



**CEFET/RJ**

# SQL

Eduardo Ogasawara  
eogasawara@ieee.org  
<https://eic.cefet-rj.br/~eogasawara>

# *Structured Query Language (SQL)*

- A Linguagem de Consulta Estruturada (SQL) é a linguagem de consulta declarativa padrão para banco de dados relacional
- Muitas das características originais do SQL foram inspiradas na álgebra relacional
- A linguagem SQL é dividida em subconjuntos de acordo com as operações que queremos efetuar sobre um banco de dados, tais como:
  - DDL - Linguagem de Definição de Dados
    - DDL é um subconjunto da linguagem SQL que permite definir tabelas e elementos associados
  - DML - Linguagem de Manipulação de Dados
    - DML é um subconjunto da linguagem SQL que é utilizado para realizar inclusões, consultas, alterações e exclusões de dados nas tabelas

## *Linguagem de definição de dados (DDL)*

## ***DDL: Linguagem de definição de dados***

- Permite a especificação dos conjuntos de relações e informações associadas, incluindo:
  - O esquema para cada relação
  - O domínio dos valores associados a cada atributo
  - As restrições de integridade
  - O conjunto dos índices a serem mantidos para cada relação
  - As informações de segurança e autorização para cada relação
  - A estrutura de armazenamento físico de cada relação no disco

## *Tipos básicos de domínio na SQL*

- **char(n)**
  - String de caracteres de tamanho fixo com tamanho n especificado pelo usuário
- **varchar(n)**
  - String de caracteres de tamanho variável com tamanho n máximo especificado pelo usuário
- **Int**
  - Inteiro (um subconjunto finito de inteiros que é dependente da máquina)
- **float, double**
  - Números de ponto flutuante e ponto flutuante de precisão dupla com precisão dependente da máquina
- Existem domínios específicos para campos longos textuais e binários que variam de banco para banco
  - Dia-hora: timestamp, date, time
  - Textuais: Text, Clob
  - Binários: Image, Blob

# SQL: Esquema e catálogo

## Esquema SQL

- Identificado por um nome de esquema
- Inclui um identificador de autorização e descritores para cada elemento

## Esquema de elementos incluem

- Tabelas, restrições, views, domínios e outras construções

Cada instrução em SQL termina com um ponto e vírgula

# SQL: Esquema e catálogo

## Instruções CREATE

- Principal comando SQL para a definição de dados

## Instrução CREATE SCHEMA

- CREATE SCHEMA EMPRESA;

## Catálogo

- Coleção nomeada de esquemas em um ambiente SQL

## Ambiente SQL

- Instalação de um SGBDR compatível com SQL em um sistema de computador

# SQL: CREATE TABLE

## Especificar nova relação:

- Dar um nome
- Especificar atributos e restrições iniciais

## Especificar o esquema:

- CREATE TABLE EMPRESA.FUNCIONARIO ...

## Ou usar esquema atual:

- CREATE TABLE FUNCIONARIO ...

## Tabelas da base (relações da base)

- A relação e suas tuplas são realmente criadas e armazenadas como um arquivo pelo SGBD

## *Esquema de projetos usado nos exemplos*

- departamento(dnome, dnumero, gerssn, gerdatainicio)
- dependente(essn, nome\_dependente, sexo, datanasc, parentesco)
- dept\_localizacoes(dnumero, dlocalizacao)
- empregado(pnome, minicial, unome, ssn, datanasc, endereco, sexo, salario, superssn, dno)
- projeto(pjnome, pnumero, plocalizacao, dnum)
- trabalha\_em(essn, pno, horas)

# Criação da tabela de empregados

```
CREATE TABLE EMPREGADO (  
    PNOOME VARCHAR(30),  
    MINICIAL VARCHAR(1),  
    UNOME VARCHAR(30),  
    CPF TIPO_CPF NOT NULL,  
    DATANASC TIMESTAMP,  
    ENDERECO TEXT,  
    SEXO VARCHAR(1),  
    SALARIO FLOAT,  
    GERENTE_CPF TIPO_CPF,  
    DNO INT,  
    PRIMARY KEY (CPF),  
    FOREIGN KEY (GERENTE_CPF) REFERENCES EMPREGADO (CPF),  
    FOREIGN KEY (DNO) REFERENCES DEPARTAMENTO (DNUMERO)  
);
```

# Criação da tabela de departamentos

```
CREATE DOMAIN TIPO_CPF CHAR(11);
```

```
CREATE TABLE DEPARTAMENTO (  
  DNOME VARCHAR(30),  
  DNUMERO INT NOT NULL,  
  GERCPF TIPO_CPF,  
  GERDATAINICIO TIMESTAMP,  
  PRIMARY KEY (DNUMERO)  
);
```

# SQL: Criação da tabela de localização dos departamentos

```
CREATE TABLE DEPT_LOCALIZACOES (  
    DNUMERO INT NOT NULL,  
    DLOCALIZACAO VARCHAR(30),  
    PRIMARY KEY (DNUMERO,DLOCALIZACAO),  
    FOREIGN KEY (DNUMERO) REFERENCES DEPARTAMENTO (DNUMERO)  
);
```

## Criação da tabela de projetos

```
CREATE TABLE PROJETO (  
  PJNOME VARCHAR(30),  
  PNUMERO INT NOT NULL,  
  PLOCALIZACAO VARCHAR(30),  
  DNUM INT,  
  PRIMARY KEY (PNUMERO),  
  UNIQUE(PJNOME),  
  FOREIGN KEY (DNUM) REFERENCES DEPARTAMENTO (DNUMERO)  
);
```

## Criação da tabela de trabalha em

```
CREATE TABLE TRABALHA_EM (  
    ECPF TIPO_CPF NOT NULL,  
    PNO INT NOT NULL,  
    HORAS FLOAT,  
    PRIMARY KEY (ECPF,PNO),  
    FOREIGN KEY (ECPF) REFERENCES EMPREGADO (CPF),  
    FOREIGN KEY (PNO) REFERENCES PROJETO (PNUMERO)  
);
```

## Criação da tabela de dependentes

```
CREATE TABLE DEPENDENTE (  
  ECPF TIPO_CPF NOT NULL,  
  NOME_DEPENDENTE VARCHAR(30) NOT NULL,  
  SEXO VARCHAR(1),  
  DATANASC TIMESTAMP,  
  PARENTESCO VARCHAR(30),  
  PRIMARY KEY (ECPF, NOME_DEPENDENTE),  
  FOREIGN KEY (ECPF) REFERENCES EMPREGADO (CPF)  
);
```

## *Referências circulares nas chaves estrangeiras*

### Problemas com chaves estrangeiras:

- Observe que algumas chaves estrangeiras podem causar erros:
  - Referências circulares
  - Ou porque dizem respeito a uma tabela que ainda não foi criada

### Solução

- Uma das soluções é utilizar um comando de alterações de esquema para colocar as restrições após a criação de todas as tabelas

## *Tipos de dados e domínios*

### Tipos de dados numérico:

- Incluem números: INTEGER ou INT e SMALLINT
- Números de ponto flutuante (reais): FLOAT ou REAL e DOUBLE PRECISION

### Tipos de dados de cadeia de caracteres:

- Tamanho fixo: CHAR(n) ou CHARACTER(n)
- Tamanho variável: VARCHAR(n) ou CHAR VARYING(n) ou CHARACTER VARYING(n)

## SQL: Tipos de dados e domínios

### Cadeia de bits:

- Tamanho fixo: BIT(n)
- Tamanho variável: BIT VARYING(n)

### Booleano:

- Valores TRUE ou FALSE ou NULL

### DATE:

- Dez posições
- Componentes são DAY, MONTH e YEAR na forma DD-MM-YYYY

## SQL: Tipos de dados e domínios

### Timestamp (TIMESTAMP):

- Inclui os campos DATE e TIME
- Mais um mínimo de seis posições para frações decimais de segundos
- Qualificador opcional WITH TIME ZONE

### INTERVAL:

- Especifica valor relativo que pode ser usado para incrementar ou decrementar um valor absoluto de uma data, hora ou timestamp

## SQL: Tipos de dados e domínios

### Domínio:

- Nome usado com a especificação de atributo
- Torna mais fácil mudar o tipo de dado para um domínio que é usado por diversos atributos
- Melhora a legibilidade do esquema

### Exemplo:

- `CREATE DOMAIN TIPO_CPF AS CHAR(11);`

## ***SQL: Especificando restrições***

- Restrições básicas:
  - Restrições de chave e integridade referencial
  - Restrições sobre domínios de atributo e NULLs
  - Restrições sobre tuplas individuais dentro de uma relação

## SQL: Restrições e valores padrão de atributos

NOT NULL

- NULL não é permitido para determinado atributo

Valor padrão

- DEFAULT <valor>

cláusula CHECK

Dnumero INT NOT NULL CHECK (Dnumero > 0 AND  
Dnumero < 21);

# SQL: *Relações virtuais*

## Relações virtuais

- Criadas por meio da instrução CREATE VIEW

# SQL: Restrições de chave e integridade referencial

## Cláusula PRIMARY KEY

- Especifica um ou mais atributos que compõem a chave primária de uma relação
- Dnumero INT PRIMARY KEY;

## Cláusula UNIQUE

- Especifica chaves alternativas (candidatas)
- Dnome VARCHAR(15) UNIQUE;

# SQL: Restrições de chave e integridade referencial

## Cláusula FOREIGN KEY

- Ação default: rejeita atualização sobre violação
- Conectado à cláusula de ação de disparo referencial
  - Opções incluem SET NULL, CASCADE e SET DEFAULT
  - Ação tomada pelo SGBD para SET NULL ou SET DEFAULT é a mesma para ON DELETE e ON UPDATE
  - Opção CASCADE adequada para relações de 'parentesco'

## SQL: Nomeando restrições (Constraint)

### Palavra-chave CONSTRAINT

- Nome de restrição
- Útil para alterações posteriores

### Exemplo:

- `CONSTRAINT PAI FOREIGN KEY (PARENT_KEY) REFERENCES Pessoa(CPF);`

# SQL: Especificando restrições sobre tuplas usando CHECK

## Cláusula CHECK ao final de uma instrução CREATE TABLE

- Aplicam a cada tupla individualmente
- CHECK (Dep\_data\_criacao <= Data\_inicio\_gerente);

## Constraint

- No exemplo acima podemos utilizar uma CONSTRAINT para referenciar a restrição:
- CONSTRAINT CRIACAO\_DEPTO CHECK (Dep\_data\_criacao <= Data\_inicio\_gerente);

## *SQL: Instruções de alteração de esquema*

- Comandos de evolução de esquema:
  - Pode ser feito enquanto o banco de dados está operando
  - Não exige recompilação do esquema

# SQL: Comando DROP

## Comando DROP

- Usado para remover elementos nomeados do esquema, como tabelas, domínios ou restrições

## Opções de comportamento de drop:

- CASCADE e RESTRICT

## Exemplo:

- DROP SCHEMA EMPRESA CASCADE;

# SQL: Comando ALTER

Ações de alteração de tabela incluem:

- Acrescentar ou remover uma coluna (atributo)
- Alterar uma definição de coluna
- Acrescentar ou remover restrições de tabela

Exemplo:

- ALTER TABLE EMPRESA.FUNCIONARIO ADD COLUMN Tarefa VARCHAR(12);

Para remover uma coluna

- Escolher CASCADE ou RESTRICT

## SQL: Comando ALTER

### Alterar as restrições

- Acrescentar ou remover uma restrição nomeada
- ALTER TABLE EMPRESA.FUNCIONARIO DROP CONSTRAINT CHESUPERFUNC CASCADE

# PostgreSQL

- Projeto iniciado na Universidade de Berkeley, Califórnia (1986)
- Atualmente na versão 13.0 (estável)
- Homepage: <http://www.postgresql.org/>
- Disponível para diversos SO
  - Windows, Linux, Mac, ...
- Suporte por uma comunidade ativa
- Baixa necessidade de manutenção
- Confiável, estável e extensível (código aberto)
- Bom desempenho com grande volume de dados
- Documentação: Boa

## *PostgreSQL – Instalação*

- Servidor
  - <http://www.postgresql.org/download/>
  - Instalador compactado
- Cliente e Gerenciador – PGAdmin
  - <http://www.pgadmin.org/download/>
- Driver JDBC e .Net
  - Java: <http://jdbc.postgresql.org/>
  - .Net: <http://www.debart.com/dotconnect/postgresql/>

# Postgresql – Interface de exploração de base de dados

The image shows the pgAdmin 4 interface for creating a new server. The 'Create - Server' dialog box is open, and the 'Connection' tab is selected. The fields are filled with the following values:

Field	Value
Host name/address	regulus.eic.cefet-rj.br
Port	5432
Maintenance database	postgres
Username	postgres
Kerberos authentication?	<input type="checkbox"/>
Password	.....
Save password?	<input checked="" type="checkbox"/>
Role	

A blue callout box points to the 'Host name/address' field with the text: "Nome da conexão (lâmina general) Servidor, Porta, Usuário e Senha (lâmina connection)".

# Postgresql – Interface de exploração de base de dados

The screenshot displays the pgAdmin 4 interface. On the left, the 'Servers' tree is expanded to show the 'regulus.eic.cefet-rj.br (bd)' server, with the 'Databases (22)' folder expanded to show the 'bd' database, and the 'Schemas (2)' folder expanded to show the 'projetos' schema. A blue callout box points to the 'projetos' schema with the text: "Esquema da base de dados da empresa: projetos".

The main area shows several performance metrics for the 'projetos' schema:

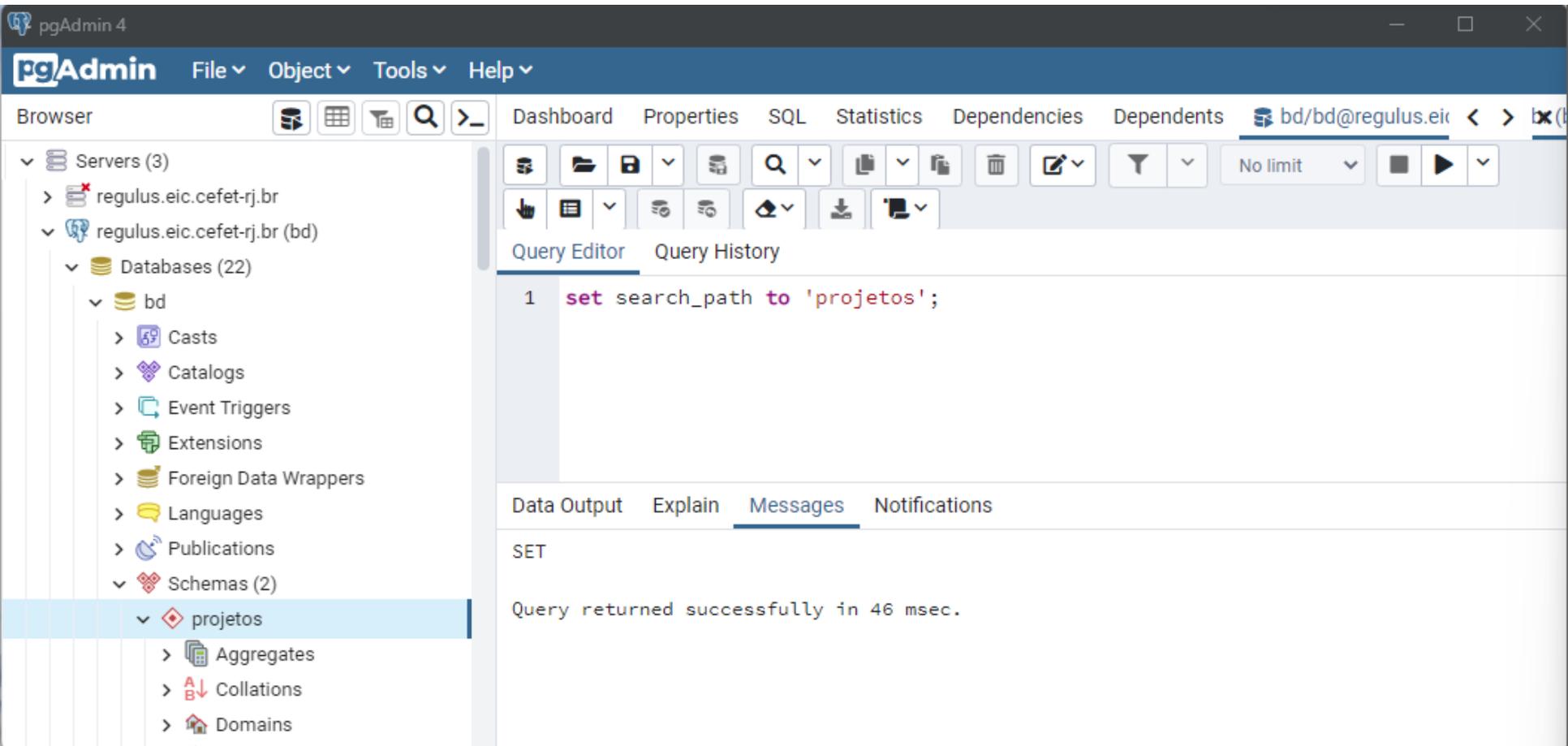
- Database sessions:** A line chart showing Total (cyan), Active (green), and Idle (red) sessions. The Y-axis ranges from 0 to 3. Total sessions are at 3, Active at 1, and Idle at 1.
- Transactions per second:** A line chart showing Transactions (cyan), Commits (green), and Rollbacks (red) per second. The Y-axis ranges from 0 to 7. There are two peaks: one at approximately 3 transactions per second and another at approximately 7 transactions per second.
- Tuples in:** A bar chart showing Inserts (cyan), Updates (green), and Deletes (red). The Y-axis ranges from 0 to 1. There is one bar at 1.
- Tuples out:** A bar chart showing Fetched (cyan) and Returned (green) tuples. The Y-axis ranges from 0 to 2500. There are two bars: one at approximately 1200 and another at approximately 2300.
- Block I/O:** A bar chart showing Reads (cyan) and Hits (green). The Y-axis ranges from 0 to 1400. There are two bars: one at approximately 100 and another at approximately 1200.

# Postgresql – abertura da janela de comandos SQL

The image shows the pgAdmin 4 interface. The 'Tools' menu is open, and 'Query Tool' is highlighted. A blue callout box points to this option with the text 'Abertura da janela de consulta para a base selecionada'. The interface also displays a tree view of databases, including 'regulus.eic.cefet-rj.br' and 'bd', and several performance monitoring charts such as 'Transactions per second', 'Tuples out', and 'Block I/O'.

Lembre-se do comando `set search_path to 'projetos';`

# Postgresql – abertura da janela de comandos SQL



The screenshot displays the pgAdmin 4 interface. On the left, the 'Servers' tree is expanded to show the 'projetos' schema under the 'bd' database. The main area is the 'Query Editor', which contains the SQL command: `1 set search_path to 'projetos';`. Below the editor, the 'Messages' pane shows the output: `SET` and `Query returned successfully in 46 msec.`

set search\_path to 'projetos';

## ***Exercícios***

- Acessar Postgresql em [regulus.eic.cefet-rj.br](http://regulus.eic.cefet-rj.br) usando o PGAdmin
- Obter o usuário e senha com o professor
- Criar as tabelas usando a script `SQL-CreateTables.sql`
  - Estude a script antes de executá-la
- Popular as tabelas usando a script `SQL-Inserts.sql`
  - Estude a script antes de executá-la

## *Linguagem de definição de consulta (DML)*

# Postgresql – consultas

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree is expanded to show the 'bd' database. The 'Query Editor' tab is active, displaying the following SQL query:

```
1 SELECT E.PNOME, E.UNOME, E.DATANASC, E.ENDERECO
2 FROM EMPREGADO E
3
```

Below the query editor, the 'Data Output' tab is selected, showing the results of the query in a table format:

	pnome character varying (30)	unome character varying (30)	datanasc timestamp without time zone	endereco text
1	James	Borg	1937-11-10 00:00:00	450 Stone, Houston, TX
2	Franklin	Wong	1955-12-08 00:00:00	638 Voss, Houston, TX
3	Ramesh	Narayan	1962-09-15 00:00:00	975 Fire Oak, Humble, TX
4	John	Smith	1965-01-09 00:00:00	731 Fondren, Houston, TX

Two blue callout boxes with white text are overlaid on the image. The first callout, labeled 'Janela de consultas SQL', points to the SQL query editor. The second callout, labeled 'Janela de resultados', points to the 'Data Output' table.

## *Mapeamento do select em algebra relacional*

- A cláusula select lista os atributos desejados no resultado de uma consulta
  - corresponde à operação projeção da álgebra relacional

- Exemplo:

Encontre os nomes de todas as agências na relação empréstimo

```
SELECT E.PNOME  
FROM EMPREGADO E
```

- Na álgebra relacional, a consulta seria:

$\pi_{PNOME}(EMPREGADO)$

# Obtenha nome, nascimento e endereço dos empregados

The screenshot shows the pgAdmin 4 interface. On the left, the server tree is expanded to show the 'bd' database. The central pane displays a SQL query in the Query Editor:

```
1 SELECT E.PNOME, E.UNOME, E.DATANASC, E.ENDERECO
2 FROM EMPREGADO E
3
```

A blue callout box points to the query with the text: "Projeção de alguns atributos de EMPREGADO".

Below the query editor, the 'Data Output' tab is active, showing the results of the query in a table:

	pnome character varying (30)	unome character varying (30)	datanasc timestamp without time zone	endereco text
1	James	Borg	1937-11-10 00:00:00	450 Stone, Houston, TX
2	Franklin	Wong	1955-12-08 00:00:00	638 Voss, Houston, TX
3	Ramesh	Narayan	1962-09-15 00:00:00	975 Fire Oak, Humble, TX
4	John	Smith	1965-01-09 00:00:00	731 Fondren, Houston, TX

## *A cláusula select all versus select distinct*

- A SQL permite duplicatas nas relações bem como nos resultados de consulta
- Para forçar a eliminação de duplicatas, insira a palavra-chave distinct após select
- Encontre os nomes de todas as agências na relação empréstimo e remova as duplicatas
  - `SELECT DISTINCT E.PNOME  
FROM EMPREGADO E`
- A palavra-chave ALL especifica que as duplicatas não são removidas (esse é o comportamento default)
- `SELECT ALL E.PNOME  
FROM EMPREGADO E`

## *A cláusula select (projeção generalizada)*

- A cláusula select pode conter expressões aritméticas envolvendo os operadores +, -, \*, / e funções operando em constantes ou atributos de tuplas

- A consulta:

```
SELECT E.PNOME, 1.05*E.SALARIO  
FROM EMPREGADO E
```

- retorna uma relação que traz o número de empréstimo e quantia em centavos

$\pi_{PNOME,1.05 \cdot SALARIO}(EMPREGADO)$

## *A operação de renomeação*

- A SQL permite renomear relações e atributos usando a cláusula as:
- nome-antigo as nome-novo
- Na consulta anterior,  $1.05 * E.SALARIO$  fica sem nome, para colocar o nome a coluna, usa-se o comando AS
  - ```
SELECT E.PNOME, 1.05 * E.SALARIO AS NOVOSALARIO  
FROM EMPREGADO E
```

## *A cláusula where*

- A cláusula where especificam condições que o resultado precisa satisfazer
  - Corresponde ao predicado de seleção da álgebra relacional
- Para encontrar todos os funcionários com salário superior a US\$1200:  
SELECT E.PNOME, E.SALARIO AS NOVOSALARIO  
FROM EMPREGADO E  
WHERE E.SALARIO > 1200
- Os resultados da comparação podem ser combinados usando os conectivos lógicos AND, OR, e NOT.
- As comparações podem ser aplicadas aos resultados das expressões aritméticas

# Obtenha a data de nascimento e endereço do John Smith

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'projetos' schema selected. The main window shows a SQL query in the Query Editor:

```
1 SELECT E.DATANASC, E.ENDERECO
2 FROM EMPREGADO E
3 WHERE E.PNOME = 'John'
4 AND E.MINICIAL = 'B'
5 AND E.UNOME = 'Smith';
6
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table:

|   | datanasc                    | endereco                 |
|---|-----------------------------|--------------------------|
|   | timestamp without time zone | text                     |
| 1 | 1965-01-09 00:00:00         | 731 Fondren, Houston, TX |

```
SELECT E.DATANASC, E.ENDERECO
FROM EMPREGADO E
WHERE E.PNOME = 'John' AND E.MINICIAL = 'B' AND E.UNOME = 'Smith';
```

# Selecionar os projetos localizados em Stafford

The screenshot shows the pgAdmin 4 interface. On the left, the 'Columns (10)' for the 'empregado' table are listed: pnome, inicial, unome, cpf, datanasc, endereco, sexo, salario, gerente\_cpf, and dno. The 'Query Editor' contains the following SQL query:

```
1 SELECT P.*
2 FROM PROJETO P
3 WHERE P.PLOCALIZACAO = 'Stafford';
4
5
6
7
```

The 'Data Output' tab shows the results of the query in a table:

|   | pjnome           | pnumero | plocalizacao | dnum |
|---|------------------|---------|--------------|------|
| 1 | Automatização    | 10      | Stafford     | 4    |
| 2 | Novos Benefícios | 30      | Stafford     | 4    |

```
SELECT P.*
FROM PROJETO P
WHERE P.PLOCALIZACAO = 'Stafford';
```

# Selecione todos os atributos do departamento de pesquisa

- O asterisco representa todos os atributos da tabela
- Evite usar select \* numa aplicação
  - Traga apenas os campos necessários

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'projetos' schema under the 'bd' database selected. The main window shows the Query Editor with the following SQL query:

```
1 SELECT *
2 FROM DEPARTAMENTO D
3 WHERE D.DNOME = 'Pesquisa';
4
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format:

|   | dnome<br>character varying (30) | dnumero<br>[PK] integer | gercpf<br>character (11) | gerdatainicio<br>timestamp without time zone |
|---|---------------------------------|-------------------------|--------------------------|----------------------------------------------|
| 1 | Pesquisa                        | 5                       | 333445555                | 1988-05-22 00:00:00                          |

```
SELECT *
FROM DEPARTAMENTO D
WHERE D.DNOME = 'Pesquisa';
```

## Consulta sem cláusula de seleção

- SQL sem a cláusula de seleção indica que não há condição de seleção; assim, todas as tuplas da relação da cláusula FROM são selecionadas

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'Columns (10)' of the 'EMPREGADO' table selected. The main window shows the 'Query Editor' with the following SQL query:

```
1 SELECT E.CPF FROM EMPREGADO E
2
3
4
5
6
7
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table:

|   | cpf<br>[PK] character (11) |
|---|----------------------------|
| 1 | 888665555                  |
| 2 | 333445555                  |
| 3 | 666884444                  |
| 4 | 123456789                  |

SELECT E.CPF FROM EMPREGADO E

# A cláusula where (between)

- A SQL inclui um operador de comparação between
- Exemplo: Encontre o nome dos empregados com salário entre US\$ 1.000 e US\$ 2.000

The screenshot shows the pgAdmin 4 interface. On the left, the 'Tables (6)' folder is expanded to show the 'departamento' table, with its 'Columns (4)' sub-folder selected. The columns listed are 'dnome', 'dnumero', 'gercpf', and 'gerdatainicio'. The 'empregado' table is also visible in the tree view.

The 'Query Editor' tab is active, displaying the following SQL query:

```
1 SELECT E.PNOME
2 FROM EMPREGADO E
3 WHERE E.SALARIO BETWEEN 1000 AND 2000
4
```

Below the query editor, the 'Data Output' tab is selected, showing the result of the query. The output is a single row with the column name 'pnome' and its data type 'character varying (30)'. The value in the row is empty.

A green notification box at the bottom right of the interface states: 'Successfully run. Total query runtime: 653 msec. 0 rows af'.

```
SELECT E.PNOME
FROM EMPREGADO E
WHERE E.SALARIO BETWEEN 1000 AND 2000
```

## *Uso de NULL*

- 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
- Consulta 14: Obter os nomes de todos os empregados quem não possuem supervisores.
- Q14: 

```
SELECT PNOOME, UNOME  
FROM EMPREGADO  
WHERE SUPERSSN IS NULL
```
- Nota: Quando há atributos, numa condição de junção, que possuem valor NULL, as tuplas desses valores não são incluídas no resultado da junção

## A cláusula from

- A cláusula from lista as relações envolvidas na consulta
  - Corresponde à operação de produto cartesiano da álgebra relacional
- Encontre o produto cartesiano empregado e dependente  
SELECT \*  
FROM EMPREGADO E, DEPENDENTE D
- Encontre o nome dos empregados com os seus respectivos dependentes  
SELECT E.PNOME, D.NOME\_DEPENDENTE  
FROM EMPREGADO E, DEPENDENTE D  
WHERE E.CPF = D.ECPF
- Junção
  - As tabelas comumente devem ser interligadas.
  - Se há n tabelas, deve-se ter n-1 cláusulas de junção

Cláusula de junção

# Produto Cartesiano

- Não há clausula de junção entre as tabelas.
- Dificilmente quer-se executar um produto cartesiano, ou seja, normalmente isto representa erro grave de consulta SQL
- As tabelas comumente devem ser interligadas. Se há n tabelas, deve-se ter n-1 cláusulas de junção

The screenshot shows the pgAdmin 4 interface. On the left, the 'empregado' table is expanded to show its columns: pnome, inicial, unome, cpf, datanasc, endereco, sexo, salario, gerente\_cpf, and dno. The 'Query Editor' is active, showing the following SQL query:

```
1 SELECT E.CPF, D.DNOME
2 FROM EMPREGADO E, DEPARTAMENTO D
3
4
5
6
7
```

Below the query editor, the 'Data Output' tab is selected, displaying the results of the Cartesian product query:

|   | cpf<br>character (11) | dnome<br>character varying (30) |
|---|-----------------------|---------------------------------|
| 1 | 888665555             | Sede administrativa             |
| 2 | 333445555             | Sede administrativa             |
| 3 | 666884444             | Sede administrativa             |
| 4 | 123456789             | Sede administrativa             |

```
SELECT E.CPF, D.DNOME
FROM EMPREGADO E, DEPARTAMENTO D
```

# Selecione empregados do departamento de pesquisa

The screenshot shows the pgAdmin 4 interface. On the left, the 'empregado' table is selected, and its columns are listed: pnome, minicial, unome, cpf, datanasc, endereco, sexo, salario, gerente\_cpf, and dno. The 'Columns (10)' folder is expanded. In the center, the Query Editor shows the following SQL query:

```
1 SELECT E.PNOME, E.UNOME, E.CPF, E.ENDereco, E.DNO
2 FROM EMPREGADO E
3 WHERE E.DNO = 5
4
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format:

|   | pnome                  | unome                  | cpf                 | endereco                 | dno     |
|---|------------------------|------------------------|---------------------|--------------------------|---------|
|   | character varying (30) | character varying (30) | [PK] character (11) | text                     | integer |
| 1 | Franklin               | Wong                   | 333445555           | 638 Voss, Houston, TX    | 5       |
| 2 | Ramesh                 | Narayan                | 666884444           | 975 Fire Oak, Humble, TX | 5       |
| 3 | John                   | Smith                  | 123456789           | 731 Fondren, Houston, TX | 5       |
| 4 | Joyce                  | English                | 453453453           | 5631 Rice, Houston, TX   | 5       |

```
SELECT E.PNOME, E.UNOME, E.CPF, E.ENDereco, E.DNO
FROM EMPREGADO E
WHERE E.DNO = 5
```

# Selecione empregados do departamento de pesquisa

The screenshot shows the pgAdmin 4 interface. On the left, the 'empregado' table is selected, and its columns are listed. The 'Query Editor' tab is active, displaying the following SQL query:

```
1 SELECT P.NOME, U.NOME, ENDERECO
2 FROM EMPREGADO E, DEPARTAMENTO D
3 WHERE D.DNOME = 'Pesquisa'
4 AND D.DNUMERO = E.DNO;
```

Below the query editor, the 'Data Output' tab shows the results of the query in a table format:

|   | pnome<br>character varying (30) | unome<br>character varying (30) | endereco<br>text         |
|---|---------------------------------|---------------------------------|--------------------------|
| 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   |

```
SELECT P.NOME, U.NOME, ENDERECO
FROM EMPREGADO E, DEPARTAMENTO D
WHERE D.DNOME = 'Pesquisa' AND D.DNUMERO = E.DNO;
```

# Liste os projetos e seus departamentos usando junção explícita

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane shows a tree view of the database structure, with 'Columns (4)' under the 'departamento' table selected. The 'Query Editor' pane contains the following SQL query:

```
1 SELECT P.PNUMERO, P.PLOCALIZACAO, P.DNUM, D.DNOME
2 FROM PROJETO P JOIN DEPARTAMENTO D ON (P.DNUM = D.DNUMERO)
3
4
5
```

Below the query editor, the 'Data Output' pane displays the results of the query in a table format:

|   | pnumero<br>integer | plocalizacao<br>character varying (30) | dnum<br>integer | dnome<br>character varying (30) |
|---|--------------------|----------------------------------------|-----------------|---------------------------------|
| 1 | 1                  | Bellaire                               | 5               | Pesquisa                        |
| 2 | 2                  | Sugarland                              | 5               | Pesquisa                        |
| 3 | 3                  | Houston                                |                 |                                 |
| 4 | 10                 | Stafford                               |                 |                                 |

A green notification box at the bottom of the results pane states: "Successfully run. Total query runtime: 4 secs 785 msec. 6 rows affected."

```
SELECT P.PNUMERO, P.PLOCALIZACAO, P.DNUM, D.DNOME
FROM PROJETO P JOIN DEPARTAMENTO D ON (P.DNUM = D.DNUMERO)
```

# Liste os projetos e seus departamentos

The screenshot shows the pgAdmin 4 interface. On the left, the 'empregado' table is expanded to show its columns: pnome, inicial, unome, cpf, datanasc, endereco, sexo, salario, gerente\_cpf, and dno. The 'Query Editor' is active, displaying the following SQL query:

```
1 SELECT P.PNUMERO, P.PLOCALIZACAO, P.DNUM, D.DNOME
2 FROM PROJETO P, DEPARTAMENTO D
3 WHERE P.DNUM = D.DNUMERO
```

A blue callout box points to the WHERE clause with the text: "DNUM=DNUMERO relaciona cada projeto com o nome do departamento de controle do projeto". Below the query editor, the 'Data Output' tab shows the results of the query in a table:

|   | pnumero | plocalizacao | dnum | dnome         |
|---|---------|--------------|------|---------------|
| 1 | 1       | Bellaire     | 5    | Pesquisa      |
| 2 | 2       | Sugarland    | 5    | Pesquisa      |
| 3 | 3       | Houston      | 5    | Pesquisa      |
| 4 | 10      | Stafford     | 4    | Administração |

```
SELECT P.PNUMERO, P.PLOCALIZACAO, P.DNUM, D.DNOME
FROM PROJETO P, DEPARTAMENTO D
WHERE P.DNUM = D.DNUMERO
```

# SQL- Apelidos

The screenshot shows the pgAdmin 4 interface. On the left, a tree view shows the database structure, with 'empregado' expanded to show its columns: pnome, inicial, unome, cpf, datanasc, endereco, sexo, salario, gerente\_cpf, and dno. The 'Columns (10)' folder is selected. In the center, the Query Editor shows the following SQL query:

```
1 SELECT E.PNOME, E.UNOME, S.PNOME, S.UNOME
2 FROM EMPREGADO E, EMPREGADO S
3 WHERE E.GERENTE_CPF = S.CPF
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table:

|   | pnome<br>character varying (30) | unome<br>character vary |          |      |
|---|---------------------------------|-------------------------|----------|------|
| 1 | Franklin                        | Wong                    | James    | Borg |
| 2 | Ramesh                          | Narayan                 | Franklin | Wong |
| 3 | John                            | Smith                   | Franklin | Wong |
| 4 | Jennifer                        | Wallace                 |          |      |

A blue callout box points to the 'E' and 'S' table aliases in the query, containing the text: "Utiliza-se apelidos (ou renomear) para se indicar claramente que tabela estamos referenciando na consulta. Os empregados E são empregados normais e empregados S representam gerentes."

At the bottom of the interface, a green status bar indicates: "Successfully run. Total query runtime: 154 msec. 7 rows affected."

```
SELECT E.PNOME, E.UNOME, S.PNOME, S.UNOME
FROM EMPREGADO E, EMPREGADO S
WHERE E.GERENTE_CPF = S.CPF
```

# Selecionar os gerentes dos departamentos que tenham projetos localizados em Stafford

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'empregado' table selected under the 'dept\_localizacoes' schema. The 'Columns (10)' section is expanded, showing fields like 'pnome', 'minicial', 'unome', 'cpf', 'datanasc', 'endereco', 'sexo', 'salario', 'gerente\_cpf', and 'dno'. The main window shows the 'Query Editor' with the following SQL query:

```
1 SELECT P.PNUMERO, P.DNUM, E.UNOME, E.DATANASC, E.ENDERECO
2 FROM PROJETO P, DEPARTAMENTO D, EMPREGADO E
3 WHERE P.DNUM = D.DNUMERO
4 AND D.GERCPF = E.CPF
5 AND P.PLOCALIZACAO = 'Stafford';
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format:

|   | pnumero | dnum    | unome                  | datanasc                    | endereco                |
|---|---------|---------|------------------------|-----------------------------|-------------------------|
|   | integer | integer | character varying (30) | timestamp without time zone | text                    |
| 1 | 10      | 4       | Wallace                | 1941-06-20 00:00:00         | 291 Berry, Bellaire, TX |
| 2 | 30      | 4       | Wallace                | 1941-06-20 00:00:00         | 291 Berry, Bellaire, TX |

```
SELECT P.PNUMERO, P.DNUM, E.UNOME, E.DATANASC, E.ENDERECO
FROM PROJETO P, DEPARTAMENTO D, EMPREGADO E
WHERE P.DNUM = D.DNUMERO AND D.GERCPF = E.CPF AND P.PLOCALIZACAO = 'Stafford';
```

## *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) and 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

- Obtenha 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

The screenshot shows the pgAdmin 4 interface. On the left, the 'Tables (6)' folder is expanded to show the 'departamento' table, with its 'Columns (4)' sub-folder selected. The columns listed are 'dnome', 'dnumero', 'gercpf', and 'gerdatainicio'. The 'Query Editor' pane contains the following SQL query:

```
1 SELECT P.PJNOME
2 FROM PROJETO P, DEPARTAMENTO D, EMPREGADO E
3 WHERE P.DNUM = D.DNUMERO AND D.GERCPF = E.CPF AND E.UNOME='Smith'
4 UNION
5 SELECT P.PJNOME
6 FROM PROJETO P, TRABALHA_EM T, EMPREGADO E
7 WHERE P.PNUMERO=T.PNO AND T.ECPF=E.CPF AND E.UNOME = 'Smith'
```

Below the query editor, the 'Data Output' pane shows the results of the query:

|   | pjnome                 |
|---|------------------------|
|   | character varying (30) |
| 1 | ProdutoY               |
| 2 | ProdutoX               |

```
SELECT P.PJNOME FROM PROJETO P, DEPARTAMENTO D, EMPREGADO E WHERE P.DNUM = D.DNUMERO AND D.GERCPF = E.CPF AND E.UNOME='Smith'
UNION
SELECT P.PJNOME FROM PROJETO P, TRABALHA_EM T, EMPREGADO E WHERE P.PNUMERO=T.PNO AND T.ECPF=E.CPF AND E.UNOME = 'Smith'
```

# Operações de Ordenação

- Obter o nome dos empregados ordenados pelo salário

The screenshot shows the pgAdmin 4 interface. On the left, the 'Columns (4)' for the 'empregado' table are listed: 'dnome', 'dnumero', 'gercpf', and 'gerdatainicio'. The 'Query Editor' tab is active, displaying the following SQL query:

```
1 SELECT E.PNOME, E.UNOME, E.SALARIO
2 FROM EMPREGADO E
3 ORDER BY E.SALARIO
4
5
6
```

Below the query editor, the 'Data Output' tab is selected, showing the results of the query in a table format:

|   | pnome<br>character varying (30) | unome<br>character varying (30) | salario<br>double precision |
|---|---------------------------------|---------------------------------|-----------------------------|
| 1 | Ahmad                           | Jabbar                          | 25000                       |
| 2 | Joyce                           | English                         | 25000                       |
| 3 | Alicia                          | Zelaya                          | 25000                       |
| 4 | John                            | Smith                           | 30000                       |

```
SELECT E.PNOME, E.UNOME, E.SALARIO
FROM EMPREGADO E
ORDER BY E.SALARIO
```

# Consulta com like

- Obter o nome dos empregados que tenham último nome que contendo a letra 'r'

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'empregado' table selected under the 'departamento' schema. The main window shows the 'Query Editor' with the following SQL query:

```
1 SELECT E.PNOME, E.UNOME, E.SALARIO
2 FROM EMPREGADO E
3 WHERE E.UNOME LIKE '%r%'
4
5
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format:

|   | pnome<br>character varying (30) | unome<br>character varying (30) | salario<br>double precision |
|---|---------------------------------|---------------------------------|-----------------------------|
| 1 | James                           | Borg                            | 55000                       |
| 2 | Ramesh                          | Narayan                         | 38000                       |
| 3 | Ahmad                           | Jabbar                          | 25000                       |

```
SELECT E.PNOME, E.UNOME, E.SALARIO
FROM EMPREGADO E
WHERE E.UNOME LIKE '%r%'
```

## *Funções agregadas*

- Essas funções operam no multiconjunto dos valores de uma coluna de uma relação e retornam um valor
- - avg: valor médio
  - min: valor mínimo
  - max: valor máximo
  - sum: soma dos valores
  - count: número de valores

# Funções agregadas – count(\*)

- Obtenha a quantidade de empregados

The screenshot shows the pgAdmin 4 web interface. The left sidebar displays a tree view of the database structure, with the 'empregado' table selected under the 'Tables (6)' category. The main area is divided into several sections: a toolbar with various icons, a 'Query Editor' tab containing the SQL query, and a 'Data Output' tab showing the query results.

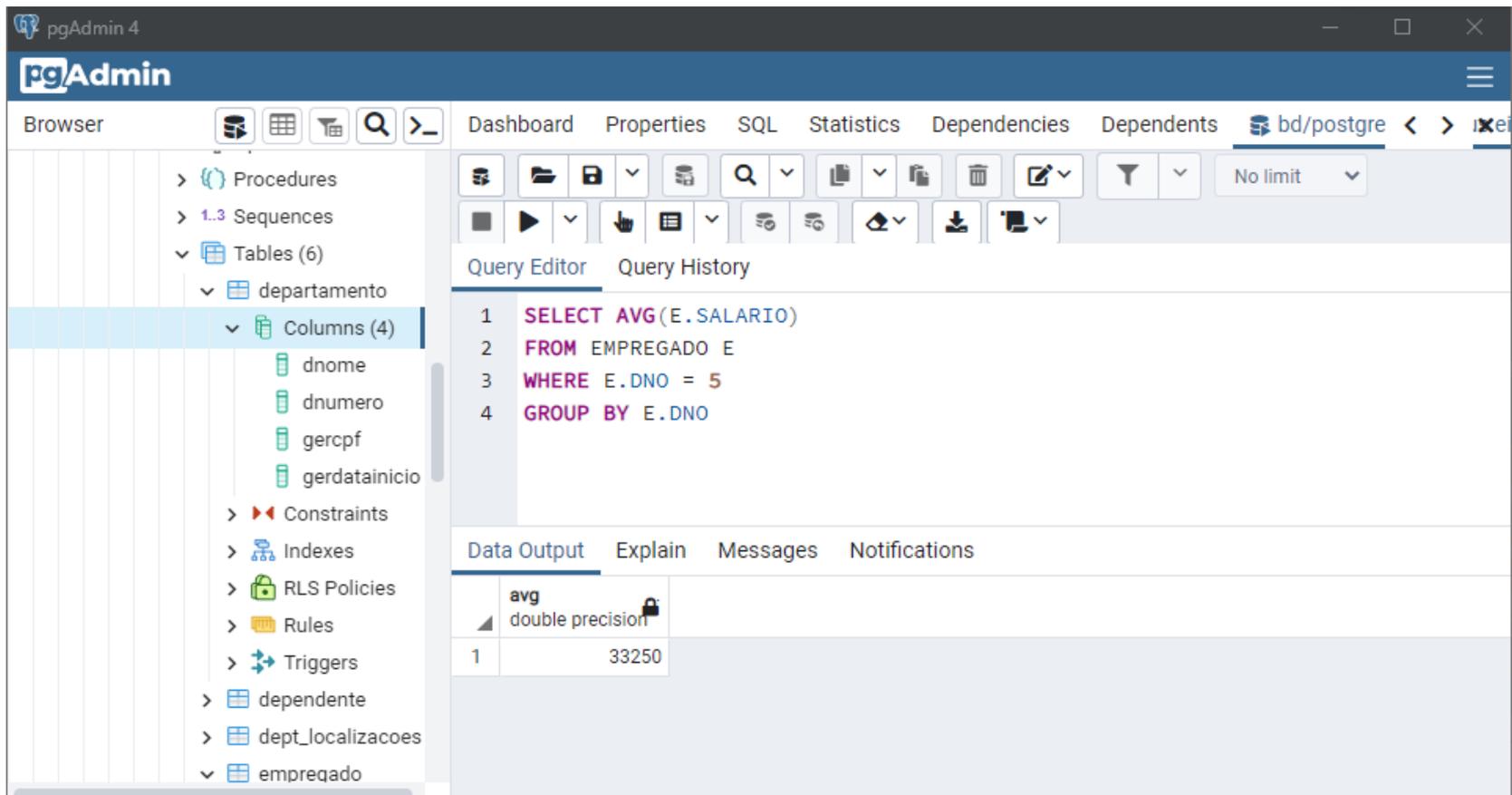
```
1 SELECT COUNT (*)
2 FROM EMPREGADO E
3
```

|   | count |
|---|-------|
| 1 | 8     |

```
SELECT COUNT(*)
FROM EMPREGADO E
```

## Funções agregadas – avg

- Obtenha o salário médio dos empregados do departamento número 5



The screenshot shows the pgAdmin 4 interface. On the left, the 'Tables (6)' folder is expanded to show the 'departamento' table, with its 'Columns (4)' sub-folder selected. The columns listed are 'dnome', 'dnumero', 'gercpf', and 'gerdatainicio'. The main window displays the 'Query Editor' with the following SQL query:

```
1 SELECT AVG(E.SALARIO)
2 FROM EMPREGADO E
3 WHERE E.DNO = 5
4 GROUP BY E.DNO
```

Below the query editor, the 'Data Output' tab is active, showing the result of the query in a table:

|   | avg              |
|---|------------------|
|   | double precision |
| 1 | 33250            |

```
SELECT AVG(E.SALARIO)
FROM EMPREGADO E WHERE E.DNO = 5
GROUP BY E.DNO
```

## Funções agregadas – cláusula group by

- Encontre a média salarial dos empregados para cada departamento
- Nota: Os atributos na cláusula select fora das funções agregadas precisam aparecer na lista group by

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'empregado' table selected. The main window shows the Query Editor with the following SQL query:

```
1 select e.dno, avg (e.salario)
2 from empregado e
3 group by e.dno
4
5
```

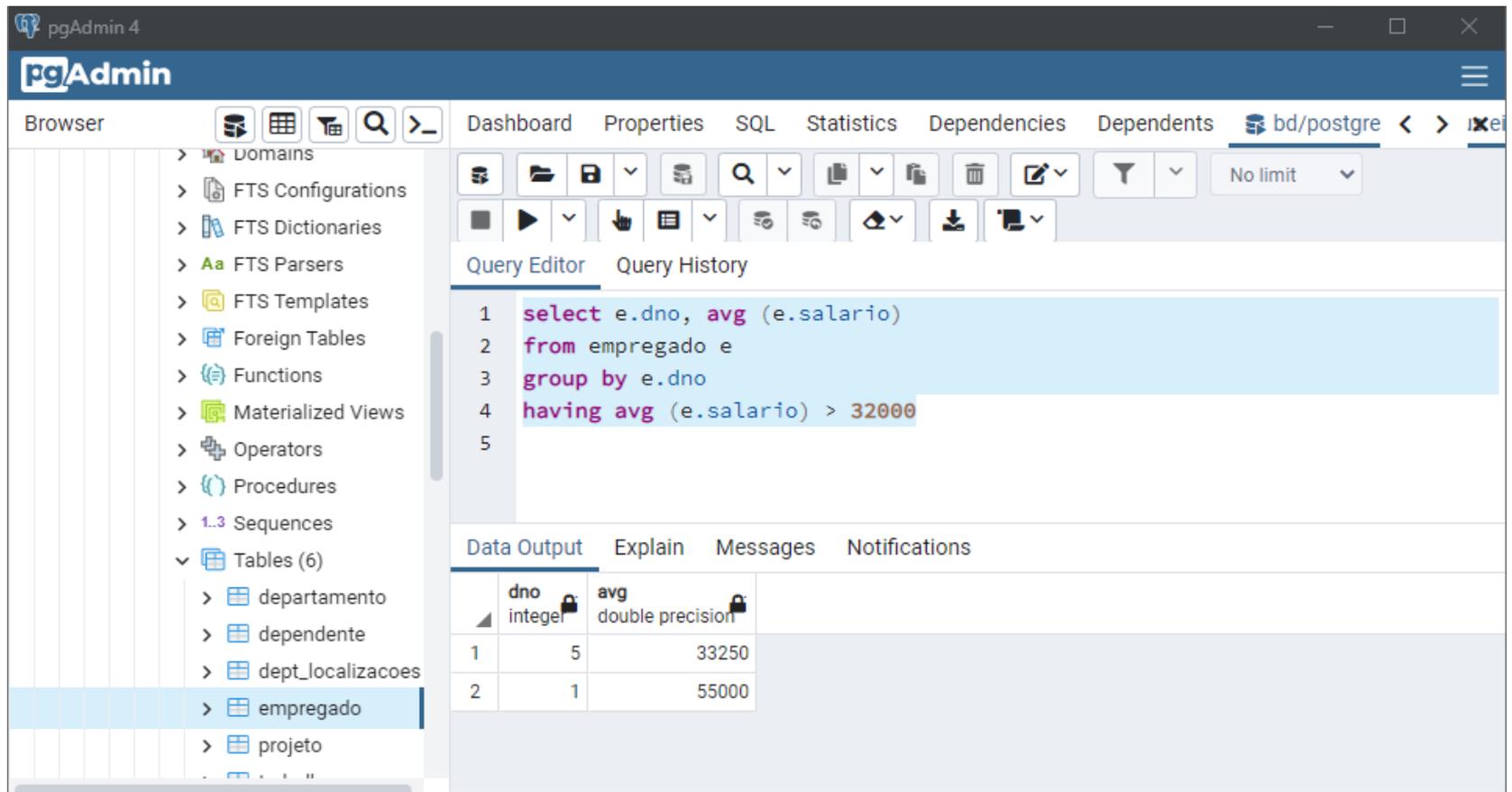
Below the query editor, the Data Output tab is active, displaying the results of the query in a table:

|   | dno     | avg              |
|---|---------|------------------|
|   | integer | double precision |
| 1 | 5       | 33250            |
| 2 | 4       | 31000            |
| 3 | 1       | 55000            |

```
select e.dno, avg (e.salario)
from empregado e
group by e.dno
```

# Funções agregadas – cláusula group by com having

- Encontre todos os departamentos cuja média salarial dos empregados seja superior a 32000
- Nota: Os predicados na cláusula having são aplicados após a formação de grupos, enquanto os predicados na cláusula where são aplicados antes da formação de grupos



The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'empregado' table selected under the 'Tables (6)' folder. The main window shows the Query Editor with the following SQL query:

```
1 select e.dno, avg (e.salario)
2 from empregado e
3 group by e.dno
4 having avg (e.salario) > 32000
5
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format:

|   | dno     | avg              |
|---|---------|------------------|
|   | integer | double precision |
| 1 | 5       | 33250            |
| 2 | 1       | 55000            |

## ***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

# Consultas aninhadas usando IN

- Obtenha o nome e endereço de todos os empregados que trabalham no departamento de 'Pesquisa'

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'empregado' table selected under the 'Tables (6)' folder. The main window shows the Query Editor with the following SQL query:

```
1 SELECT E.PNOME, E.UNOME, E.ENDereco
2 FROM EMPREGADO E
3 WHERE DNO IN (SELECT D.DNUMERO
4               FROM DEPARTAMENTO D
5               WHERE D.DNOME='Pesquisa')
```

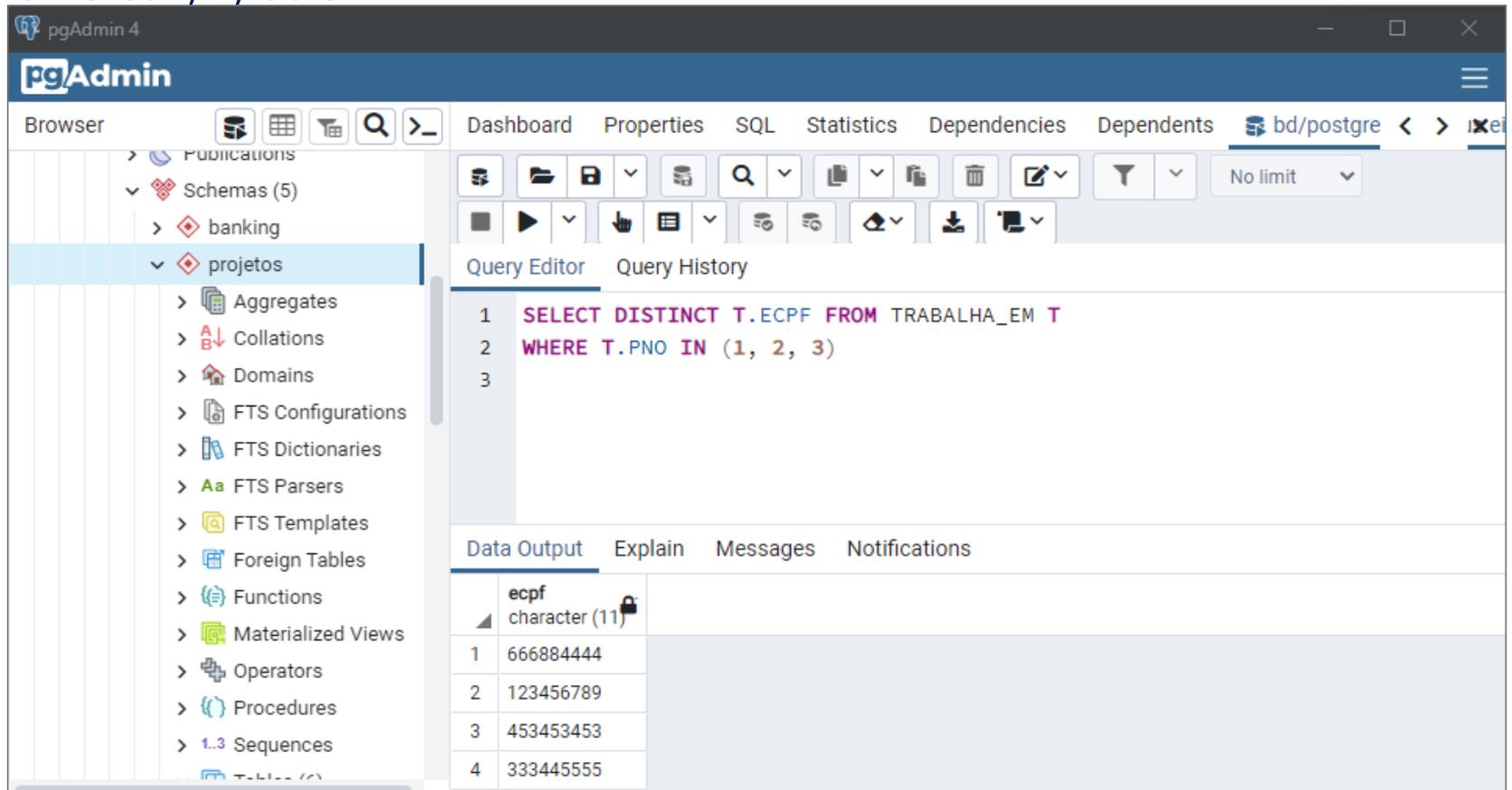
Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format:

|   | pnome<br>character varying (30) | unome<br>character varying (30) | endereco<br>text         |
|---|---------------------------------|---------------------------------|--------------------------|
| 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   |

```
SELECT E.PNOME, E.UNOME, E.ENDereco
FROM EMPREGADO E
WHERE DNO IN (SELECT D.DNUMERO FROM DEPARTAMENTO D WHERE D.DNOME='Pesquisa')
```

# Conjuntos explícitos

- É também possível usar um conjunto de valores explícito (enumerado) na cláusula WHERE, ao invés de uma consulta aninhada
- Obtenha o CPF de todos os empregados que trabalham em projetos de números 1, 2, ou 3



The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'projetos' schema selected. The main window shows a SQL query in the Query Editor:

```
1 SELECT DISTINCT T.ECPF FROM TRABALHA_EM T
2 WHERE T.PNO IN (1, 2, 3)
3
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format:

|   | eCPF<br>character (11) |
|---|------------------------|
| 1 | 666884444              |
| 2 | 123456789              |
| 3 | 453453453              |
| 4 | 333445555              |

```
SELECT DISTINCT T.ECPF
FROM TRABALHA_EM T
WHERE T.PNO IN (1, 2, 3)
```

## ***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 da relação da consulta externa

# Uso do EXISTS

- EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada (contém uma ou mais tupla associadas)

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane shows the database structure for 'bd/postgre', with 'dependente' selected under 'Tables (6)'. The 'Columns (5)' for 'dependente' are listed: 'ecpf', 'nome\_dependente', 'sexo', 'datanasc', and 'parentesco'. The 'Query Editor' pane contains the following SQL query:

```
1 SELECT E.PNOME, E.UNOME
2 FROM EMPREGADO E
3 WHERE EXISTS (SELECT * FROM DEPENDENTE D
4               WHERE D.ECPF=E.CPF AND E.PNOME=D.NOME_DEPENDENTE)
```

Below the query editor, the 'Data Output' pane shows the results of the query:

| pnome                  | unome                  |
|------------------------|------------------------|
| character varying (30) | character varying (30) |

```
SELECT E.PNOME, E.UNOME
FROM EMPREGADO E
WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF=E.CPF AND E.PNOME=D.NOME_DEPENDENTE)
```

# Uso do NOT EXISTS

- NOT EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio (não contém tupla associada)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with the 'projetos' schema selected. The main window shows the Query Editor with the following SQL query:

```
1 SELECT E.PNOME, E.UNOME
2 FROM EMPREGADO E WHERE NOT EXISTS (
3     SELECT *
4     FROM DEPENDENTE D
5     WHERE D.ECPF=E.CPF
6 )
```

Below the query editor, the 'Data Output' tab is active, showing the results of the query in a table:

|   | pnome<br>character varying (30) | unome<br>character varying (30) |
|---|---------------------------------|---------------------------------|
| 1 | James                           | Borg                            |
| 2 | Ramesh                          | Narayan                         |
| 3 | Alicia                          | Zelaya                          |
| 4 | Ahmad                           | Jabbar                          |

A green notification box at the bottom right of the results area states: "Successfully run. Total query runtime: 160 msec. 5 rows affected."

```
SELECT E.PNOME, E.UNOME
FROM EMPREGADO E
WHERE NOT EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF=E.CPF)
```

# Referências

