhas da tabela EMPREGADO.

Teremos uma resposta semelhante à figura 1.

	id numeric (7)	ult_nome character varying (20)	prim_nome character varying (20)	cargo character varying (30)	salario numeric (7,2)	dt_admissao date	cpf character (11)	id_depto numeric (7)	id_gerente numeric (7)
1	1	Velasques	Carmen	Presidente	29500.00	2009-05-05	34567890125	10	[null]
2	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1
3	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1
4	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2
5	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3
6	6	Ugarte	Marlene	Vendedor	3500.00	2009-03-03	87654345678	20	3

 Figura 1 – Retorno do Comando

Observe que não há nenhuma ordem ou seleção de linhas ou colunas.

Retornando colunas específicas

Na Álgebra Relacional, vimos que existe a operação de projeção que permite retornar apenas algumas colunas da tabela, mas todas as linhas. O mesmo pode ser obtido em SQL. Para isso basta que se liste as colunas desejadas na cláusula SELECT, separando-as por virgulas.

Veja um exemplo:

Acesse o SGBD e digite o seguinte comando:

SELECT ID, PRIM_NOME, ULT_NOME FROM EMPREGADO.

No comando acima, são selecionadas apenas três colunas e todas as linhas da tabela EMPREGADO, e o seu retorno pode ser observado na Fig2.

	id numeric (7)	prim_nome character varying (20)	ult_nome character varying (20)
1	1	Carmen	Velasques
2	2	Lauro	Neves
3	3	Ernane	Nogueira
4	4	Mark	Queiroz
5	5	Alberto	Rodrigues
6	6	Marlene	Ugarte


 Figura 2 – Retorno do Comando

Comentário

Nesse segundo caso, não são exibidas as colunas CARGO, SALARIO, DT_ADMISSAO, CPF,ID_DEPTO e ID_GERENTE.

No primeiro comando analisado, um asterisco substitui a lista de colunas desejadas, indicando que todas as colunas devem ser informadas.

Exemplos:


Como seria o comando para exibir todo o conteúdo da tabela departamento, cujo retorno é exibido na fig3?

	id numeric (7)	nome character varying (40)	id_regiao numeric (7)
1	10	Administrativo	1
2	20	Vendas	1
3	30	Compras	2


 Figura 3 – Retorno do Comando

Dica: tente dar o comando no SGBD antes de ver a SOLUÇÃO.

CONTEÚDO SOLUÇÃO


 empresa on postgres@postgres

1 **SELECT** *
 2 **FROM** DEPARTAMENTO

Data Output
 Explain
 Messages
 Notifications
 Query History

	id numeric (7)	nome character varying (40)	id_regiao numeric (7)
1	10	Administrativo	1
2	20	Vendas	1
3	30	Compras	2


Como seria o comando para exibir todo o NOME e o ID de todos os clientes, cujo retorno é exibido na fig4?

	nome character varying (40)	id numeric (7)
1	Ponto Quente	110
2	Casa Supimpa	120
3	Coisas e Tralhas	130
4	Casa Desconto	140

Figura 4 – Retorno do Comando

Dica: tente dar o comando no SGBD antes de ver a SOLUÇÃO.

CONTEÚDO SOLUÇÃO



empresa on postgres@postgres

1

2

```
SELECT NOME, ID
FROM CLIENTE
```

Data Output

Explain

Messages

Notifications

Q


	nome character varying (40)	id numeric (7)
1	Ponto Quente	110
2	Casa Supimpa	120
3	Coisas e Tralhas	130
4	Casa Desconto	140

Dica

Note que:

- Quando você utiliza *, as colunas retornam na ordem em que foram criadas na tabela;
- Quando você lista as colunas no SELECT, elas retornam na ordem em que as listou;
- As colunas no SELECT devem estar separadas por vírgula;
- Não deve existir vírgula antes da cláusula FROM.

 SELECT

 Clique no botão acima.

Incrementando a consulta

Para efetuar consultas mais complexas e derivar dados a partir das informações contidas nas tabelas, você pode construir expressões na cláusula SELECT.

As expressões podem ser aritméticas ou alfanuméricas, fazendo concatenações por exemplo.

Uma expressão aritmética pode conter os seguintes operadores:

- * multiplicação;
- / divisão;
- + adição;
- subtração.

Para concatenarmos duas colunas, utilizamos o operador || OU + dependendo do SGBD.

Escrevendo expressões aritméticas em comando Select

Em uma expressão, podemos especificar não apenas uma coluna, mas um dado derivado de uma ou mais colunas.


Exemplo

A figura 5 mostra a tabela Empregado. Como ficaria então o comando que listaria o ID, o Ult_NOME, o Salário e o salário anual (consideramos que o salário anual é o salário mensal multiplicado por doze) de todos os empregados?

	id numeric (7)	ult_nome character varying (20)	prim_nome character varying (20)	cargo character varying (30)	salario numeric (7,2)	dt_admissao date	cpf character (11)	id_depto numeric (7)	id_gerente numeric (7)
1	1	Velasques	Carmen	Presidente	29500.00	2009-05-05	34567890125	10	[null]
2	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1
3	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1
4	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2
5	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3
6	6	Ugarte	Marlene	Vendedor	3500.00	2009-03-03	87654345678	20	3

 Figura 5 – TABELA EMPREGADO

O comando pode ser visto na figura 6:



empresa on postgres@postgres

1

2

SELECT ID, ULT_NOME, SALARIO,SALARIO * 12

FROM EMPREGADO

Data Output

Explain

Messages

Notifications

Query History

	id numeric (7)	ult_nome character varying (20)	salario numeric (7,2)	?column? numeric
1	1	Velasques	29500.00	354000.00
2	2	Neves	19500.00	234000.00
3	3	Nogueira	18000.00	216000.00
4	4	Queiroz	8000.00	96000.00
5	5	Rodrigues	4000.00	48000.00
6	6	Ugarte	3500.00	42000.00

Figura 6 – COMANDO E RETORNO

Esse comando funciona da mesma forma no PostGreSql, SqlServer e Oracle.

Escrevendo expressões de concatenação

No PostGreSql, o operador de concatenação é o ||. Se desejássemos retornar o PRIM_NOME do empregado com o ULT_NOME, o comando seria:

```
SELECT PRIM_NOME || ULT_NOME  
  
FROM EMPREGADO
```

O retorno seria o exibido na Fig7.

	?column? text
1	CarmenVelasques
2	LauroNeves
3	ErnaneNogueira
4	MarkQueiroz
5	AlbertoRodrigues
6	MarleneUgarte


 Figura 7 – RETORNO DO COMANDO

Analise a figura 7. Notou que os nomes estão colados?

Isso decorre do fato de que, depois do PRIM_NOME ou antes do ULT_NOME, não existe espaço em branco armazenado na coluna.

Como resolver isso?

Basta concatenar as colunas com um espaço em branco entre elas, conforme mostra a Figura 8.

 empresa on postgres@postgres

1

2

SELECT

PRIM_NOME || ' ' ||

ULT_NOME

FROM

EMPREGADO

Data Output

Explain

Messages

Notifications

Query Hist

	<div>?column?</div> <div>text</div>
1	Carmen Velasques
2	Lauro Neves
3	Ernane Nogueira
4	Mark Queiroz
5	Alberto Rodrigues
6	Marlene Ugarte

 Figura 8 – COMANDO E RETORNO

Mas você pode se perguntar: como fica isso no Oracle e no Sql Server?

No Oracle, o operador é o mesmo (Figura 9).

	<pre>SELECT PRIM_NOME ' ' ULT_NOME FROM EMPREGADO</pre>														
<div> Resultado da Consulta × </div> <div> SQL Todas as Linhas Extraídas: 6 em 0,004 segundos </div>															
	<table> <tr> <th></th><th>PRIM_NOME " " ULT_NOME</th></tr> <tr> <td>1</td><td>Carmen Velasques</td></tr> <tr> <td>2</td><td>Lauro Neves</td></tr> <tr> <td>3</td><td>Ernane Nogueira</td></tr> <tr> <td>4</td><td>Mark Queiroz</td></tr> <tr> <td>5</td><td>Alberto Rodrigues</td></tr> <tr> <td>6</td><td>Marlene Ugarte</td></tr> </table>		PRIM_NOME " " ULT_NOME	1	Carmen Velasques	2	Lauro Neves	3	Ernane Nogueira	4	Mark Queiroz	5	Alberto Rodrigues	6	Marlene Ugarte
	PRIM_NOME " " ULT_NOME														
1	Carmen Velasques														
2	Lauro Neves														
3	Ernane Nogueira														
4	Mark Queiroz														
5	Alberto Rodrigues														
6	Marlene Ugarte														

Figura 9 – COMANDO E RETORNO

Já no SqlServer, o operador de concatenação é o + (Figura 10).

<div> Run Cancel Disconnect Change Connection empresa </div>															
1	SELECT PRIM_NOME + ' ' + ULT_NOME														
2	FROM EMPREGADO														
<div> RESULTS </div> <table> <tr> <th></th><th>(No column name)</th></tr> <tr> <td>1</td><td>Carmen Velasques</td></tr> <tr> <td>2</td><td>Lauro Neves</td></tr> <tr> <td>3</td><td>Ernane Nogueira</td></tr> <tr> <td>4</td><td>Mark Queiroz</td></tr> <tr> <td>5</td><td>Alberto Rodrigues</td></tr> <tr> <td>6</td><td>Marlene Ugarte</td></tr> </table>			(No column name)	1	Carmen Velasques	2	Lauro Neves	3	Ernane Nogueira	4	Mark Queiroz	5	Alberto Rodrigues	6	Marlene Ugarte
	(No column name)														
1	Carmen Velasques														
2	Lauro Neves														
3	Ernane Nogueira														
4	Mark Queiroz														
5	Alberto Rodrigues														
6	Marlene Ugarte														

Figura 10 – COMANDO E RETORNO

Criando Alias

Quando são utilizadas expressões, o cabeçalho da coluna, normalmente, fica sem significado.

Dependendo do SGBD, ele pode ser a própria expressão, como acontece no ORACLE, pode ser (No column name), como acontece no SQLSERVER, ou pode ser? column?

Como acontece no PostgreSQL (Figura 11).

Seja como for, seria mais interessante se fosse possível nomear as colunas de forma a manterem o seu significado. Para isso, existem os alias de coluna.

PostGreSql

	?column? text
1	Carmen Velasques
2	Lauro Neves
3	Ernane Nogueira
4	Mark Queiroz
5	Alberto Rodrigues
6	Marlene Ugarte

Figura 11 – NOME DAS COLUNAS NAS EXPRESSÕES

Para você criar um alias após a coluna, você deve colocar ‘AS’ e em seguida a palavra - sem espaços em branco e sem caracteres em português - que usará para ser o cabeçalho da coluna.

Para chamar a concatenação do PRIM_NOME com o ULT_NOME como NOME_COMPLETO, você deve comandar:

SELECT PRIM_NOME || ' ' || ULT_NOME AS NOME_completo

FROM EMPREGADO

PostGreSql


	nome_completo text
1	Carmen Velasques
2	Lauro Neves
3	Ernane Nogueira
4	Mark Queiroz
5	Alberto Rodrigues
6	Marlene Ugarte

Figura 12 - Retorno dos SGBD

Atenção

1. Repare que no comando o alias está com NOME em maiúsculo e completo em minúsculo. Analise agora o retorno e note que no PostGreSql o alias fica todo em minúsculo, no Oracle todo em maiúsculo e no SqlServer da forma que você digitou o alias.
2. O AS nos 3 SGBD é opcional. Se você escrever o comando sem o AS, ele funciona. Teste para ver.

Colocando espaço em branco no Alias

 Clique no botão acima.

Se você desejar utilizar espaço em branco no Alias, no Oracle e no PostgreSql, então deverá colocar o alias entre aspas duplas (" ").

O comando seria:

```
SELECT PRIM_NOME || ' ' || ULT_NOME "NOME completo"
```

```
FROM EMPREGADO
```

PostGreSql

	NOME completo text
1	Carmen Velasques
2	Lauro Neves
3	Ernane Nogueira
4	Mark Queiroz
5	Alberto Rodrigues
6	Marlene Ugarte

Figura 12 - Retorno dos SGBD

Note que, agora, para os dois o nome da coluna, além de ter o espaço em branco está escrito da forma exata que digitamos o alias.

E no SqlServer?

Neste SGBD, além do alias também ser opcional, ele pode estar entre aspas duplas(“ ”), apóstrofes (‘ ’) ou colchetes ([]). Veja a figura 13.

```
1  SELECT PRIM_NOME + ' ' + ULT_NOME "NOME completo"
2  FROM EMPREGADO
```

RESULTS

	NOME completo
1	Carmen Velasques
2	Lauro Neves
3	Ernane Nogueira
4	Mark Queiroz
5	Alberto Rodrigues
6	Marlene Ugarte

```
1  SELECT PRIM_NOME + ' ' + ULT_NOME 'NOME completo'
2  FROM EMPREGADO
```

RESULTS

	NOME completo
1	Carmen Velasques
2	Lauro Neves
3	Ernane Nogueira
4	Mark Queiroz
5	Alberto Rodrigues
6	Marlene Ugarte

```
1 SELECT PRIM_NOME + ' ' + ULT_NOME [NOME completo]
2 FROM EMPREGADO
```

RESULTS	
	NOME completo
1	Carmen Velasques
2	Lauro Neves
3	Ernane Nogueira
4	Mark Queiroz
5	Alberto Rodrigues
6	Marlene Ugarte

Figura 15 - ALIAS SQLSERVER

Sugestão: tente dar o comando de ALIAS no PòstGreSql ou no ORACLE utilizando apóstrofes ou colchetes e veja o que acontece.

Exemplo 1

Dica


Como seria o comando para mostrar os últimos nomes dos empregados com o cabeçalho Sobrenome, cujo retorno é exibido na figura 14?

	sobrenome character varying (20)
1	Carmen
2	Lauro
3	Ernane
4	Mark
5	Alberto
6	Marlene

 Figura 14 – Retorno do Comando.

Tente dar o comando no SGBD antes de ver a SOLUÇÃO.

CONTEÚDO SOLUÇÃO

 empresa on postgres@postgres

1

2

SELECT

PRIM_NOME

SOBRENOME

FROM

EMPREGADO

Data Output		Explain	Messages	Notifications
	<div>sobrenome</div> <div>character varying (20)</div>			
1	Carmen			
2	Lauro			
3	Ernane			
4	Mark			
5	Alberto			
6	Marlene			

Exemplo 2

Dica

Como seria o comando para mostrar a concatenação do ID do cliente com o seu Nome com o cabeçalho 'Dados dos Clientes'? O Id deverá vir separado do nome por um hífen (-),cujo retorno é exibido na fig.15?

	Dados dos Clientes text
1	110-Ponto Quente
2	120-Casa Supimpa
3	130-Coisas e Tralhas
4	140-Casa Desconto

Figura 15 – Retorno do Comando

Tente dar o comando no SGBD antes de ver a SOLUÇÃO.

CONTEÚDO SOLUÇÃO

1

2

SELECT ID || '-' || NOME AS "Dados dos Clientes"

FROM CLIENTE

Data Output

Explain

Messages

Notifications


Query History

	Dados dos Clientes text
1	110-Ponto Quente
2	120-Casa Supimpa
3	130-Coisas e Tralhas
4	140-Casa Desconto

Utilizando SELECT sem FROM

A cláusula SELECT, além de permitir a realização da projeção das colunas da tabela, pode ser utilizada para exibir resultados de operações aritméticas, retorno de funções ou textos.

Vejamos um exemplo no PostGreSql:



empresa on postgres@postgres

1

2

SELECT 'ALO', 5+9 , NOW()
|

Data Output

Explain

Messages

Notifications

Query History


	?column? unknown	?column? integer	now timestamp with time zone
1	ALO	14	2019-06-17 01:57:43.917509-03

Atenção

Note que:

1. O comando primeiro exibe um texto (ALO), em seguida o resultado de uma operação aritmética (9+5) e finalmente retorna o valor da data da data/hora do sistema (FUNÇÃO NOW());
2. Não existe cláusula FROM, pois desejamos retornar apenas uma linha com os valores.

O que aconteceria se fosse acrescentada uma cláusula FROM com um nome de tabela?

 empresa on postgres@postgres

1

2

SELECT 'ALO', 9 + 5, NOW()
FROM CLIENTE

Data Output

Explain

Messages

Notifications

Query History





	?column? unknown	?column? integer	now timestamp with time zone
1	ALO	14	2019-06-17 04:52:47.991499-03
2	ALO	14	2019-06-17 04:52:47.991499-03
3	ALO	14	2019-06-17 04:52:47.991499-03
4	ALO	14	2019-06-17 04:52:47.991499-03

Note na figura acima que retornaram 4 linhas todas iguais. Isso acontece porque tanto as expressões como a função NOW() são de linha, ou seja, retornam uma linha para cada linha da tabela da cláusula FROM. Como a tabela CLIENTE possui 4 linhas, o retorno tem essa quantidade de linhas.

Por isso é omitida a cláusula FROM, já que desejamos apenas uma linha de retorno.

E o SqlServer, como fica?

Muito Similar. A única diferença é que a função que retorna a data/hora se chama GETDATE().

 Run  Cancel  Disconnect  Change Connection

teste

1

SELECT 'ALO', 5+4, GETDATE()

RESULTS

	(No column name)	(No column name)	(No column name)
1	ALO	9	2019-06-17 02:02:10.327

Como no PostgreSQL, se tiver FROM, retornará uma linha para cada linha da tabela.

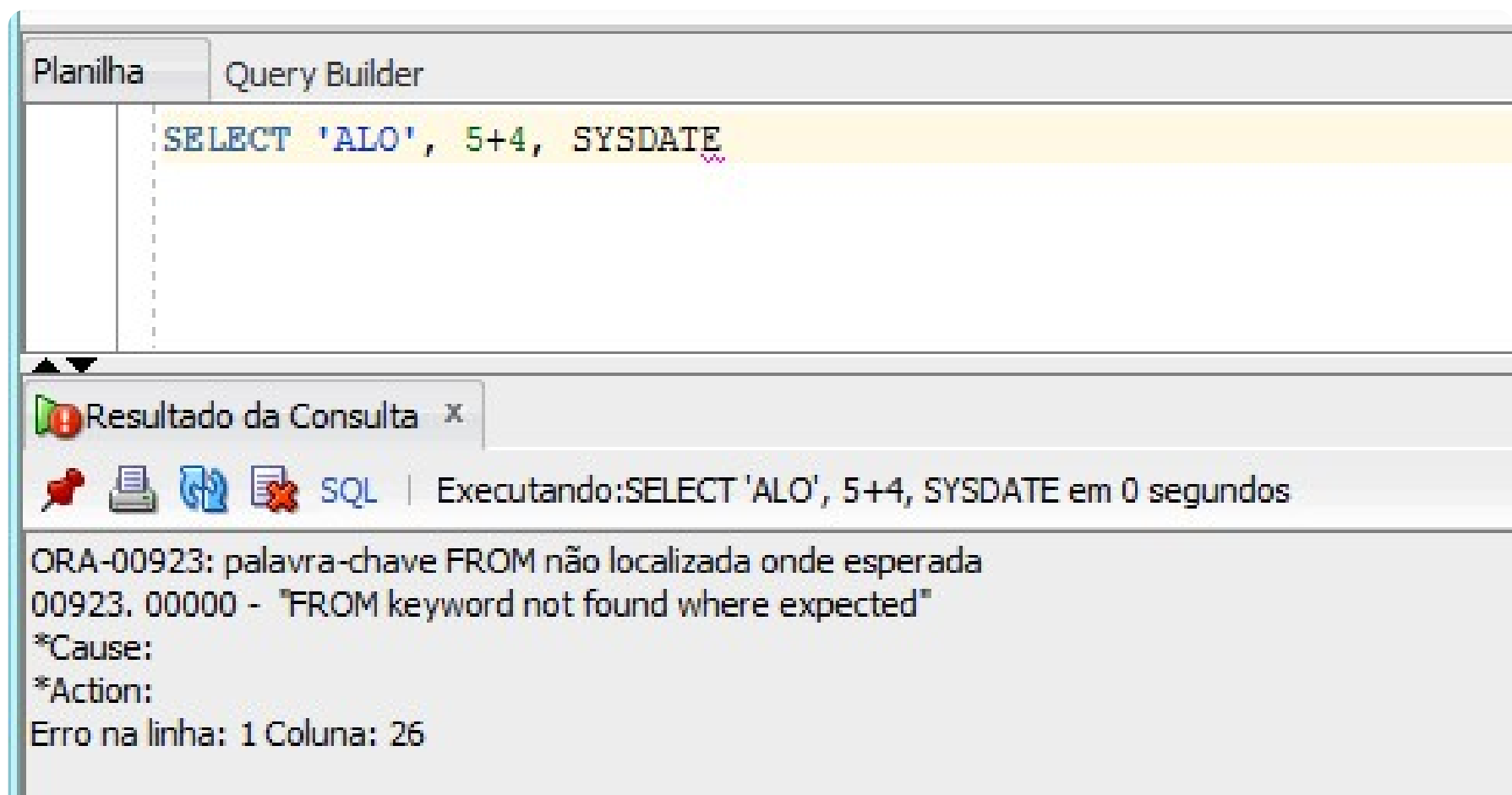

```
1 SELECT 'ALO', 5+4, GETDATE()  
2 FROM REGIAO
```

RESULTS

	(No column name)	(No column name)	(No column name)
1	ALO	9	2019-06-17 02:11:51.197
2	ALO	9	2019-06-17 02:11:51.197

Faltou o Oracle.

A função de data do Oracle é SYSDATE. Veja na figura abaixo o retorno do comando



Note que deu o erro, pois a cláusula From não foi encontrada. Por que isso acontece?

Ao contrário do PostgreSQL e do SqlServer, no Oracle a cláusula FROM é obrigatória. Como fazemos então para conseguir listar os valores? Colocamos a Cláusula FROM com a tabela DUAL, conforme a figura abaixo.

Planilha

Query Builder





SELECT 'ALO', 5+4, SYSDATE

FROM DUAL

▶

Resultado da Consulta

x



SQL

Todas as Linhas Extraídas: 1 em 0

	'ALO'	5+4	SYSDATE
1	ALO	9	17/06/19

Atenção

Observações:

- 1. Sysdate também pode retornar a hora mas temos que fazer algumas configurações que ultrapassam o nosso escopo aqui.
- 2. Como vimos antes no SqlServer e no PostgreSql também podemos colocar a cláusula FROM com um nome de tabela, só que retornaram uma linha para cada linha da tabela, mas como a cláusula FROM não é obrigatória não é um problema.
- 3. No Oracle se colocarmos uma tabela com 4 linhas na FROM retornaram 4 linhas, como a cláusula FROM é obrigatório isto poderia gerar um problema. Para eliminar esta dificuldade existe uma tabela de sistema chamada DUAL que possui uma única linha e que deverá ser colocada na cláusula FROM sempre que se desejar retornar uma única linha com expressões ou funções de linha.

Se por curiosidade você quiser saber o conteúdo de DUAL, basta comandar:

```
SELECT *  
FROM DUAL
```



Resultado da Consulta

X



SQL

|

Todas as Linhas Extraídas: 1 em 0,0



DUMMY

1

X

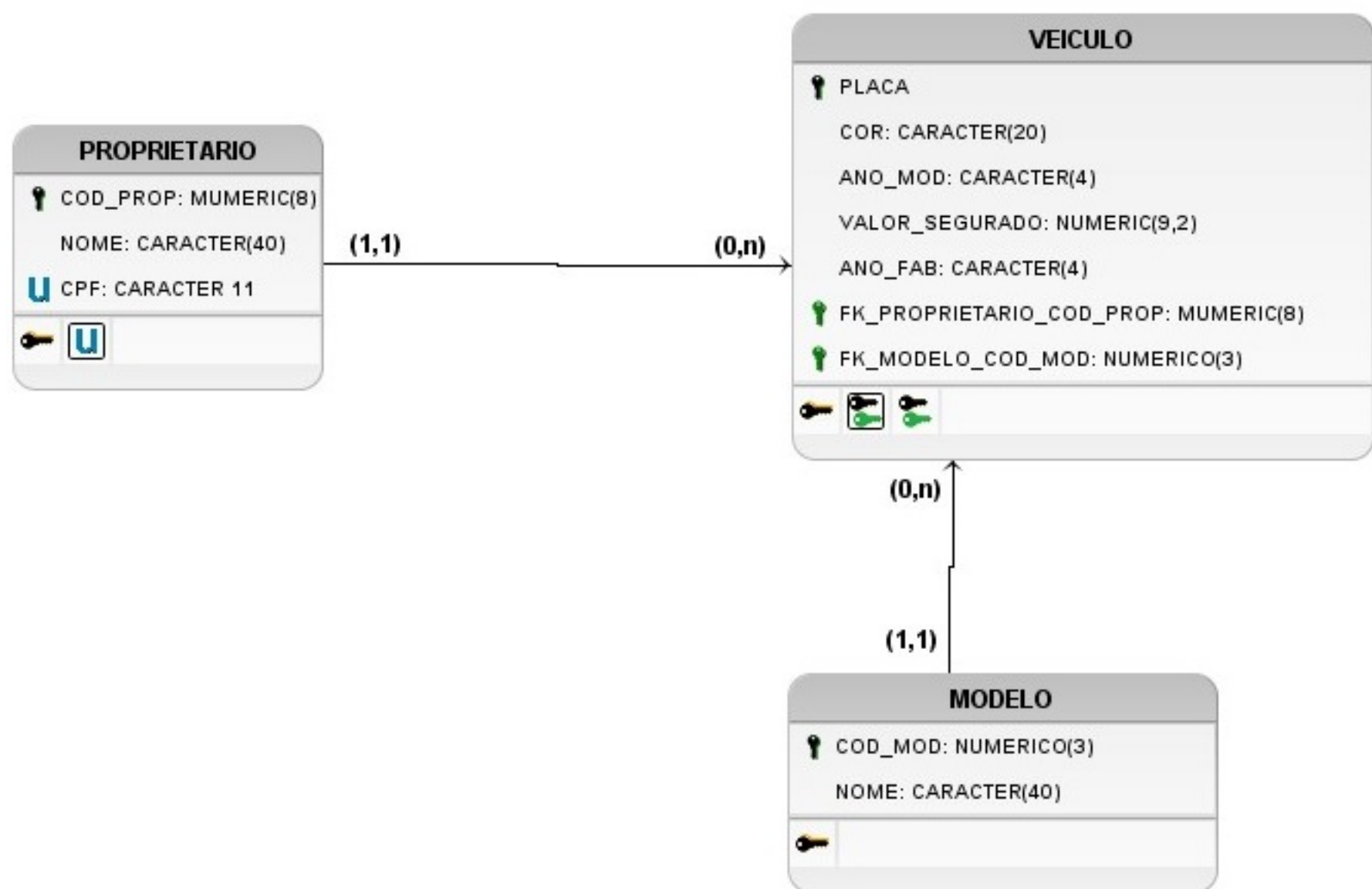
Você verá que a tabela possui uma única coluna chamada DUMMY e uma Linha com o valor X.

Atenção


Importante: DUAL é uma tabela de sistema em que nenhum usuário pode DROPAR ou INSERIR linhas, tampouco ALTERAR ou ELIMINAR sua única linha.

Atividade

1. A partir do Modelo Lógico abaixo, crie as tabelas no PostGreSql em um DataBase chamado "Seguradora".



2. Insira os dados nas tabelas criadas na atividade 1 de forma que as tabelas fiquem conforme as figuras abaixo.

seguradora on postgres@PostgreSQL 11

1

2

SELECT *

FROM PROPRIETARIO

Data Output


Explain

Messages

Notifications

Query History

	cod_prop numeric (8)	nome character varying (110)	cpf character varying (11)
1	10812	DILMA NEVES	51230611266
2	10816	JAQUELINE MEIRELES	[null]
3	10819	IVONE NEVILLE	21233622471
4	10821	MARIANA ROSA	41293524158
5	10823	MARIA CHINALIA	62421381231

 seguradora on postgres@PostgreSQL 11

1

2













SELECT


*

FROM

MODELO

Data Output	Explain	Messages	Notifications	Query History
	cod_mod numeric (3)	nome character varying (80)		
1	101	CLASSIC		
2	102	SANDERO		
3	103	KA		
4	104	PALIO		
5	105	GOL		

        No limit    

 seguradora on postgres@PostgreSQL 11

1

2

SELECT

*

FROM

VEICULO

Data Output								Explain	Messages	Notifications	Query History
	placa character (7)	cor character varying (20)	modelo numeric (3)	proprietario numeric (8)	ano_fab character (4)	ano_mod character (4)	valor_segurado numeric (9,2)				
1	LGA4674	BRANCO	101	10816	2010	2011	22849.40				
2	KTM2693	PRETO	101	10823	2010	2010	20584.68				
3	KTM6449	VERMELHO	102	10823	2018	2018	52584.68				
4	NTA3259	VERDE	103	10819	2013	2014	33000.93				
5	LVA3600	PRETO	104	10823	2015	2015	38979.67				
6	KTM8642	PRETO	105	10812	2009	2009	172584.68				
7	LTU4580	AMARELO	105	10816	2014	2015	35732.11				
8	LVA8755	VERMELHO	104	10812	2011	2011	23452.93				
9	NTA4860	PRETO	103	10819	2016	2016	39017.24				
10	KTM9642	VERMELHO	102	10823	2014	2014	31510.30				

3. Utilizando o banco de dados da Seguradora, emita os comandos abaixo escolhendo as colunas e escrevendo as expressões.

- a. SELECIONE TODOS OS DADOS DE VEÍCULOS;
- b. SELECIONE A PLACA, A COR, O ANO DE FABRICAÇÃO E O ANO DO MODELO DE TODOS OS VEÍCULOS;
- c. EMITA UM COMANDO QUE CONCATENE O NOME DO PROPRIETÁRIO E O SEU CÓDIGO, SEPRANDO-OS COM UM ' - ' E CHAME DA COLUNA DE NOME_COD;
- d. EMITA UM COMANDO QUE RETORNE A PLACA DO VEÍCULO E O SEU VALOR SEGURADO MAJORADO EM 10% E CHAME A COLUNA DE SEGURO MAJORADO;
- e. EMITA UM COMANDO QUE RETORNE A PLACA DO VEÍCULO, O CÓDIGO DO PROPRIETÁRIO, O CÓDIGO DO MODELO E A SOMA DOS DOIS CÓDIGOS.

Notas

Ethernet ¹

Ethernet é uma arquitetura de rede local que entre suas especificações possui o cabeamento UTP.

Backoff ²

Backoff refere-se ao tempo em que um dispositivo de transmissão de dados aguarda para realizar uma nova transmissão após a ocorrência de um problema na primeira tentativa.

Feeds de notícias ³

O que é um feed de notícias?

É um tipo de ferramenta da internet que faz com que não seja preciso correr atrás de determinado site para descobrir quando ele vai atualizar um conteúdo específico. Eles vão até os endereços que você mais gosta e trazem até a sua área de trabalho aquele texto, vídeo ou música tão esperados. Servem para manter você conectado com pessoas, locais e assuntos importantes. As publicações que aparecem primeiro são influenciadas por suas conexões e atividades no Facebook. São atualizados a partir daquilo que a internet entende ser de seu maior interesse.

Fonte: <https://www.tecmundo.com.br/rss/252-o-que-sao-feeds-.htm>

Referências

DATE, C. J. **Introdução a sistemas de banco de dados**. 7. ed. Rio de Janeiro: Campus, 2000.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Addison Wesley, 2015.

Próxima aula

- Comando Select para filtrar linhas.

Explore mais

- [PostgreSQL Prático](#);
- [Postgresql.org](#);
- [Tipos de dados no Postgresql e Sql Server](#).