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 (
   PNOME 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)
);
```

Pode-se incluir as chaves primárias e estrangeiras durante ou após a criação

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)
);
```

Alguns atributos podem ser obrigatórios

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)
);
```

Chaves candidatas podem ser indicadas com o unique

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

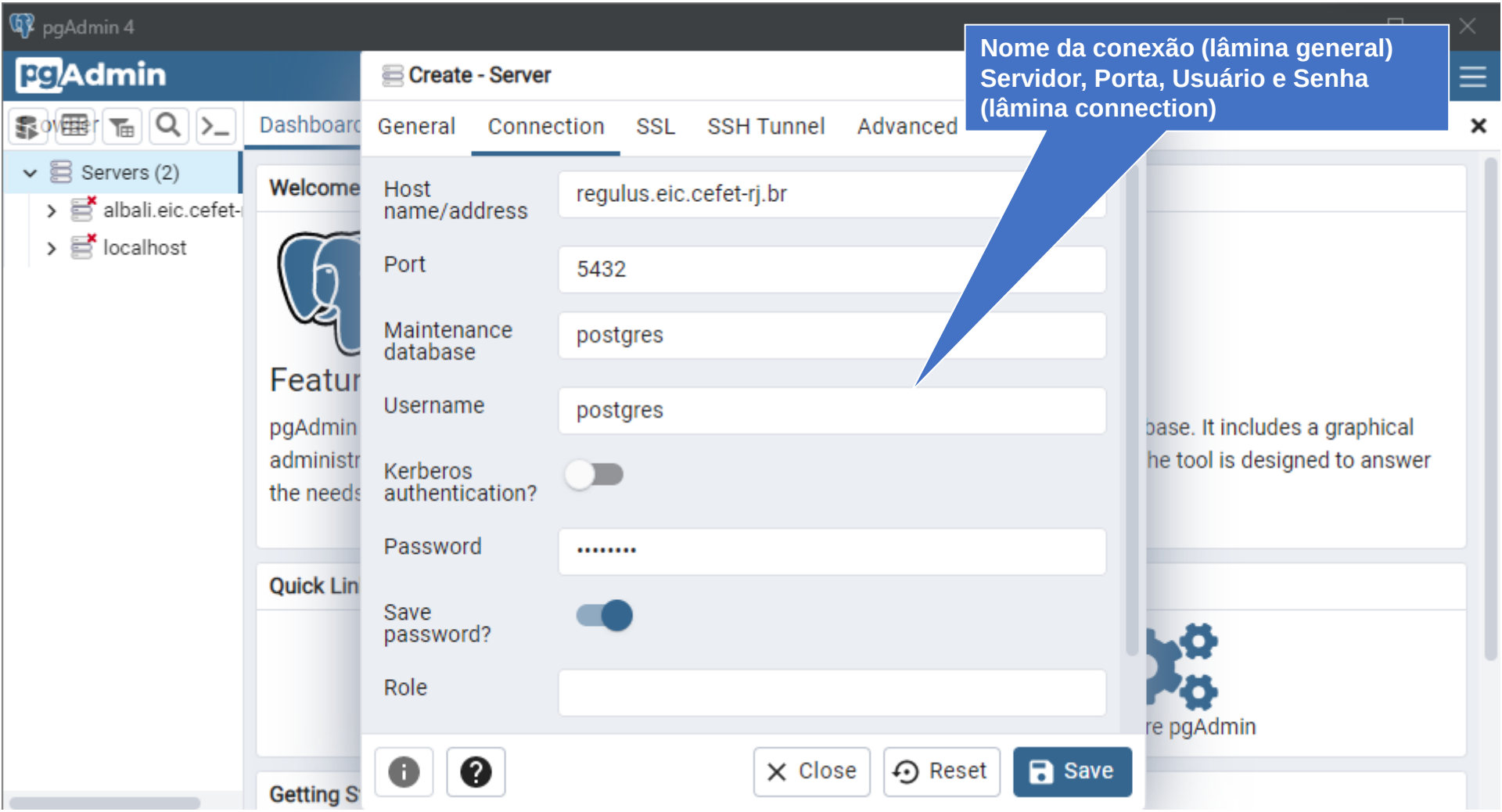
PostgreSQL

- 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

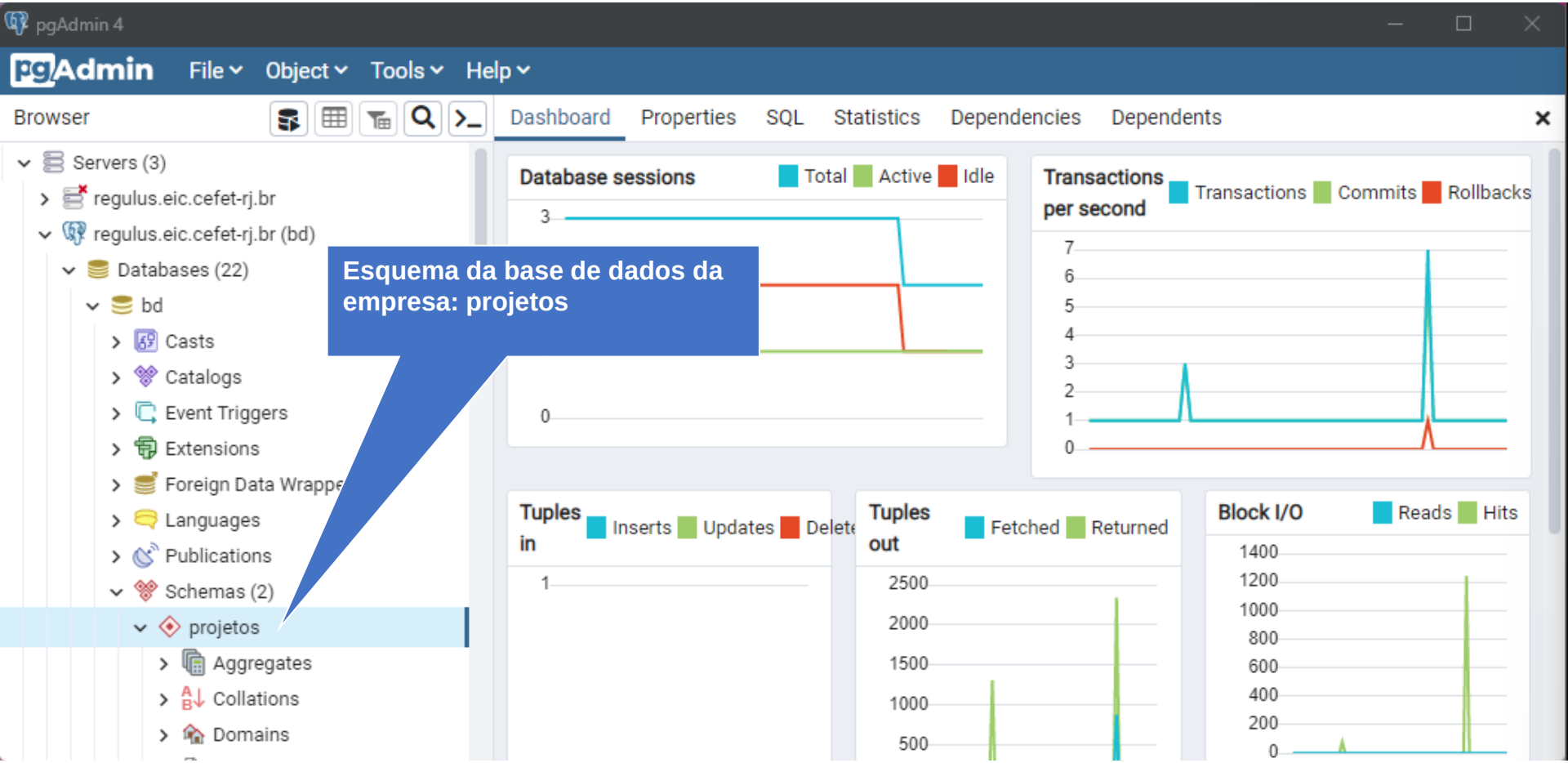
PostgreSQL – Instalação

- 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.devart.com/dotconnect/postgresql/

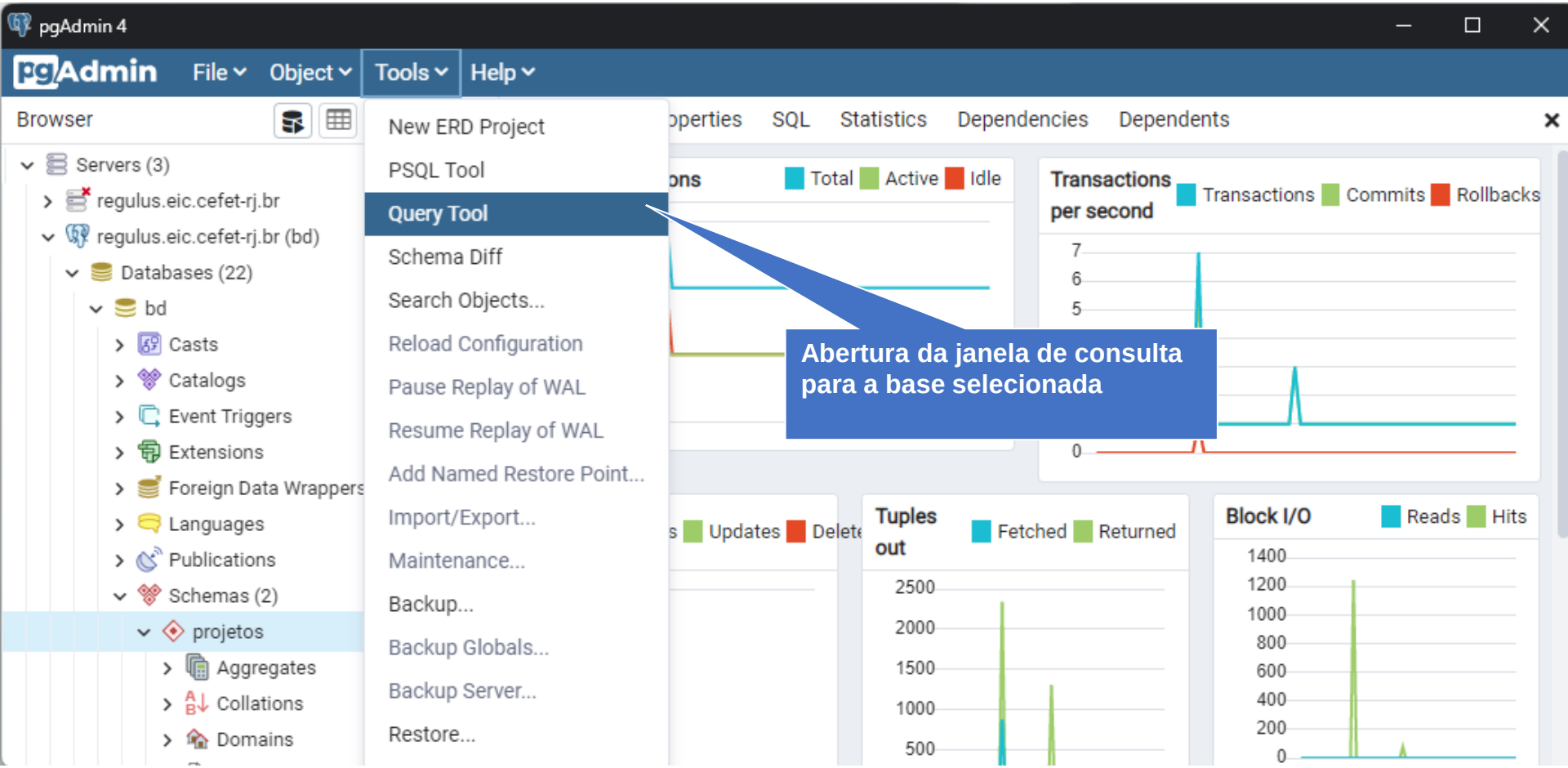
Postgresql – Interface de exploração de base de dados



Postgresql – Interface de exploração de base de dados

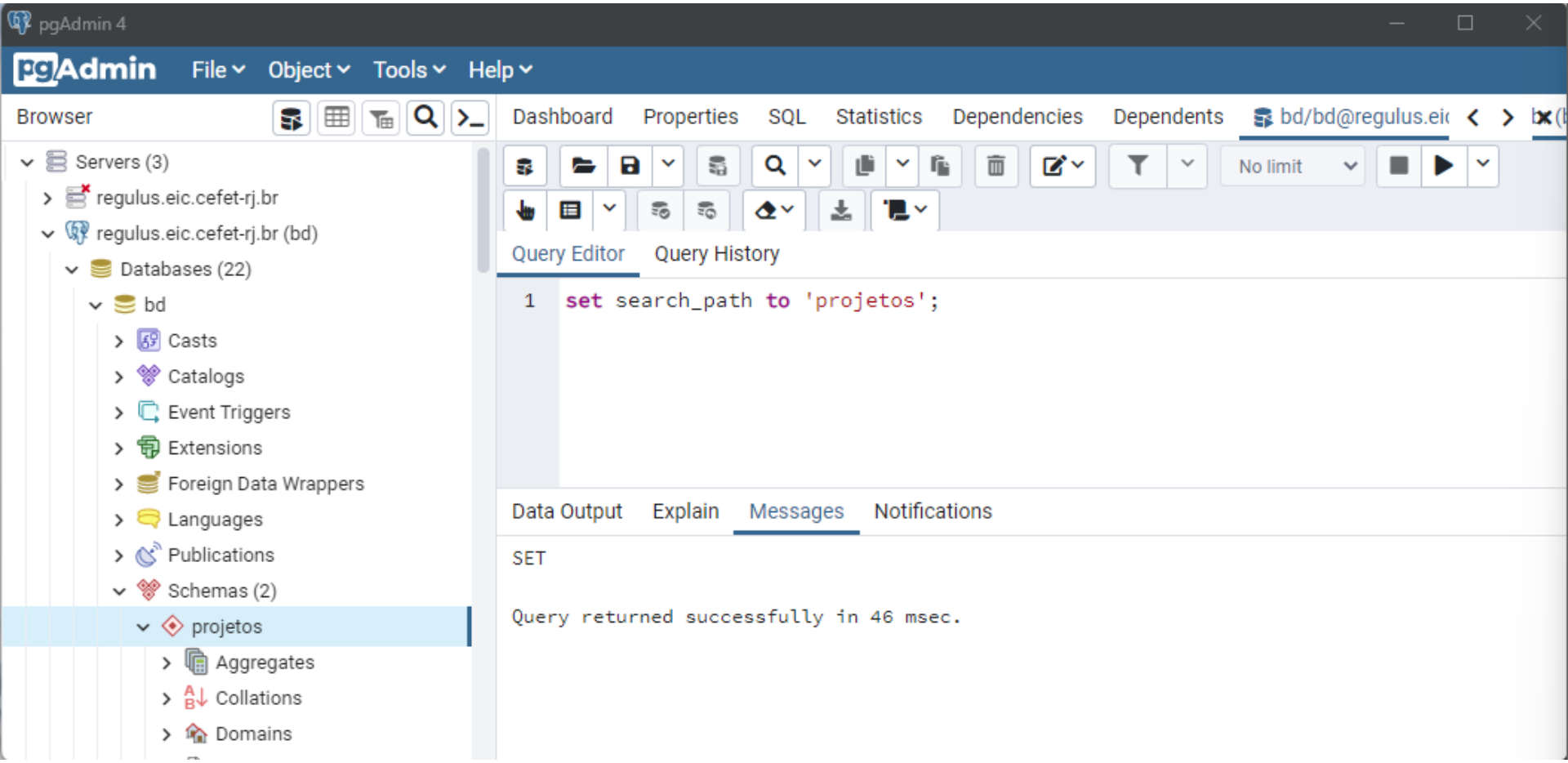


Postgresql – abertura da janela de comandos SQL



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

Postgresql – abertura da janela de comandos SQL



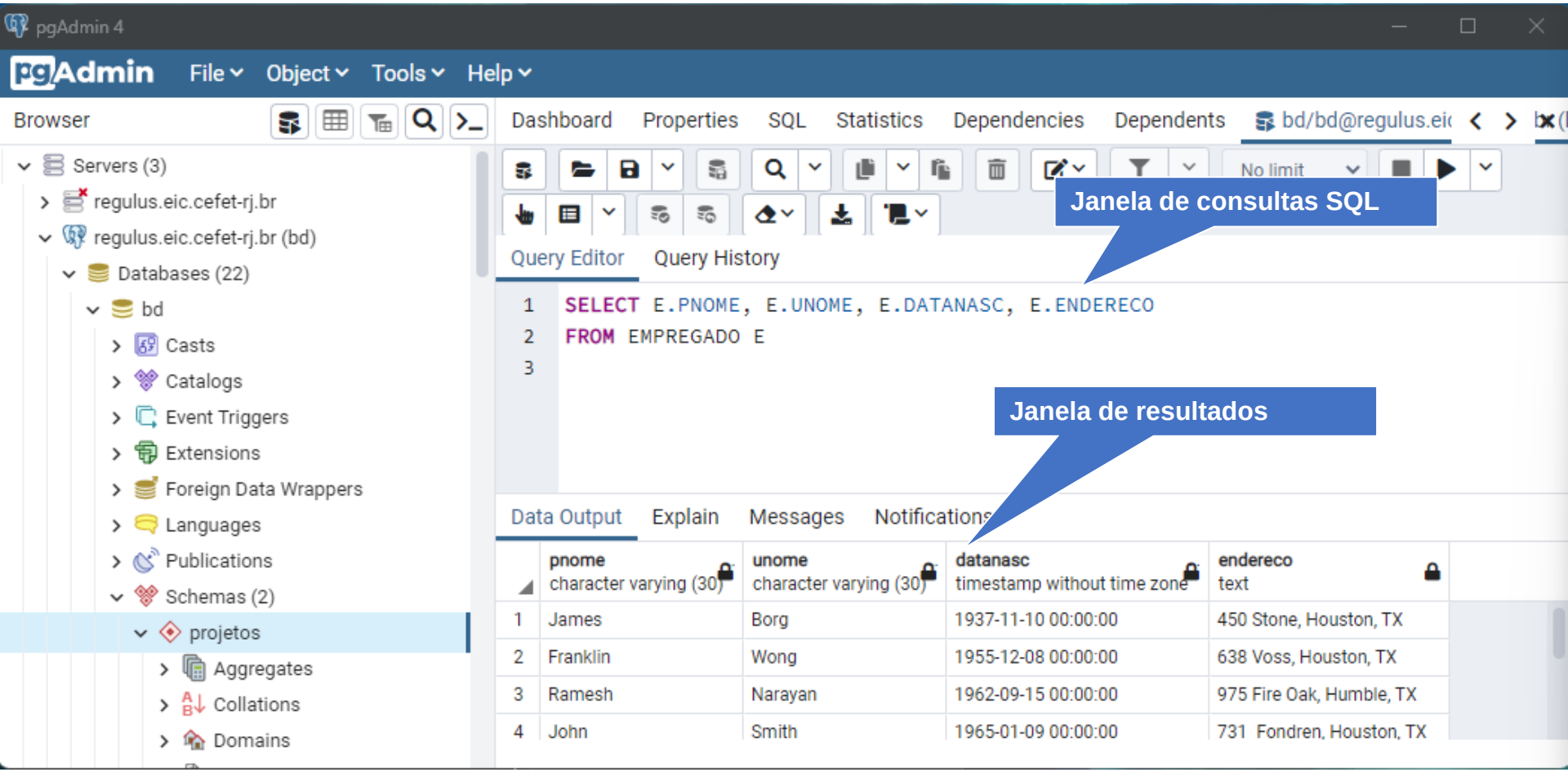
set search_path to 'projetos';

Exercícios

- Acessar Postgresql em 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



O apelido para empregado adotado no exemplo é a letra E

Mapeamento do select em algebra relacional

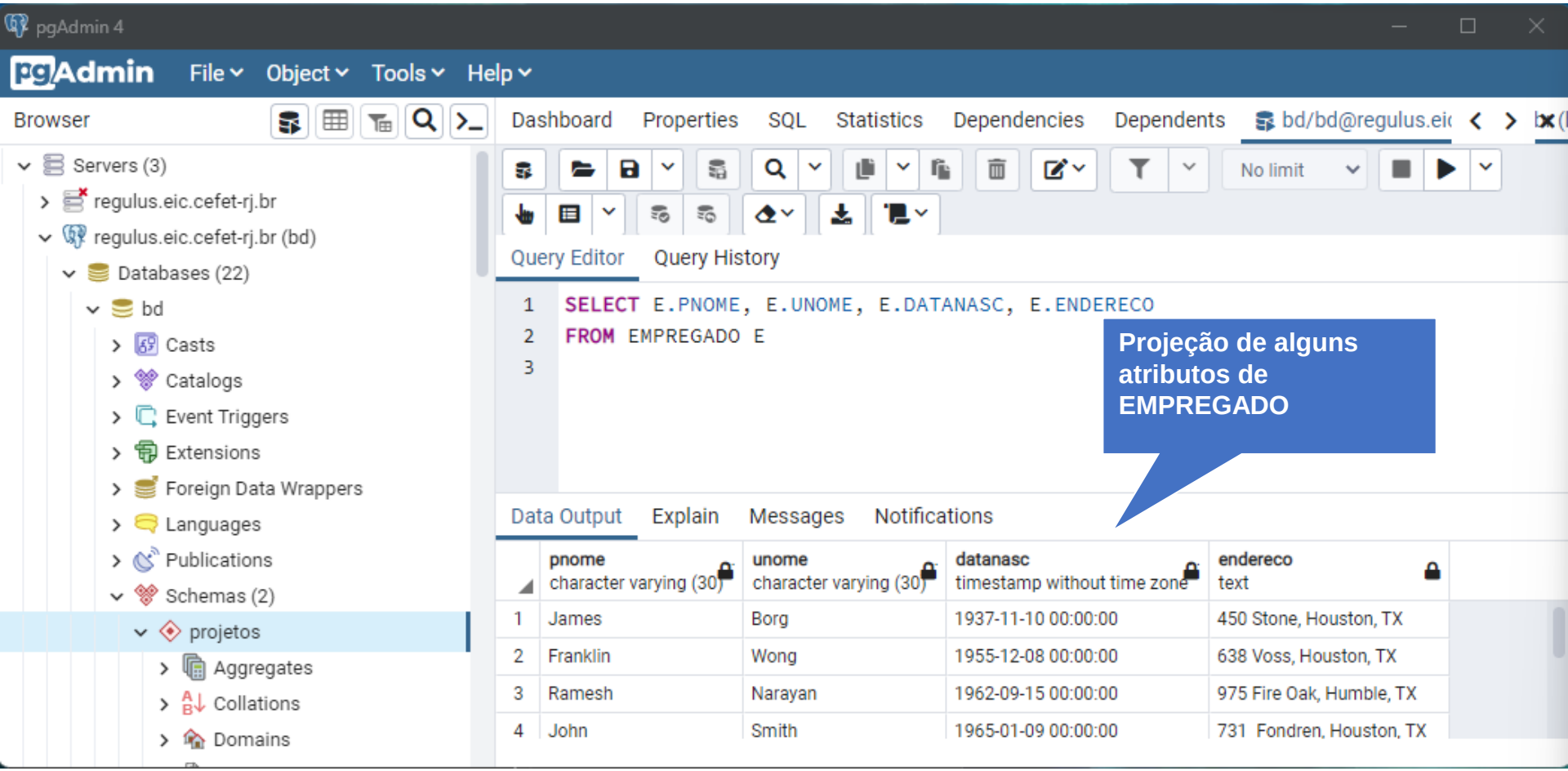
- A cláusula select lista os atributos desejados no resultado de uma consulta
  - corresponde à operação projeção da álgebra relacional
- Exemplo:
  Encontre os nomes de todas as agências na relação empréstimo

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

- Na álgebra relacional, a consulta seria:

$$\pi_{PNOME}(EMPREGADO)$$

Obtenha nome, nascimento e endereço dos empregados



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

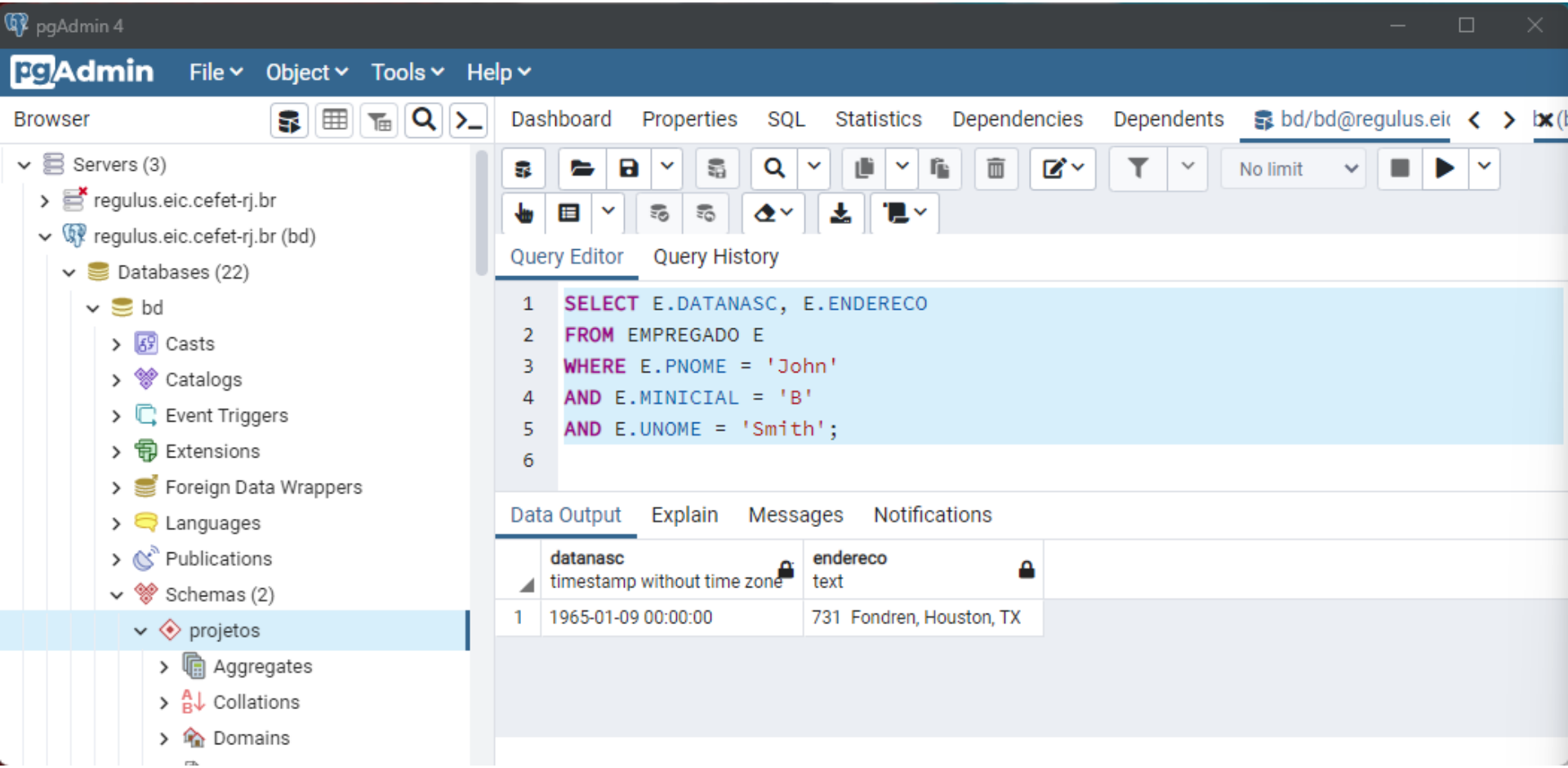
A cláusula where

- 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
```

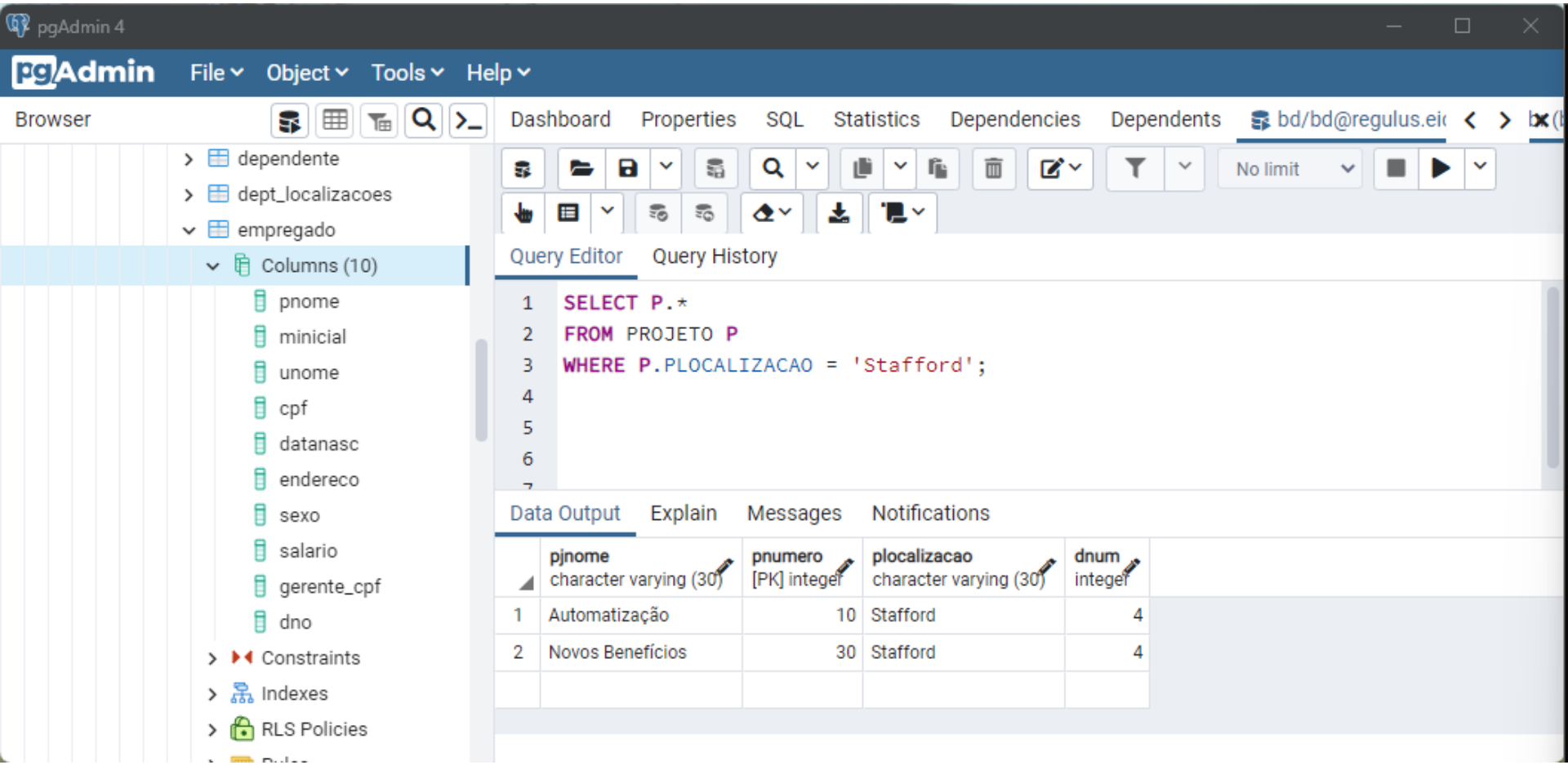
- Os resultados da comparação podem ser combinados usando os conectivos lógicos AND, OR, e NOT.
- As comparações podem ser aplicadas aos resultados das expressões aritméticas

Obtenha a data de nascimento e endereço do John Smith




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

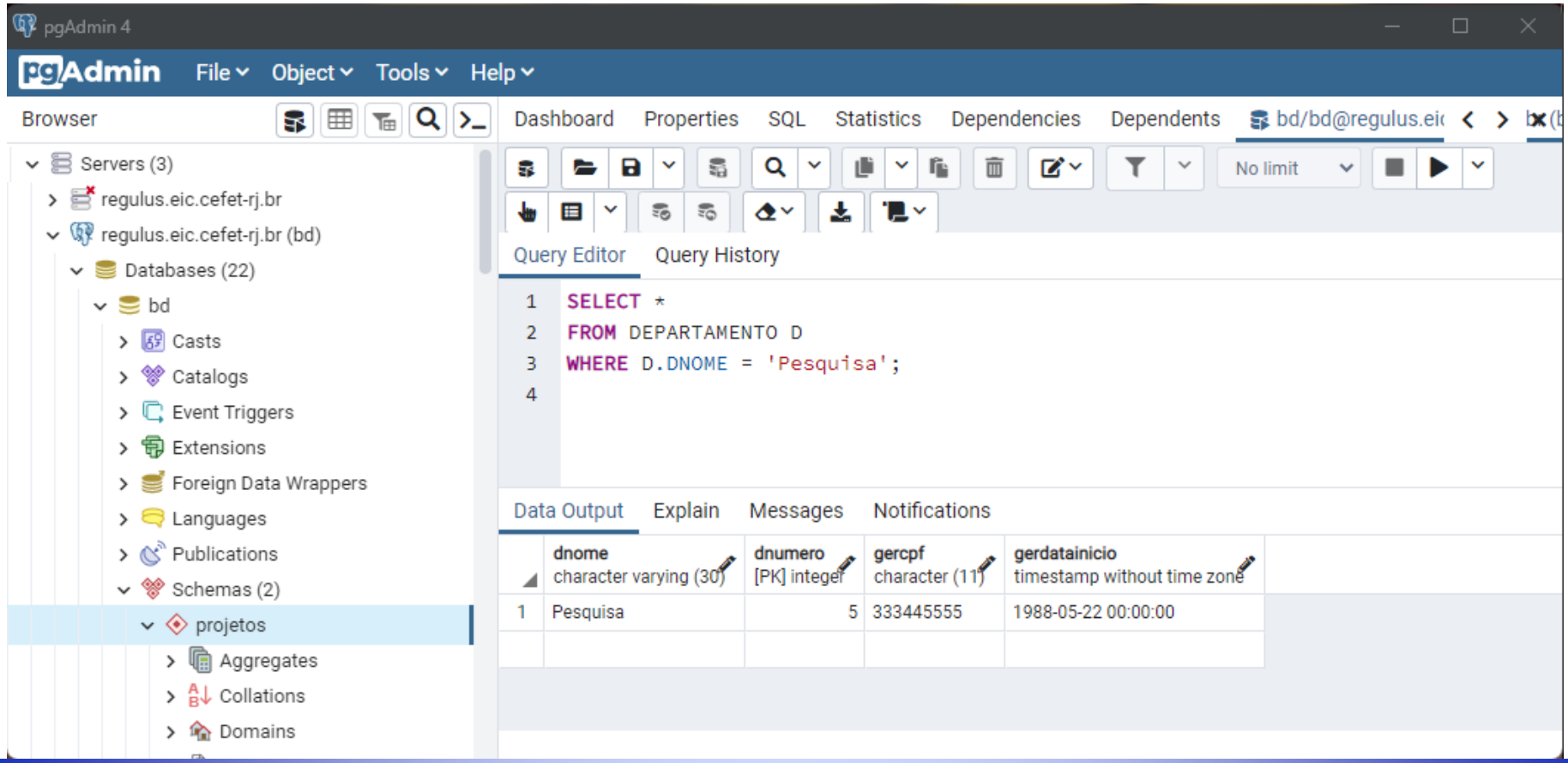
Selecionar os projetos localizados em Stafford




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

Selecione todos os atributos do departamento de pesquisa

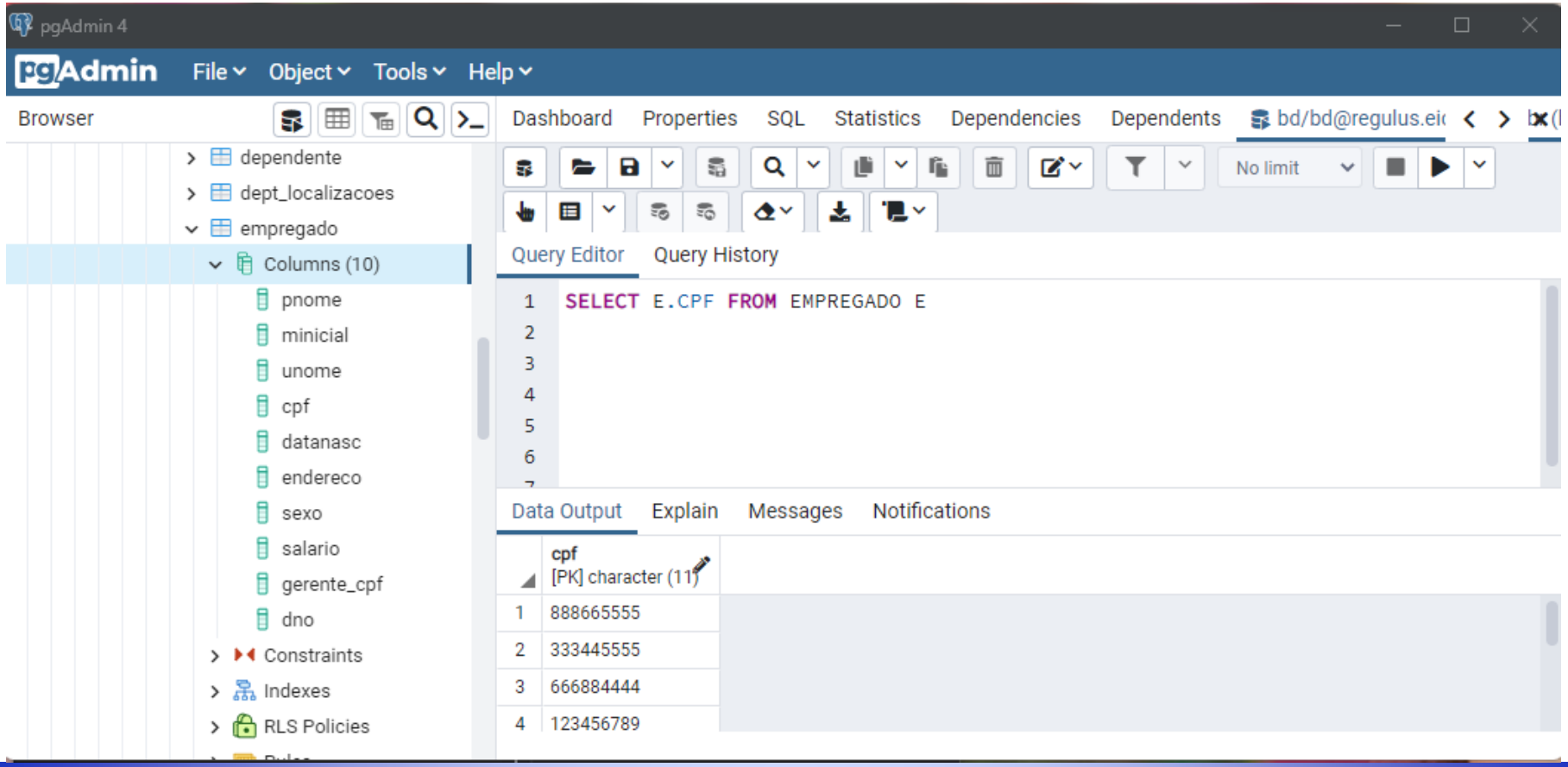
- O asterisco representa todos os atributos da tabela
- Evite usar select * numa aplicação
  - Traga apenas os campos necessários




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

Consulta sem cláusula de seleção

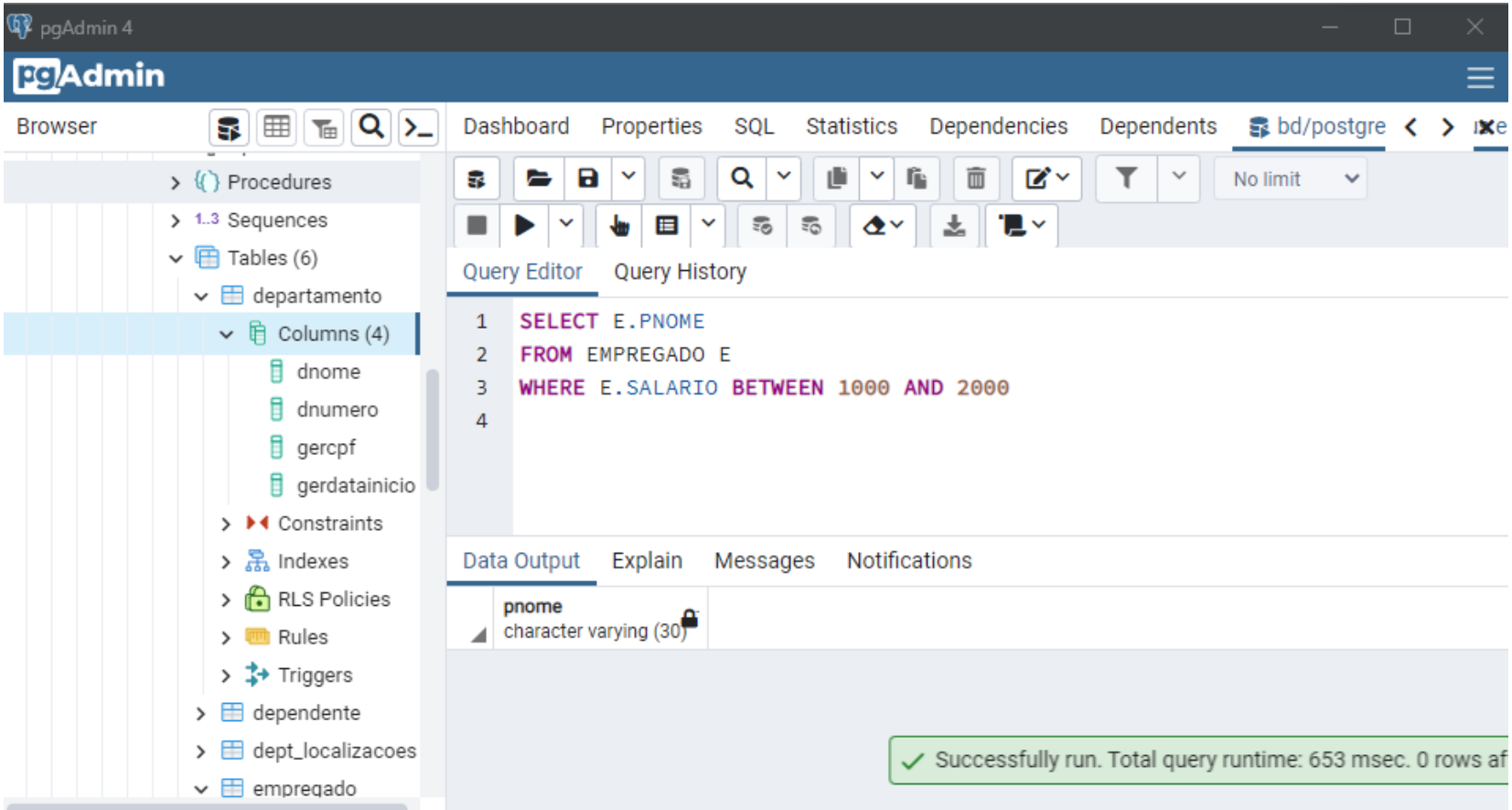
- SQL sem a cláusula de seleção indica que não há condição de seleção; assim, todas as tuplas da relação da cláusula FROM são selecionadas



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



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 PNOME, 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

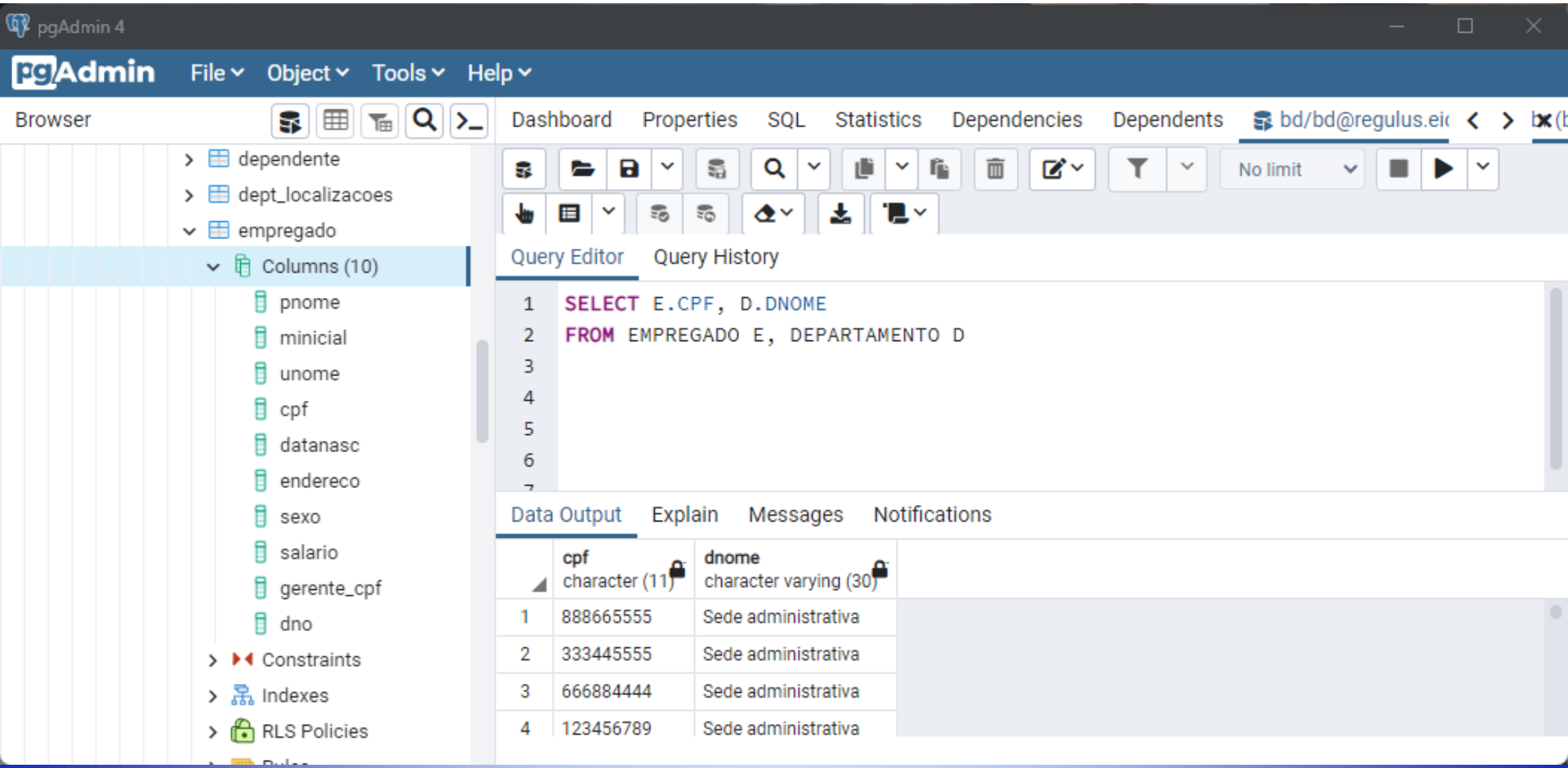
```
SELECT E.PNOME, D.NOME_DEPENDENTE
FROM EMPREGADO E, DEPENDENTE D
WHERE E.CPF = D.ECPF
```

Cláusula de junção

- Junção
  - As tabelas comumente devem ser interligadas.
  - Se há n tabelas, deve-se ter n-1 cláusulas de junção

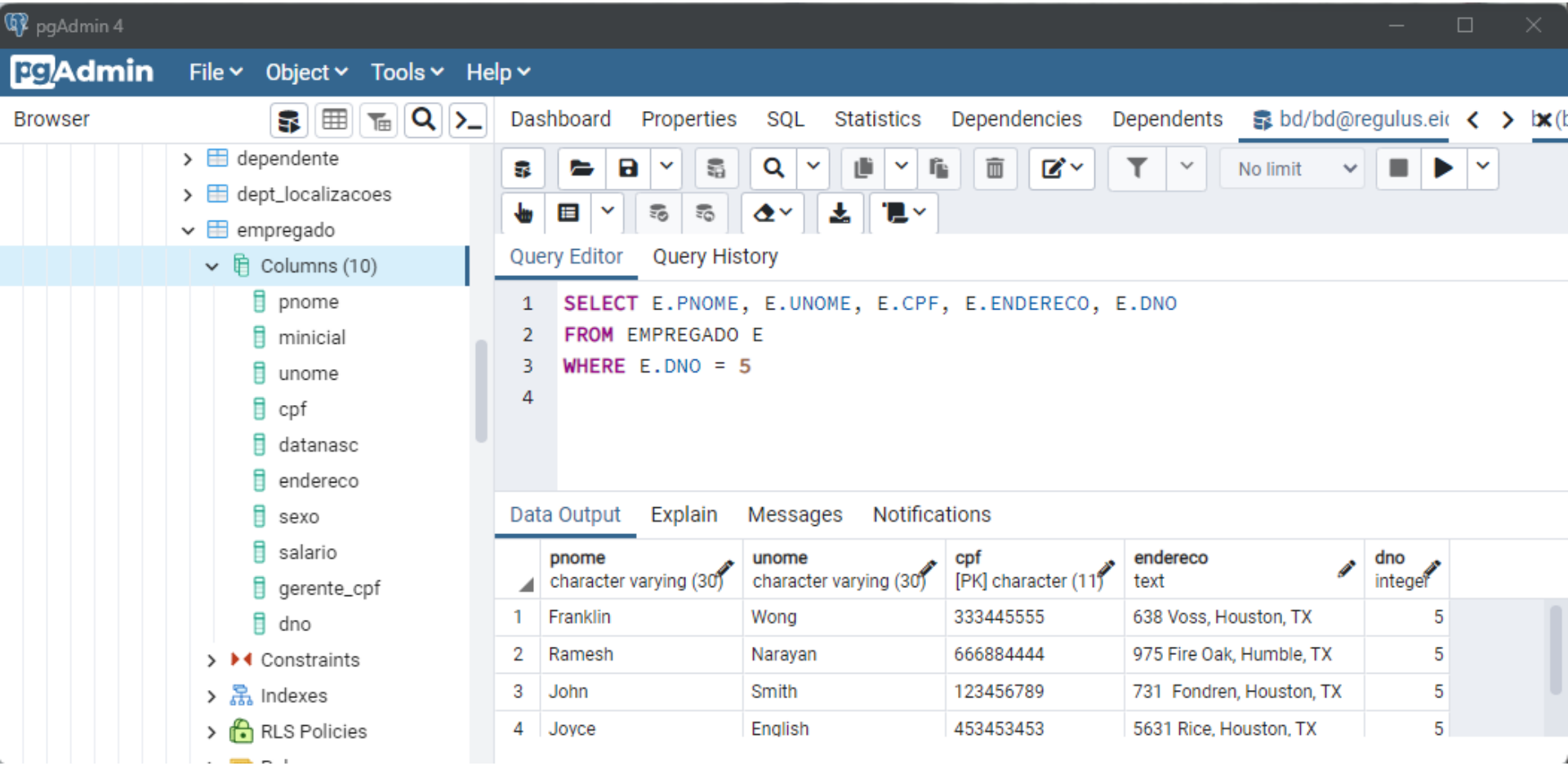
Produto Cartesiano

- 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



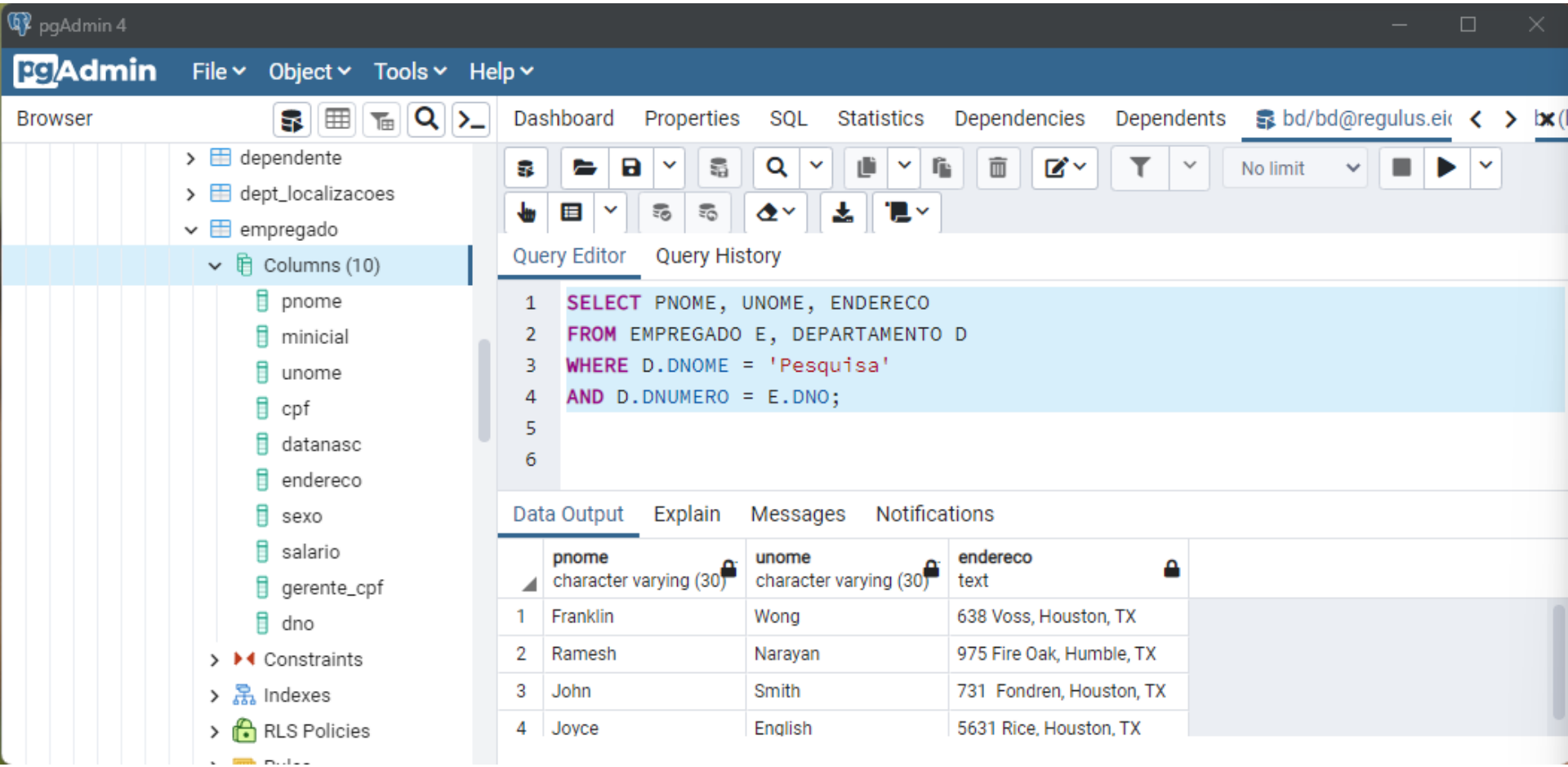

```
SELECT E.CPF, D.DNOME
FROM EMPREGADO E, DEPARTAMENTO D
```

Selecione empregados do departamento de pesquisa



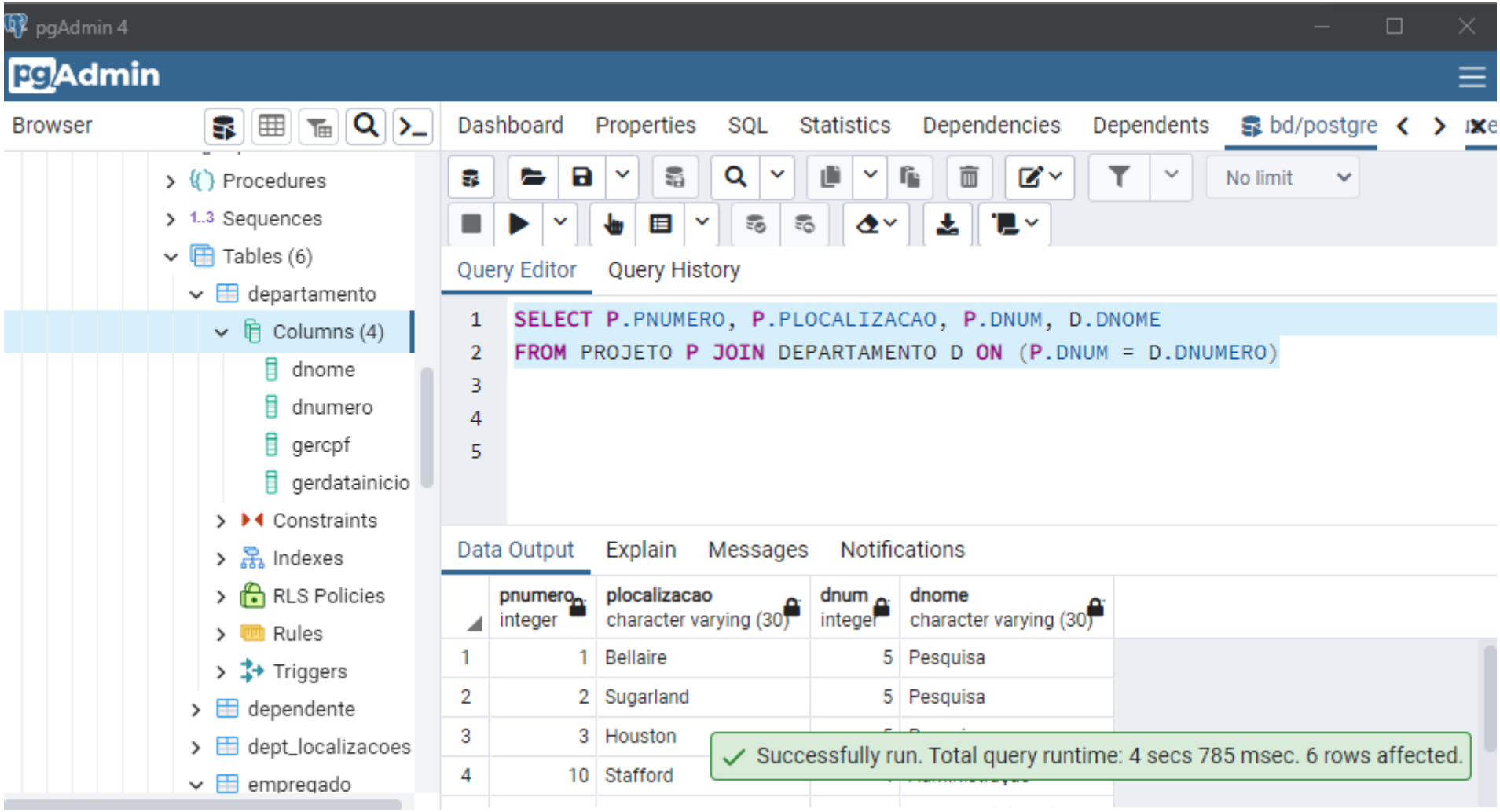

```
SELECT E.PNOME, E.UNOME, E.CPF, E.ENDERECO, E.DNO
FROM EMPREGADO E
WHERE E.DNO = 5
```

Selecione empregados do departamento de pesquisa



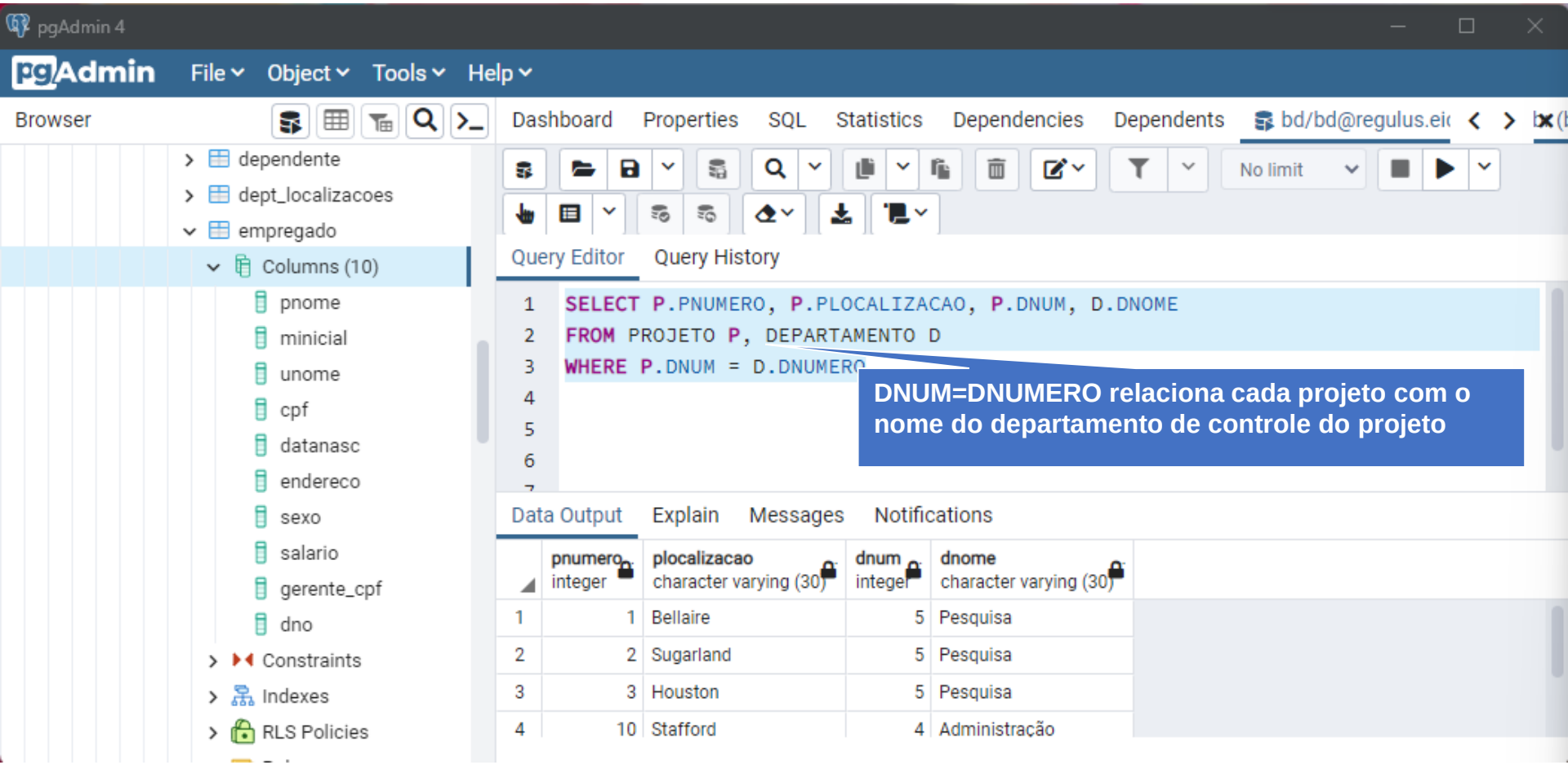

```
SELECT PNOME, UNOME, ENDERECO
FROM EMPREGADO E, DEPARTAMENTO D
WHERE D.DNOME = 'Pesquisa' AND D.DNUMERO = E.DNO;
```

Liste os projetos e seus departamentos usando junção explicita



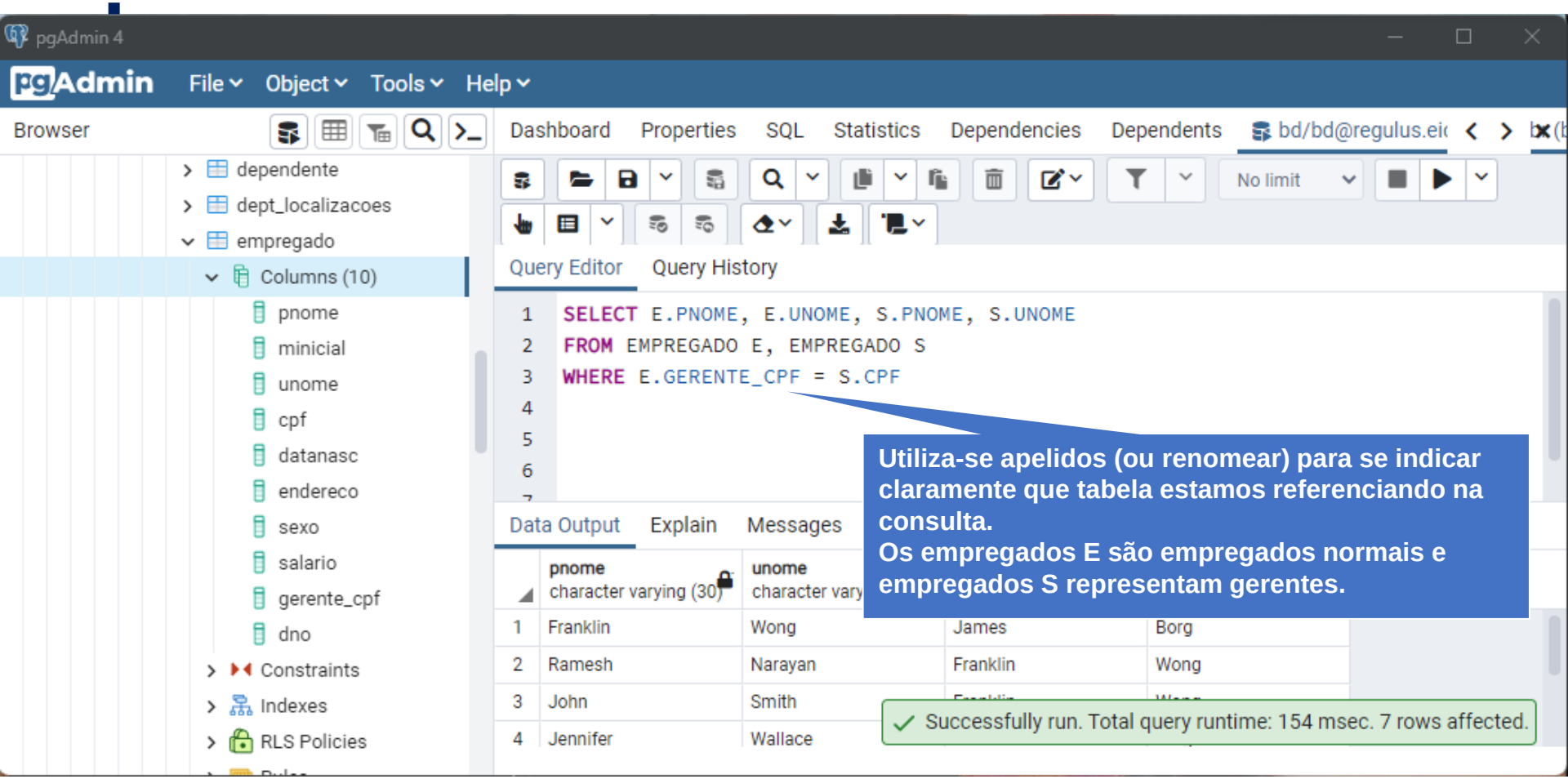

```
SELECT P.PNUMERO, P.PLOCALIZACAO, P.DNUM, D.DNOME
FROM PROJETO P JOIN DEPARTAMENTO D ON (P.DNUM = D.DNUMERO)
```

Liste os projetos e seus departamentos



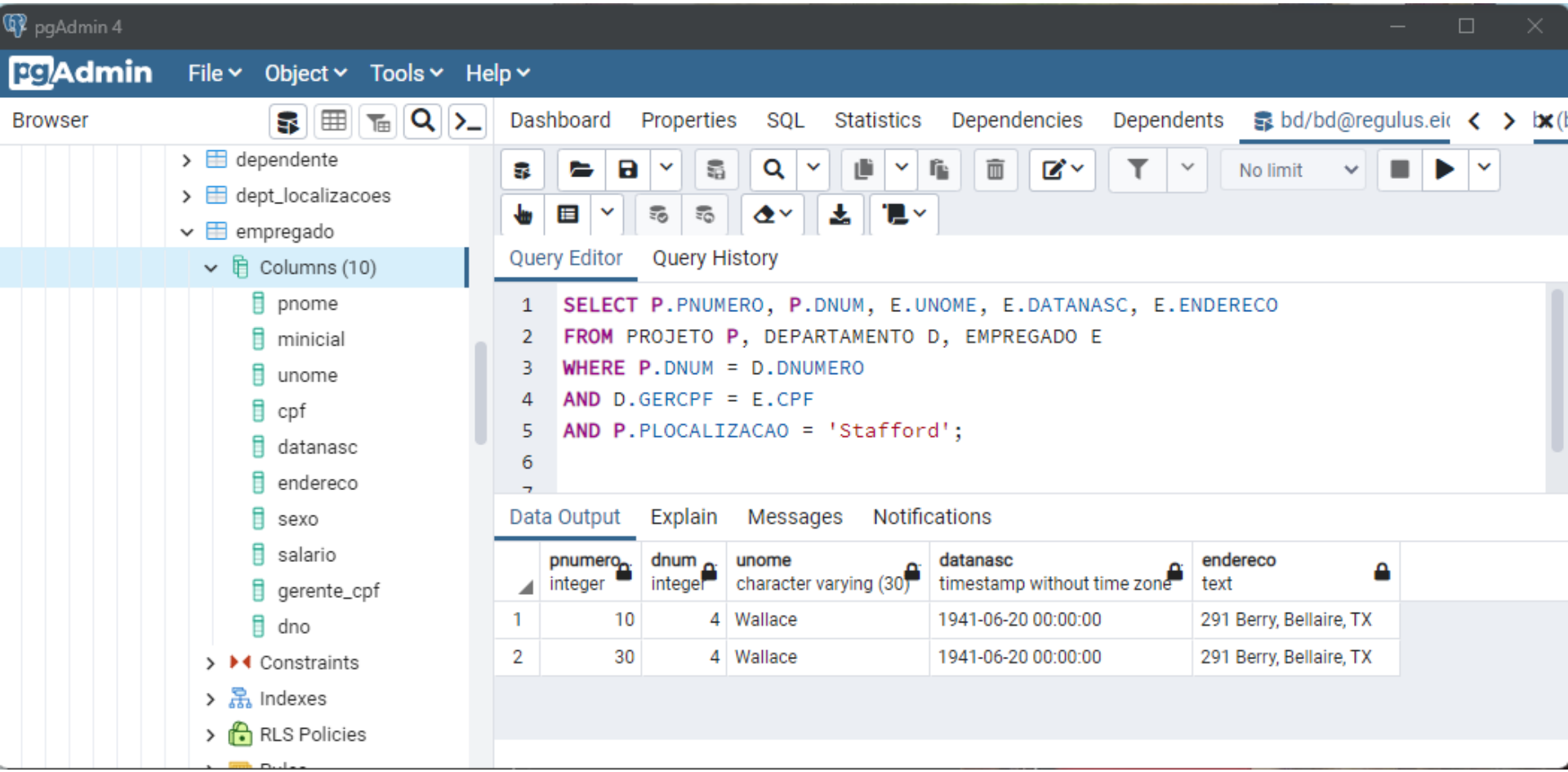

```
SELECT P.PNUMERO, P.PLOCALIZACAO, P.DNUM, D.DNOME
FROM PROJETO P, DEPARTAMENTO D
WHERE P.DNUM = D.DNUMERO
```

SQL- Apelidos




```
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



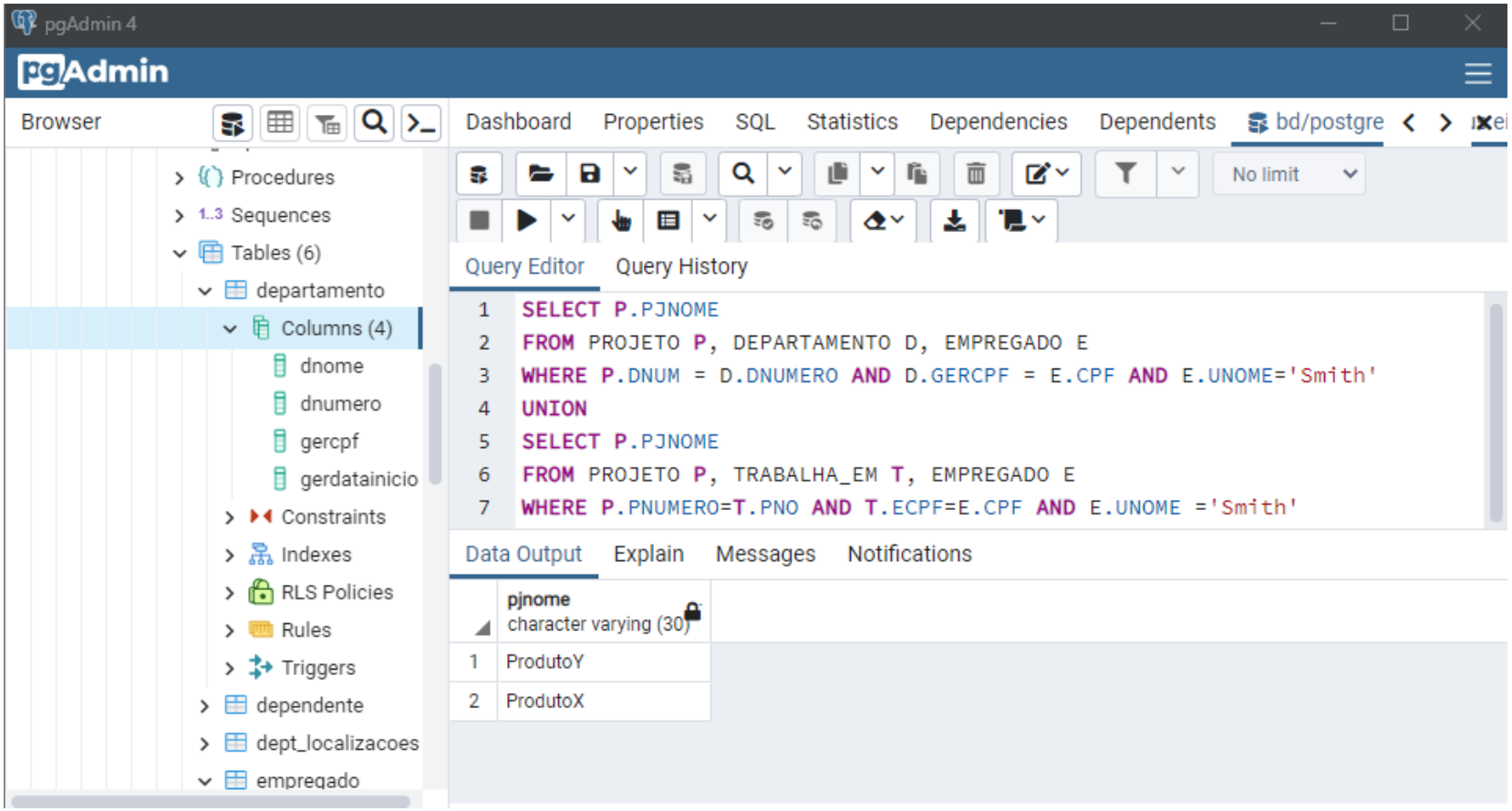

```
SELECT P.PNUMERO, P.DNUM, E.UNOME, E.DATANASC, E.ENDERECO
FROM PROJETO P, DEPARTAMENTO D, EMPREGADO E
WHERE P.DNUM = D.DNUMERO AND D.GERCPF = E.CPF AND P.PLOCALIZACAO = 'Stafford';
```

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

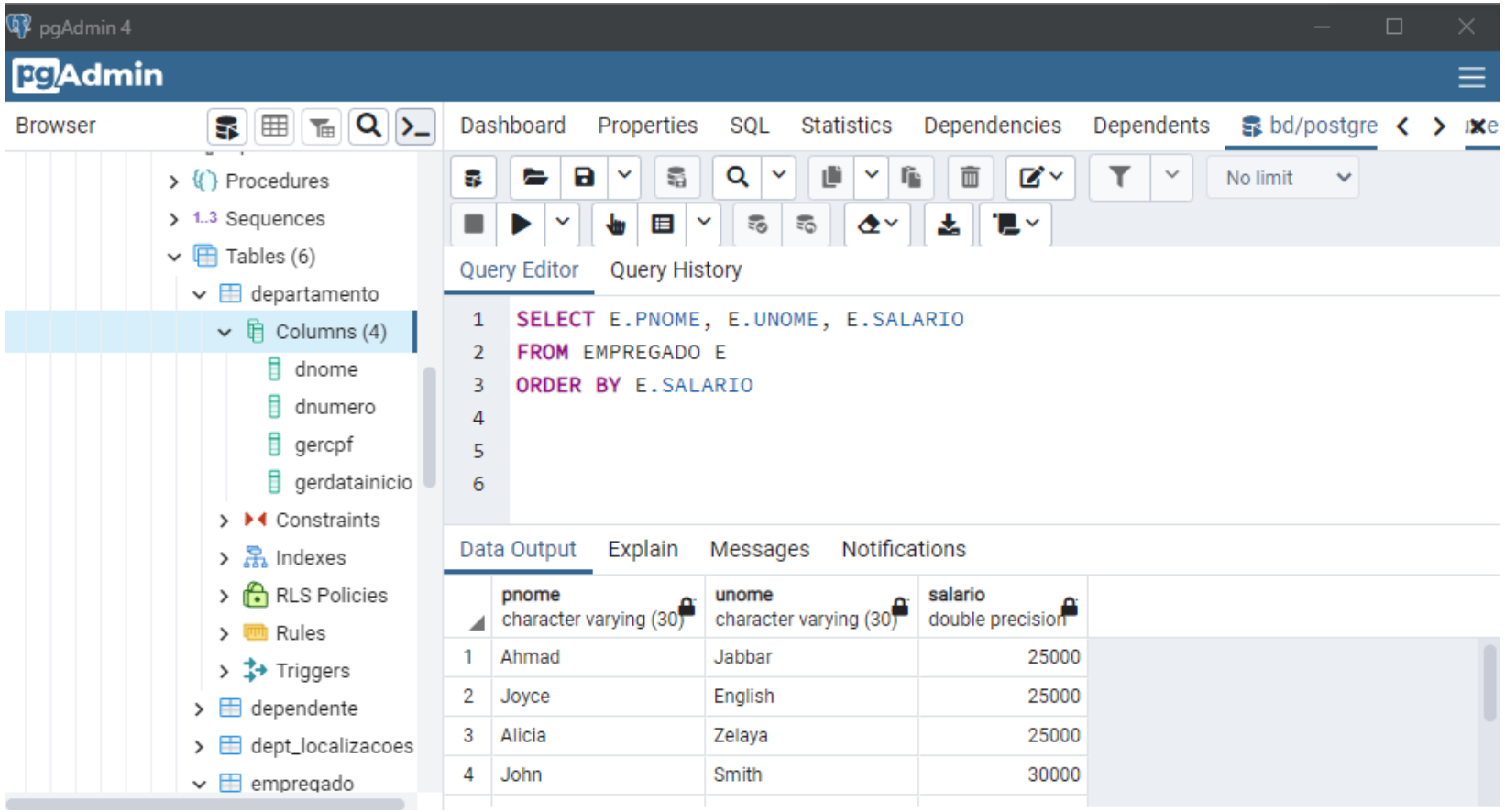
- Obtenha a lista do nome de todos os projetos que envolvem algum empregado cujo sobrenome é 'Smith' como trabalhador ou como gerente do departamento que controla o projeto



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

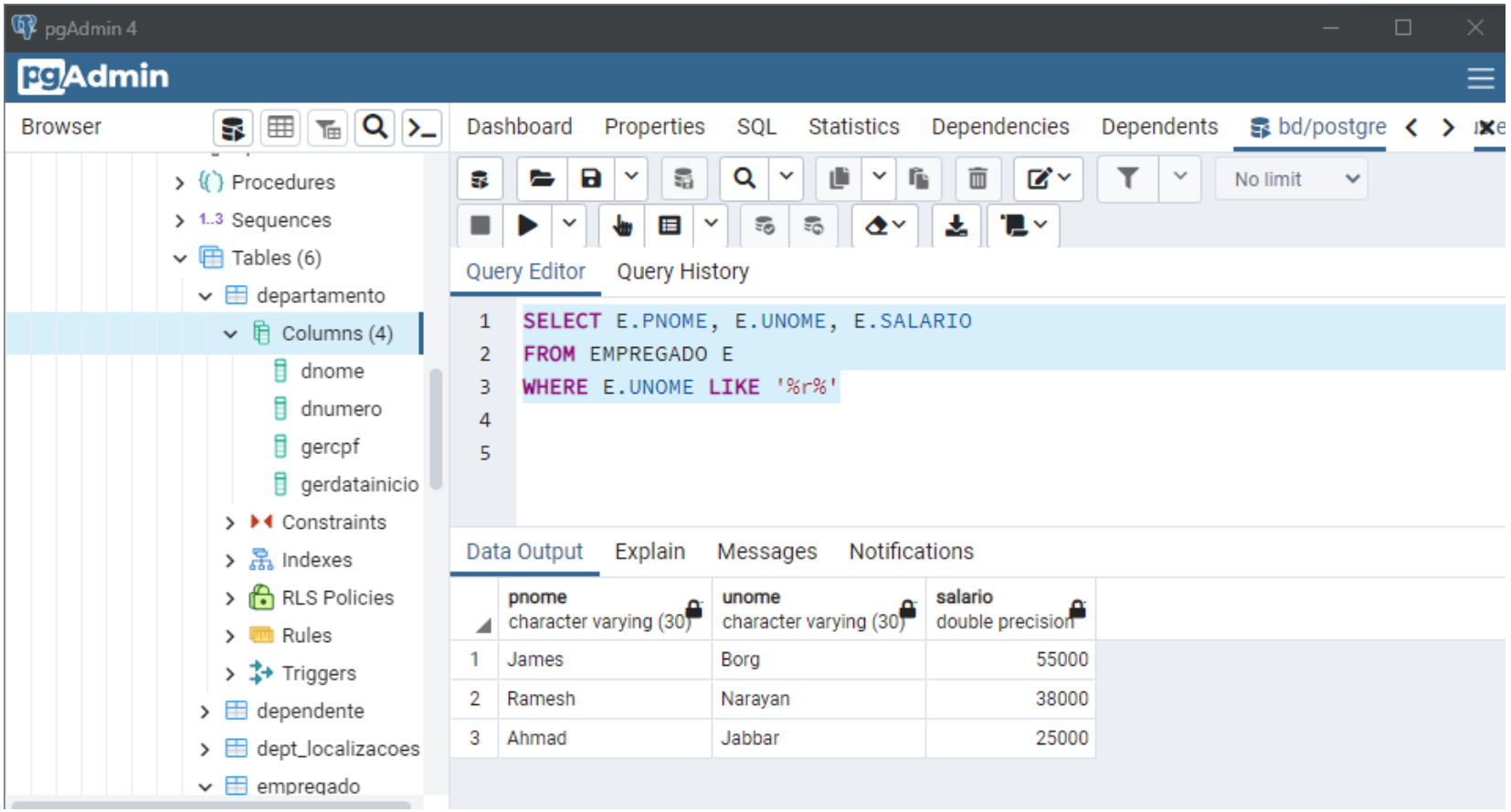
- Obter o nome dos empregados ordenados pelo salário




```
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’



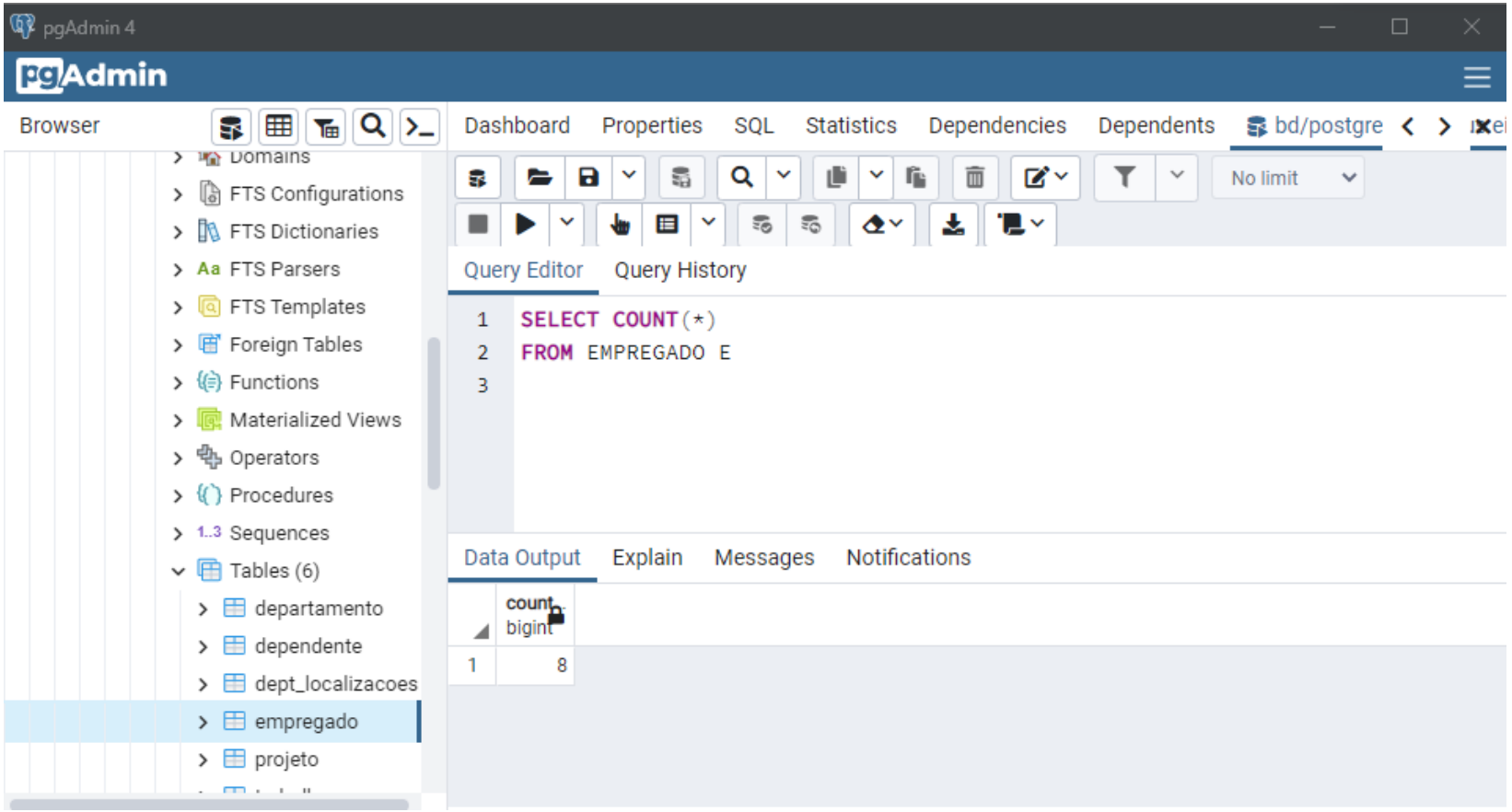

```
SELECT E.PNOME, E.UNOME, E.SALARIO
FROM EMPREGADO E
WHERE E.UNOME LIKE '%r%'
```

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(*)

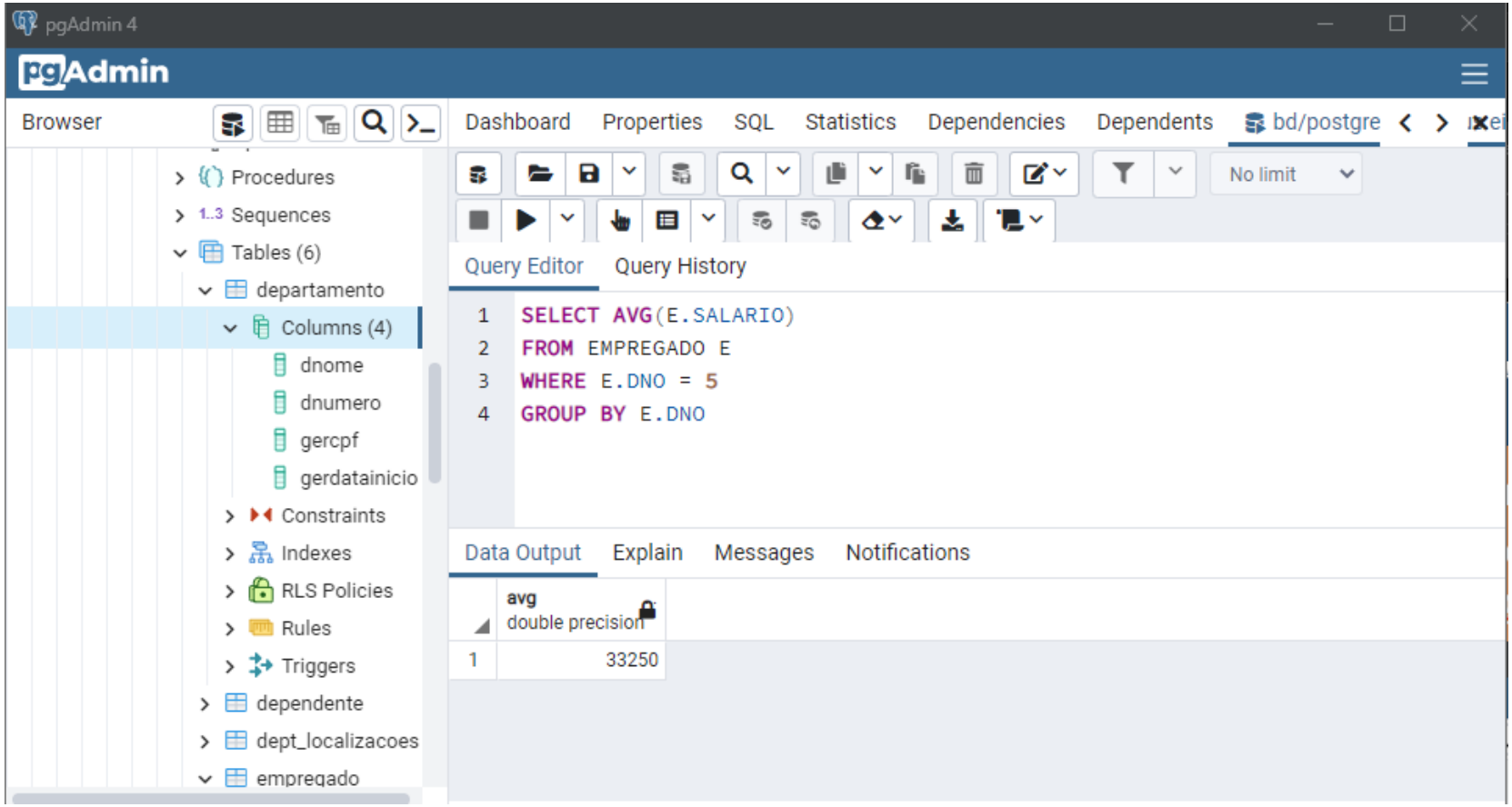
- Obtenha a quantidade de empregados



SELECT COUNT(*)
FROM EMPREGADO E

Funções agregadas – avg

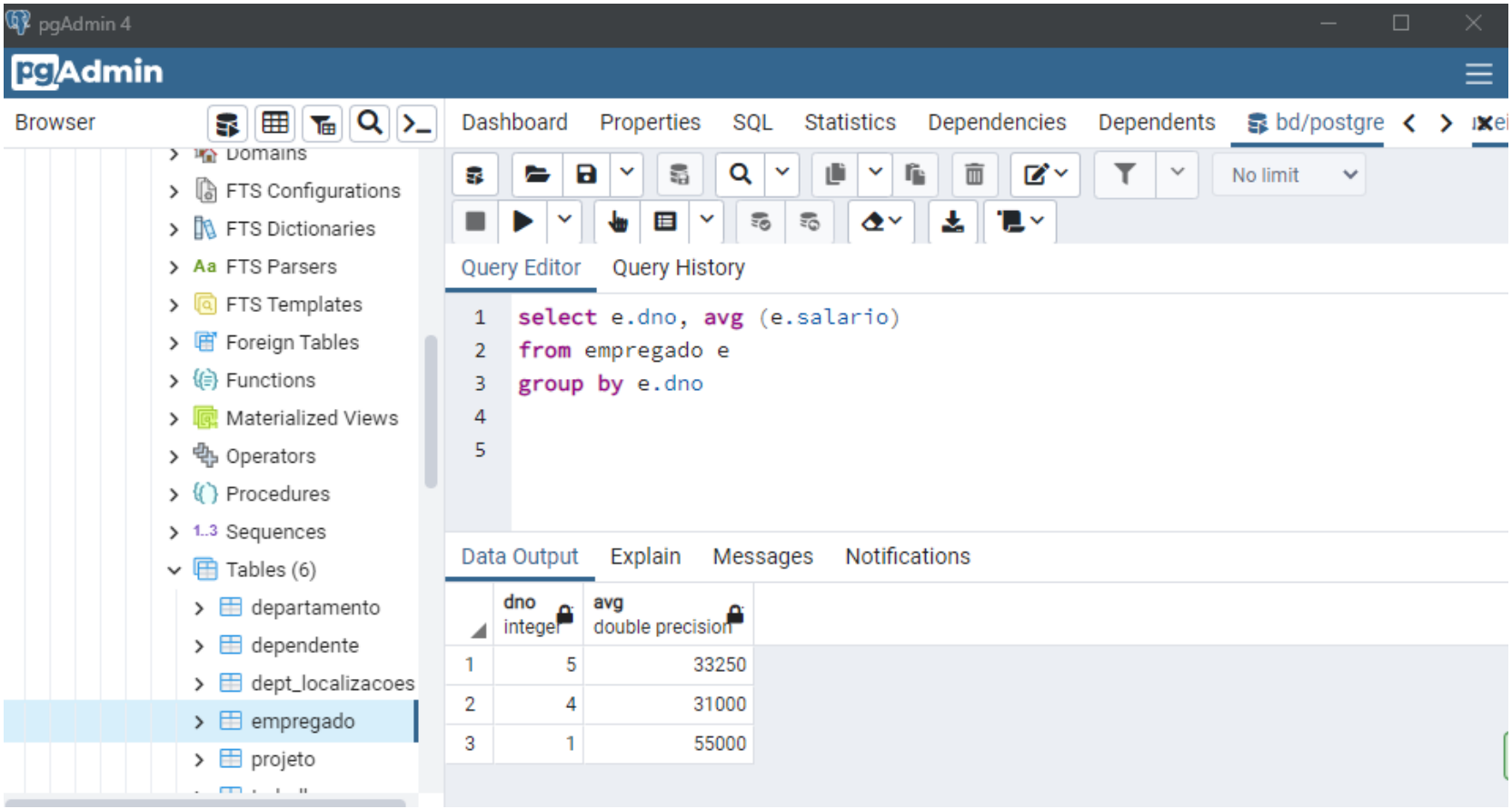
- Obtenha o salário médio dos empregados do departamento número 5




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

Funções agregadas – cláusula group by

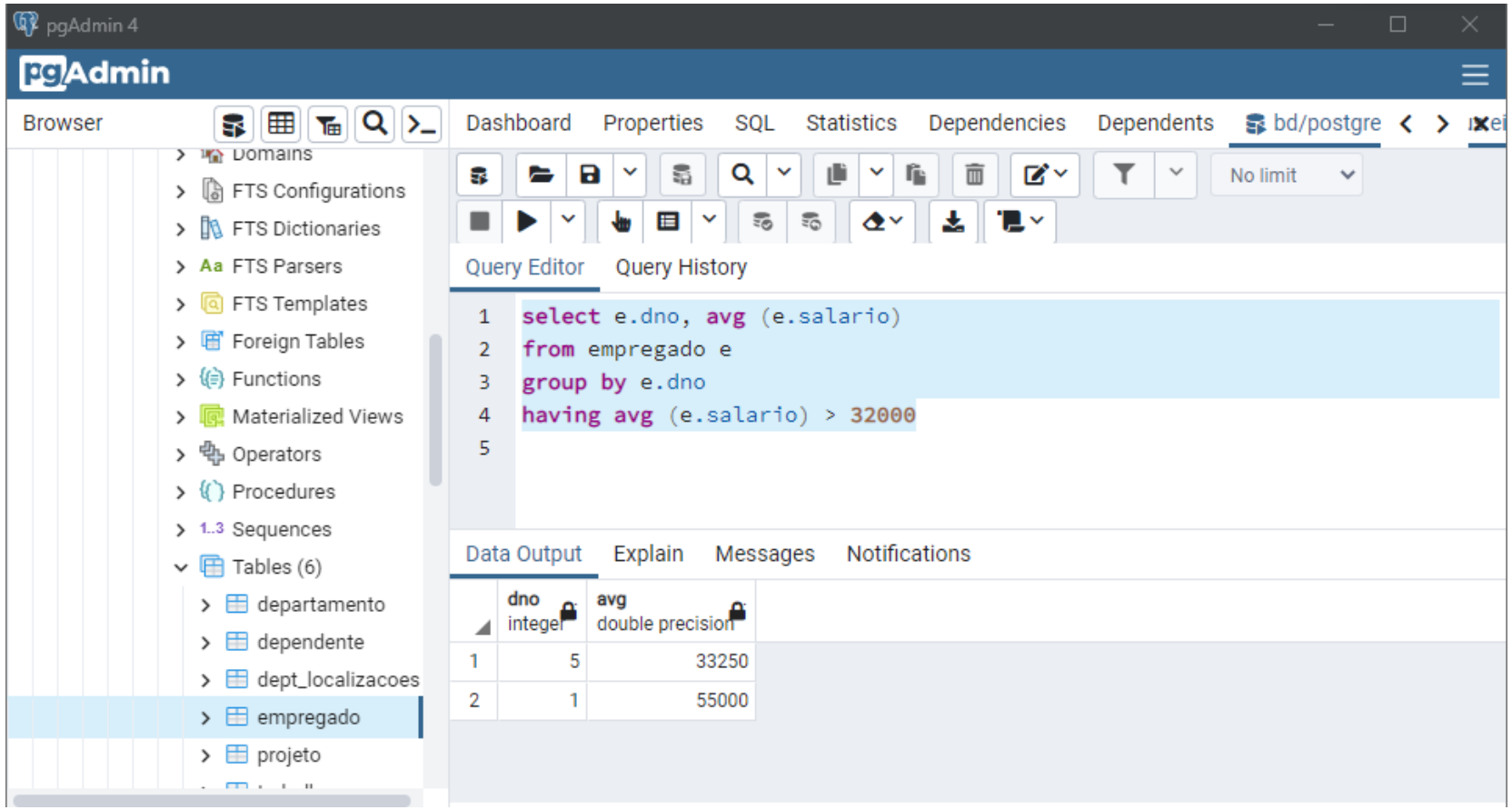
- Encontre a média salarial dos empregados para cada departamento
- Nota: Os atributos na cláusula select fora das funções agregadas precisam aparecer na lista group by




```
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

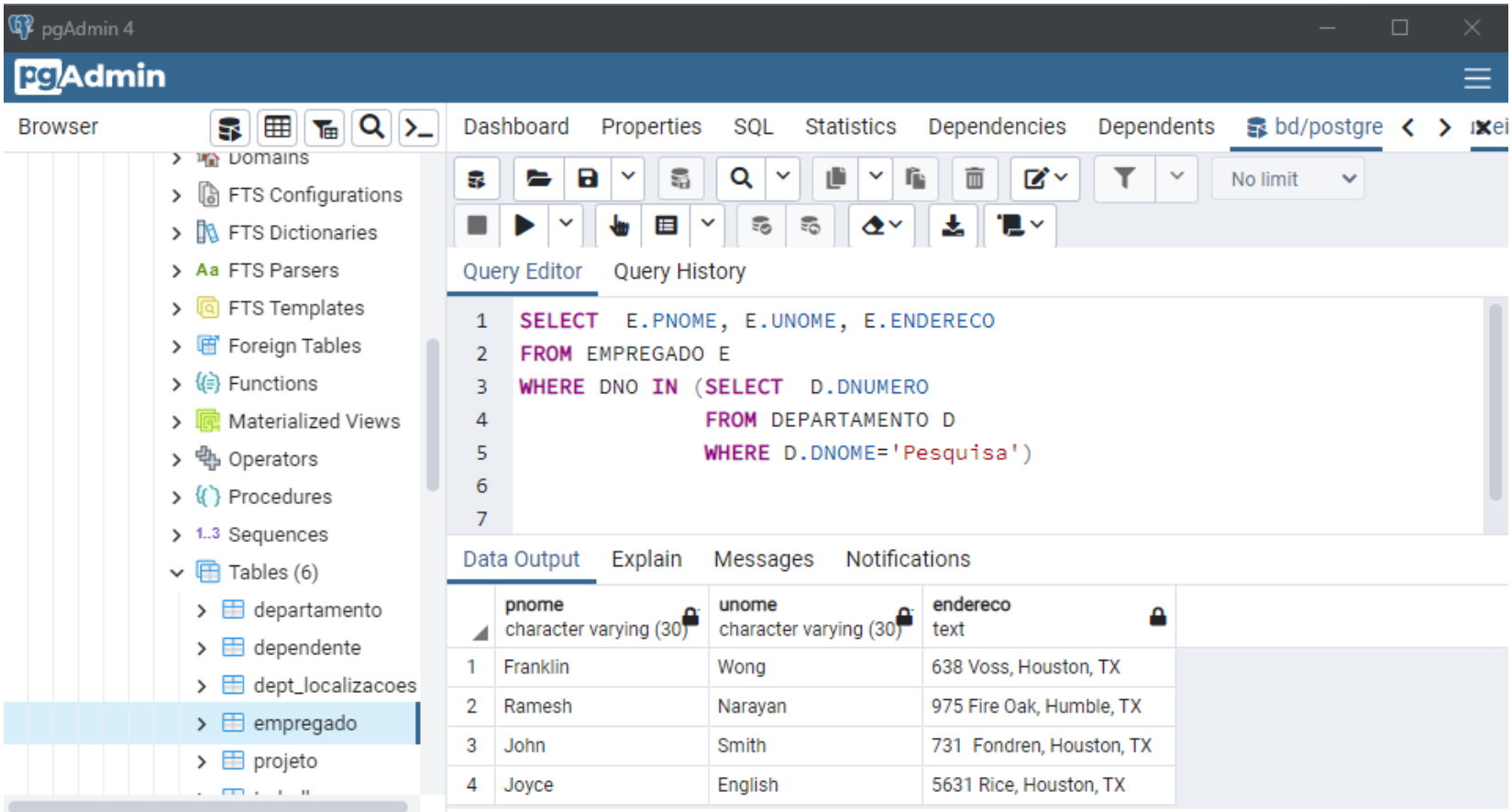


Consultas aninhadas

- 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

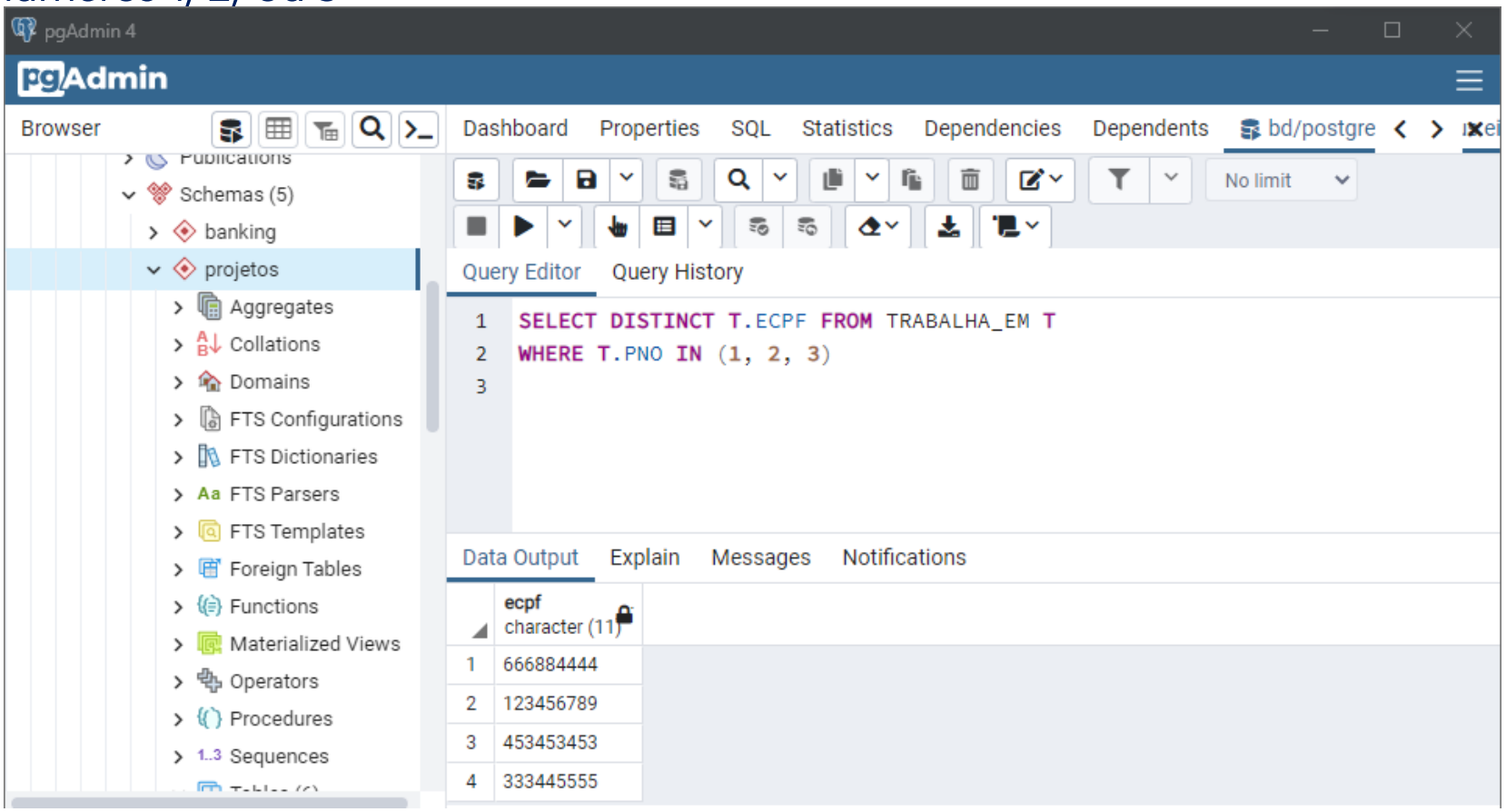
- Obtenha o nome e endereço de todos os empregados que trabalham no departamento de 'Pesquisa'




```
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



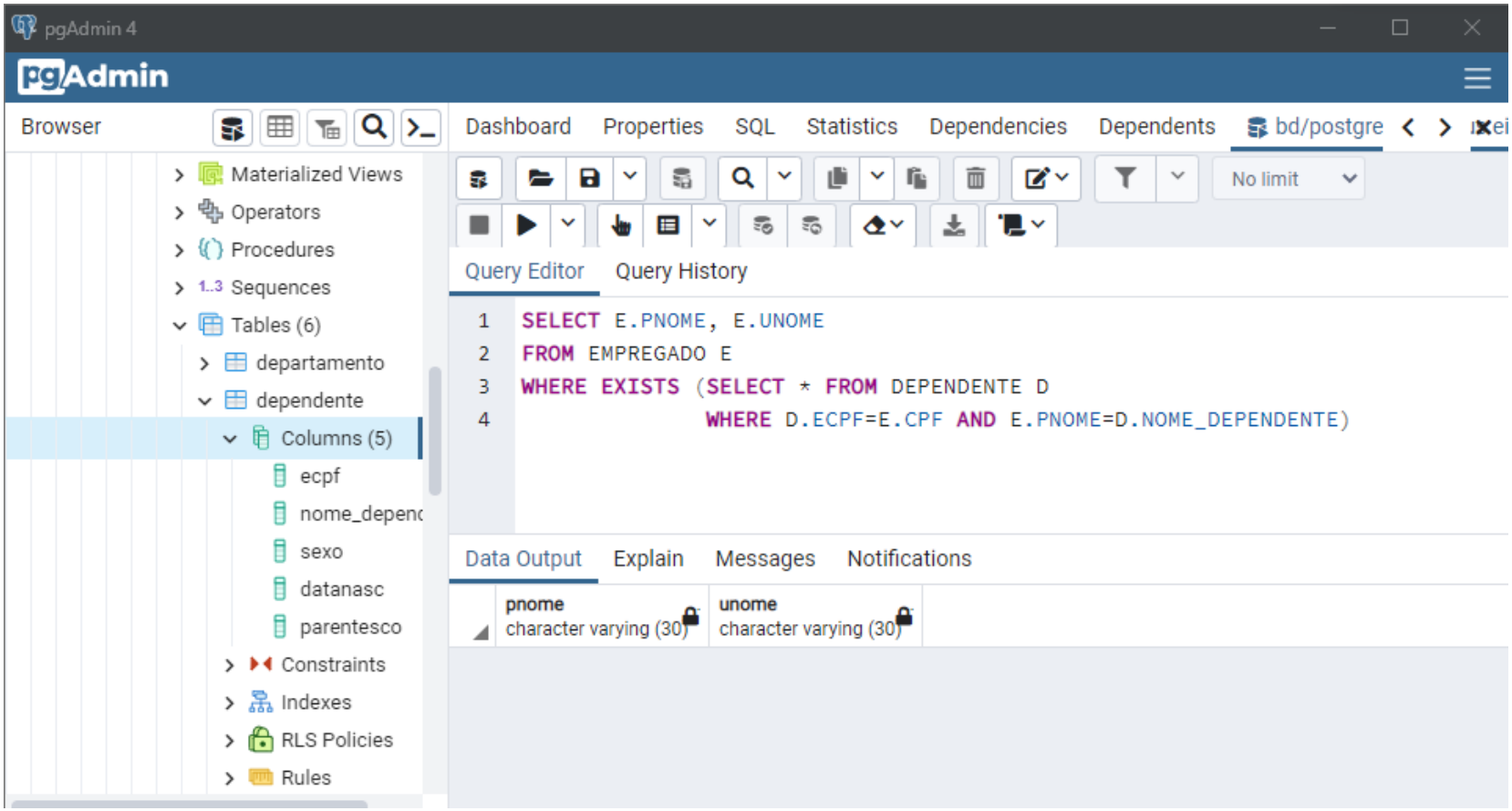

```
SELECT DISTINCT T.ECPF
FROM TRABALHA_EM T
WHERE T.PNO IN (1, 2, 3)
```

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

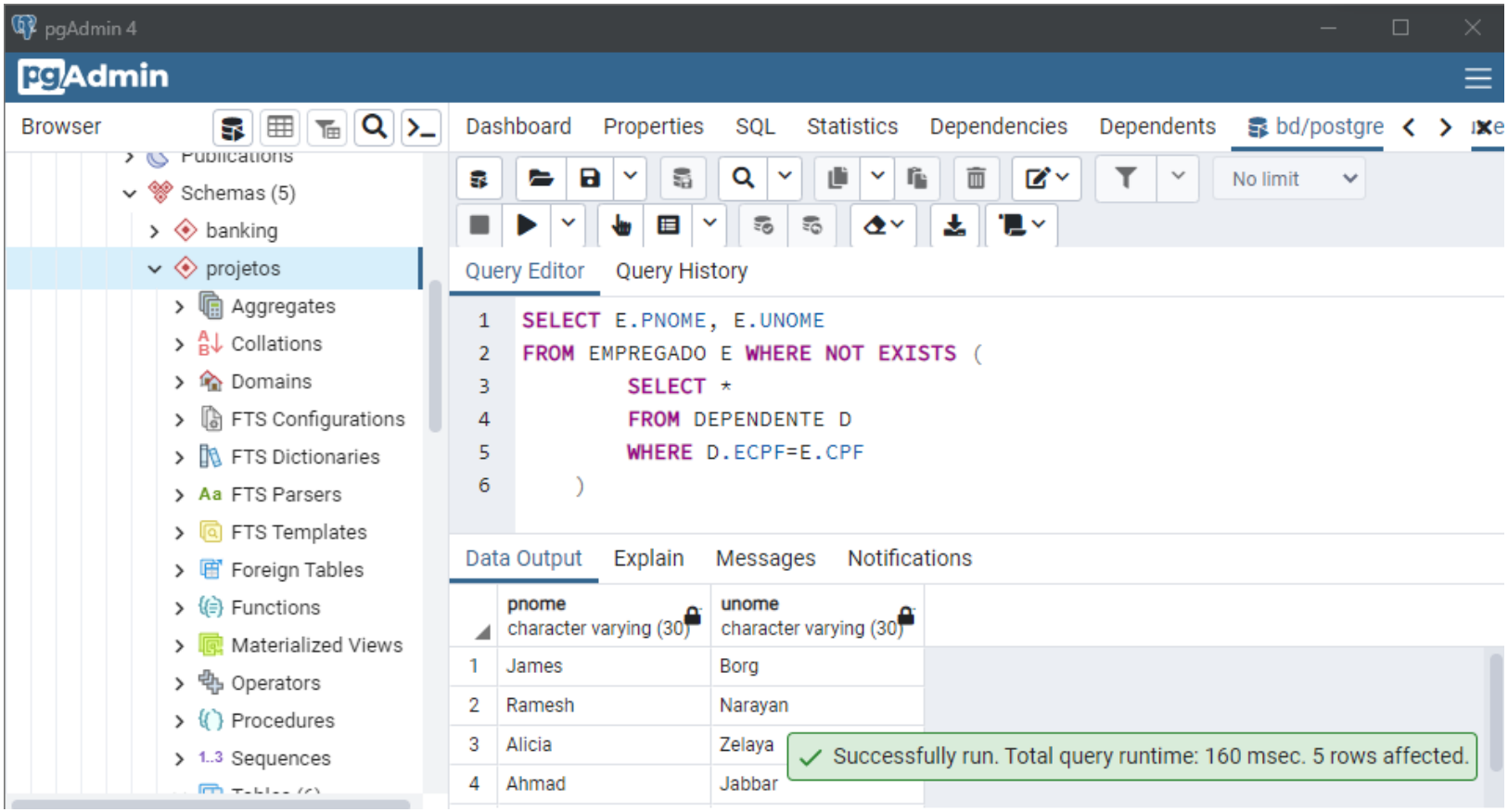
- EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada (contém uma ou mais tupla associadas)



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)

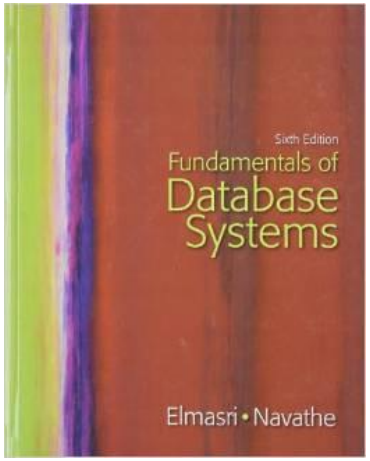
Uso do NOT EXISTS

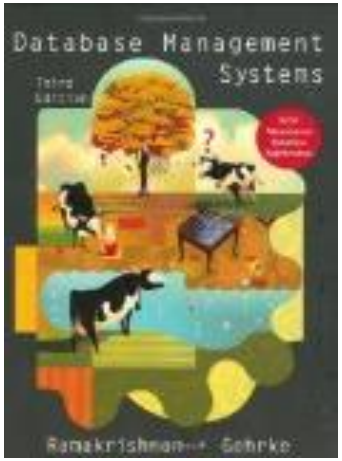
- NOT EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio (não contém tupla associada)

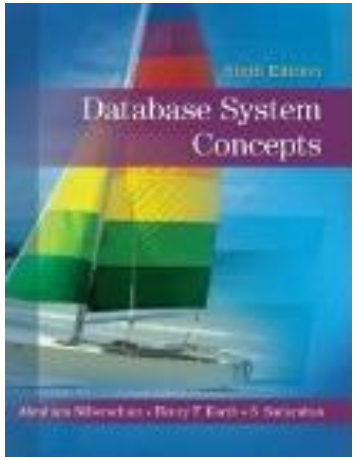



```
SELECT E.PNOME, E.UNOME
FROM EMPREGADO E
WHERE NOT EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF=E.CPF )
```

Referências







A maioria dos slides apresenta exemplos do livro Database System Concepts