



# Disciplina: Fundamentos de Banco de Dados

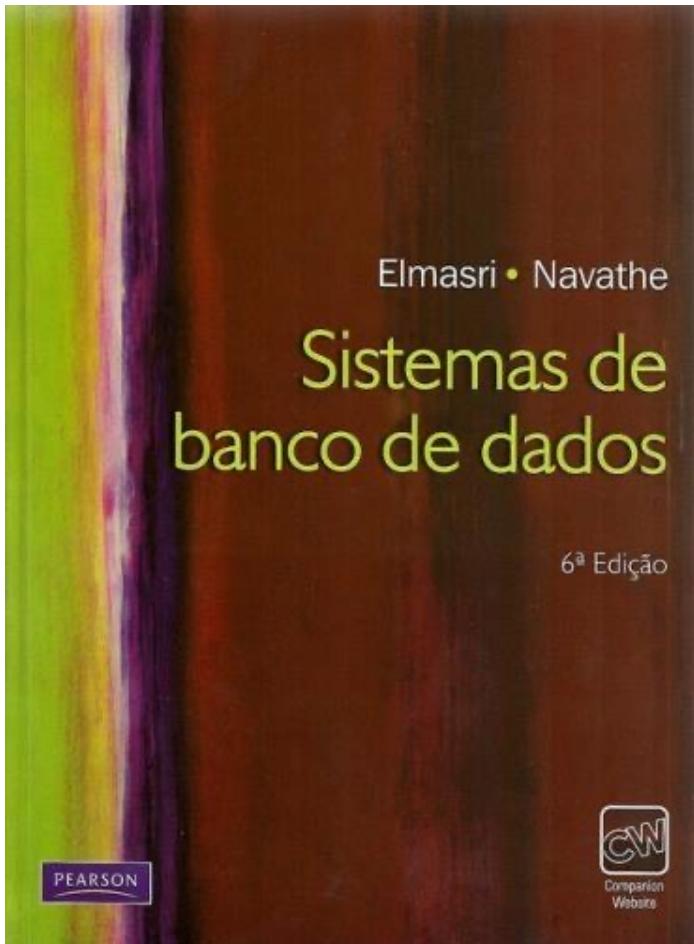
## 12. SQL (Parte II e III) Consultas

---

Professora: Marília S. Mendes  
E-mail: marilia.mendes@ufc.br

# Onde encontrar a matéria?

---



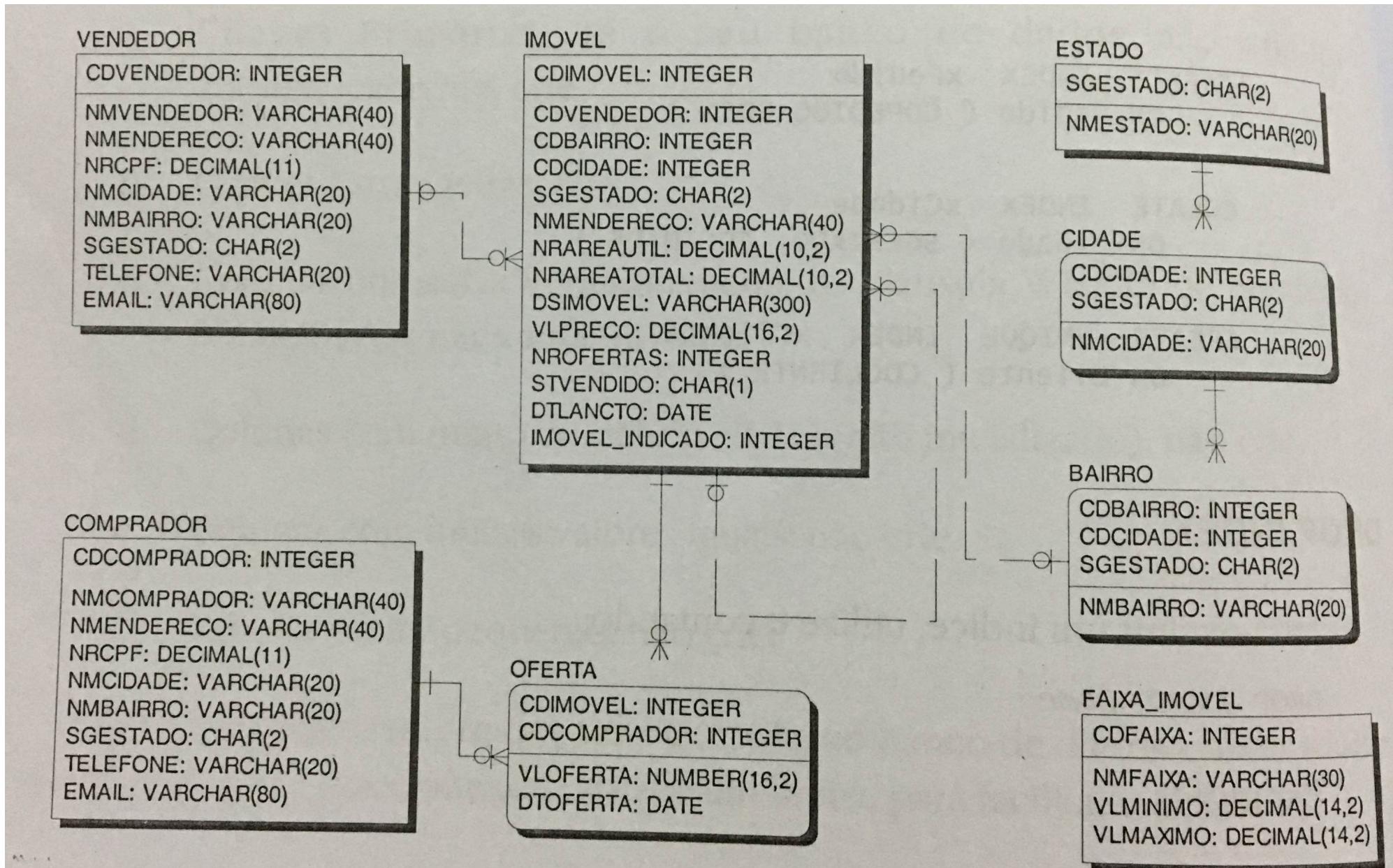
- ▶ Capítulo 4
- ▶ Consultas básicas, aninhadas e correlacionadas; operadores relacionais, lógicos e especiais; renomeação, operações com conjuntos e aritméticas, junções e variações.

# Aprendendo exercitando no PostgreSQL

---

- ▶ Abra a interface de consulta do PostgreSQL
  
- ▶ Serão usados as seguintes bases de dados para os exercícios em sala:
  - ▶ ImovelNet
  - ▶ Empresa

# BD – Database RedeImoveis



EMPREGADO	PNAME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
John	B	Smith		123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong		333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya		999887777	1968-01-19	3321 Castle, Soring, TX	F	25000	987654321	4
Jennifer	S	Wallace		987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan		666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English		453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar		987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg		888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPARTAMENTO	DNAME	<u>DNUMERO</u>	GERSSN	GERDATAINICIO	DEPTO_LOCALIZACOES	<u>DNUMERO</u>	<u>DLOCALIZACAO</u>
					1	Houston	
	Pesquisa	5	333445555	1988-05-22		4	Stafford
	Administração	4	987654321	1995-01-01		5	Bellaire
	Sede administrativa	1	888665555	1981-06-19		5	Sugarland
						5	Houston

TRABALHA_EM	ESSN	PNO	HORAS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJETO	PJNOME	<u>PNUMERO</u>	PLOCALIZACAO	DNUM
	ProdutoX	1	Bellaire	5
	ProdutoY	2	Sugarland	5
	ProdutoZ	3	Houston	5
	Automatização	10	Stafford	4
	Reorganização	20	Houston	1
	Novos Benefícios	30	Stafford	4

DEPENDENTE	ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
	333445555	Alice	F	1986-04-05	FILHA
	333445555	Theodore	M	1983-10-25	FILHO
	333445555	Joy	F	1958-05-03	CÔNJUGE
	987654321	Abner	M	1942-02-28	CÔNJUGE
	123456789	Michael	M	1988-01-04	FILHO
	123456789	Alice	F	1988-12-30	FILHA
	123456789	Elizabeth	F	1967-05-05	CÔNJUGE

# Consultas de recuperação básicas em SQL

---

- ▶ **SQL –DQL (*Data Query Language*)**
  
- ▶ É a parte da SQL que trata das consultas e muitas vezes está embutida na DML.
- ▶ É uma linguagem de consulta baseada na Álgebra Relacional e no cálculo relacional de tupla.
- ▶ O resultado é uma tabela, que pode ter uma ou mais linhas, com um ou mais atributos.

# Consultas SQL

- SQL possui um comando básico para obter informações da base de dados: o comando SELECT
- Ele *não é o mesmo* da operação de SELEÇÃO da álgebra relacional  
O comando SELECT embute várias operações da álgebra relacional, como:  $\sigma$ ,  $\pi$ ,  $\bowtie$  , etc.
- Existe uma diferença importante entre SQL e o modelo relacional formal; com SQL é permitido que uma tabela (relação) tenha duas ou mais tuplas idênticas

# Consultas SQL



A forma básica do comando SELECT da SQL é chamado de bloco *SELECT-FROM-WHERE*

**SELECT**      < lista atributos >  
**FROM**        < lista tabelas >  
**WHERE**      <condição>



< lista atributos > é uma lista de nomes de atributos cujos valores serão obtidos pela consulta



< lista tabelas > é uma lista de nomes de relações necessárias para processar a consulta



<condição> é uma expressão condicional (Booleana) que define as tuplas a serem obtidas pela consulta

# Consultas SQL Simples

---

- ➔ Exemplo de uma consulta simples sobre *uma* relação
- ➔ Consulta 0: Obtenha a data de nascimento e o endereço do empregado cujo nome é 'John B. Smith'.
- ➔ Consulta 0 em álgebra:



# Consultas SQL Simples

- Exemplo de uma consulta simples sobre *uma* relação
- Consulta 0: Obtenha a data de nascimento e o endereço do empregado cujo nome é 'John B. Smith'.

**Q0:SELECT** DATANASC, ENDERECO

**FROM** EMPREGADO

A cláusula **FROM**  
especifica a relação da  
consulta

**WHERE** PNOME='John'

**AND** MINICIAL='B'

**AND** UNOME='Smith'

# Consultas SQL Simples

- Exemplo de uma consulta simples sobre *uma* relação
- Consulta 0: Obtenha a data de nascimento e o endereço do empregado cujo nome é 'John B. Smith'.

**Q0:**SELECT DATANASC, ENDERECO

**FROM** EMPREGADO

**WHERE** PNOME='John'

**AND** MINICIAL='B'

**AND** UNOME='Smith'

A cláusula WHERE especifica a condição de seleção

# Consultas SQL Simples

- Exemplo de uma consulta simples sobre *uma* relação
- Consulta 0  
empregado  

A cláusula SELECT  
especifica os atributos  
da projeção

 cimento e o endereço do  
mith'.

**Q0:** **SELECT** DATANASC, ENDERECO

**FROM** EMPREGADO

**WHERE** PNOME='John'

**AND** MINICIAL='B'

**AND** UNOME='Smith'

# Consultas SQL Simples

---

```
Q0: SELECT      DATANASC, ENDERECO  
        FROM        EMPREGADO  
        WHERE       PNOME='John'  
        AND          MINICIAL='B'  
        AND          UNOME='Smith'
```

- Embora semelhante à expressão algébrica que possui a SELEÇÃO e PROJEÇÃO, o resultado do SQL *pode conter* tuplas duplicadas



## Exercício 1 – Database ImovelNet

---

- ▶ Liste as cidades do estado do Ceará.
-

## SQL sem cláusula Where

---

- ➔ SQL *sem a cláusula WHERE* indica que não há condição de seleção; assim, *todas as tuplas* da relação da cláusula FROM são selecionadas
- ➔ É equivalente à condição WHERE verdadeira
- ➔ Consulta 9: Obter o SSN para todos os empregados.

**Q9:**      **SELECT**      SSN  
                **FROM**      EMPREGADO



## Exercício 2 – Database ImovelNet

---

- ▶ Liste os nomes dos vendedores.
-

## **Uso do \***

→ Para obter todos os atributos das tuplas selecionadas, o \* é usado,

Exemplos:

**Q1C:**    **SELECT**        \*  
                **FROM**        EMPREGADO  
                **WHERE**      DNO=5



## Exercício 3 – Database ImovelNet

---

- ▶ Liste as informações dos bairros do estado do Rio de Janeiro.

## Uso de DISTINCT

---

- ➔ SQL não considera a relação como um conjunto; *tuplas duplicadas podem ser geradas*
- ➔ Para eliminar tuplas duplicadas no resultado da consulta, a palavra-chave **DISTINCT** é usada



## Uso de DISTINCT

---

→ Por exemplo, o resultado da consulta **Q11** pode gerar valores duplicados de SALARIO, enquanto que **Q11A** não permite valores duplicados nas tuplas do resultado

**Q11:**    **SELECT**    SALARIO  
              **FROM**      EMPREGADO

**Q11A:**    **SELECT**    **DISTINCT** SALARIO  
              **FROM**      EMPREGADO



# Operadores relacionais

---

Operador	Significado	Exemplo
=	Igual	Codigo_autor = 2
<	Menor que	Preco_venda < 10
<=	Menor ou igual a	Preco_venda <= 10
>=	Maior ou igual a	Preco_venda >= 10
!= ou <>	Diferente	Codigo_autor != 2 ou codigo_autor <> 2

# Operadores lógicos

---

Operador	Significado	Exemplo
AND	e	Condicao1 AND condicao2
OR	Ou	Condicao1 OR condicao2
NOT ou !	Não / negação	NOT Condicao1

## Exercício 4 – Database ImovelNet

---

- 1) Liste quais são os estados das cidades cadastradas.
- 2) Liste os estados *distintos* das cidades cadastradas

# Exercício 5 – Database ImovelNet

- ▶ Liste todos os campos e linhas da tabela BAIRRO.

Output pane

Data Output Explain Messages History

	cdbairro	nmbairro	cdcidade	sestado
	integer	character varying(20)	integer	character(2)
1		1 Jardins	1	SP
2		2 Morumbi	1	SP
3		3 Aeroporto	1	SP
4		1 Aeroporto	2	RJ
5		2 Flamengo	2	RJ

OK.

## Exercício 6 – Database ImovelNet

- ▶ Liste todas as linhas e os campos CDCOMPRADOR, NMCOMPRADOR e EMAIL da tabela COMPRADOR.

Data Output			
	cdcomprador	nmcomprador	email
	integer	character varying(40)	character varying(80)
1		1 Emmanuel Antunes	eantunes@ufc.br
2		2 Joana Pereira	jpereira@novatec.br
3		4 Manfred Augusto	manfred@ufc.br
4		5 Zé Pereira	zezim@ufc.br
5		3 Ronaldo Campelo	rcampelo@novatec.br

# Exercício 7 – Database ImovelNet

- ▶ Liste todos os bairros do estado de SP.

Output pane

Data Output Explain Messages History

	cdbairro	nmbairro	cdcidade	sgestado
	integer	character varying(20)	integer	character(2)
1		1 Jardins	1	SP
2		2 Morumbi	1	SP
3		3 Aeroporto	1	SP



## Exercício 8 – Database ImovelNet

- ▶ Liste as colunas CDIMOVEL, CDVENDEDOR e VLPRECO de todos os imóveis do vendedor 2.



## Exercício 9 – Database ImovelNet

- ▶ Liste as colunas CDIMOVEL, CDVENDEDOR, VLPRECO e SGESTADO dos imóveis cujo preço de venda seja inferior a 150 mil e sejam do estado do RJ.



## Exercício 10 – Database ImovelNet

Liste as colunas CDIMOVEL, CDVENDEDOR, VLPRECO e SGESTADO dos imóveis cujo preço de venda seja inferior a 150 mil e o vendedor não seja 2.



## ORDER BY

---

- ▶ Determinar a ordem em que são mostradas as linhas de uma tabela.

```
SELECT SALARIO FROM EMPREGADO ORDER BY  
SALARIO
```

```
SELECT SALARIO FROM EMPREGADO ORDER BY  
SALARIO DESC
```

```
SELECT PNOME, SALARIO FROM EMPREGADO  
ORDER BY SALARIO, PNOME
```

---

# Exercício 11 – Database ImovelNet

- ▶ Liste todas as linhas e os campos CDVENDEDOR, NMVENDEDOR e EMAIL da tabela VENDEDOR em ordem alfabética.

	cdvendedor integer	nmvendedor character varying(40)	email character varying(80)
1	3	André Cardoso	acardoso@novatec.com.br
2	8	Gal Costa	gal@novatec.com.br
3	7	Jorge Benjor	jben@novatec.com.br
4	5	Luiz Gonzaga	gonzagao@novatec.com.br
5	2	Marcos Andrade	mandrade@novatec.com.br
6	1	Maria da Silva	msilva@novatec.com.br
7	6	Renato Russo	rrusso@novatec.com.br
8	4	Tatiana Souza	tsouza@novatec.com.br

OK.

# Exercício 12 – Database ImovelNet

- ▶ Repita o comando anterior em ordem alfabética decrescente.

Data Output			
	cdvendedor integer	nmvendedor character varying(40)	email character varying(80)
1	4	Tatiana Souza	tsouza@novatec.com.br
2	6	Renato Russo	rrusso@novatec.com.br
3	1	Maria da Silva	msilva@novatec.com.br
4	2	Marcos Andrade	mandrade@novatec.com.br
5	5	Luiz Gonzaga	gonzagao@novatec.com.br
6	7	Jorge Benjor	jben@novatec.com.br
7	8	Gal Costa	gal@novatec.com.br
8	3	André Cardoso	acardoso@novatec.com.br

OK.

# Consultas SQL em mais de uma tabela

---

<b>SELECT</b>	Atributo1, atributo2, atributo3
<b>FROM</b>	Tabela1, tabela2
<b>WHERE</b>	condicao1
<b>AND</b>	condicao2



# Consultas SQL em mais de uma tabela

---



Consulta 1: Obtenha o nome e endereço de todos os empregados que trabalham para o departamento de 'Pesquisa'.



# Consultas SQL Simples

→ Consulta 1: Obtenha o nome e endereço de todos os empregados que trabalham para o departamento de 'Pesquisa'.

Q1: **SELECT** PNAME, UNOME, ENDERECO  
**FROM** EMPREGADO, DEPARTAMENTO  
**WHERE** DNAME='Pesquisa'  
**AND** DNUMERO=DNO

Projeção

Seleção

Junção

```
DEPTO_PESQUISA ← σDNAME='Pesquisa' (DEPARTAMENTO)
EMPS_PESQ ← (DEPTO_PESQUISA DNUMERO= DNO EMPREGADO
RESULT ← πPNAME, UNOME, ENDERECO (EMPS_PESQ)
```

# Consultas SQL em mais de uma tabela

---

- ➡ Consulta 2: Para cada projeto localizado em 'Stafford', listar o número do projeto, o número do departamento de controle, e o sobrenome, endereço e data de nascimento do gerente do departamento.



# Consultas SQL em mais de uma tabela

**Q2:** **SELECT** PNUMERO, DNUM, UNOME, DATANASC, ENDEREC  
**FROM** PROJETO, DEPARTAMENTO, EMPREGADO  
**WHERE** DNUM=DNUMERO  
**AND** GERSSN=SSN  
**AND** PLOCALIZACAO='Stafford'

- ➡ Em Q2, existem *duas* condições de junção
- ➡ A condição de junção DNUM=DNUMERO relaciona o projeto ao seu departamento de controle
- ➡ A condição de junção GERSSN=SSN relaciona o departamento de controle ao empregado que gerencia aquele departamento



# Consultas SQL - atributos iguais

- ▶ Liste o nome dos departamento e o nome de suas localizações

ERRO: referência à coluna "dnumero"  
é ambígua  
LINE 3: where dnumero=dnumero

```
select dnome, dlocalizacao  
from departamento,  
dept_localizacoes  
where dnumero=dnumero
```



```
select departamento.dnome, dept_localizacoes.dlocalizacao  
from departamento, dept_localizacoes  
where departamento.dnumero=dept_localizacoes.dnumero
```

ou

```
select a.dnome, b.dlocalizacao  
from departamento as a,  
dept_localizacoes as b  
where a.dnumero=b.dnumero
```

## SQL - renomear

---

- Em SQL, pode-se usar o mesmo nome para dois (ou mais) atributos, desde que os atributos sejam de *relações diferentes*
- Uma consulta que referencia dois ou mais atributos com o mesmo nome precisa *qualificar* o nome do atributo com o nome da relação através de um *prefixo*, com o nome da relação, para o nome do atributo.
- Exemplo:

TRABALHA\_EM.ESSN, DEPENDENTE.ESSN



## SQL - renomear

---

- ➔ Algumas consultas precisam referenciar a mesma relação duas vezes
- ➔ Nesse caso, *apelidos (alias)* são dados ao nome da relação (renomear)
- ➔ Consulta 8: Para cada empregado, obtenha o nome do empregado e o nome do seu supervisor



# SQL - renomear

```
SELECT      E.PNOME, E.UNOM  
           , S.PNOME, S.UNOM  
FROM        EMPREGADO E S  
WHERE       E.SUPERSSN = S.SSN
```

E e S são nomes alternativos à relação EMPREGADO, também chamados de variáveis de tuplas



## SQL - renomear

---

```
SELECT      E.PNOME, E.UNOME,  
                  S.PNOME, S.UNOME  
FROM        EMPREGADO E S  
WHERE       E.SUPERSSN = S.SSN
```

E e S podem ser vistos como *cópias diferentes* de EMPREGADO ;  
E representa os empregados no papel de "supervisionados" e  
S representa empregados no papel de *supervisores*



## SQL - renomear

- ➔ Renomear também pode ser usado em qualquer consulta SQL por simplicidade.  
O termo AS também pode ser usado para renomear

```
SELECT      E.PNOME, E.UNOME,  
            S.PNOME, S.UNOME  
FROM        EMPREGADO AS E,  
            EMPREGADO AS S  
WHERE       E.SUPERSSN = S.SSN
```

PostgreSQL

# Operações de conjuntos

---

- ➡ SQL apresenta algumas operações de conjuntos:
- ➡ A operação de união (**UNION**), e em *algumas versões* da SQL há também as operações de diferença (**MINUS**) e interseção (**INTERSECT**)
- ➡ As relações resultantes dessas operações de conjuntos são de fato conjuntos de tuplas; *tuplas duplicadas são eliminadas do resultado*
- ➡ As operações de conjuntos se aplicam apenas a *relações união compatíveis*; as duas relações tem que ter os mesmos atributos que precisam aparecer na mesma ordem



# Operações de conjuntos

---

→ Obter o SSN dos empregados do depto 5 e dos seus supervisores, em álgebra:



# Operações de conjuntos

---



Consulta 4: Obter a lista do nome de todos os projetos que envolvem algum empregado cujo sobrenome é 'Smith' como trabalhador ou como gerente do departamento que controla o projeto.

# Operadores especiais

---

- ▶ **Between**
  - ▶ **Like**
  - ▶ **In**
  - ▶ **Is null**
  - ▶ **Exists**
-

## BETWEEN

---

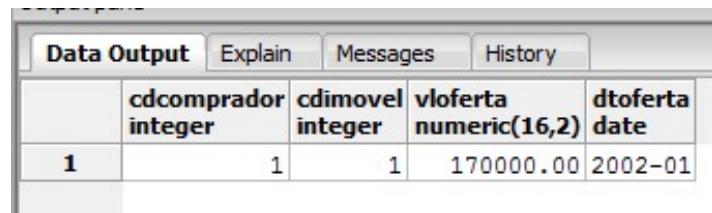
- ▶ Esse operador serve para determinar um intervalo de busca. Assim, sempre quisermos realizar buscas que indiquem um intervalo de números, datas, etc., podemos utilizar o **between** para simplificar a forma de escrevermos o comando.
- ▶ Exemplos:

```
SELECT PNOME, SALARIO FROM EMPREGADO WHERE  
SALARIO BETWEEN 30000 AND 50000;
```

```
SELECT DNOME FROM DEPARTAMENTO WHERE  
GERDATAINICIO BETWEEN '1990-01-01' AND '2000-01-  
01';
```

## Exercício 13 – Database ImovelNet

- ▶ Liste todas as ofertas cujo valor esteja entre 150 mil e 170 mil.



The screenshot shows a PostgreSQL Data Output window with four tabs: Data Output (selected), Explain, Messages, and History. The Data Output tab displays a single row of data in a table format:

	<b>cdcomprador</b> <b>integer</b>	<b>cdimovel</b> <b>integer</b>	<b>vloferta</b> <b>numeric(16,2)</b>	<b>dtoferta</b> <b>date</b>
1	1	1	170000.00	2002-01



## Exercício 14 – Database ImovelNet

- ▶ Liste todas as ofertas cuja data da oferta esteja entre '01/02/01' e '11/03/11'

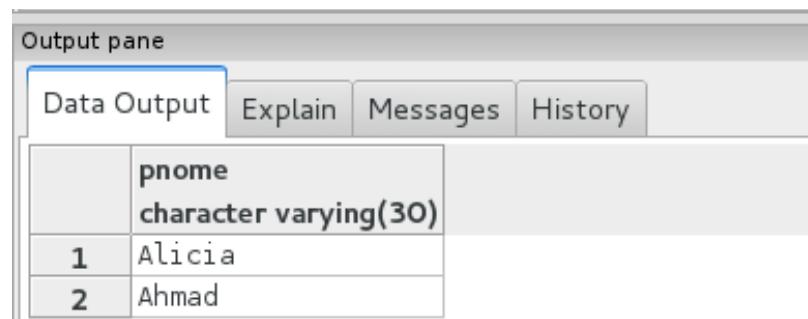
Data Output				
	cdcomprador integer	cdimovel integer	vloferta numeric(16,2)	dtoferta date
1	1	1	170000.00	2002-01-10



# Like

- ▶ O comando **LIKE** é utilizado quando queremos comparar *strings* em uma consulta. No entanto, diferentemente dos operadores relacionais de igual (=) e diferente (<>) que comparam a *string* exata, o comando **LIKE** permite que utilizemos operadores que comparam se parte de uma string está contida em algum registro de uma tabela.

```
SELECT PNAME FROM EMPREGADO  
WHERE PNAME LIKE 'A%'
```



The screenshot shows a software interface for running SQL queries. At the top, there's a menu bar with tabs: 'Output pane' (which is selected), 'Data Output' (highlighted in blue), 'Explain', 'Messages', and 'History'. Below this is a table with two rows of data. The table has one column labeled 'pnome'. The first row contains the value 'Alicia' and is labeled '1' in the row number column. The second row contains the value 'Ahmad' and is labeled '2' in the row number column. The data type for the column is listed as 'character varying(30)'.

	pnome
1	Alicia
2	Ahmad

# Exercício 15 – Database ImovelNet

- ▶ Liste todos os vendedores que comecem com a letra M

	cdvendedor integer	nmvendedor character varying(40)	nmendereco character varying(40)
1	1	MARIA DA SILVA	RUA DO GRITO, 45
2	2	MARCOS ANDRADE	AV DA SAUDADE, 325
3	5	MARIANA DA FONSECA	RUA DO DESPERO,



# Variações do LIKE

- ▶ Com este operador, podemos comparar cadeia de caracteres, utilizando padrões de comparação para um ou mais caracteres.

Expressão	Explicação
LIKE 'A%'	Todas as palavras que iniciem com a letra A.
LIKE '%A'	Todas as palavras que terminem com a letra A.
LIKE '%A%'	Todas as palavras que tenham A em qualquer posição.
LIKE 'A_'	String de dois caracteres que tenham a primeira letra A e o segundo caractere seja qualquer outro.
LIKE '_A'	String de dois caracteres cujo primeiro caractere seja qualquer um e a última letra seja A.
LIKE '_A_'	String de três caracteres cuja segunda letra seja A, independentemente do primeiro ou do último caractere.
LIKE '%A_'	Todos que tenham a letra A na penúltima posição e a última seja qualquer outro caractere.
LIKE '_A%	Todos que tenham a letra A na segunda posição e o primeiro caractere seja qualquer um.

## Exercício 16 – Database ImovelNet

- ▶ Liste todos os vendedores que tenham a letra A na quinta posição do nome

	<b>cdvendedor</b> integer	<b>nmvendedor</b> character varying(40)	<b>nmendereco</b> character varying(40)
1	1	MARIA DA SILVA	RUA DO GRITO, 45
2	4	TATIANA SOUZA	RUA DO IMPERADOR,
3	5	MARIANA DA FONSECA	RUA DO DESESPERO,



## Exercício 17 – Database ImovelNet

- ▶ Liste todos os compradores que tenham a letra U em qualquer posição do endereço



# Exercício 18 – Database Empresa

- ▶ Liste os empregados (nome e endereço) que moram em Houston, TX

Data Output			
	pnome character varying(30)	unome character varying(30)	endereco character varying(100)
1	James	Borg	450 Stone, Houston,
2	Franklin	Wong	638 Voss, Houston,
3	John	Smith	731 Fondren, Houst
4	Ahmad	Jabbar	980 Dallas, Houston
5	Joyce	English	5631 Rice, Houston,



## Exercício 19 – Database **Empresa**

- ▶ Liste os empregados (nome e data de nascimento) que fazem aniversário em janeiro.

## IN

- ▶ Permite comparar o valor de uma coluna com um conjunto de valores.
- ▶ Normalmente, utilizamos o IN para substituir uma série de comparações seguidas da cláusula OR.

Consulta 13: Obter o SSN de todos os empregados que trabalham em projetos de números 1, 2, ou 3.

**Q13:**    **SELECT**    **DISTINCT ESSN**  
              **FROM**        **TRABALHA\_EM**  
              **WHERE**      **PNO IN (1, 2, 3)**

A maior utilização do IN é em subquery

## Exercício 20 – Database ImovelNet

- ▶ Liste todas as ofertas cujo imóvel seja 1 ou 2.



## Exercício 21 – Database ImovelNet

- ▶ Liste todos os imóveis cujo código seja 2 ou 3 em ordem alfabética de endereço.



# Uso do NULL em SQL

---

- ➔ SQL permite que a consulta verifique se o valor de um atributo é NULL (ausente ou indefinido ou não se aplica)
- ➔ SQL usa **IS** ou **IS NOT** para comparar NULLs pois considera que cada valor NULL é distinto de outros valores NULL, assim, comparação via igualdade não é apropriado .



# Uso do NULL em SQL



Consulta 14: Obter os nomes de todos os empregados que não possuem supervisores.

# Consultas aninhadas

---

- Uma consulta com SELECTs embutidos ou aninhados é chamada de *consulta aninhada*
- Esse tipo de consulta pode ser especificado dentro da cláusula WHERE de uma outra consulta, chamada de *consulta externa*
- Diversas das consultas anteriores podem ser especificadas de modo alternativo usando aninhamento
- Em geral, é possível haver vários níveis de consultas aninhadas

# SQL sem cláusula Where

---

→ Exemplo:

**Q10:**      **SELECT**      SSN, DNOME  
                 **FROM**        EMPREGADO, DEPARTAMENTO

- Nesse caso, será aplicado o produto cartesiano para obter o resultado



# Q1 – Passo a passo

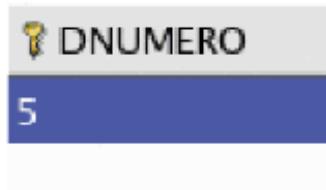


Consulta 1: Obter o nome e endereço de todos os empregados que trabalham no departamento de 'Pesquisa'.

```
Q1:  SELECT PNOME, UNOME, ENDERECO  
      FROM EMPREGADO  
      WHERE DNO IN (SELECT DNUMERO  
                     FROM DEPARTAMENTO  
                     WHERE DNOME='Pesquisa' )
```

**DNUMERO = 5**

# Q1 – Passo a passo

Q1: **SELECT** PNOME, UNOME, ENDERECO  
**FROM** EMPREGADO  
**WHERE** DNO **IN**  )

O operador de comparação **IN** compara o valor **v** com um conjunto de valores **V**, e retorna **VERDADEIRO** caso **v** seja um dos elementos de **V**

# Consultas aninhadas

➡ Consulta 1: Obter o nome e endereço de todos os empregados que trabalham no departamento de 'Pesquisa'.

**Q1:**

Output pane

Data Output Explain Messages History

	pnome character varying(30)	unome character varying(30)	endereco character varying(100)
1	Franklin	Wong	638 Voss, Houston, TX
2	Ramesh	Narayan	975 Fire Oak, Humble, TX
3	John	Smith	731 Fondren, Houston, TX
4	Joyce	English	5631 Rice, Houston, TX

OK.

# Consultas aninhadas correlacionadas

---

- ▶ Caso a condição da cláusula WHERE da *consulta interna* referencie um atributo de uma relação declarada na *consulta externa*, as duas consultas são ditas **correlacionadas**
  
- ▶ O resultado de uma consulta aninhada correlacionada é *diferente para cada tupla (ou combinação de tuplas) da relação(ões) da consulta externa*



# Consultas aninhadas correlacionadas

- Caso a condição da cláusula WHERE da consulta interna referencie um atributo de uma relação declarada na consulta externa, as duas consultas são ditas **correlacionadas**
- O resultado de uma consulta aninhada correlacionada é *diferente para cada tupla (ou combinação de tuplas) da relação(ões) da consulta externa*

No ex. anterior o resultado da consulta interna era fixo. Independende das tuplas da relação da consulta externa

# Consultas aninhadas correlacionadas

---

- Consulta Qc: Obter o nome de cada empregado que possui um dependente com o mesmo sexo do empregado (o responsável ).



# Consultas aninhadas correlacionadas

**EMPREGADO**

PNAME	SSN	SEXO
John	123456789	M
Franklin	333445555	M
Joyce	453453453	F
Ramesh	666884444	M
James	888665555	M
Jennifer	987654321	F
Ahmad	987987987	M
Alicia	999887777	F

**DEPENDENTE**

ESSN	SEXO
123456789	F
123456789	F
123456789	M
333445555	F
333445555	F
333445555	M
987654321	M

**Consultas correlacionadas**, como Qc, geram um resultado diferente no select interno para cada tupla da relação externa.

# Passo a passo

EMPREGADO

PNOME	SSN	SEXO
John	123456789	M
Franklin	333445555	M
Joyce	453453453	F
Ramesh	666884444	M
James	888665555	M
Jennifer	987654321	F
Ahmad	987987987	M
Alicia	999887777	F

DEPEN

ESSN	SEXO
123456789	F
123456789	F
123456789	M
333445555	F
333445555	F
333445555	M
987654321	M

Tuplas da relação interna que satisfazem a condição de junção:  
 $ESSN=E.SSN$   
Para  
 $E.SSN=123456789$



▶ **Consultas correlacionadas**, como Qc, geram um resultado diferente no select interno para cada tupla da relação externa.

# Passo a passo

## EMPREGADO

PNAME	SSN	SEXO
John	123456789	M
Franklin	333445555	M
Joyce	453453453	F
Ramesh	666884444	M
James	888665555	M
Jennifer	987654321	F
Ahmad	987987987	M
Alicia	999887777	F

## DEPENDENTE

ESSN	SEXO
123456789	F
123456789	F
123456789	M
333445555	F
333445555	F
333445555	M
987654321	M

**Consultas correlacionadas, como Qc, geram um resultado diferente no select interno para cada tupla da relação externa.**

# Passo a passo

**EMPREGADO**

PNAME	SSN	SEXO
John	123456789	M
Franklin	333445555	M
Joyce	453453453	F
Ramesh	666884444	M
James	888665555	M
Jennifer	987654321	F
Ahmad	987987987	M
Alicia	999887777	F

**DEPENDENTE**

ESSN	SEXO
123456789	F
123456789	F
123456789	M
333445555	F
333445555	F
333445555	M
987654321	M

**Consultas correlacionadas**, como Qc, geram um resultado diferente no select interno para cada tupla da relação externa.

# Passo a passo

**EMPREGADO**

PNAME	SSN	SEXO
John	123456789	M
Franklin	333445555	M
Joyce	453453453	F
Ramesh	666884444	M
James	888665555	M
Jennifer	987654321	F
Ahmad	987987987	M
Alicia	999887777	F

**DEPENDENTE**

ESSN	SEXO
123456789	F
123456789	F
123456789	M
333445555	F
333445555	F
333445555	M
987654321	M



F

**Consultas correlacionadas**, como Qc, geram um resultado diferente no select interno para cada tupla da relação externa.

# Consultas aninhadas correlacionadas

- ➡ Consulta Qc: Obter o nome de cada empregado que possui um dependente com o mesmo sexo do empregado .
- ➡ Uma consulta escrita com o bloco `SELECT... FROM... WHERE...` aninhado e que usa as operações de comparação = ou **IN** pode ser **sempre** expressa como um único bloco de consulta. Por exemplo, Qc pode ser escrita como Qca:

Nesse caso, atributos da "relação interna" podem fazer parte do resultado.

```
Qca:    SELECT E.PNOME, E.UNOM,  
              NOME_DEPENDENTE  
        FROM EMPREGADO AS E,  
              DEPENDENTE AS D  
   WHERE E.SSN=D.ESSN  
     AND E.SEXO=D.SEXO
```

# Consultas aninhadas correlacionadas

Qca:

```
SELECT E.PNOME, E.UNOME, E.SEXO,  
        NOME_DEPENDENTE  
FROM    EMPREGADO AS E,  
        DEPENDENTE AS D  
WHERE   E.SSN=D.ESSN  
AND      E.SEXO=D.SEXO
```

Output pane

Data Output Explain Messages History

	pnome character varying(30)	unome character varying(30)	sexo character varying(1)	nome_dependente character varying(30)
1	Franklin	Wong	M	Theodore
2	John	Smith	M	Michael



## Consultas aninhadas correlacionadas

---

- ➡ Consulta 12: Obter o nome de cada empregado que possui um dependente com o mesmo primeiro nome do empregado (o responsável).



## A função EXISTS

- ▶ Modifique a consulta interna para saber apenas quais funcionários possuem dependentes:

	pnome character varying(30)	unome character varying(30)
1	Jennifer	Wallace
2	Franklin	Wong
3	John	Smith

## A função EXISTS

- ▶ E para saber quem não tem dependentes?

	pnome character varying(30)	unome character varying(30)
1	James	Borg
2	Ramesh	Narayan
3	Alicia	Zelaya
4	Ahmad	Jabbar
5	Joyce	English

## A função EXISTS

- EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio (não contém nenhuma tupla) ou não
- A consulta Q6 obtém o nome dos empregados que não tem dependentes com NOT EXISTS

```
Q6: SELECT E.PNOME, E.UNOME
      FROM EMPREGADO
      WHERE NOT EXISTS (SELECT *
                        FROM DEPENDENTE
                        WHERE ESSN=SSN )
```

## A função EXISTS

- EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio (não contém nenhuma tupla) ou não
- A consulta Q6 obtém o nome dos empregados dependentes com NOT EXISTS

```
Q6: SELECT E.PNOME, E.UNOM  
      FROM EMPREGADO  
      WHERE NOT EXISTS (SELECT *  
                        FROM DEPENDENTE  
                        WHERE ESSN=SSN )
```

Em Q6, a consulta aninhada correlacionada obtém todas as tuplas de DEPENDENTE relacionadas à tupla corrente de EMPREGADO. Se *não existe nenhuma*, a tupla de EMPREGADO é selecionada.

A função EXISTS é fundamental para o poder de expressão da SQL

# Junções

- A relação operando da junção pode ser especificada na cláusula FROM
- Parece com qualquer outra relação, mas é o resultado de uma junção

```
C1A: SELECT Pnome, Unome, Endereco  
        FROM   (FUNCIONARIO JOIN  
                  DEPARTAMENTO  
                ON Dnr=Dnumero)  
        WHERE  Dnome='Pesquisa';
```

# Junções

---

## ▶ Tipos

- ▶ Theta join permite comparações arbitrárias de relacionamentos como  $=, \geq, \leq, >, <$ .
- ▶ Equijoin é um theta join usando somente o operador de igualdade.
- ▶ Natural join é um equijoin em atributos que possuem o mesmo nome na relações usadas na junção.
  - ▶ Uma junção natural remove colunas duplicadas envolvidas na comparação.
    - Somente uma das colunas comparadas permanece no resultado da junção.

# Variações em junções

```
SELECT *
FROM DEPT_LOCALIZACOES
NATURAL JOIN DEPARTAMENTO
```

A condição de junção  
implícita é  
**DNUMERO=DNUMERO**

	dnumero	dlocalizacao	dnome	gerssn	gerdatainicio
	integer	character varying(30)	character varying(30)	integer	date
1	1	Houston	Sede administrativa	888665555	1981-06-19
2	4	Staffor	Administração	987654321	1995-01-01
3	5	Bellaire	Pesquisa	333445555	1988-05-22
4	5	Sugarland	Pesquisa	333445555	1988-05-22

# Variações em junções

```
SELECT DNOME, DLOCALIZACAO,
DEPT_LOCALIZACOES, DNUMERO
FROM DEPT_LOCALIZACOES
NATURAL JOIN DEPARTAMENTO
```

	dnome character varying(30)	dlocalizacao character varying(30)	dept_localizacoes dept_localizacoes	dnumero integer
1	Sede administrativa	Houston	(1,Houston)	1
2	Administração	Staffor	(4,Staffor)	4
3	Pesquisa	Bellaire	(5,Bellaire)	5
4	Pesquisa	Sugarland	(5,Sugarland)	5



# Variações em junções

```
SELECT DNOME, DLOCALIZACAO,  
DEPT_LOCALIZACOES, DNUMERO  
FROM DEPT_LOCALIZACOES  
NATURAL JOIN DEPARTAMENTO  
WHERE DNOME = 'Pesquisa'
```

dnome	dlocalizacao	dept_localizacoes	dnumero
character varying(30)	character varying(30)	dept_localizacoes	integer
Pesquisa	Bellaire	(5, Bellaire)	5
Pesquisa	Sugarland	(5, Sugarland)	5



# Variações em junções

---

- ▶ NATURAL JOIN //sem condições, atributos iguais
- ▶ JOIN, INNER JOIN //com condições, junção interna (default)
- ▶ OUTER JOIN // com condições – junção externa
  - ▶ LEFT OUTER JOIN
  - ▶ RIGHT OUTER JOIN
  - ▶ FULL OUTER JOIN

## Junções – inner join

- ▶ Tipo padrão de junção em uma tabela de junção.
- ▶ A tupla é incluída no resultado somente se uma tupla correspondente existir na outra relação.

```
select pnome, dnome  
from funcionario  
      inner join departamento  
    on dnr=dnumero;
```

Explícito

```
select pnome, dnome  
from funcionario, departamento  
  where dnr=dnumero;
```

Implícito

## Junções – LEFT OUTER JOIN

- ▶ Toda tupla na tabela esquerda deve aparecer no resultado.
- ▶ Se não houver tupla correspondente:
  - ▶ Preenchido com valores NULL para atributos da tabela da direita.

```
select pnome, dnome
from funcionario
left outer join departamento
on dnr=dnumero;
```

## Junções – RIGHT OUTER JOIN

- ▶ Toda tupla na tabela direita deve aparecer no resultado.
  - ▶ Se não houver tupla correspondente:
    - ▶ Preenchido com valores NULL para atributos da tabela da esquerda.

```
select pnome, dnome
from funcionario
right outer join departamento
on dnr=dnumero;
```

## Junções – FULL OUTER JOIN

---

- ▶ Toda tupla nas tabelas direita e esquerda deve aparecer no resultado.

```
select pnome, dnome  
from funcionario  
      full outer join departamento  
    on dnr=dnumero;
```

## Junções – exemplos

---

- ▶ `select pnome, nome_dependente from empregado, dependente where ssn=essn`
- ▶ `select pnome, nome_dependente from empregado inner join dependente on ssn=essn`
- ▶ `select pnome, nome_dependente from empregado left outer join dependente on ssn=essn`
- ▶ `select pnome, nome_dependente from empregado right outer join dependente on ssn=essn`
- ▶ `select pnome, nome_dependente from empregado full outer join dependente on ssn=essn`

# Operações Aritméticas

---



Consulta 27: Mostre o efeito de dar a todos os empregados que trabalham no projeto 'ProdutoX' um aumento de 10%.



# Operações combinadas - resultados

---

```
select pnome, unome, salario  
from empregado, trabalha_em, projeto  
where ssn=essn and pno=pnumero and pjnome='ProdutoX'
```

	<b>pnome</b> <b>character varying(30)</b>	<b>unome</b> <b>character varying(30)</b>	<b>salario</b> <b>double precision</b>
<b>1</b>	John	Smith	30000
<b>2</b>	Joyce	English	25000



# Operações combinadas - resultados

```
select pnome, unome, 1.1*salario  
from empregado, trabalha_em, projeto  
where ssn=essn and pno=pnumero and pjnome='ProdutoX'
```

	pnome character varying(30)	unome character varying(30)	?column? double precision
1	John	Smith	33000
2	Joyce	English	27500



# Operações combinadas - variações

```
select pnome, unome, 1.1*salario  
from empregado, trabalha_em, projeto  
where ssn=essn and pno=pnumero and pjnome like 'Produto%'
```

	pnome character varying(30)	unome character varying(30)	?column? double precision
1	John	Smith	33000
2	John	Smith	33000
3	Ramesh	Narayan	41800
4	Joyce	English	27500
5	Joyce	English	27500
6	Franklin	Wong	44000
7	Franklin	Wong	44000



# Operações combinadas - variações

```
select pnome, unome, 1.1*salario  
from empregado, trabalha_em, projeto  
where ssn=essn and pno=pnumero and pjnome like 'Produto%'  
order by salario
```

	pnome character varying(30)	unome character varying(30)	?column? double precision
1	Joyce	English	27500
2	Joyce	English	27500
3	John	Smith	33000
4	John	Smith	33000
5	Ramesh	Narayan	41800
6	Franklin	Wong	44000
7	Franklin	Wong	44000

# Operações combinadas - variações

```
select pnome, unome, 1.1*salario  
from empregado, trabalha_em, projeto  
where ssn=essn and pno=pnumero and pjnome like 'Produto%'  
order by pjnome, salario
```

	pnome character varying(30)	unome character varying(30)	?column? double precision
1	Joyce	English	27500
2	John	Smith	33000
3	Joyce	English	27500
4	John	Smith	33000
5	Franklin	Wong	44000
6	Ramesh	Narayan	41800
7	Franklin	Wong	44000



# Bibliografias utilizada nesta aula

---

- ▶ ELMASRI, R.; NAVATHE, S. B. Sistemas de banco de dados. 6 ed. Pearson/Addison-Wesley, 2011. ISBN: 9788579360855
- ▶ SILBERSCHATZ,A.; SUDARSHAN, S. Sistema de banco de dados. Campus, 2006. ISBN: 9788535211078
- ▶ OLIVEIRA, C.H. SQL: Curso prático. Novatec, 2002. ISBN: 9788575220245
- ▶ DATE, C. J. Introdução a Sistemas de Bancos de Dados. 8<sup>a</sup> ed. Rio de Janeiro, Campus, 2004.
- ▶ CARDOSO, Virgínia M; CARDOSO, Giselle C. Sistemas de Banco de Dados: uma Abordagem Introdutória e Aplicada. São Paulo: Saraiva, 2012