Implementação de banco de dados

Aula 06: Linguagem SQL – junção

Apresentação

Até agora, todas as consultas que fizemos retornavam dados de uma única tabela. E se você precisar retornar dados de mais de uma tabela? Por exemplo: o nome do departamento armazenado na tabela Departamento e o nome da região armazenado na tabela Região.

Para resolver este tipo de situação é que existem os comandos de junção, que serão objetos de estudo desta nossa aula.

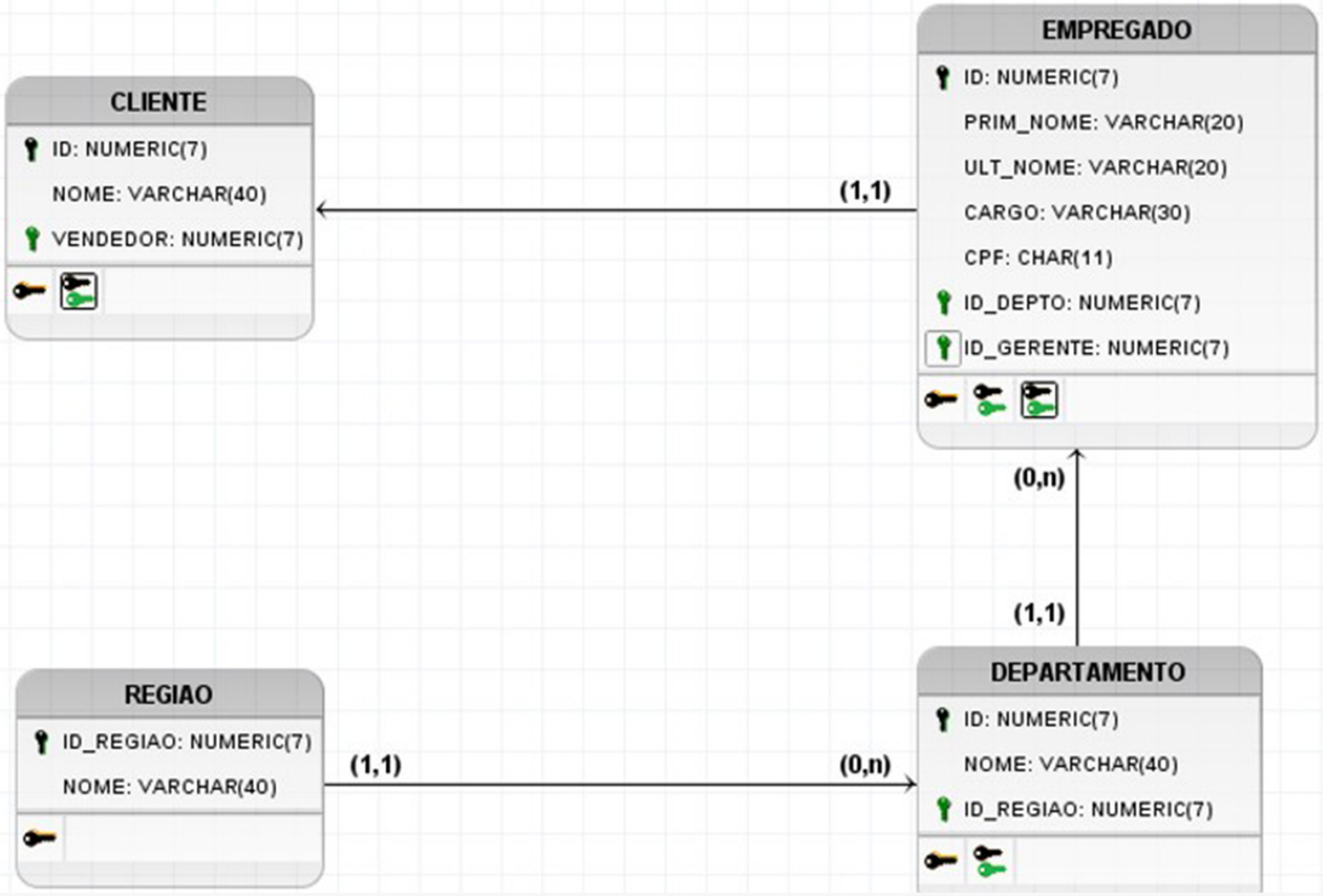
Objetivos

- Elaborar comandos de junção interior na sintaxe tradicional e na ANSI.

Banco de dados de exemplo

Nesta aula, continuaremos utilizando o banco de dados da empresa para os exemplos.

Modelo Lógico:



Clique nos botões para ver as informações.

Região

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

Departamento

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

Empregado

	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

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]

Tendo este banco em mente, é extremante recomendável que você execute os comandos de exemplo no PostGreSQL.

Comentário

Foi escolhido como base o PostgreSql, por ser um SGBD mais leve e fácil de instalar; porém, você pode usar o SQLServer ou o Oracle. Quando houver diferença entre os SGBDs, você receberá um alerta.

Junções de tabelas

O banco de dados da EMPRESA possui, na tabela DEPARTAMENTO, os dados dos departamentos e, na tabela REGIÃO, os dados regiões.

Se você reparar no esquema da tabela DEPARTAMENTO, vai notar a existência de uma chave estrangeira (ID_REGIAO) para a tabela REGIÃO que nos permite saber a região a que o departamento pertence.

Se você deseja pegar dados dos departamentos e de sua região, por exemplo, vai utilizar a FK ID_REGIAO para ligar as linhas das duas tabelas, fazendo uma junção.

Atenção

Lembre-se que, na modelagem lógica, os relacionamentos são mapeados em chaves estrangeiras. Desta forma, fazer junção nada mais é que recuperar os dados das entidades que estão relacionadas nas duas tabelas.

Tipos de junção:

1

Junção cruzada

Retorna o produto cartesiano das duas tabelas, ou seja, a combinação de todas as linhas de uma tabela com todas as linhas de outra tabela.

2

Junção interior

Neste tipo, retornam as linhas que estão relacionadas nas duas tabelas.

3

Junção exterior

Neste tipo, retornam as linhas relacionadas e as não relacionadas em uma ou duas tabelas.

Sintaxe da junção:

Os comandos de junção interior e junção cruzada podem ser escritos de duas formas diferentes, ou seja, em duas sintaxes diferentes. Uma delas é a tradicional, mais antiga, e a segunda é a sintaxe ANSI.

Tradicional

Na cláusula **from**, listamos as tabelas com seus nomes separados por vírgula e, no caso da junção interior, na cláusula **where** deve ser escrita a condição de junção [join condition], que as linhas devem atender para participarem da resposta.

ANSI

Na cláusula **from**, deve ser especificado o tipo de junção que estamos realizando com a condição de junção sem ser estabelecida na subcláusula **on**.

Os comandos de **junção exterior**, na maioria do SGBDs, serão escritos na sintaxe ANSI.

Ao longo desta aula, exemplificaremos as duas sintaxes.

Junção cruzada (cross join)

Este tipo de junção gera a combinação de todas as linhas de uma tabela com todas as linhas da outra tabela; é o chamado **produto cartesiano**.

Como a junção cruzada, combinadas todas as linhas das duas tabelas, não existe uma condição de junção.

Observe o conteúdo das tabelas Departamento e Região:

Departamento

```
1  SELECT *
2  FROM REGIAO
3
4
5
```

Data Output | Explain | Messages | Notificatic

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

Região

```
1  SELECT *
2  FROM DEPARTAMENTO
3
4
5
```

Data Output | Explain | Messages | Notifications | Query His

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

Se você comandar a junção cruzada das duas tabelas, o que irá acontecer?

Note que a tabela Veículo possui 10 linhas e modelo 5. O resultado final terá então 50 linhas, já que cada uma das linhas da tabela será combinada com todas as linhas de Modelo.

Na sintaxe tradicional, o comando será:

```
SELECT *

FROM DEPARTAMENTO, REGIÃO
```

```
1  SELECT *
2  FROM DEPARTAMENTO, REGIAO
3
4
```

Data Output | Explain | Messages | Notifications | Query History

	id	nome	id_regiao	id_regiao	nome
	numeric (7)	character varying (40)	numeric (7)	numeric (7)	character varying (40)
1	10	Administrativo	1	1	Norte
2	20	Vendas	1	1	Norte
3	30	Compras	2	1	Norte
4	10	Administrativo	1	2	Sul
5	20	Vendas	1	2	Sul
6	30	Compras	2	2	Sul

Note que:

1

Retornam seis linhas.

2

Se você olhar a última coluna (nome da região primeiro), temos a região Norte, associada a todos os três departamentos, e depois Sul associada a todos os departamentos.

3

Se existissem mais regiões, o descrito no item 2 se repetiria para todas.

4

As linhas 1 e 4 são do mesmo departamento, já que é o início de um novo conjunto de linhas ligadas à mesma região.

5

Não existe condição de junção.

6

O esquema do retorno, ou seja a ordem das colunas, é a justaposição das colunas da primeira tabela da cláusula **from** com as da segunda. Se você inverter a ordem na cláusula **from**, o retorno também será invertido, como mostra a figura a seguir.

```
1  SELECT *
2  FROM REGIAO , DEPARTAMENTO
3
4
5
```

Data Output | Explain | Messages | Notifications | Query History

	id_regiao numeric (7)	nome character varying (40)	id numeric (7)	nome character varying (40)	id_regiao numeric (7)
1	1	Norte	10	Administrativo	1
2	1	Norte	20	Vendas	1
3	1	Norte	30	Compras	2
4	2	Sul	10	Administrativo	1
5	2	Sul	20	Vendas	1
6	2	Sul	30	Compras	2

Já na sintaxe ANSI, devemos escrever na cláusula **from** o tipo de junção comandado.

```
SELECT *

FROM DEPARTAMENTO CROSS JOIN REGIAO
```

```
SELECT *
FROM DEPARTAMENTO CROSS JOIN REGIAO
```

Data Output | Explain | Messages | Notifications | Query History

	id numeric (7)	nome character varying (40)	id_regiao numeric (7)	id_regiao numeric (7)	nome character varying (40)
1	10	Administrativo	1	1	Norte
2	20	Vendas	1	1	Norte
3	30	Compras	2	1	Norte
4	10	Administrativo	1	2	Sul
5	20	Vendas	1	2	Sul
6	30	Compras	2	2	Sul

Observe agora o seguinte comando:

```
SELECT *

FROM EMPREGADO

CROSS JOIN DEPARTAMENTO

CROSS JOIN REGIAO
```

```
SELECT *
FROM EMPREGADO
     CROSS JOIN DEPARTAMENTO
     CROSS JOIN  REGIAO
```

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)	id numeric (7)	nome character varying (40)	id_regiao numeric (7)	id_regiao numeric (7)	nome character varying (40)
1	1	Velasques	Carmen	Presidente	29500.00	2009-05-05	34567890125	10	[null]	10	Administrativo	1	1	Norte
2	1	Velasques	Carmen	Presidente	29500.00	2009-05-05	34567890125	10	[null]	10	Administrativo	1	2	Sul
3	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1	10	Administrativo	1	1	Norte
4	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1	10	Administrativo	1	2	Sul
5	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1	10	Administrativo	1	1	Norte
6	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1	10	Administrativo	1	2	Sul
7	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2	10	Administrativo	1	1	Norte
8	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2	10	Administrativo	1	2	Sul
9	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3	10	Administrativo	1	1	Norte
10	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3	10	Administrativo	1	2	Sul
11	6	Ugarte	Marlene	Vendedor	3500.00	2009-03-03	87654345678	20	3	10	Administrativo	1	1	Norte
12	6	Ugarte	Marlene	Vendedor	3500.00	2009-03-03	87654345678	20	3	10	Administrativo	1	2	Sul
13	1	Velasques	Carmen	Presidente	29500.00	2009-05-05	34567890125	10	[null]	20	Vendas	1	1	Norte
14	1	Velasques	Carmen	Presidente	29500.00	2009-05-05	34567890125	10	[null]	20	Vendas	1	2	Sul
15	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1	20	Vendas	1	1	Norte
16	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1	20	Vendas	1	2	Sul
17	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1	20	Vendas	1	1	Norte
18	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1	20	Vendas	1	2	Sul
19	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2	20	Vendas	1	1	Norte
20	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2	20	Vendas	1	2	Sul
21	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3	20	Vendas	1	1	Norte
22	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3	20	Vendas	1	2	Sul

Ele faz o produto cartesiano das três tabelas, gerando 36 linhas (6 empregados X 3 departamentos X 2 regiões).

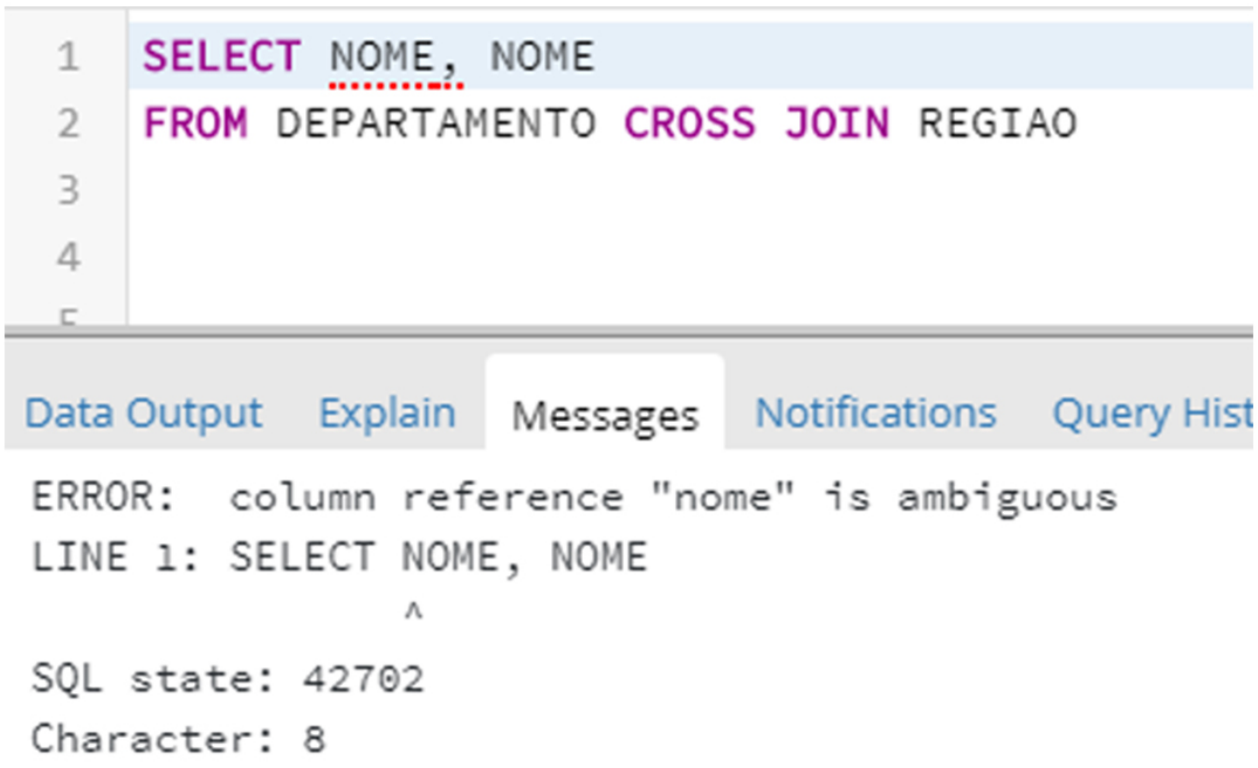
Em qualquer comando de **junção**, independentemente da sintaxe, podemos determinar as colunas que desejamos, basta listá-las no **Select**.

Se você desejasse retornar apenas o **nome** do departamento e o **nome** da região, poderia comandar:

```
SELECT NOME, NOME

FROM DEPARTAMENTO CROSS JOIN REGIAO
```

Correto? Não; veja o erro do comando:



Analise a mensagem de erro. Ela diz que a coluna **nome** é ambígua. Como assim, ambígua?

Acontece que tanto a tabela **Departamento** como a tabela Região têm uma coluna com o título **Nome**, e o SGBD não sabe qual delas utilizar no comando. A solução é qualificar a coluna, ou seja, colocar o nome da tabela seguido do nome da coluna ligados por um ponto, da seguinte forma:

```
1  SELECT DEPARTAMENTO.NOME, REGIAO.NOME
2  FROM DEPARTAMENTO CROSS JOIN REGIAO
3
4
5
```

Data Output | Explain | Messages | Notifications | Query Hi

	nome character varying (40)	nome character varying (40)
1	Administrativo	Norte
2	Vendas	Norte
3	Compras	Norte
4	Administrativo	Sul
5	Vendas	Sul
6	Compras	Sul

Esta solução, a qualificação como o nome da tabela, pode ser trabalhosa, já que os nomes de tabela podem ter até 40 caracteres. Uma forma para resolvermos isso é apelidarmos as tabelas e usarmos este apelido para qualificarmos as colunas.

Um apelido (também chamado de alias) de tabela nada mais é que uma ou mais letras colocadas logo depois do nome da tabela na cláusula FROM.

Observe este comando e reflita:

```
1  SELECT D.NOME, R.NOME
2  FROM DEPARTAMENTO D CROSS JOIN  REGIAO R
3
4
5
```

Data Output | Explain | Messages | Notifications | Query His

	nome character varying (40)	nome character varying (40)
1	Administrativo	Norte
2	Vendas	Norte
3	Compras	Norte
4	Administrativo	Sul
5	Vendas	Sul
6	Compras	Sul

Atenção

Depois que você cria o apelido, não pode mais utilizar o nome da tabela para qualificar as colunas.

Junção interior (inner join)

Este tipo de junção recupera as linhas relacionadas nas tabelas envolvidas na junção.

Se você desejasse retornar os dados do **departamento** e os da região a que o departamento pertence, seria este tipo de junção que deveria comandar. O fato de retornarem as linhas relacionadas apenas acarreta que, se existem departamentos sem região ou região que não tem departamentos, estes não aparecem no resultado.

O Inner join é também chamado de Equijoin quando a condição de junção utiliza a igualdade, ou seja, quando o valor da coluna de uma tabela (normalmente a chave estrangeira) é igual ao valor da coluna na outra tabela (normalmente a chave primária).

Essa junção pode ser comandada tanto na sintaxe tradicional como na ANSI. Vamos estudá-las.

Junção interior – sintaxe tradicional

A sintaxe tradicional do inner join tem o seguinte formato:

sintaxe tradicional do inner join

```
select <colunas>
from tabela1, tabela2
Where <condição de junção>;
```

Condição de junção é uma expressão de comparação entre as colunas das duas tabelas envolvidas, normalmente a chave de primária de uma com a chave estrangeira da outra.

Na realidade, a sintaxe tradicional pode ser entendida como o produto cartesiano das duas tabelas seguido de uma seleção utilizando a cláusula **where**.

Vamos comandar a junção de Departamento e Região.

A coluna ID_REGIAO em departamento é uma FK para a tabela Região, que tem como PK a coluna ID_REGIAO. Desta forma, o comando seria:

```
SELECT *

FROM DEPARTAMENTO D, REGIAO R

WHERE D.ID_REGIAO = R.ID_REGIAO
```

```
1  SELECT *
2  FROM DEPARTAMENTO D ,  REGIAO R
3  WHERE D.ID_REGIAO = R.ID_REGIAO
4
```

Data Output | Explain | Messages | Notifications | Query History

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

Junção interior – sintaxe ANSI

Na sintaxe ANSI, junções interiores são indicadas com inner join:

sintaxe ANSI

```
select <colunas>
from tabela1 INNER JOIN tabela2 ON <condição de junção;>
```

No caso anterior, o comando seria:

```
SELECT *

FROM DEPARTAMENTO D

INNER JOIN REGIAO R ON D.ID_REGIAO = R.ID_REGIAO
```

```
1  SELECT *
2  FROM DEPARTAMENTO D
3       INNER JOIN   REGIAO R  ON D.ID_REGIAO = R.ID_REGIAO
4
```

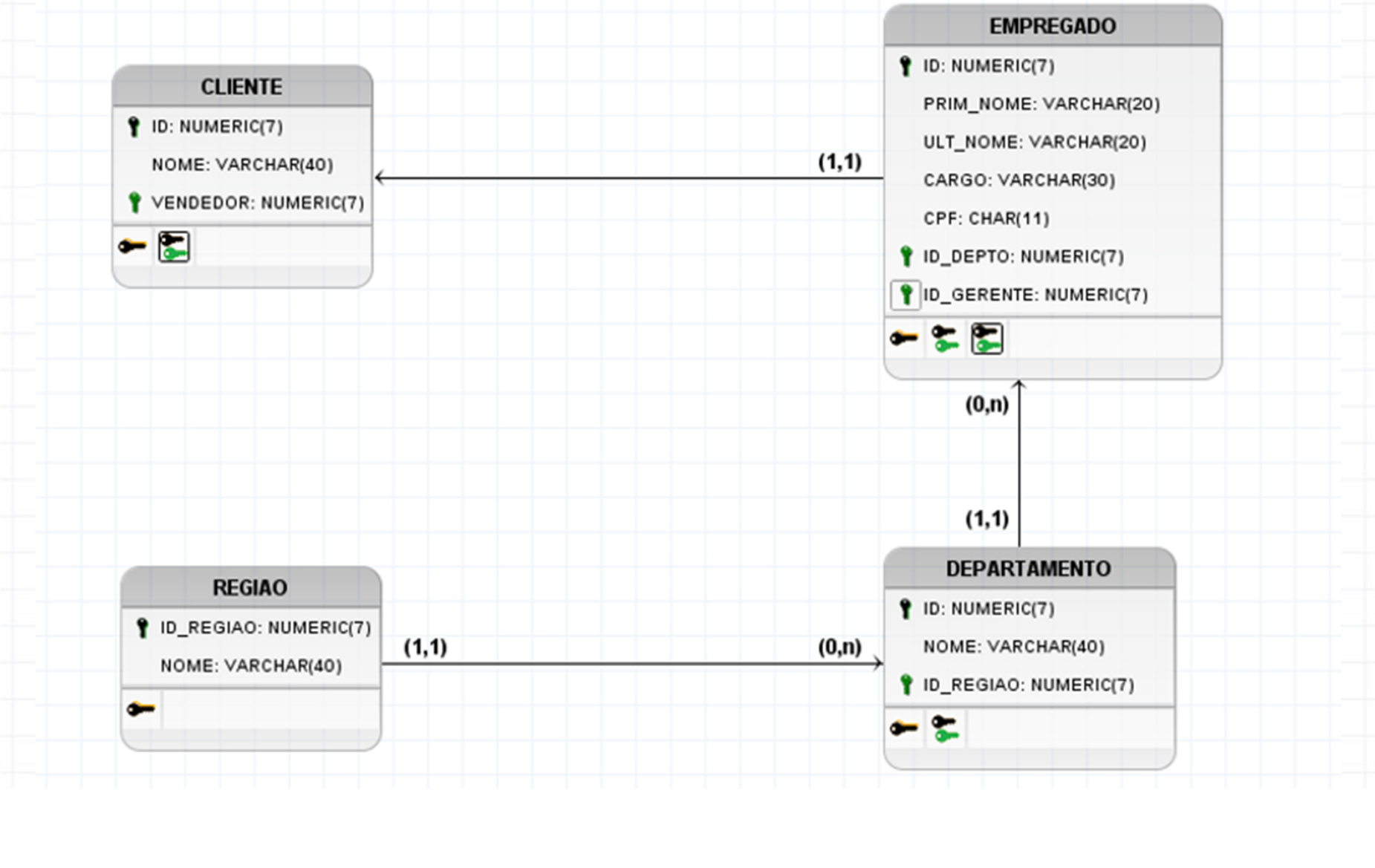
Data Output | Explain | Messages | Notifications | Query History

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

Junção interior de mais de duas tabelas

Podemos fazer a junção de mais de duas tabelas. Para isto, basta acrescentarmos uma nova tabela com uma nova cláusula de junção.

Vejamos um exemplo inicialmente na sintaxe ANSI. A primeira coisa a fazer é identificar quais tabelas podem se juntar. Analise o modelo do banco:



Temos uma FK de Empregado para Departamento e uma FK de Departamento para Região. Desta forma, temos que comandar a junção de Empregado com Departamento, que gera uma tabela intermediária em memória, e juntar este resultado com Região.

```
1  SELECT E.ID, E.ULT_NOME, D.ID, D.NOME, R.NOME
2  FROM EMPREGADO E
3       INNER JOIN DEPARTAMENTO D  ON E.ID_DEPTO = D.ID
4       INNER JOIN REGIAO R ON D.ID_REGIAO = R.ID_REGIAO
```

Data Output | Explain | Messages | Notifications | Query History

	id numeric (7)	ult_nome character varying (20)	id numeric (7)	nome character varying (40)	nome character varying (40)
1	1	Velasques	10	Administrativo	Norte
2	2	Neves	30	Compras	Sul
3	3	Nogueira	20	Vendas	Norte
4	4	Queiroz	30	Compras	Sul
5	5	Rodrigues	20	Vendas	Norte
6	6	Ugarte	20	Vendas	Norte

O mesmo comando na sintaxe tradicional seria:

```
1  SELECT E.ID, E.ULT_NOME, D.ID, D.NOME, R.NOME
2  FROM EMPREGADO E , DEPARTAMENTO D, REGIAO R
3  WHERE E.ID_DEPTO = D.ID AND D.ID_REGIAO = R.ID_REGIAO
```

Data Output | Explain | Messages | Notifications | Query History

	id numeric (7)	ult_nome character varying (20)	id numeric (7)	nome character varying (40)	nome character varying (40)
1	1	Velasques	10	Administrativo	Norte
2	2	Neves	30	Compras	Sul
3	3	Nogueira	20	Vendas	Norte
4	4	Queiroz	30	Compras	Sul
5	5	Rodrigues	20	Vendas	Norte
6	6	Ugarte	20	Vendas	Norte

Neste caso, colocamos as tabelas na cláusula **from** e as condições de junção no **where**, ligadas por **and**.

Saiba mais

Utilizando o nosso banco de dados de exemplo, vamos fazer alguns exercícios.

Notas

Título modal [1]

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

Título modal [1]

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

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

- Subconsultas;
- Operadores de conjunto.

Explore mais

Veja:

- Junções entre tabelas
- Introdução ao SQL/ Junções
- Left join e inner join: junção em consultas SQL
- Inner, cross, left, rigth e full joins
- Joins em SQL
- PostgreSQL Prático/DML/Consultas Join