Criando Bancos e Tabelas no PostgreSQL

14.5

Aula 3

Introdução a linguagem SQL

Introdução a linguagem SQL

SQL significa Structured Query Language, ou Linguagem de Consulta Estruturada.

SQL é a linguagem padrão* dos Bancos de Dados Relacionais.

Grande parte dos bancos de dados utilizam a sigla da linguagem no seu próprio nome:

- MySQL;
- PostgreSQL;
- Microsoft SQL Server;
- SQLite;
- outros...

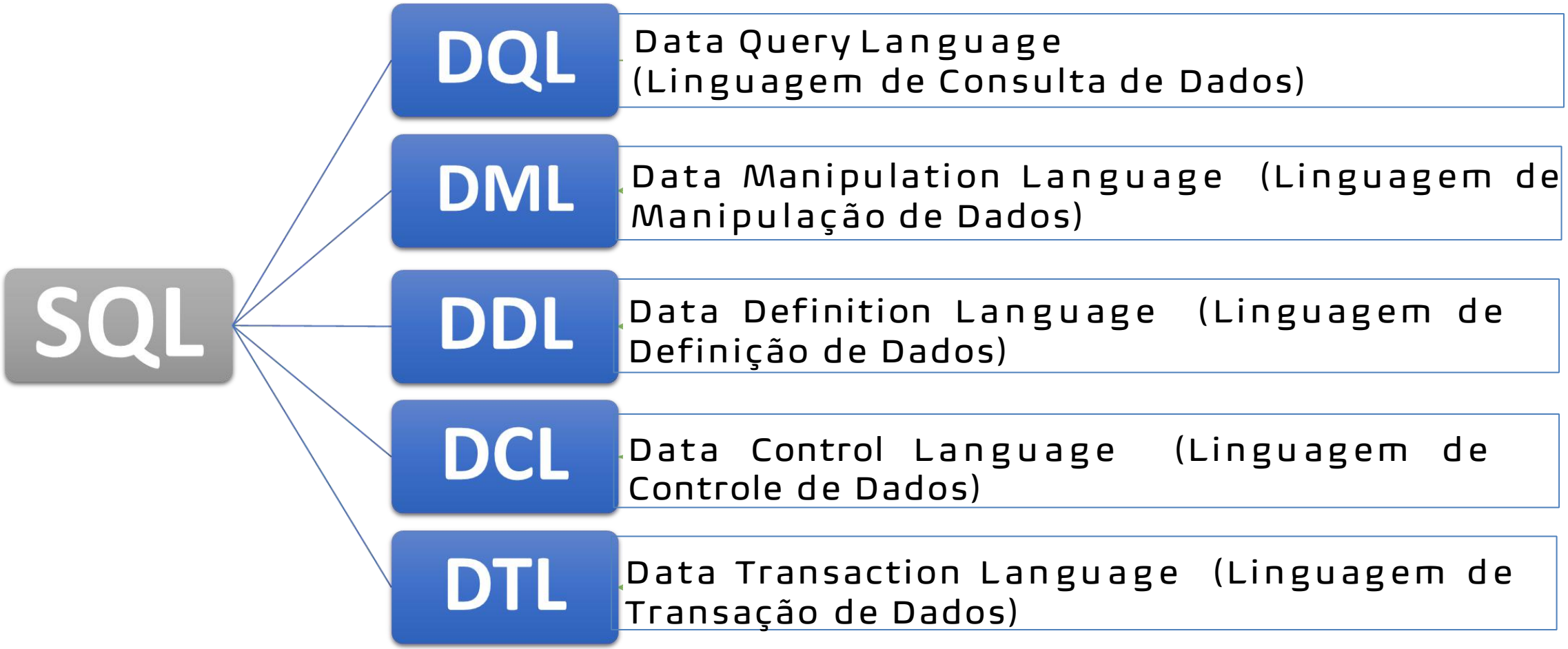
Introdução a linguagem SQL

A linguagem SQL foi inspirada na **Álgebra Relacional**

Os bancos de dados possuem um dialeto um pouco diferente em alguns comandos.

Introdução à Linguagem SQL

A linguagem SQL se divide em 5 subgrupos:



Introdução a linguagem SQL

Cada subgrupo **SQL** possui comandos próprios de execução e ao executar estes comandos sempre temos como resultado duas coisas:

- - O resultado da execução do comando;
- - Uma mensagem* de execução que pode ser de sucesso ou de erro;

OBSERVAÇÃO: Apesar de não obrigatório, costumamos a escrever os comandos SQL todos em letra maiúscula, o que ajuda a entender melhor nosso código já que o que for referente aos comandos SQL estarão destacados em maiúsculo e o que for referente aos nossos dados/tabelas e etc estarão em minúsculo.

Modelagem Conceitual, Lógica e Física

Os modelos de banco de dados são usados para descrever, mais detalhadamente, a estrutura de um banco de dados. Eles servem então como parte importante da documentação dos sistemas que auxiliam não somente os desenvolvedores que estão trabalhando no projeto mas também servem como documentação que pode ser entregue ao cliente que contratou o serviço de desenvolvimento.

Atenção: Um modelo de dados não informa quais dados estão armazenados em um banco de dados, mas sim apenas quais e que tipos de dados contém.

Estes modelos são baseados em três níveis: **Conceitual**, **Lógico** e **Físico**

Modelagem Conceitual, Lógica e Física

Modelo Conceitual:

- Este é o modelo de mais alto nível, ou seja, que está mais próximo dos usuários.
- O nível conceitual é desenvolvido com alto nível de abstração, a partir dos requisitos do sistema, extraídos na fase de levantamento de requisitos pelos analista de sistemas.
- Esse modelo pode ser elaborado de forma textual ou por meio de dois diagramas: Diagrama de Entidade e Relacionamento e/ou Diagrama de Classes*.

* Diagramas UML - Unified Modeling Language (Linguagem de Modelagem Unificada)

Modelagem Conceitual, Lógica e Física

Modelo Conceitual:

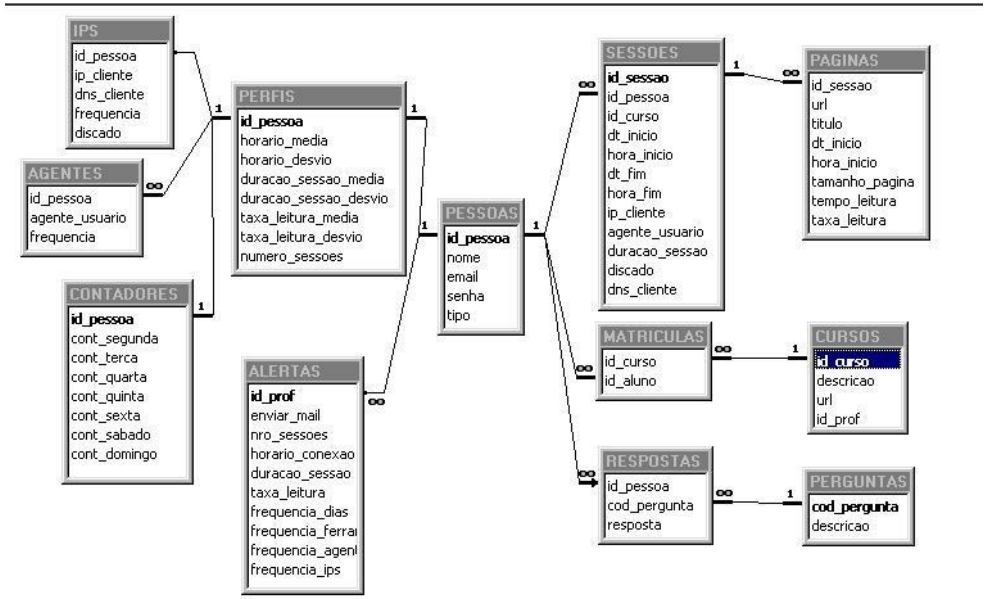
1)Clientes
Dados necessários: nome completo, tipo de pessoa (física ou jurídica), endereço, bairro, cidade, estado, telefone, email, nome de contato.

2)Pedido
Dados necessários: código do produto, quantidade, código do cliente, código do vendedor.

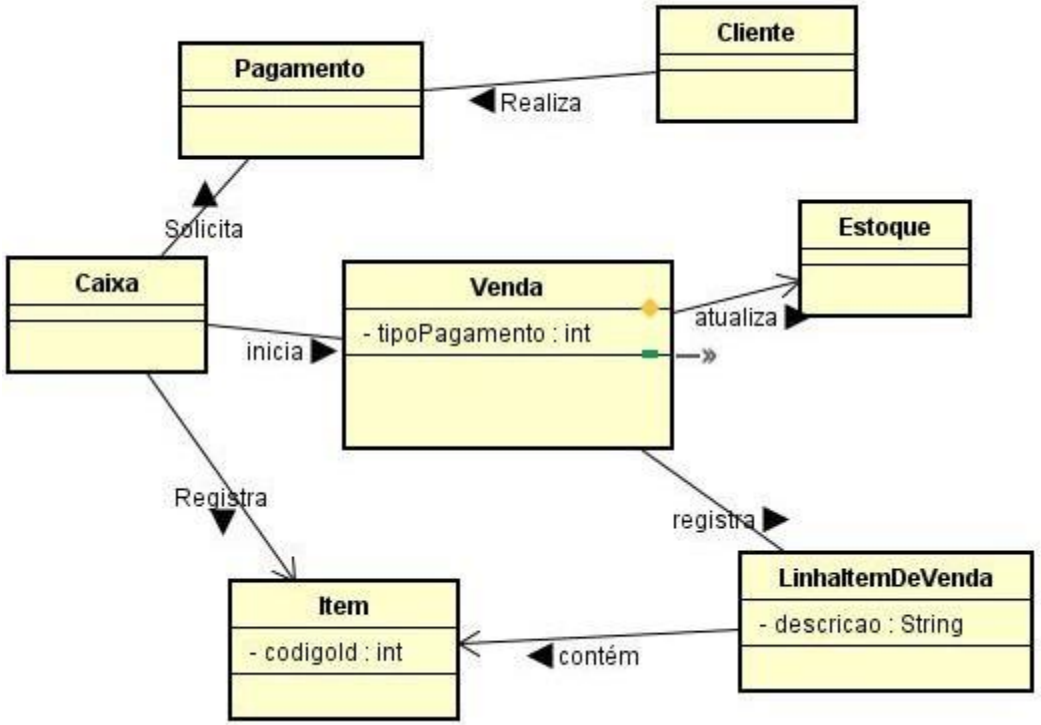
Exemplo Modelo Conceitual Textual

Modelagem Conceitual, Lógica e Física

Modelo Conceitual:

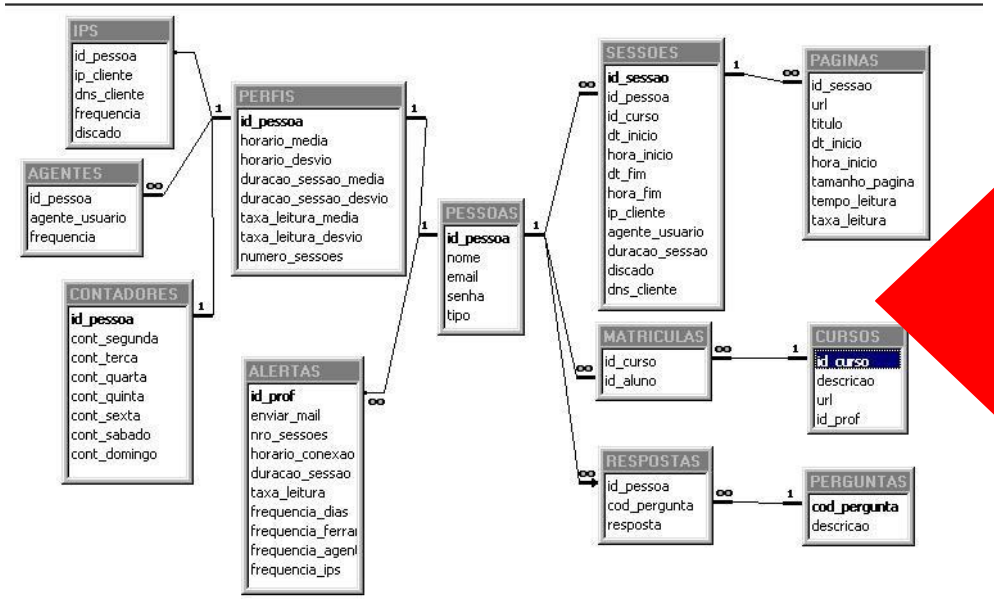


Exemplo de um DER - Diagrama de Entidade e Relacionamento / Modelo de Entidade e Relacionamento.



Exemplo de um Diagrama de Classes

Modelagem Conceitual, Lógica e Física



Este modelo é independente de implementação!

Modelagem Conceitual, Lógica e Física

Modelo Lógico:

- Este modelo descreve como os dados serão armazenados no banco de dados e também seus relacionamentos.
- Neste modelo ainda pode ser definida a tecnologia que será utilizada para armazenagem dos dados: Bancos de Dados Relacionais ou Bancos de Dados Não Relacionais

Modelagem Conceitual, Lógica e Física

Modelo Lógico:

Tipo de produto

Código	Descrição
1	Computador
2	Impressora

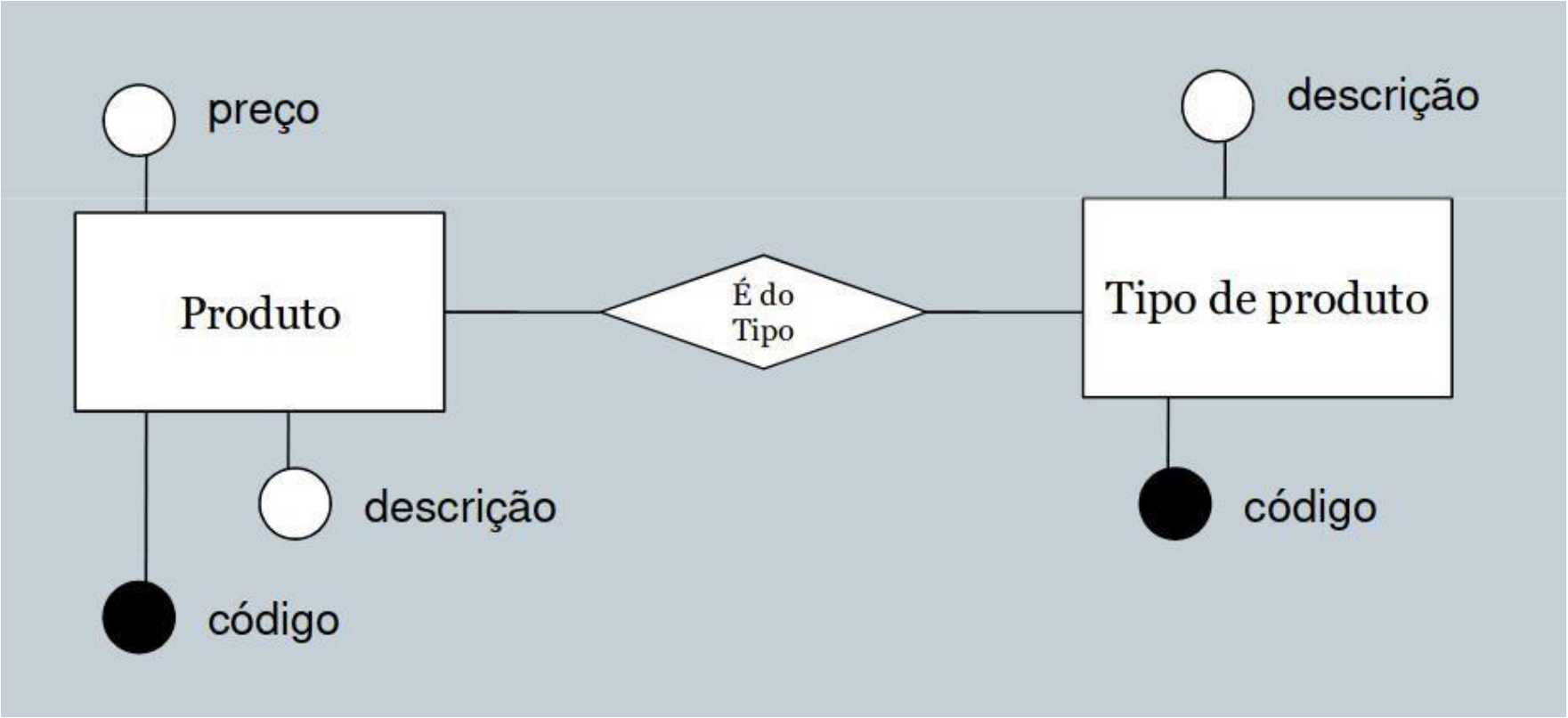
Produto

Código	Descrição	Preço	CódigoDoTipo
10	Desktop	1.200,00	1
20	Laptop	1.800,00	1
30	Impr. Jato Tinta	300,00	2
40	Impr. Laser	500,00	2

Forma de Representação 1: Exemplo de um Banco de Dados Relacional. Note que temos basicamente a definição dos nomes de tabela, suas colunas e exemplificação dos dados que serão armazenados.

Modelagem Conceitual, Lógica e Física

Modelo Lógico:



Forma de Representação 2: Exemplo de um Banco de Dados Relacional. Note que temos basicamente a definição dos nomes de tabela, suas colunas e definição das chaves de cada tabela.

Modelagem Conceitual, Lógica e Física

Modelo Físico:

- Também chamado de Modelo de Implementação, descreve, por meio de alguma linguagem (comumente SQL), como será feita a armazenagem do banco.
- Neste nível se escolhe qual Sistema Gerenciador de Banco de Dados (SGBD) será utilizado, levando em consideração o modelo lógico adotado.

Modelagem Conceitual, Lógica e Física

Modelo Físico:

Cadastro de Paciente

Nome do campo	Tipo de Dado	Tamanho do campo
Código do Paciente	Numérico	5 dígitos
Nome do Paciente	Alfanumérico	50 caracteres
Endereço	Alfanumérico	50 caracteres
Bairro	Alfanumérico	40 caracteres
Cidade	Alfanumérico	40 caracteres
Estado	Alfanumérico	2 caracteres
CEP	Alfanumérico	9 caracteres
Data de Nascimento	Data	10 caracteres

Exemplo de detalhamento de colunas (campos) de uma tabela na preparação para o modelofísico.

Modelagem Conceitual, Lógica e Física

Modelo Físico:

```sql
CREATE TABLE `turma` (
`idturma` INTEGER(4) NOT NULL AUTO_INCREMENT,
`capacidade` INTEGER(2) NOT NULL,
`idProfessor` INTEGER(4) NOT NULL,
PRIMARY KEY (`idturma`),
FOREIGN KEY(`idProfessor`) REFERENCES professor(idProfessor),
UNIQUE KEY `idturma` (`idturma`)
)
```

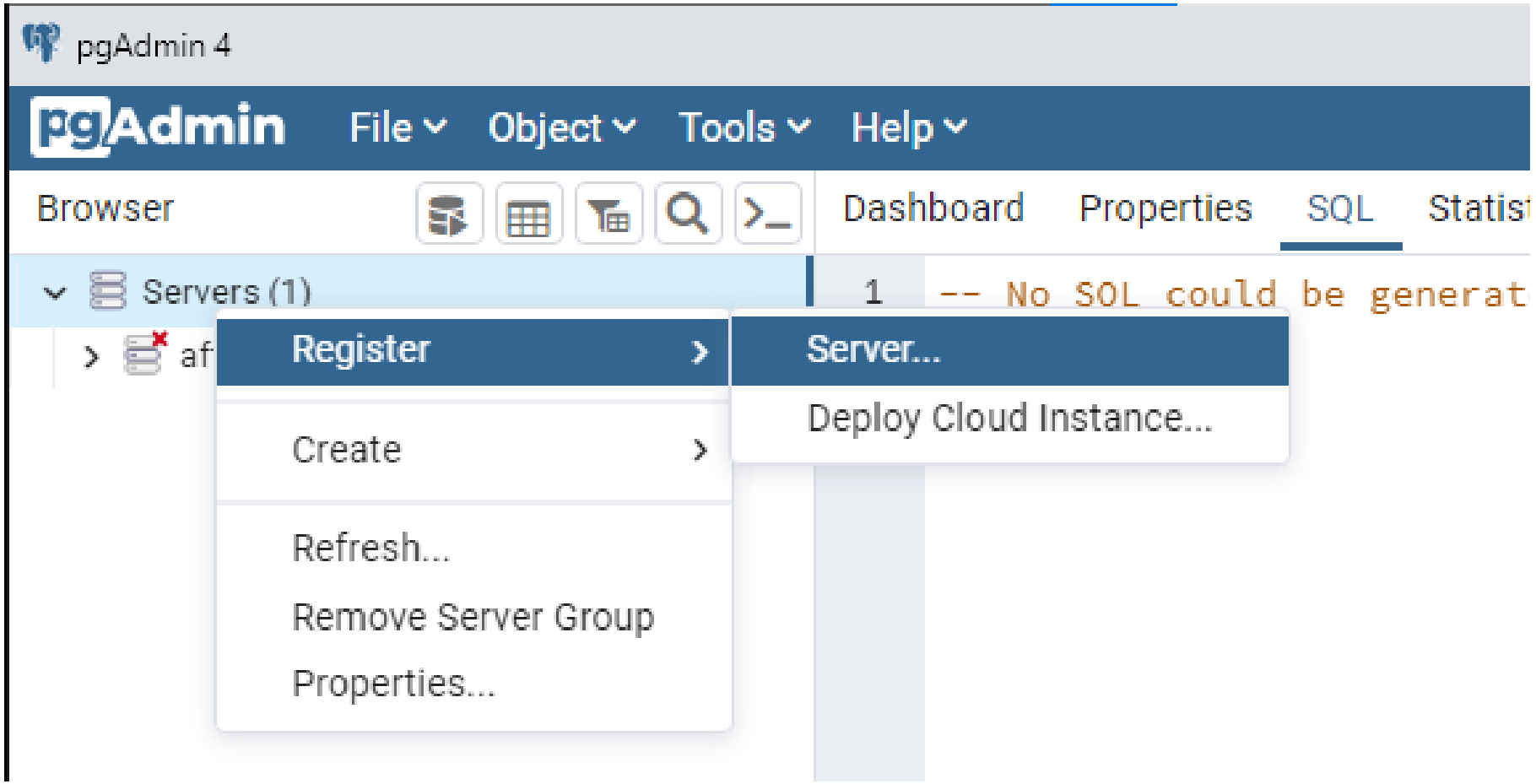
```sql
CREATE TABLE `professor` (
`idProfessor` INTEGER(4) NOT NULL AUTO_INCREMENT,
`telefone` INTEGER(10) NOT NULL,
`nome` CHAR(80) COLLATE NOT NULL DEFAULT '',
PRIMARY KEY (`idProfessor`),
UNIQUE KEY `idProfessor` (`idProfessor`)
)
```

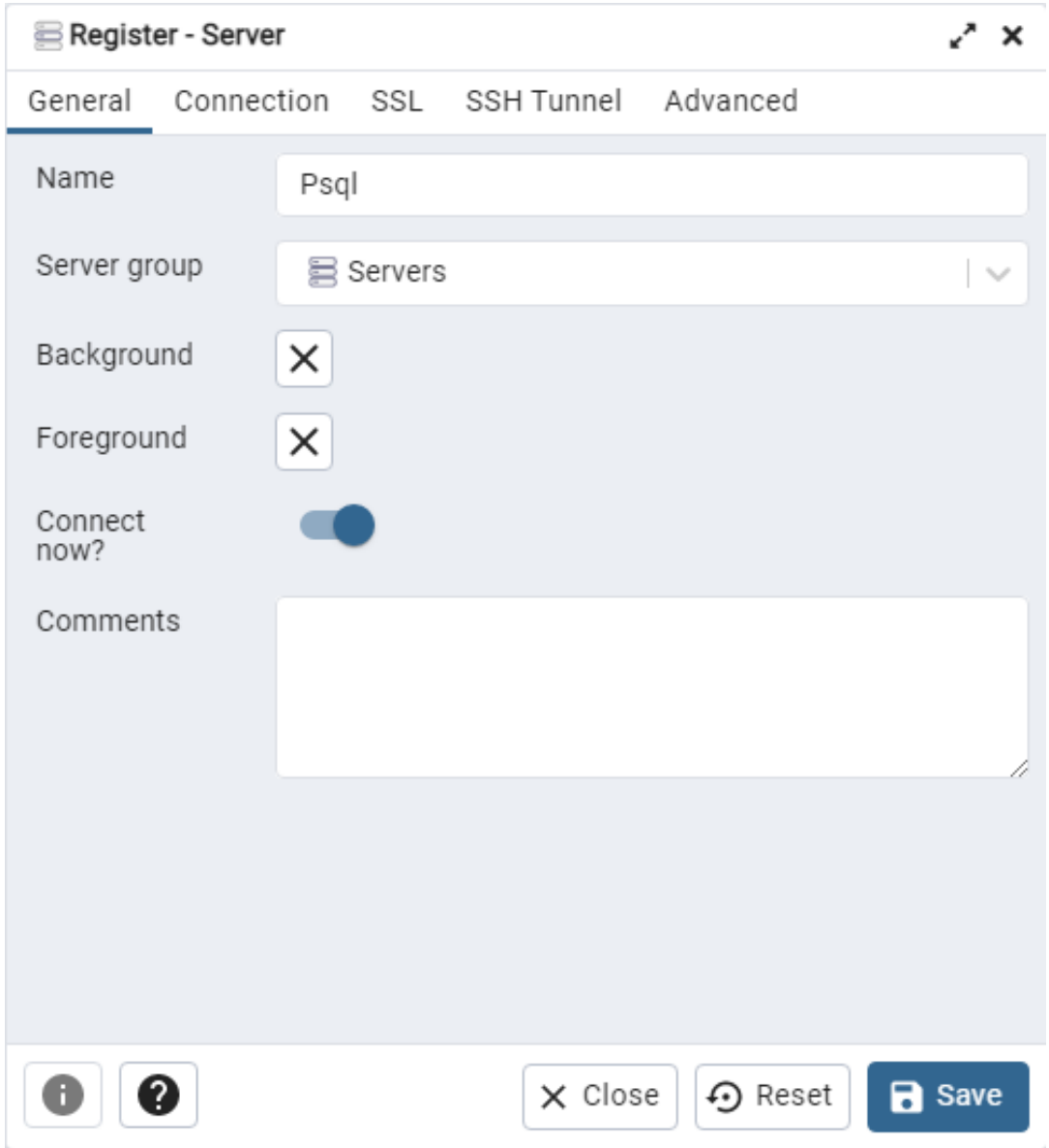
Exemplo de utilização da linguagem SQL para criação de tabelas, com suas colunas, tipos de dados e relacionamentos.

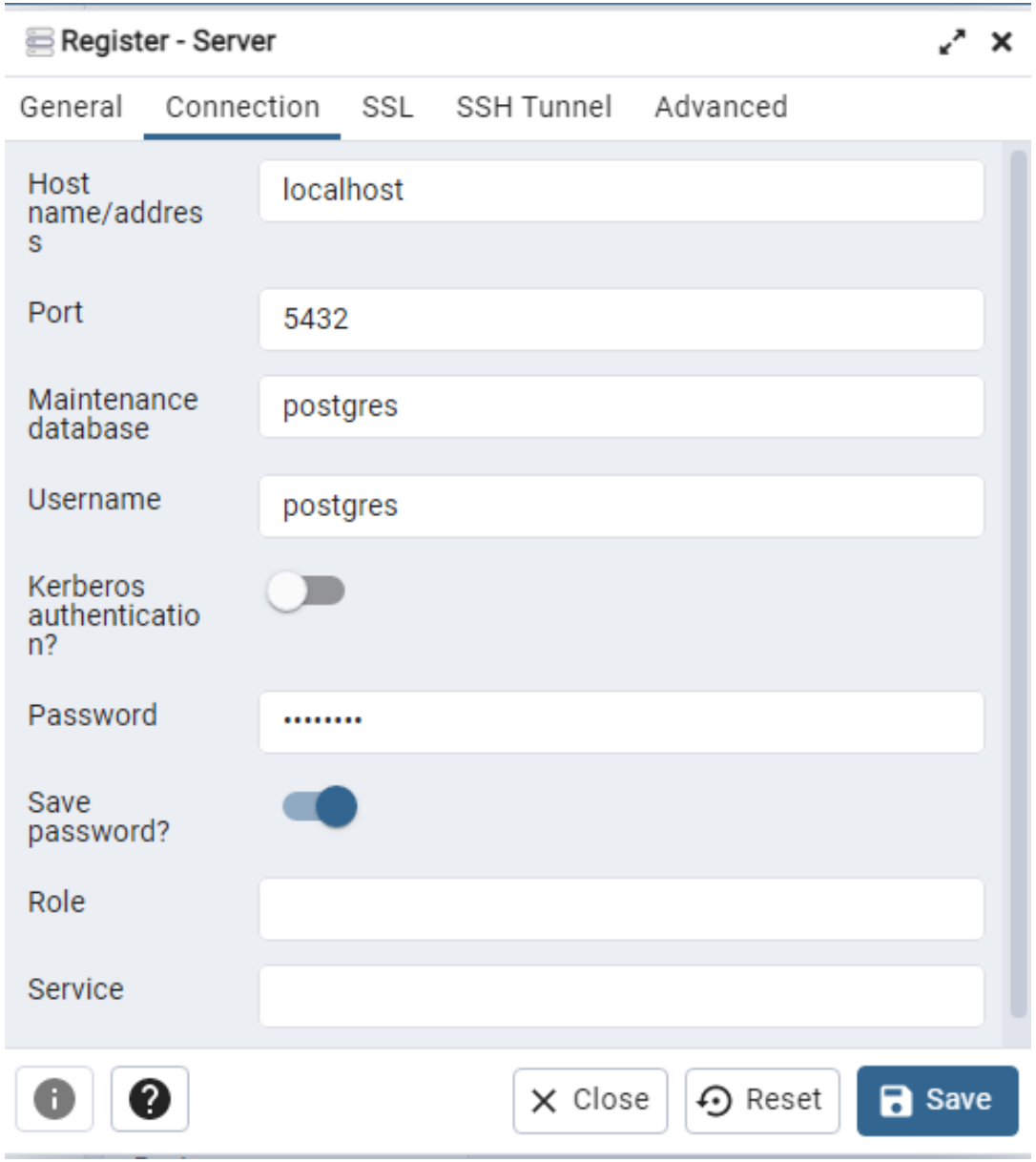
Exemplo prático

```
CREATE TABLE tipos_produto(
    id SERIAL PRIMARY KEY,
    descricao CHARACTER VARYING(50) NOT NULL
);
CREATE TABLE Tipo_produtos(
     id SERIAL PRIMARY KEY,
     descricao CHARACTER VARYING(50) NOT NULL
);
```

Lembrando

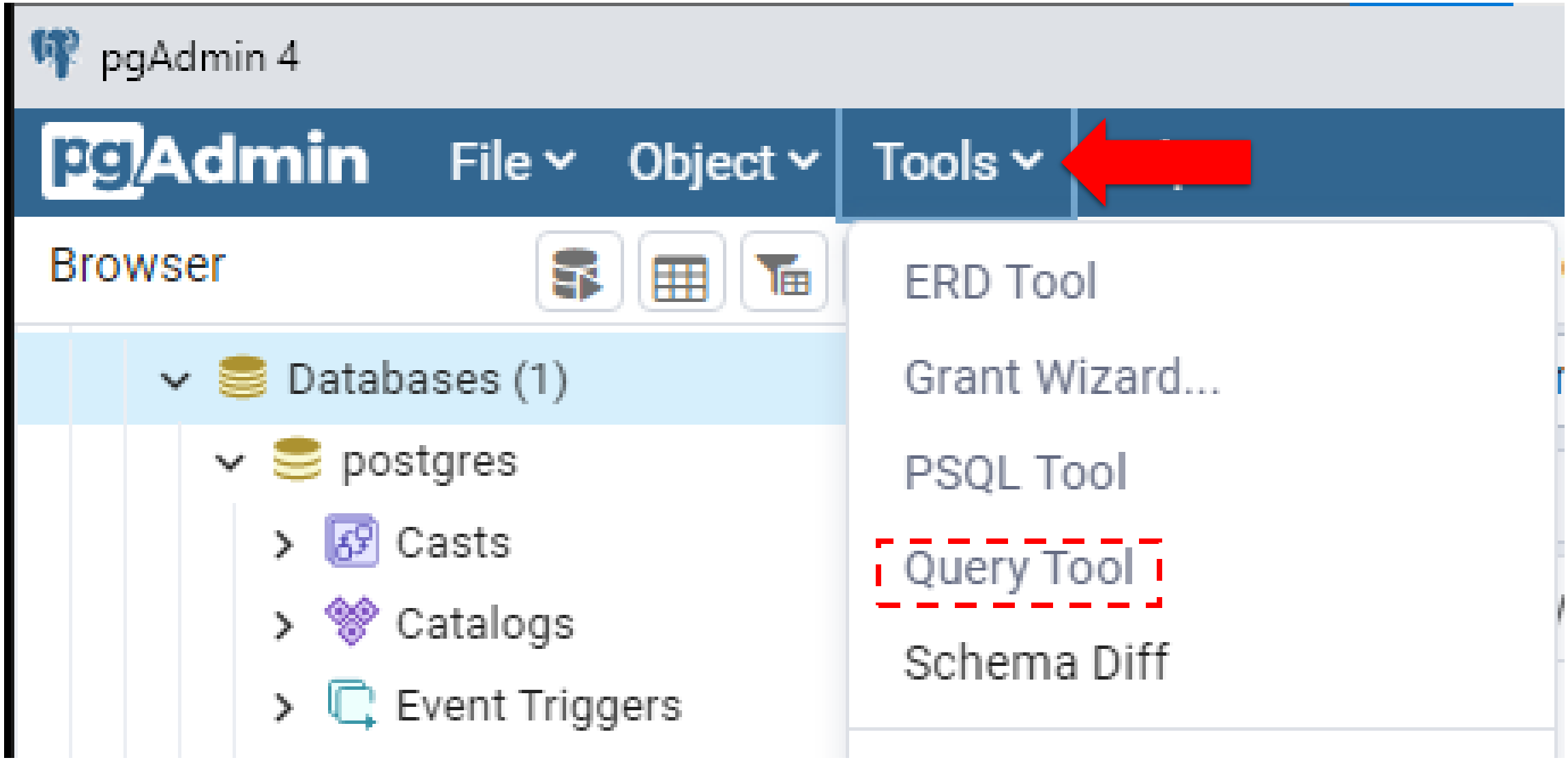

pgAdmin 4
pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Servers (1)
1
-- No SQL could be generat
Register
Server...
Deploy Cloud Instance...
Create
Refresh...
Remove Server Group
Properties...

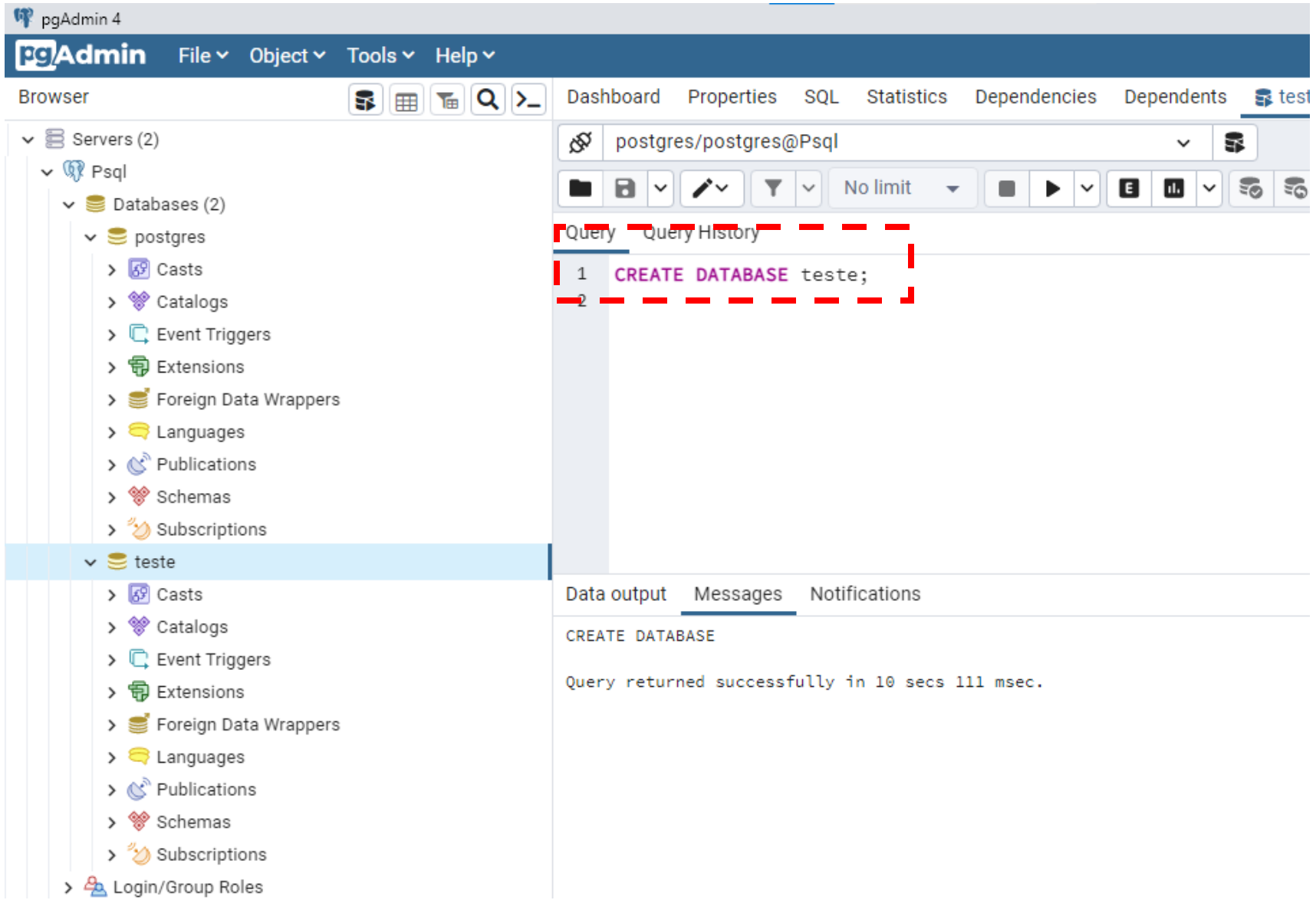

Register - Server
General
Connection
SSL
SSH Tunnel
Advanced
Name
Psql
Server group
Servers
Background
Foreground
Connect now?
Comments
Close
Reset
Save


Register - Server
General
Connection
SSL
SSH Tunnel
Advanced
Host name/address
localhost
Port
5432
Maintenance database
postgres
Username
postgres
Kerberos authentication?
Password
Save password?
Role
Service
Close
Reset
Save

```
CREATE TABLE Tipo_produtos(
id SERIAL PRIMARY KEY,
descricao CHARACTER VARYING(50) NOT NULL
);
```

```
CREATE TABLE  produtos(
    id SERIAL PRIMARY KEY,
    descricao CHARACTER VARYING(50) NOT NULL,
    preco MONEY NOT NULL,
    id_tipo_produto INT REFERENCES tipos_produto(id) NOT NULL
);
```


pgAdmin 4
pgAdmin
File
Object
Tools
Browser
ERD Tool
Grant Wizard...
PSQL Tool
Query Tool
Schema Diff
Databases (1)
postgres
Casts
Catalogs
Event Triggers


pgAdmin 4
pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
test
Servers (2)
Psql
Databases (2)
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
teste
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
Login/Group Roles
postgres/postgres@Psql
No limit
Query
Query History
1 CREATE DATABASE teste;
2
Data output
Messages
Notifications
CREATE DATABASE
Query returned successfully in 10 secs 111 msec.

Dashboard Properties SQL Statistics Dependenci

postgres/postgres@Psql

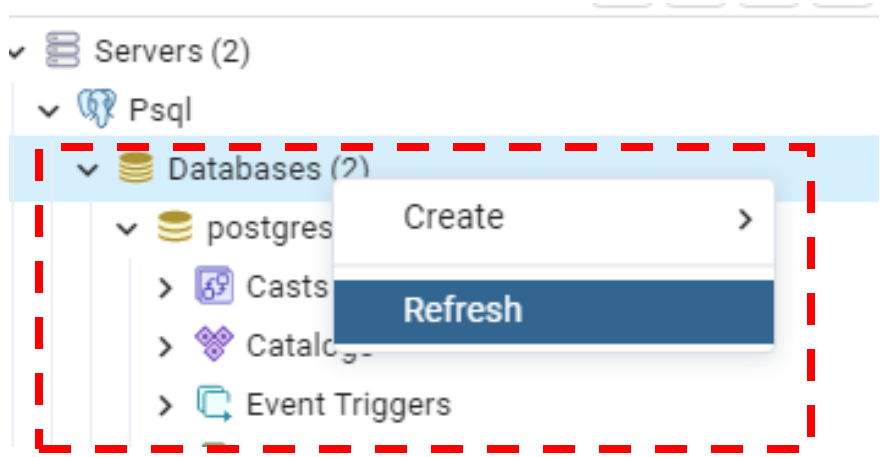
No limit

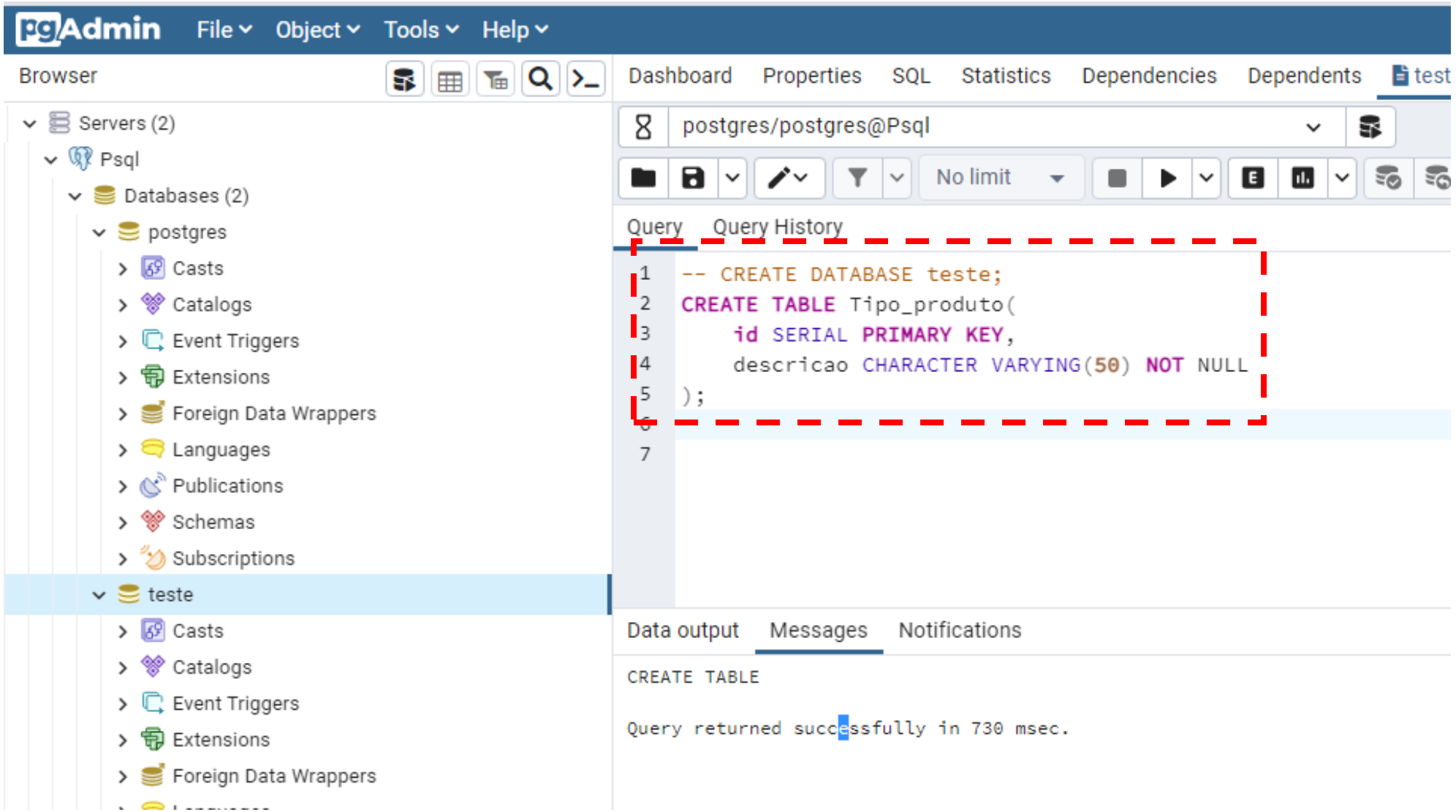
Query Query History

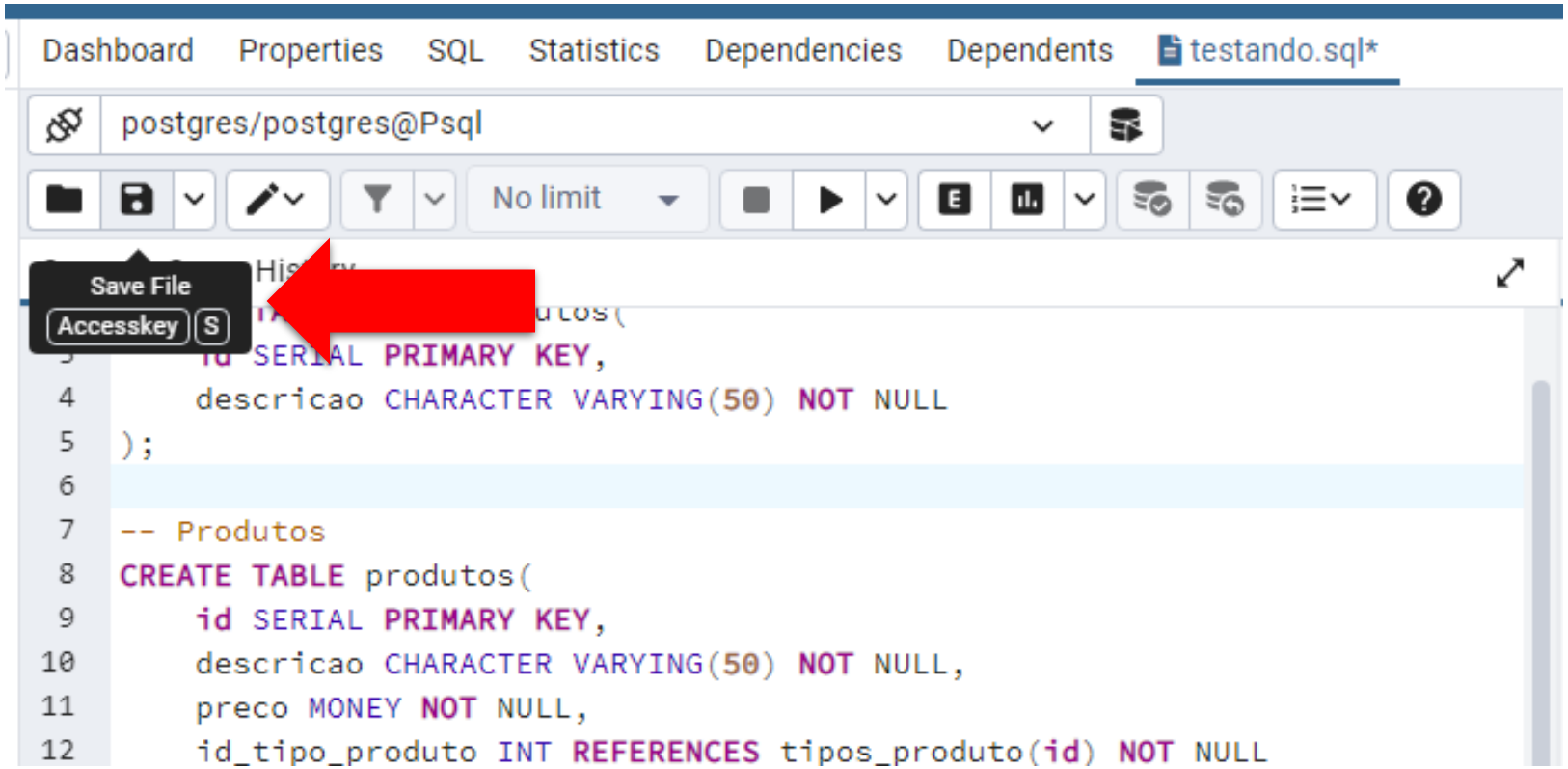
```
1  CREATE DATABASE teste;
2
```

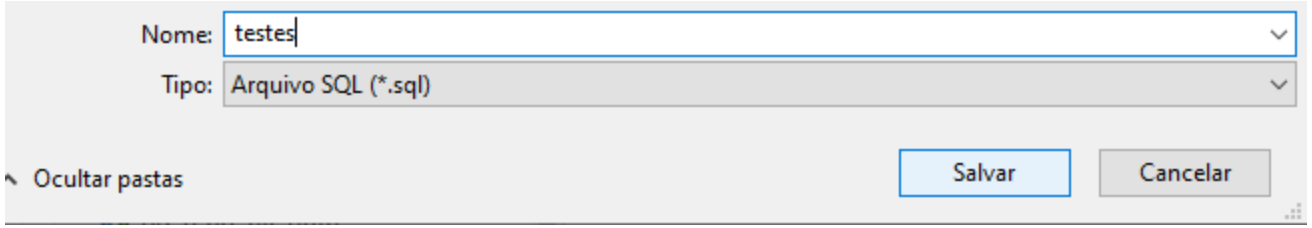
Data output Messages Notifications

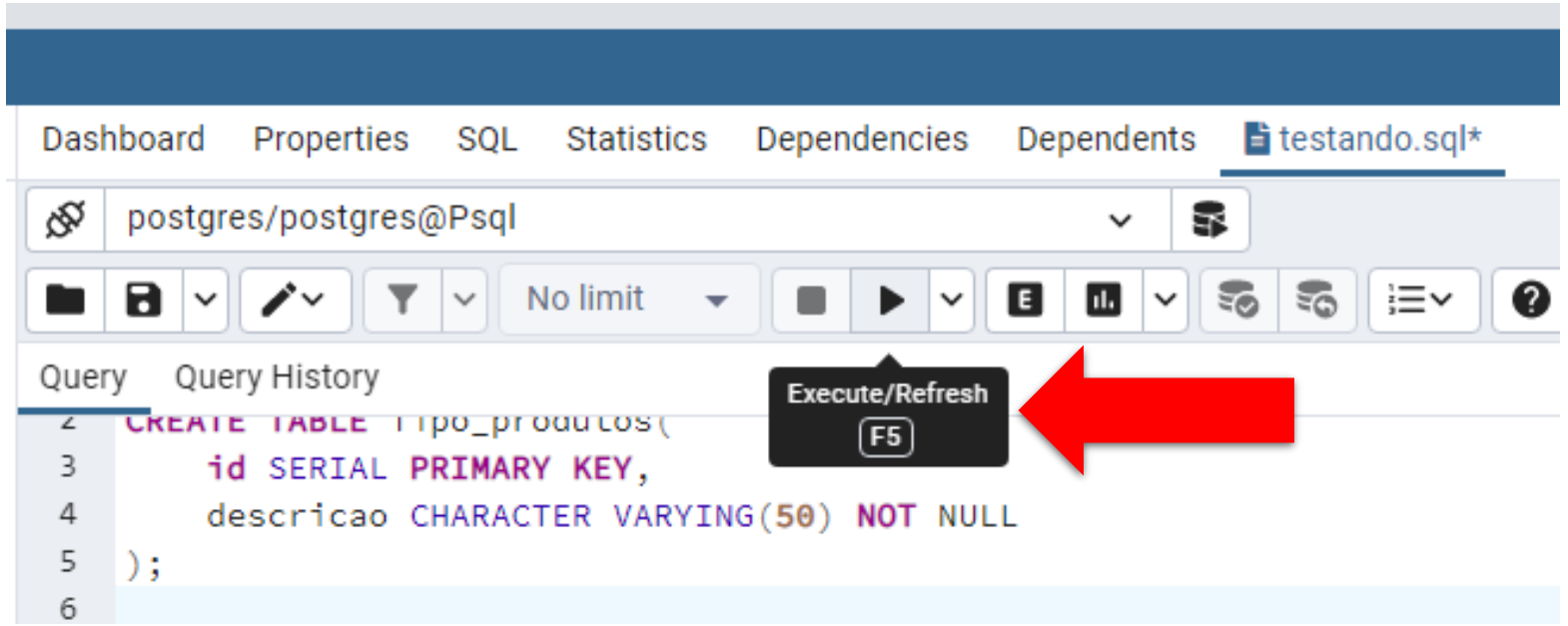
```
CREATE DATABASE

Query returned successfully in 10 secs 111 msec.
```

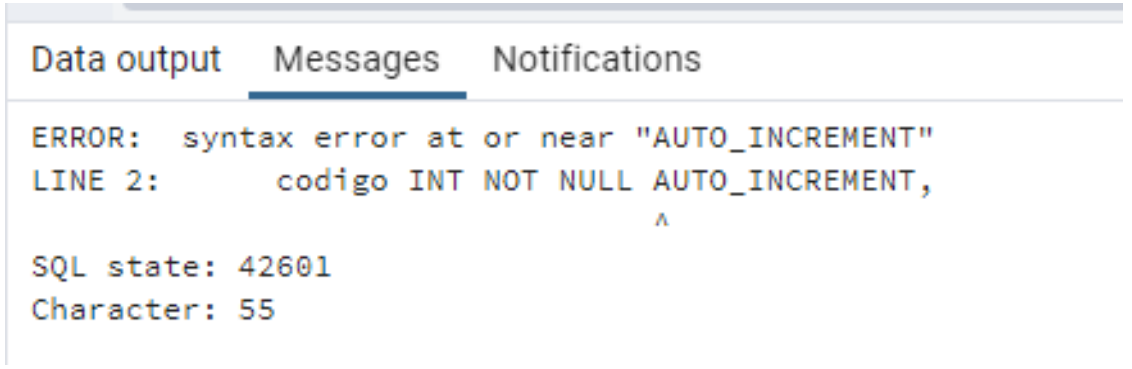


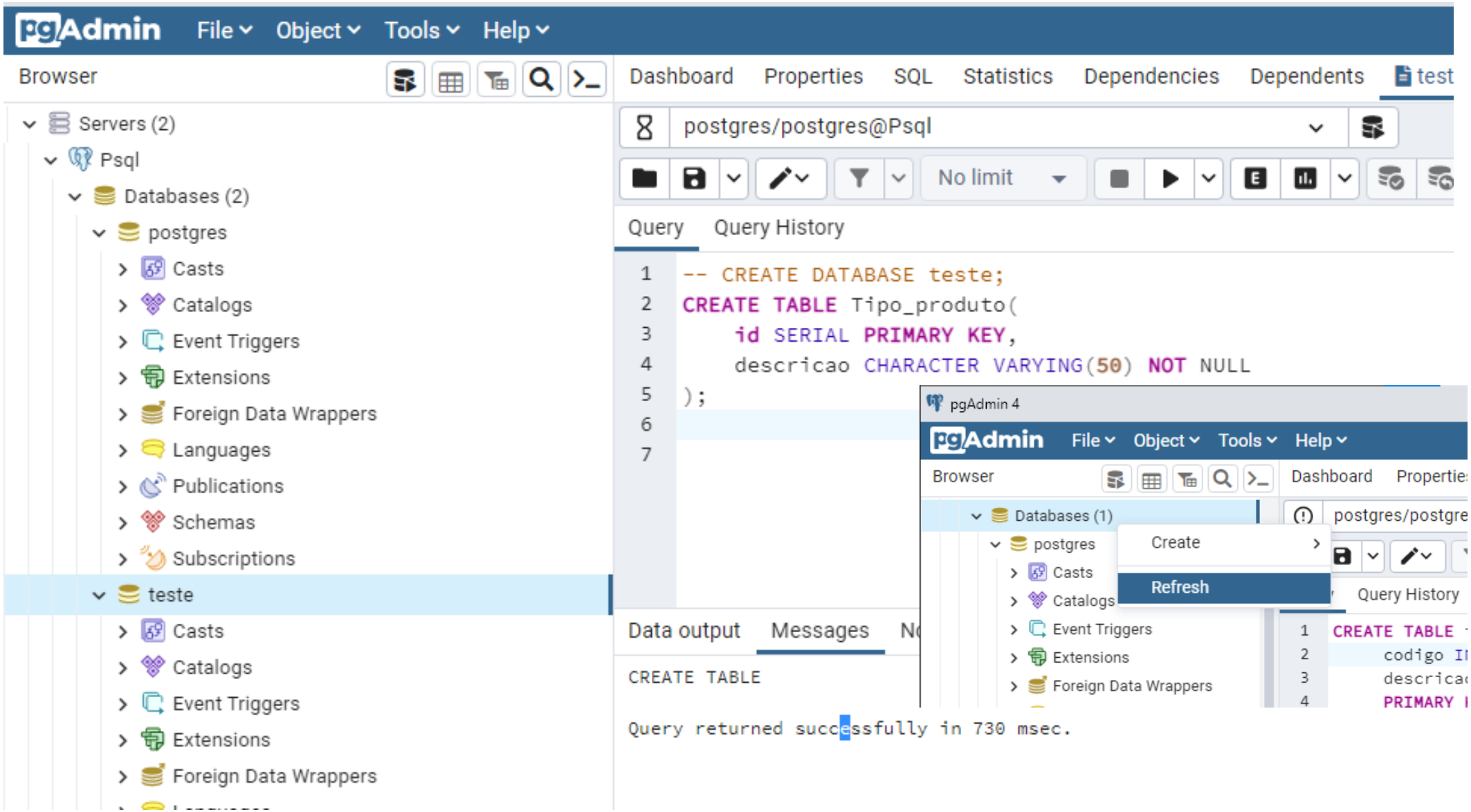

pgAdmin
File
Object
Tools
Help
Browser
Servers (2)
Psql
Databases (2)
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
teste
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
test
postgres/postgres@Psql
No limit
Query
Query History
1 -- CREATE DATABASE teste;
2 CREATE TABLE Tipo_produto(
3 id SERIAL PRIMARY KEY,
4 descricao CHARACTER VARYING(50) NOT NULL
5);
6
7
Data output
Messages
Notifications
CREATE TABLE
Query returned successfully in 730 msec.

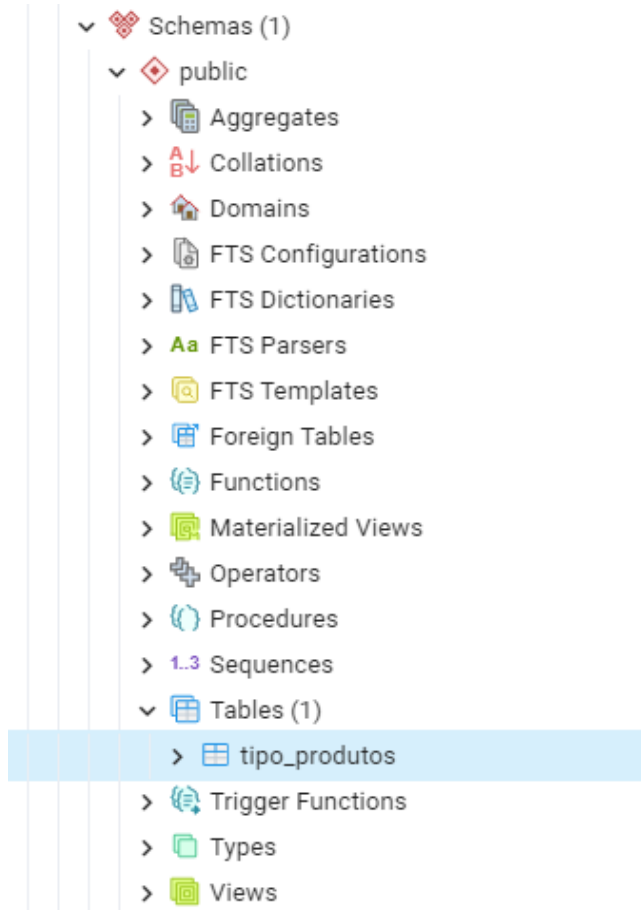

Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
testando.sql*
postgres/postgres@Psql
No limit
Save File
Accesskey S
id SERIAL PRIMARY KEY,
descricao CHARACTER VARYING(50) NOT NULL
);
-- Produtos
CREATE TABLE produtos(
id SERIAL PRIMARY KEY,
descricao CHARACTER VARYING(50) NOT NULL,
preco MONEY NOT NULL,
id_tipo_produto INT REFERENCES tipos_produto(id) NOT NULL

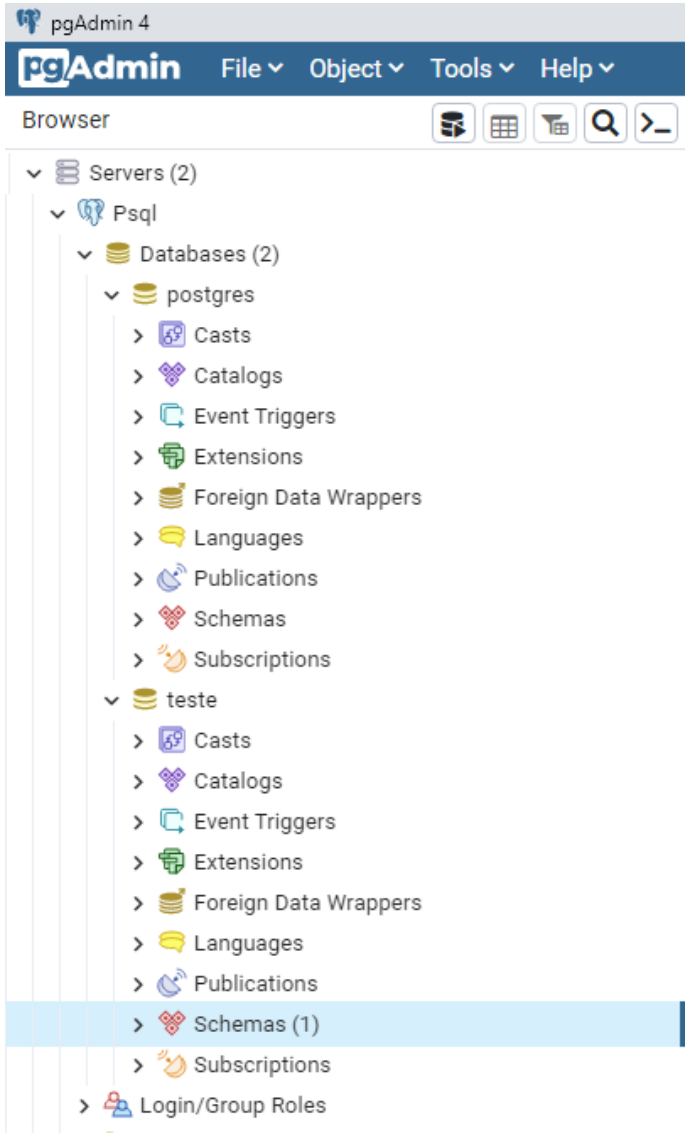

Nome: testes
Tipo: Arquivo SQL (*.sql)
Ocultar pastas
Salvar
Cancelar

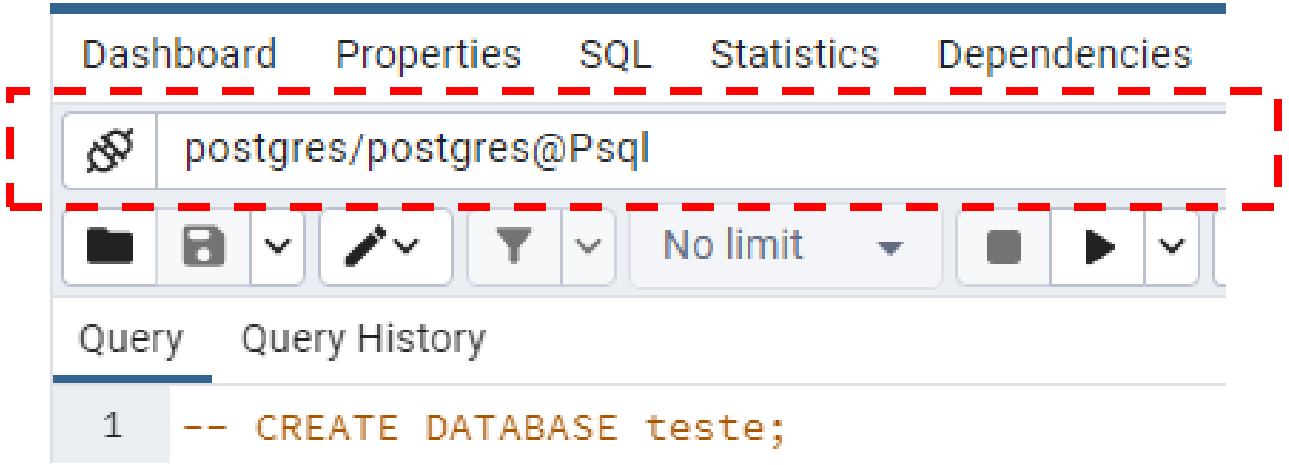

Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
testando.sql*
postgres/postgres@Psql
No limit
Query
Query History
Execute/Refresh
F5
3 id SERIAL PRIMARY KEY,
4 descricao CHARACTER VARYING(50) NOT NULL
5);
6


Data output
Messages
Notifications
ERROR: syntax error at or near "AUTO_INCREMENT"
LINE 2: codigo INT NOT NULL AUTO_INCREMENT,
^
SQL state: 42601
Character: 55


pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
test
Servers (2)
Psql
Databases (2)
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
teste
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
postgres/postgres@Psql
No limit
Query
Query History
1 -- CREATE DATABASE teste;
2 CREATE TABLE Tipo_produto(
3 id SERIAL PRIMARY KEY,
4 descricao CHARACTER VARYING(50) NOT NULL
5);
6
7
Data output
Messages
CREATE TABLE
Query returned successfully in 730 msec.
pgAdmin 4
Databases (1)
postgres
Create
Refresh
Query History
CREATE TABLE

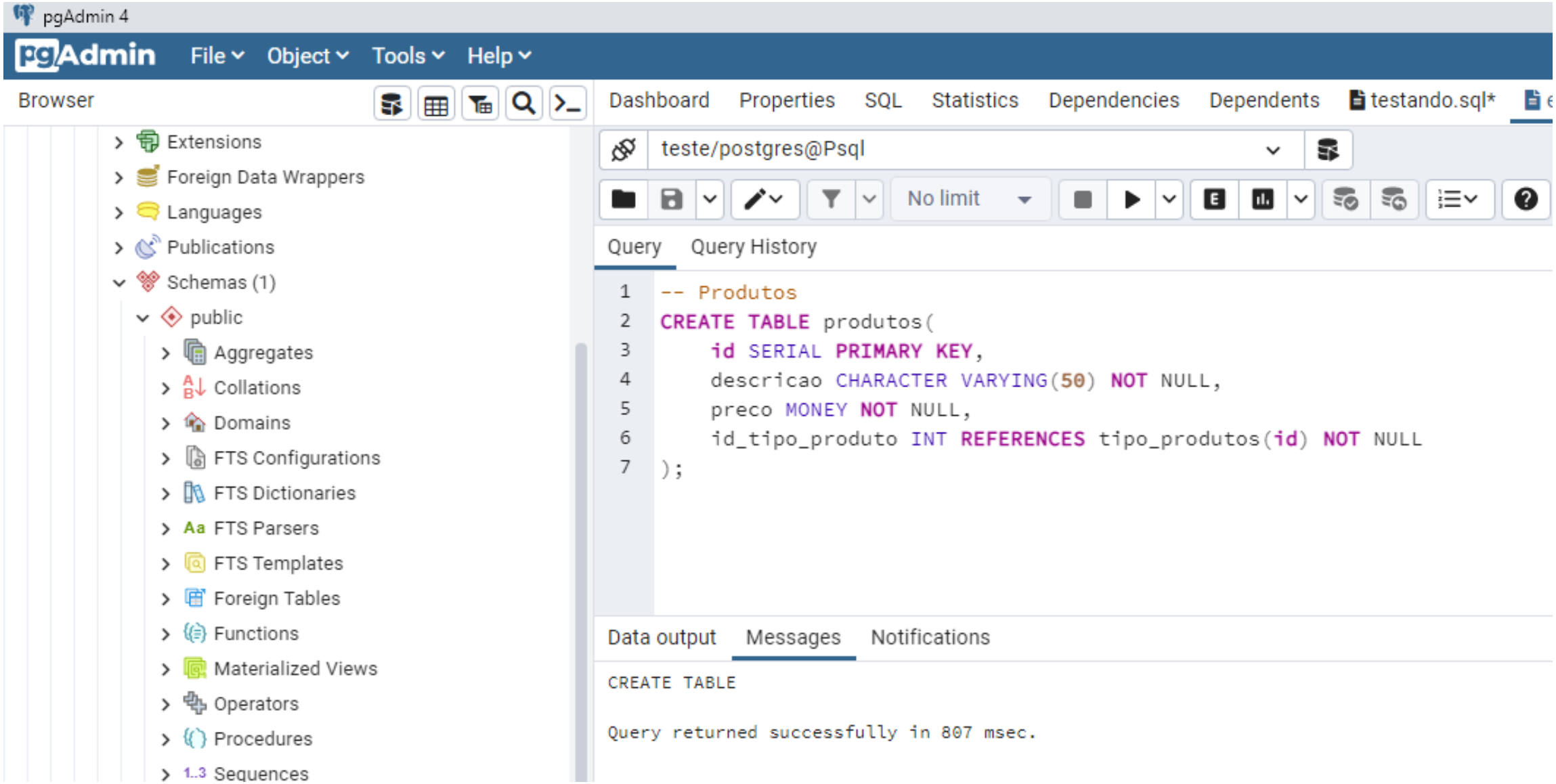

Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
tipo_produtos
Trigger Functions
Types
Views


pgAdmin 4
pgAdmin
File
Object
Tools
Help
Browser
Servers (2)
Psql
Databases (2)
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
teste
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
Subscriptions
Login/Group Roles


Dashboard
Properties
SQL
Statistics
Dependencies
postgres/postgres@Psql
No limit
Query
Query History
1
-- CREATE DATABASE teste;

Outro exemplo (para fazer)

```
-- Produtos
CREATE TABLE produtos(
    id SERIAL PRIMARY KEY,
    descricao CHARACTER VARYING(50) NOT NULL,
    preco MONEY NOT NULL,
    id_tipo_produto INT REFERENCES tipos_produto(id) NOT NULL
);
```

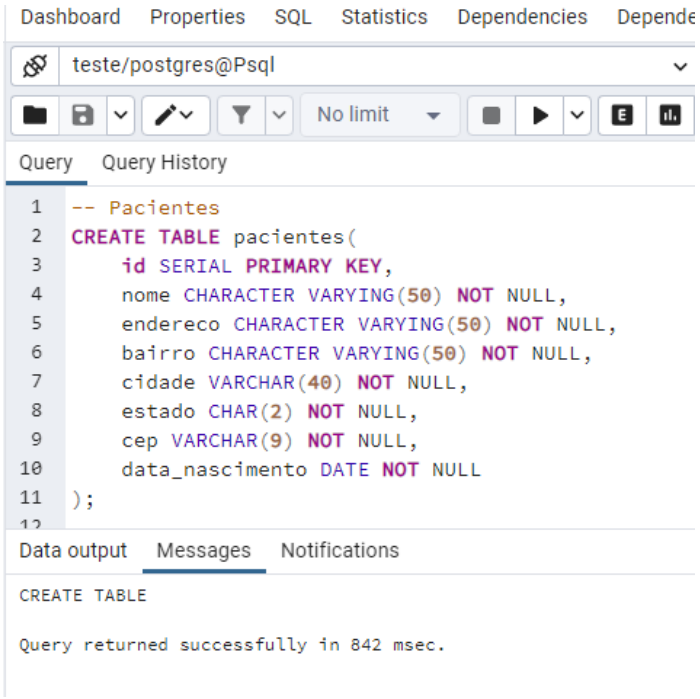

pgAdmin 4
pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
testando.sql*
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
teste/postgres@Psql
No limit
Query
Query History
1 -- Produtos
2 CREATE TABLE produtos(
3 id SERIAL PRIMARY KEY,
4 descricao CHARACTER VARYING(50) NOT NULL,
5 preco MONEY NOT NULL,
6 id_tipo_produto INT REFERENCES tipo_produtos(id) NOT NULL
7);
Data output
Messages
Notifications
CREATE TABLE
Query returned successfully in 807 msec.

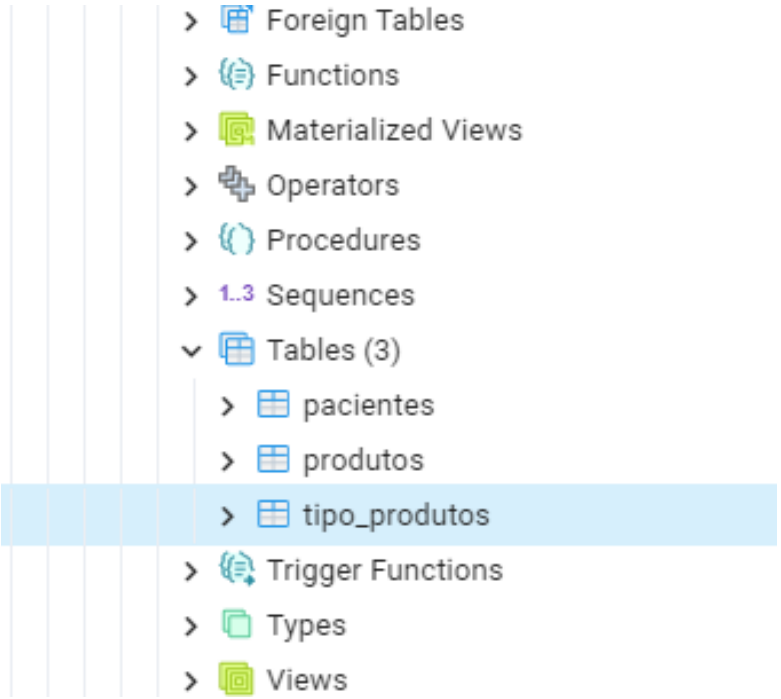
...

- -- Pacientes

```
CREATE TABLE pacientes(
        id SERIAL PRIMARY KEY,
        nome CHARACTER VARYING(50) NOT NULL,
        endereco CHARACTER VARYING(50) NOT NULL,
        bairro CHARACTER VARYING(50) NOT NULL,
        cidade VARCHAR(40) NOT NULL,
        estado CHAR(2) NOT NULL,
        cep VARCHAR(9) NOT NULL,
        data_nascimento DATE NOT NULL
);
```

Resultados


Dashboard
Properties
SQL
Statistics
Dependencies
Depende
teste/postgres@Psql
No limit
Query
Query History
1 -- Pacientes
2 CREATE TABLE pacientes(
3 id SERIAL PRIMARY KEY,
4 nome CHARACTER VARYING(50) NOT NULL,
5 endereco CHARACTER VARYING(50) NOT NULL,
6 bairro CHARACTER VARYING(50) NOT NULL,
7 cidade VARCHAR(40) NOT NULL,
8 estado CHAR(2) NOT NULL,
9 cep VARCHAR(9) NOT NULL,
10 data_nascimento DATE NOT NULL
11);
Data output
Messages
Notifications
CREATE TABLE
Query returned successfully in 842 msec.


Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (3)
pacientes
produtos
tipo_produtos
Trigger Functions
Types
Views

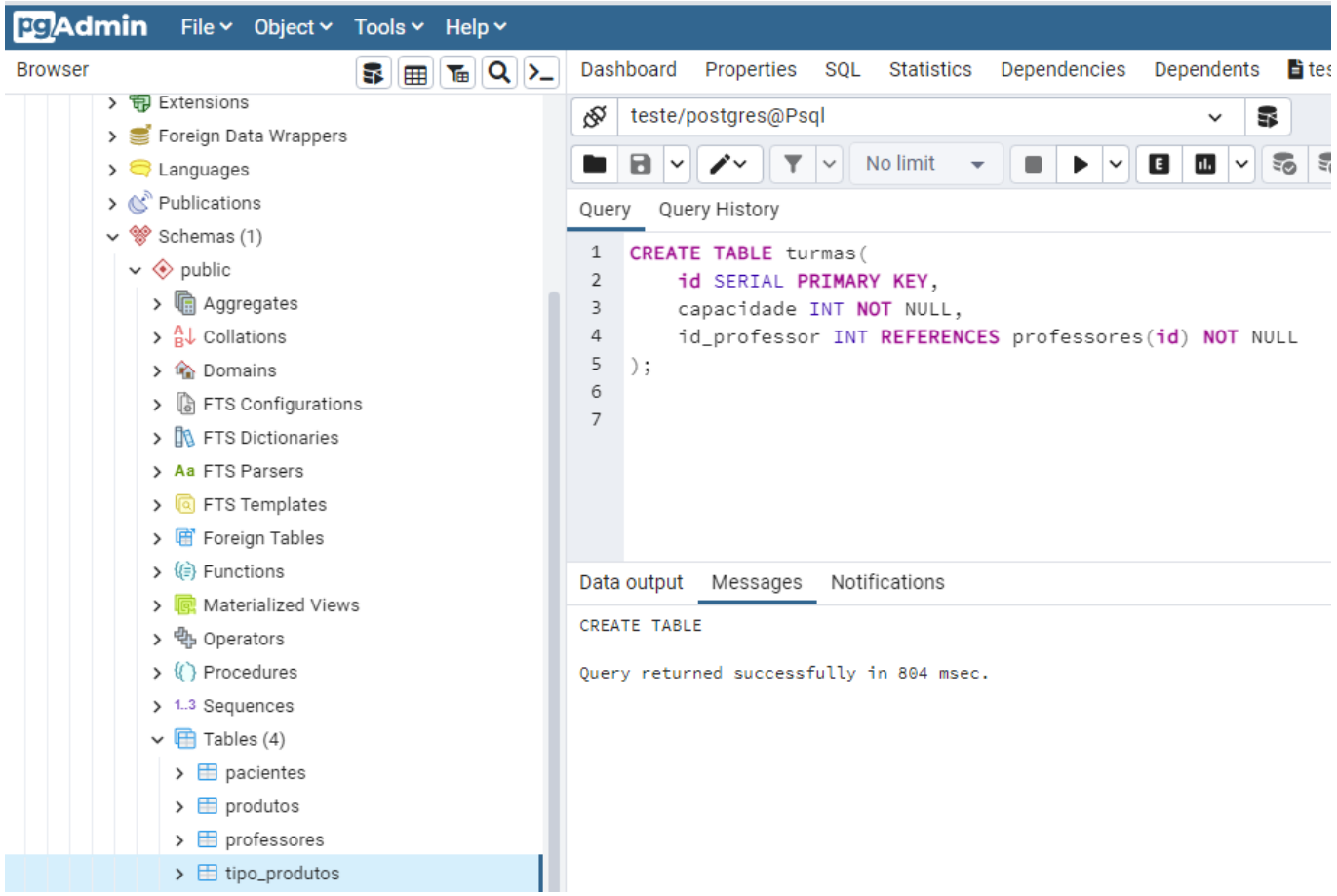
...

```
CREATE TABLE professores(
        id SERIAL PRIMARY KEY,
        telefone INT NOT NULL,
        nome VARCHAR(80) NOT NULL
);
```

...

```
CREATE TABLE turmas(
        id SERIAL PRIMARY KEY,
        capacidade INT NOT NULL,
        id_professor INT REFERENCES professores(id) NOT NULL
);
```

Resultados


pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
teste/postgres@Psql
No limit
Query
Query History
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
pacientes
produtos
professores
tipo_produtos
1 CREATE TABLE turmas(
2 id SERIAL PRIMARY KEY,
3 capacidade INT NOT NULL,
4 id_professor INT REFERENCES professores(id) NOT NULL
5);
6
7
Data output
Messages
Notifications
CREATE TABLE
Query returned successfully in 804 msec.