

Implementação de Banco de Dados

Aula 4: LINGUAGEM SQL – SELECT – PARTE 2

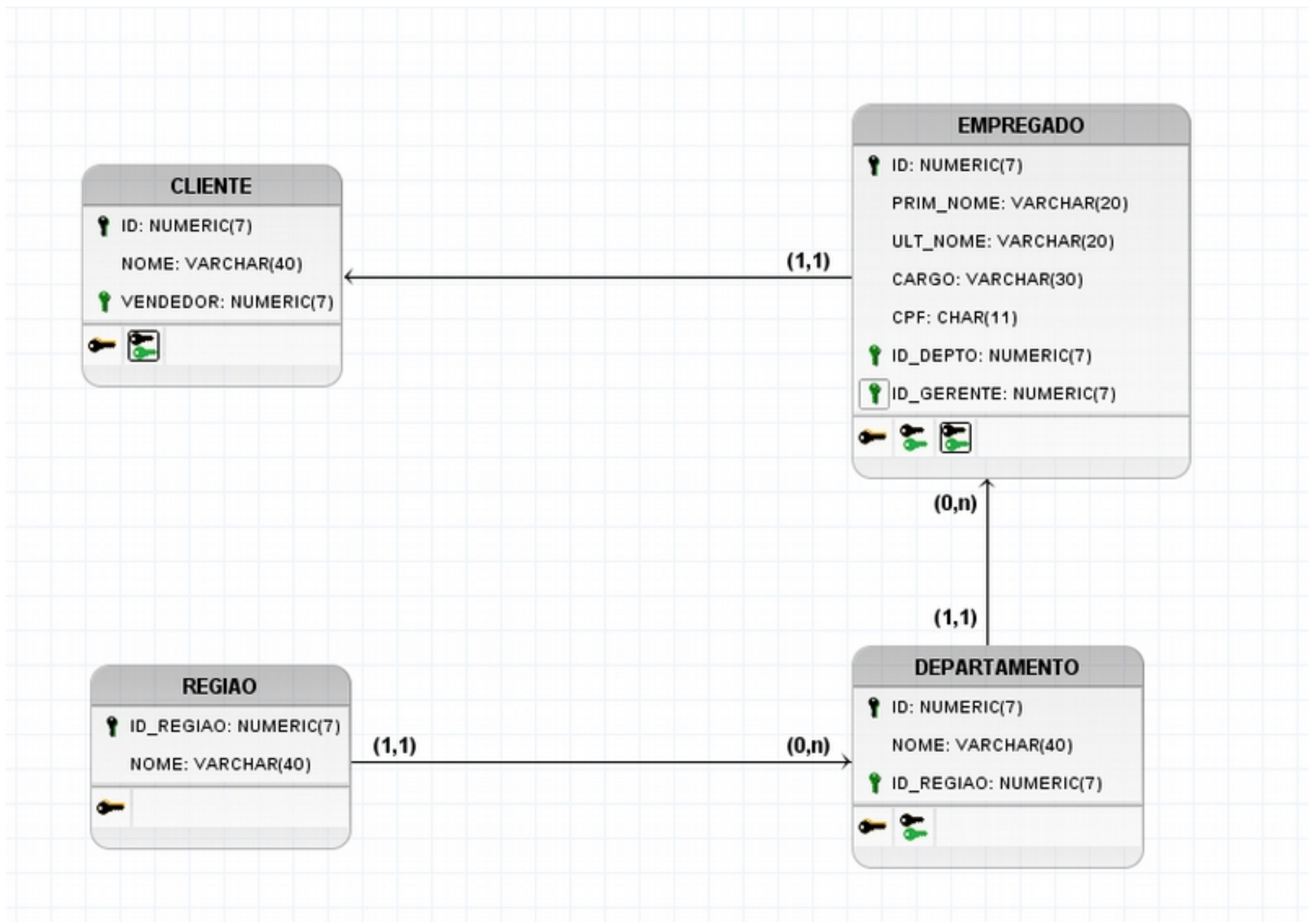
Apresentação

Na última aula, você começou o seu estudo do comando Select vendo a cláusula Select e From. Nesta aula, aprenderá a filtrar as linhas das tabelas utilizando a cláusula Where.

Objetivos

- Analisar a Cláusula Where;
- Formular Consultas.

Nesta aula, continuaremos utilizando o Banco de Dados da Empresa para os exemplos de Modelo Lógico.



As tabelas possuem os seguintes dados:

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

Região

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

Departamento

	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

Empregado

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



Cliente

Comentário

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

Gerando restrições às consultas

Até agora, todos os comandos que demos retornaram todas as linhas da tabela. Mas e se você desejar filtrar as tabelas e retornar apenas as linhas que atendam a uma condição?

Nesse caso, você deve acrescentar a cláusula Where ao comando de Select.

Saiba mais

A cláusula Where estabelece uma condição que a linha deverá obedecer para que faça parte do conjunto resposta da consulta. No caso, apenas retornam as linhas cujo teste da condição dê como resposta verdadeiro.

Ao selecionar os dados para visualização ou outra necessidade, podemos, além de ordená-los, restringir o espectro de visualização utilizando a cláusula Where.

1

2

3

SELECT

*

FROM

EMPREGADO

WHERE

ID

<

3

Data Output

Explain

Messages

Notifications

Query History

	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

Repare que, das seis linhas que a tabela possui “empregado com ID de 1 até 6”, somente retornam as três que possuem ID maior que 3 (4,5 e 6).

Na construção das condições, você pode utilizar os seguintes operadores relacionais:

_____ = _____

igual

_____ <> _____

diferente

_____ < _____

menor que

_____ > _____

maior que

_____ >= _____

maior ou igual a

_____ <= _____

menor ou igual a

Um cuidado que você deve tomar é com o tipo de dado que está utilizando para filtrar. No caso anterior, era um dado numérico (ID) e bastava escrevê-lo. Mas e se fosse um texto?

Veja os dados da tabela CLIENTE.

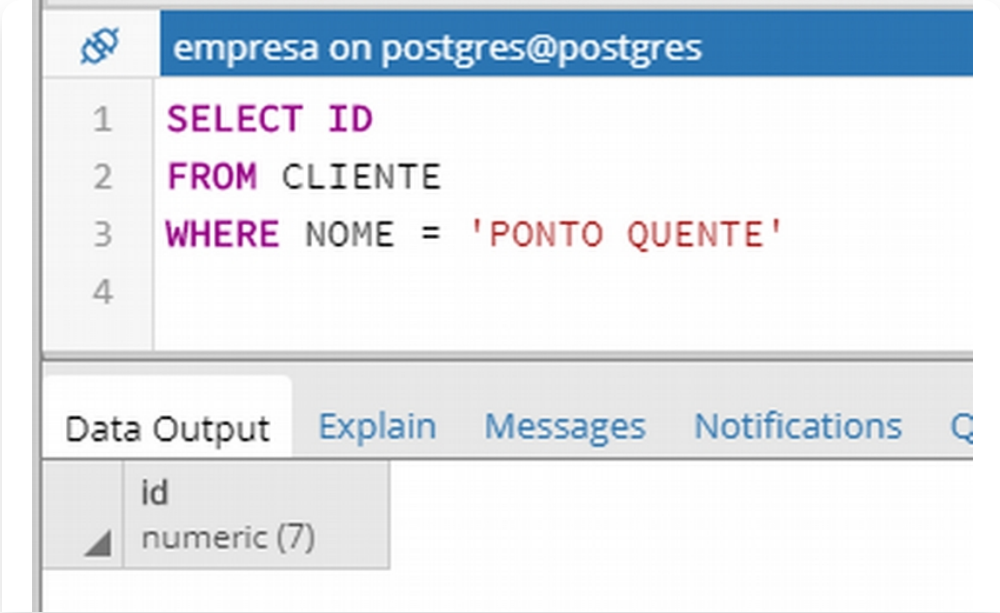
	Id numeric (7)	nome character varying (40)	vendedor numeric (7)
1	110	Ponto Quente	5
2	120	Casa Supimpa	6
3	130	Coisas e Tralhas	5
4	140	Casa Desconto	[null]

 Cliente

Você deseja o ID do cliente Ponto Quente, cujo valor é 110. O Comando seria então:

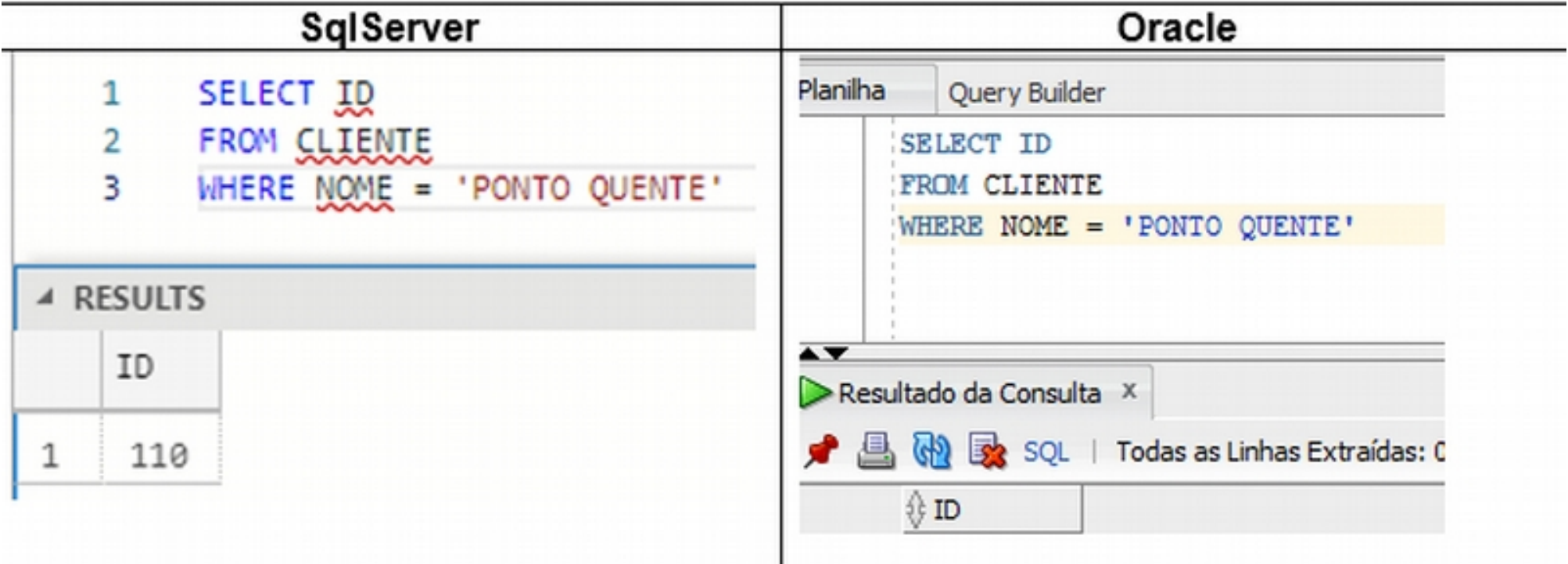
- SELECT ID
- FROM CLIENTE
- WHERE NOME = 'PONTO QUENTE'

Veja o retorno.



 Retorno do Comando.

Por que não voltou o ID do cliente, já que existe esse nome na tabela? Veja o mesmo comando no Oracle e no SqlServer.



Retorno do Comando

Repare que, no SqlServer, retornou o valor 110 e no Oracle, não. O que está acontecendo?

Alguns SGBD são `case sensitive` para os dados, ou seja, fazem diferenciação entre letra maiúscula e letra minúscula. . Dessa forma, temos que respeitar isso, ou a consulta poderá não retornar as linhas.

No caso do nosso banco de dados, o nome do Cliente está com a primeira letra de cada palavra em maiúsculo, portanto, temos que escrever dessa forma no comando.

- SELECT ID
- FROM CLIENTE
- WHERE NOME = 'Ponto Quente'

PostGreSql

1

SELECT ID

2

FROM CLIENTE

3

WHERE NOME = 'Ponto Quente'

4

5

Data Output

Explain

Messages

Notifications

	id	
	numeric (7)	
1		110

Oracle

SELECT ID

FROM CLIENTE

WHERE NOME = 'Ponto Quente'

Resultado da Consulta

SQL

Todas as Linhas Extraídas: 1 em

	ID	
1		110

 Retorno do Comando.

Você pode estar pensando:

Significa, então, que eu tenho que saber como está escrito no Banco de Dados?

E se eu não souber?

E se em uma linha estiver tudo maiúsculo e na outra, tudo minúsculo? E se o banco não tiver um padrão?

Neste caso, você deve padronizar a consulta utilizando uma função que leve o valor existente no banco de dados para maiúsculo ou para minúsculo antes de fazer a comparação.

Veja os exemplos a seguir.

PostGreSql

```
1 SELECT ID
2 FROM CLIENTE
3 WHERE UPPER(NOME) = 'PONTO QUENTE'
4
5
```

Data Output

Explain

Messages

Notifications

Qu

	id
	numeric (7)
1	110

Oracle

```
SELECT ID
FROM CLIENTE
WHERE UPPER(NOME) = 'PONTO QUENTE'
```

Resultado da Consulta x

SQL

 | Todas as Linhas Extraídas: 1 em 0,005 seg

	ID
1	110

Retorno do Comando

Repare que:

- Nos comandos PONTO QUENTE, está em maiúsculo e foi utilizada a função UPPER para levar o conteúdo da coluna NOME para maiúsculo antes da comparação;
- UPPER atua apenas na comparação, não altera o valor do existente no banco de dados.

Veja o exemplo abaixo, onde, ao pedirmos para retornar também o nome, ele vem como está no banco de dados.

```
1 SELECT ID, NOME
2 FROM CLIENTE
3 WHERE UPPER(NOME) = 'PONTO QUENTE'
4
5
```

Data Output

Explain

Messages

Notifications

Qu

	id	nome
	numeric (7)	character varying (40)
1	110	Ponto Quente

 Retorno do Comando

Dois cuidados que você deve tomar ao trabalhar com *string*:

- A *string* deve vir entre apóstrofes ‘PONTO QUENTE’;
- Se o SGBD for case sensitive, você deve escrever o comando como os dados que estão no banco ou utilizar uma função para padronizar a forma de comparação.

Trabalhando com Datas

Ao trabalhar com datas, devemos colocá-las entre aspas simples, no formato dd/mm/aaaa, onde “dd” é o dia em dois dígitos, “mm” o mês em dois dígitos e “aaaa” é o ano em quatro dígitos.

Exemplo

Mostrar sobrenome e senha dos empregados admitidos em 3/3/2009.

- SELECT ULT_NOME, DT_ADMISSAO
- FROM EMPREGADO
- WHERE DT_ADMISSAO = '3/3/2009';

PostGreSql

1

SELECT

ULT_NOME, DT_ADMISSAO

2

FROM

EMPREGADO

3

WHERE

DT_ADMISSAO = '3/3/2009'

4

5

Data Output

Explain

Messages

Notification

ult_nome

character varying (20)

dt_admissao

date

1

Neves

2009-03-03

2

Ugarte

2009-03-03

SqlServer

1

SELECT

ULT_NOME, DT_ADMISSAO

2

FROM

EMPREGADO

3

WHERE

DT_ADMISSAO = '3/3/2009'

RESULTS

ULT_NOME

DT_ADMISSAO

1

Neves

2009-03-03

2

Ugarte

2009-03-03

Oracle

SELECT

ULT_NOME, DT_ADMISSAO

FROM

EMPREGADO

WHERE

DT_ADMISSAO = '3/3/2009'

Resultado da Consulta

SQL

Todas as Linhas Extra

ULT_NOME

DT_ADMISSAO

1

Neves

03/03/09

2

Ugarte

03/03/09

Retorno do Comando

Teste também os seguintes comandos:

SELECT ULT_NOME, DT_ADMISSAO

FROM EMPREGADO

WHERE DT_ADMISSAO = '3/MAR/2009'

ou

SELECT ULT_NOME, DT_ADMISSAO

FROM EMPREGADO

WHERE DT_ADMISSAO = '3/MARÇO/2009'

SELECT ULT_NOME, DT_ADMISSAO

FROM EMPREGADO

WHERE DT_ADMISSAO = '3/MAR/09'

Saiba mais

Estes formatos também são aceitos, mas, no PostgreSql, deve-se ter um cuidado: o nome/abreviatura dos meses devem ser em inglês:

Março - March

Abr - Apr

Já o Oracle e o SqlServer dão suporte aos nomes em português.

Consultando dados com várias condições

Você pode especificar critérios complexos combinando várias condições de pesquisa.

A utilização dos operadores lógicos AND e OR permite montar expressões lógicas para filtrar as linhas. Como toda expressão lógica, o operador AND somente retorna Verdadeiro (TRUE) se ambas as condições forem verdadeiras, enquanto o operador OR somente retorna FALSO (FALSE) se as duas condições forem falsas.

Veja a tabela EMPREGADO.

	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

Vamos supor que você deseja mostrar os empregados que tenham sido contratados após primeiro de janeiro de 2010 e que tenham salário maior que R\$10.000.

O comando e o resultado estão na figura a seguir.

empresa on postgres@postgres

1

2

3

4

5

SELECT *

FROM EMPREGADO

WHERE DT_ADMISSAO > '1/1/2010' AND SALARIO > 10000

Data Output

Explain

Messages

Notifications

Query History

	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	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1

Comando e Retorno

Repare que a utilização do AND obriga que as duas condições sejam verdadeiras para que a linha retorne. Se tivéssemos utilizado OR, bastaria uma ser verdadeira para que a linha retornasse.

empresa on postgres@postgres

1

2

3

4

5

SELECT *

FROM EMPREGADO

WHERE DT_ADMISSAO > '1/1/2010' OR SALARIO > 10000

Data Output

Explain

Messages

Notifications

Query History

	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

Comando e Retorno

Atividade

1. Utilizando o nosso banco de dados de exemplo para fazer alguns exercícios.

Mostrar primeiro nome e sobrenome dos empregados lotados no departamento, cuja identificação é 20. O retorno esperado é exibido na figura.

	prim_nome character varying (20)	ult_nome character varying (20)
1	Ernane	Nogueira
2	Alberto	Rodrigues
3	Marlene	Ugarte

Retorno do comando

2. Utilizando o nosso banco de dados de exemplo para fazer alguns exercícios.

Mostrar sobrenome e cargo dos empregados admitidos após 3/3/2009. O retorno esperado é exibido na figura.

	ult_nome character varying (20)	cargo character varying (30)
1	Velasques	Presidente
2	Nogueira	Diretor de Vendas
3	Queiroz	Gerente de Compras

Retorno do comando

3. Utilizando o nosso banco de dados de exemplo para fazer alguns exercícios.

Mostrar sobrenome, cargo e salário dos empregados que não sejam Vendedores. O retorno esperado é exibido na figura.

	ult_nome character varying (20)	cargo character varying (30)	salario numeric (7,2)
1	Velasques	Presidente	29500.00
2	Neves	Diretor de Compras	19500.00
3	Nogueira	Diretor de Vendas	18000.00
4	Queiroz	Gerente de Compras	8000.00

Retorno do comando

4. Utilizando o nosso banco de dados de exemplo para fazer alguns exercícios.

Mostrar sobrenome, cargo e salário dos empregados que ganham menos que R\$18.000 e que não sejam vendedores. O retorno esperado é exibido na figura.

	ult_nome character varying (20)	cargo character varying (30)	salario numeric (7,2)
1	Queiroz	Gerente de Compras	8000.00

Retorno do comando

5. Utilizando o nosso banco de dados de exemplo para fazer alguns exercícios.

Mostrar sobrenome, cargo e salário dos empregados que são vendedores ou que ganham pelo menos R\$18.000. O retorno esperado é exibido na figura.

	ult_nome character varying (20)	cargo character varying (30)	salario numeric (7,2)
1	Velasques	Presidente	29500.00
2	Neves	Diretor de Compras	19500.00
3	Nogueira	Diretor de Vendas	18000.00
4	Rodrigues	Vendedor	4000.00
5	Ugarte	Vendedor	3500.00

Retorno do comando

6. Utilizando o nosso banco de dados de exemplo para fazer alguns exercícios.

Altere o comando:

SELECT ULT_NOME, CARGO, SALARIO

FROM EMPREGADO

WHERE UPPER(CARGO) = 'VENDEDOR' OR SALARIO >= 18000

Para mostrar sobrenome, cargo e salário dos empregados que ganham menos que R\$18.000 e que não sejam vendedores. Essa alteração deverá ser feita pelo acréscimo do operador lógico NOT e de parênteses onde for o caso. O retorno esperado é exibido na figura.

	ult_nome character varying (20)	cargo character varying (30)	salario numeric (7,2)
1	Queiroz	Gerente de Compras	8000.00

Retorno do comando

7. Utilizando o nosso banco de dados de exemplo para fazer alguns exercícios.

Mostrar todos os dados dos empregados contratados que trabalham no departamento 10 ou no departamento 30 (resolver utilizando operadores lógicos). O retorno esperado é exibido na figura.

	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	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2

Retorno do comando

Operadores da Linguagem SQL

A linguagem SQL possui um conjunto próprio de operadores para testar condições.

Esses operadores são:

IN CONTIDO EM (LISTA)

BETWEEN ENTRE VALORES

LIKE STRING SEMELHANTE

IS NULL Testa valores nulos

Vamos agora estudá-los mais detalhadamente.

Operador IN

O operador IN permite comparar o valor da coluna com uma lista de valores e retorna verdadeiro se, em uma determinada linha, o valor da coluna for igual a um dos valores da lista.

Exemplo

Desejamos listar os empregados que trabalham no departamento 10 ou no departamento 30. Como ficaria o comando?

```
SELECT *  
FROM EMPREGADO  
WHERE ID_DEPTO IN (10,30)  
Observe o retorno na figura.
```

	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	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2

Agora, para ver os empregados que NÃO trabalham nestes departamentos, basta acrescentar NOT ao comando:

```
SELECT *  
  
FROM EMPREGADO  
  
WHERE ID_DEPTO NOT IN (10,30)  
  
Observe o retorno na figura.
```

	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	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1
2	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3
3	6	Ugarte	Marlene	Vendedor	3500.00	2009-03-03	87654345678	20	3

Retorno do comando

8. Utilizando o nosso banco de dados de exemplo para fazer alguns exercícios.

Mostrar sobrenome, cargo e código do departamento dos empregados que NÃO são Gerentes de Vendas ou Vendedores (resolver utilizando IN). O retorno esperado é exibido na figura.

	ult_nome character varying (20)	cargo character varying (30)	id_depto numeric (7)
1	Velasques	Presidente	10
2	Neves	Diretor de Compras	30
3	Nogueira	Diretor de Vendas	20
4	Queiroz	Gerente de Compras	30

Retorno do comando

Operador Between

O operador Between And permite verificar se o valor de um campo está contido em uma faixa de valores. Por exemplo, desejamos retornar o id ult_nome e cargo dos empregados com salários entre R\$8.000 e R\$19.500 inclusive.

O comando seria:

SELECT *

FROM EMPREGADO

WHERE SALARIO BETWEEN 8000 AND 19500

Observe o retorno na figura.

	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	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1
2	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1

 Retorno do comando.

Repare que retornam os empregados com salários de R\$8.000 e de R\$19.500. Isso mostra que o Between cria um intervalo fechado, ou seja, os limites fazem parte dos valores aceitáveis.

Para fazer a condição inversa, basta utilizarmos Not Between.

SELECT *

FROM EMPREGADO

WHERE SALARIO BETWEEN 8000 AND 19500

Observe o retorno na figura.

	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	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3
3	6	Ugarte	Marlene	Vendedor	3500.00	2009-03-03	87654345678	20	3

Retorno do comando

Podemos utilizar Between com Datas.

Por exemplo, desejamos os empregados contratados em 2009.

O comando seria:

SELECT *

FROM EMPREGADO

WHERE DT_ADMISSAO BETWEEN '1/1/2009' AND '31/12/2009'

Observe o retorno na figura.

	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	6	Ugarte	Marlene	Vendedor	3500.00	2009-03-03	87654345678	20	3

Retorno do comando

Atividade

9. Mostrar Ult_nome, data de admissão e salário dos empregados contratados no ano de 2010 (resolver utilizando Between). O retorno esperado é exibido na figura.

	ult_nome character varying (20)	dt_admissao date	salario numeric (7,2)
1	Nogueira	2010-04-07	18000.00
2	Queiroz	2010-11-11	8000.00

Retorno do comando

10. Mostrar Ult_nome, data de admissão e salário dos empregados que ganham menos que R\$8.000 ou mais que R\$18.000 (resolver utilizando Between). O retorno esperado é exibido na figura.

	ult_nome character varying (20)	dt_admissao date	salario numeric (7,2)
1	Velasques	2009-05-05	29500.00
2	Neves	2009-03-03	19500.00
3	Rodrigues	2008-10-10	4000.00
4	Ugarte	2009-03-03	3500.00

Retorno do comando

Operador Like

O operador Like é utilizado para fazer casamento de padrão, ou seja, procurar um conjunto de caracteres que existe em uma string.

Esta operação de comparação, para poder ser eficiente, necessita do uso de caracteres coringa, que no caso do SQL são dois:

% - Curinga para representar uma quantidade arbitrária de caracteres (inclusive nenhum);

_ - Curinga para indicar a existência obrigatória de um caracter naquela posição.

Exemplo

Desejamos saber os dados dos empregados com o ult_nome começado com N.

O comando seria:

```
SELECT *  
FROM EMPREGADO  
WHERE UPPER(ULT_NOME) LIKE 'N%'
```

Observe o retorno na figura.

	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	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1
2	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1

 Retorno do comando.

Observações

- **Repare o uso da função UPPER.** Esta função leva a string para maiúsculo antes de fazer a comparação, sendo muito útil para contornar a limitação do Oracle e do Postgresql, por serem Case Sensitive. Uma outra função possível de ser utilizada é lower, que leva o texto todo para minúsculo.
- **No SqlServer, você deve eliminar UPPER,** já que ele não é Case Sensitive.
- **Note o % depois do N** informando que, após essa letra, pode existir uma quantidade arbitrária de caracteres.

Outro exemplo.

Exemplo

Desejamos saber os dados dos empregados com o ult_nome terminado com S.

O comando seria:

```
SELECT *
FROM EMPREGADO
WHERE UPPER(ULT_NOME) LIKE '%S'
```

Observe o retorno na figura.


	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	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3

Retorno do comando

E se você quiser retornar os PRIMEIROS NOMES dos empregados que possuem E em qualquer lugar do nome? Como seria o comando? O retorno esperado é exibido na figura.

	prim_nome character varying (20)
1	Carmen
2	Ernane
3	Alberto
4	Marlene

Retorno do comando.

 empresa on postgres@postgres

1

2

3

4

5

SELECT

PRIM_NOME

FROM

EMPREGADO

WHERE

UPPER

(

PRIM_NOME

)

LIKE

'

%E%

'

Data Output

Explain

Messages

Notifications

Queries

	prim_nome character varying (20)
1	Carmen
2	Ernane
3	Alberto
4	Marlene

Solução.

Repare no retorno. Temos Carmen com E no meio, mas temos também Ernane que possui E no início e no final, pois o % autoriza a ter uma quantidade arbitrária de caracteres, inclusive nenhum.

Se você desejasse retornar os sobrenomes que possuem E como segunda letra, não adiantaria utilizar %. Para isso temos que

informar que OBRIGATORIAMENTE deve existir uma letra antes do E, utilizando nosso outro curinga, o ‘_’.

O comando seria então:

```
SELECT ULT_NOME

FROM EMPREGADO

WHERE UPPER(ULT_NOME) LIKE '_E%'
```

Observe o retorno na figura.

	ult_nome character varying (20)
1	Velasques
2	Neves

Retorno do comando.

Note que temos que ter um _ no antes do E para informar que o E é a segunda letra e que, após ela, podemos ter uma quantidade qualquer de caracteres.

Alguns cuidados:

O ‘_’ deve estar colado no ‘E’. Não pode haver espaço entre eles;

Você deve colocar um ‘_’ para cada caractere. Por exemplo, para L na terceira, o comando seria o da figura abaixo.

1	SELECT ULT_NOME
2	FROM EMPREGADO
3	WHERE UPPER(ULT_NOME) LIKE '__L%'
4	
5	

Data Output		Explain	Messages	Notifications	Qu
	ult_nome character varying (20)				
1	Velasques				

Comando e retorno.

Agora, quando você deseja quem não tem R no nome, deve acrescentar o NOT antes do LIKE. Veja a figura.

empresa on postgres@postgres	
1	SELECT ULT_NOME
2	FROM EMPREGADO
3	WHERE UPPER(ULT_NOME) NOT LIKE '%R%'
4	
5	
Data Output Explain Messages Notifications Query	
	ult_nome character varying (20)
1	Velasques
2	Neves

Comando e retorno.

ILIKE

Devido ao fato do PostgreSQL ser Case Sensitive, nos comandos utilizamos o UPPER, porém este SGBD possui um operador de LIKE proprietário que permite que façamos o teste sem o uso da função UPPER. É o ILIKE, ou seja, Insensitive-LIKE.

Quando o utilizamos, não precisamos nos preocupar com maiúsculas ou minúsculas. Veja a figura.

empresa on postgres@postgres	
1	SELECT ULT_NOME
2	FROM EMPREGADO
3	WHERE ULT_NOME ILIKE '_E%'
4	
5	
Data Output Explain Messages Notification	
	ult_nome character varying (20)
1	Velasques
2	Neves

Comando e retorno.

Atividade

11. Mostrar primeiro nome e sobrenome dos empregados cujo nome comece pela letra L. O retorno esperado é exibido na figura.

	prim_nome character varying (20)	ult_nome character varying (20)
1	Lauro	Neves

Retorno do comando

12.Mostrar todos os dados dos clientes cujo nome possua a palavra Casa. O retorno esperado é exibido na figura.

	id numeric (7)	nome character varying (40)	vendedor numeric (7)
1	120	Casa Supimpa	6
2	140	Casa Desconto	[null]

Retorno do comando

13.Mostrar todos os dados dos clientes que possuem pelo menos 13 letras no nome. O retorno esperado é exibido na figura.

	id numeric (7)	nome character varying (40)	vendedor numeric (7)
1	130	Coisas e Tralhas	5
2	140	Casa Desconto	[null]

Retorno do comando

14.Mostrar todos os dados dos clientes que possuem 12 letras ou menos no nome. O retorno esperado é exibido na figura.

	id numeric (7)	nome character varying (40)	vendedor numeric (7)
1	110	Ponto Quente	5
2	120	Casa Supimpa	6

Retorno do comando

Operador Is Null

O operador Is Null visa determinar se, no campo, existe valor ou não (o campo é nulo).

Um valor nulo é um valor que está indisponível, não foi atribuído, é desconhecido ou inaplicável, tornando inviável usar '=' no teste. Como nulo não é valor, mas sim ausência de valor, ele não pode ser igual ou diferente de qualquer outro valor. Considere a tabela CLIENTE.

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

Se você desejasse retornar todos os dados dos CLIENTES que não são atendidos por um vendedor, o comando seria:


```
SELECT *  
  
FROM CLIENTE  
  
WHERE VENDEDOR IS NULL
```

Observe o retorno na figura.

	id numeric (7)	nome character varying (40)	vendedor numeric (7)
1	140	Casa Desconto	[null]

Retorno do comando.

Caso contrário, se você quisesse os que são atendidos por vendedor, utilizaria IS NOT NULL. Veja o comando e o retorno na figura.

 empresa on postgres@postgres

1

2

3

4

5

SELECT

*

FROM

CLIENTE

WHERE

VENDEDOR

IS

NOT

NULL

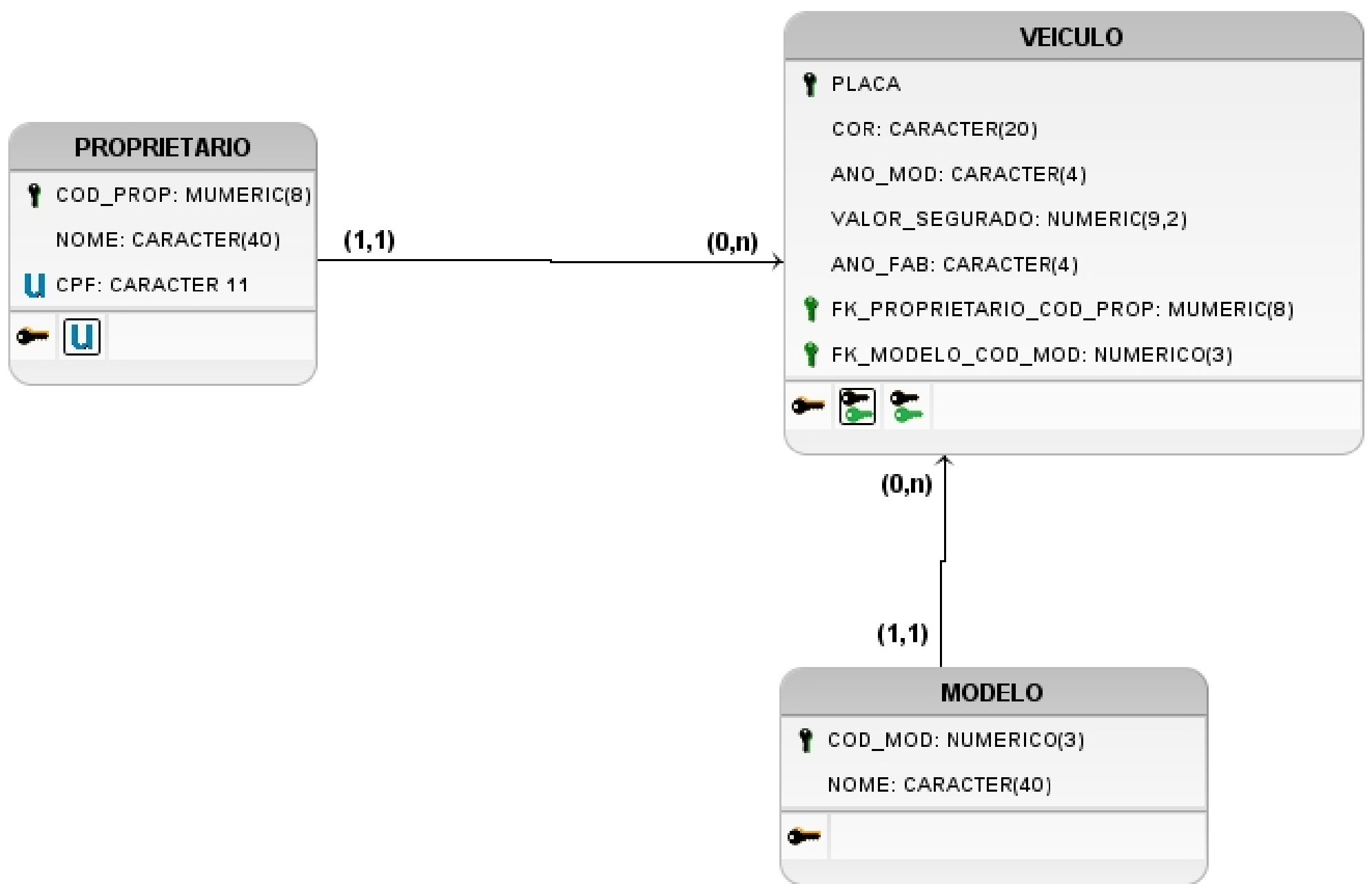
Data Output	Explain	Messages	Notifications	Query History
-------------	---------	----------	---------------	---------------

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

Comando e retorno.

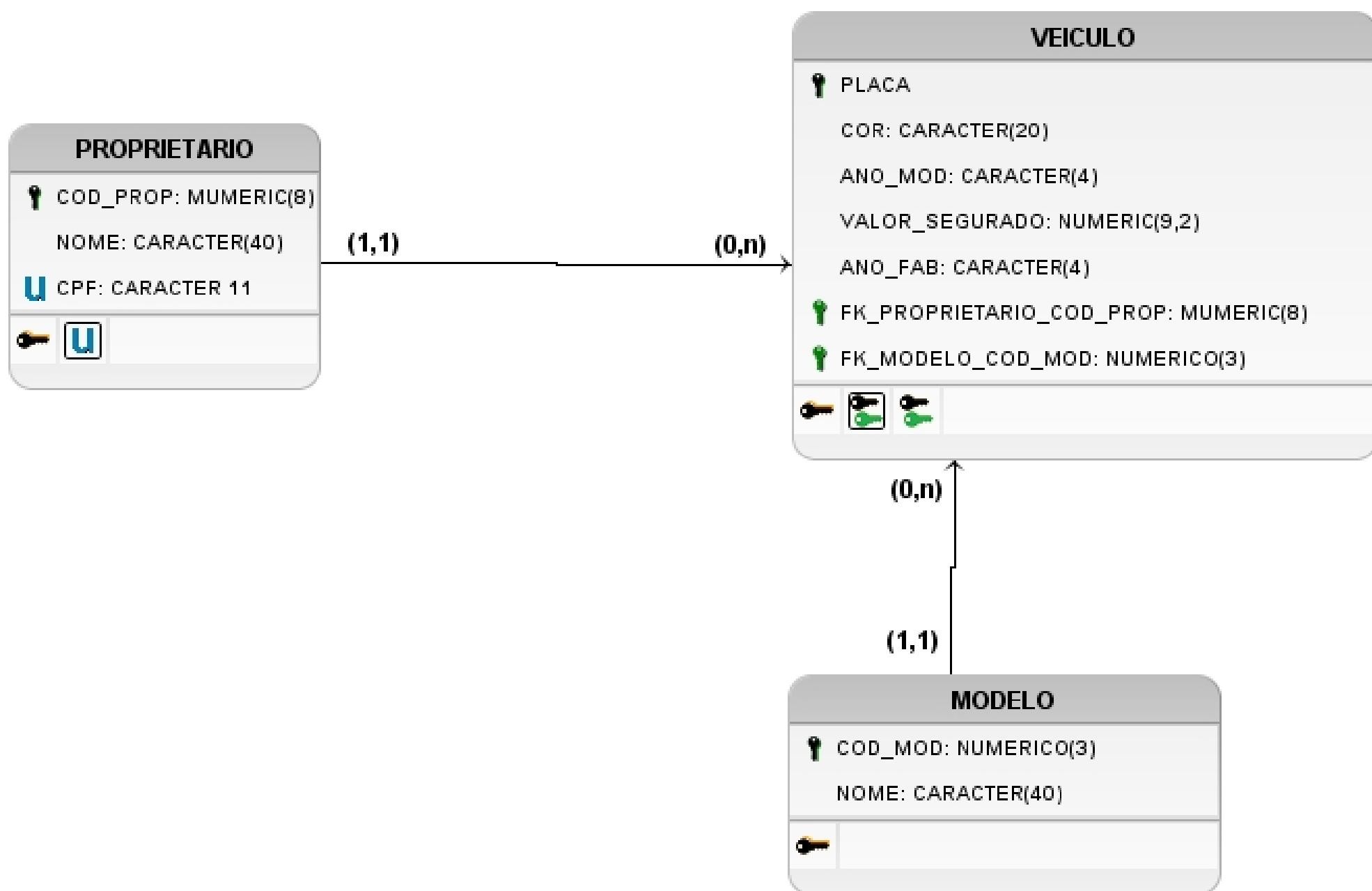
Atividade

15.Utilizando o banco de dados da seguradora, emita os comandos abaixo utilizando operadores relacionais e lógicos.



- Selecione todos os dados dos proprietários que têm código maior que 10816.
- Selecione todos os dados dos modelos cujos nomes vêm alfabeticamente depois de 'ka'.
- Selecione todos os dados dos veículos cujo ano de fabricação é diferente do ano do modelo.
- Selecione todos os dados dos veículos cujo valor segurado seja menor de R\$30.000.
- Selecione todos os dados dos veículos do modelo 104.
- Selecione a placa dos veículos do proprietário 10.823.
- Selecione a placa, o proprietário e a cor dos veículos pretos ou do proprietário 10.812.
- Selecione a placa, a cor e o modelo dos veículos pretos do modelo 105.

16. Utilizando o banco de dados da seguradora, emita os comandos abaixo utilizando operadores do SQL.



- Selecione o código do proprietário que não possui CPF cadastrado.
- Selecione todos os dados dos modelos cujo nome inicie por 'P'.
- Selecione todos os dados dos veículos cujo ano de fabricação esteja entre 2010 e 2013.
- Selecione todos os dados dos veículos que não são dos modelos 102 ou 105.
- Selecione todos os dados dos modelos cujo nome não esteja entre Gol e Palio inclusive.
- Selecione a placa e o código dos proprietários dos veículos pertencentes aos proprietários 10.823 ou 10.812.
- Selecione todos os dados dos proprietários cujo CPF não tenha a sequência '33'.
- Selecione todos os dados dos modelos cuja segunda letra seja 'A'.

Notas

Case sensitive

Você deve considerar que o SqlServer, a princípio, não faz distinção entre maiúsculas e minúsculas, enquanto o Oracle e o PostGreSql fazem. Na realidade, o SqlServer pode ou não fazer. É uma configuração que o DBA faz no servidor, mas o padrão é não fazer tal distinção.

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.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistemas de banco de dados**. 5. ed. Rio de Janeiro: Campus, 2006.

Próxima aula

- Ordenamento do resultado de consulta;
- Funções de grupo.

Explore mais

- [Tipos de dados no Postgresql e Sql Server](#)