C.R.U.D.

Básico de Bancos

C.R.U.D. - Criar Ler Atualizar e Apagar

CRUD é um acrônimo em inglês que significa "**Create**, **Read**, **Update**, **Delete**" (ou, em português, "Criar, Ler, Atualizar, Excluir").

O CRUD corresponde às ações básicas que podem ser realizadas em um banco de dados:

- **Create (criar):** criar um novo registro em uma tabela do banco de dados.
- **Read (ler):** ler os dados de um ou mais registros existentes em uma tabela do banco de dados.
- **Update (atualizar):** atualizar os dados de um registro existente em uma tabela do banco de dados.
- **Delete (excluir):** excluir um registro existente em uma tabela do banco de dados.

Criando uma tabela no PostgreSQL

Um exemplo completo

Camadas do banco de dados: esquema

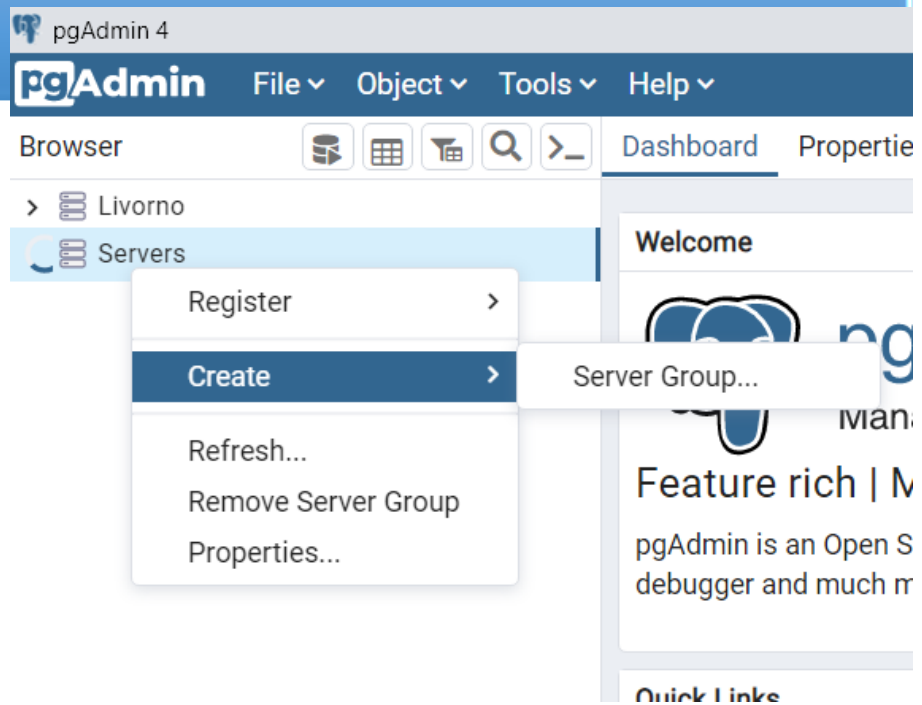
Antes de descobrir como usar esquemas, você precisa saber qual é a finalidade de um esquema. Para entender isso, primeiro dê uma olhada em como o PostgreSQL está estruturado:

- Instance - Instância
- Database - Base de dados
- Schema - Esquema
- Table
- Row - Linha

Para criar uma tabela no PostgreSQL, pode-se executar os seguintes passos:

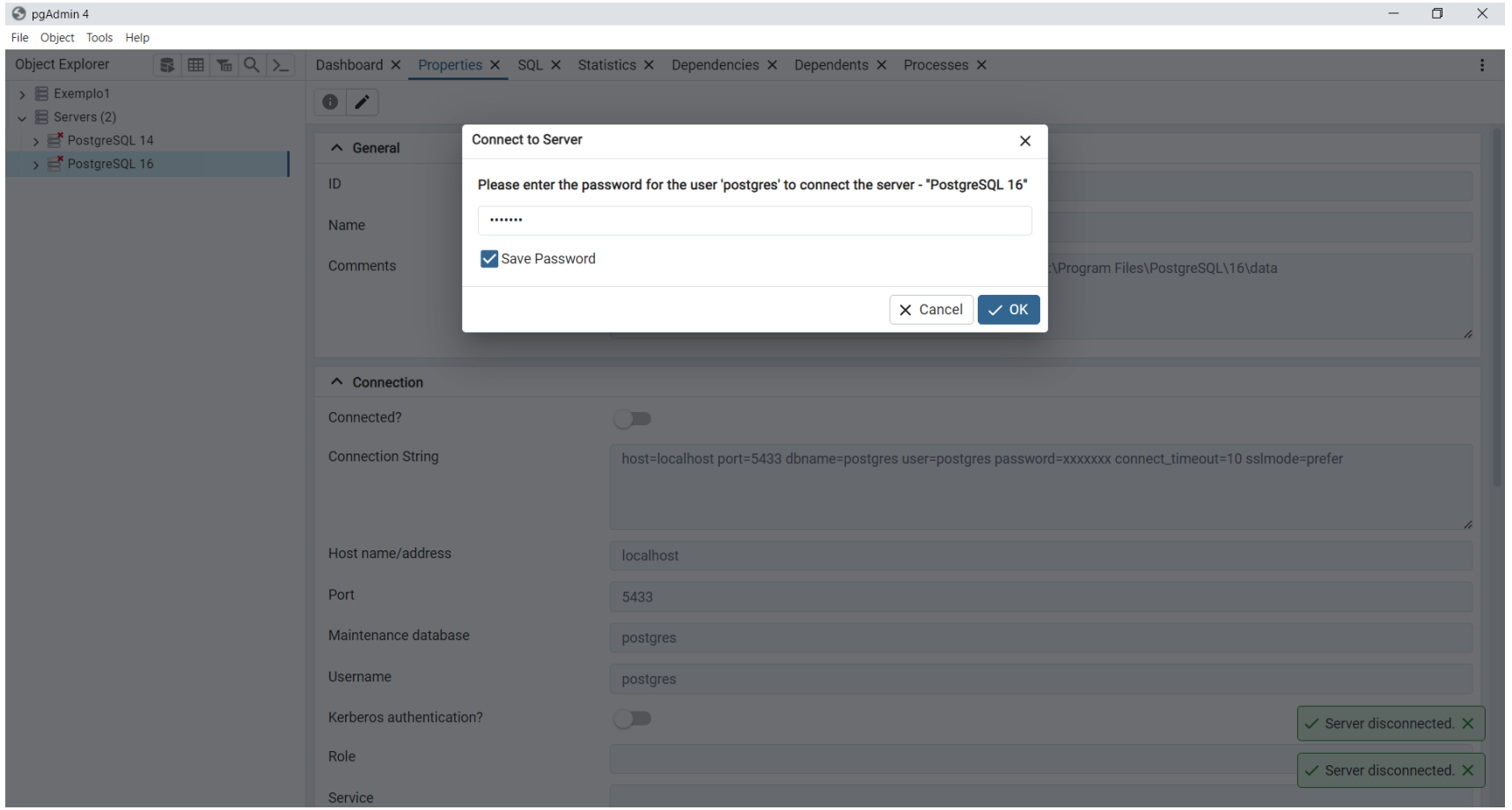
Certifique-se de que você esteja conectado ao banco de dados desejado.

Ou crie um Servidor para estar conectado

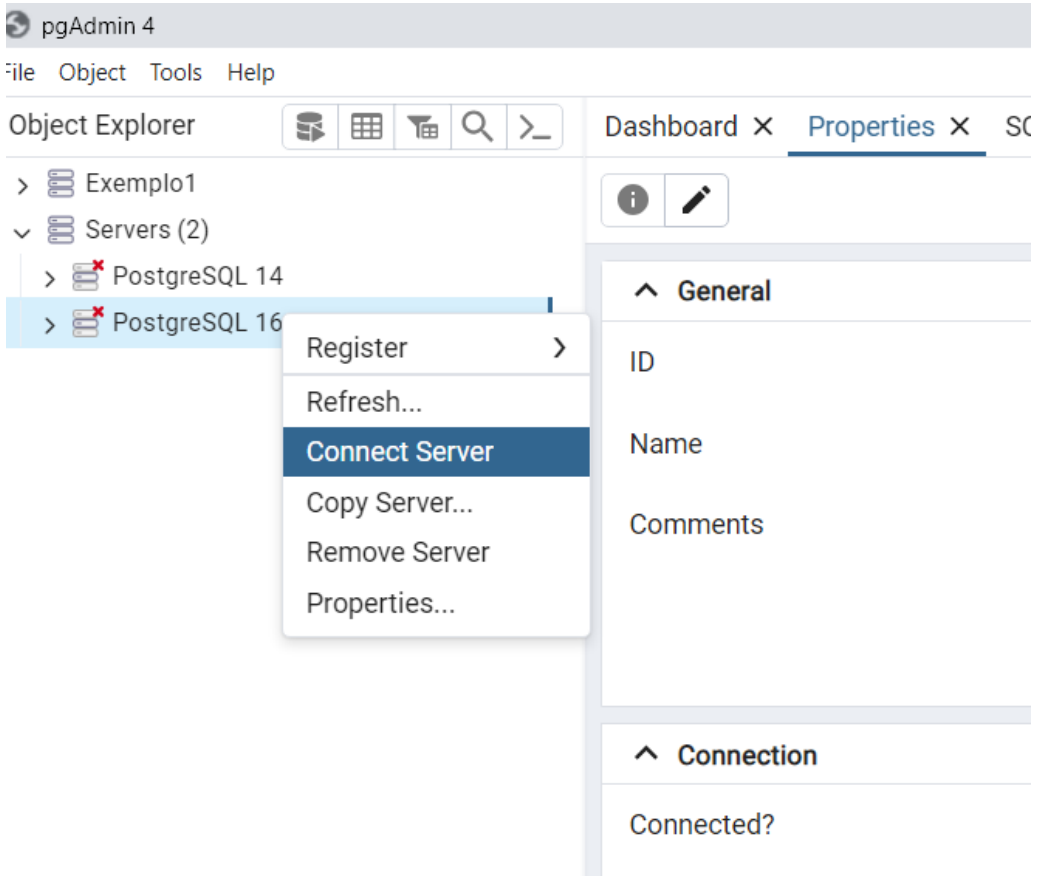


Exemplo geral

Instância - conecte ao servidor



Conete a base ao servidor:



O objetivo de um esquema

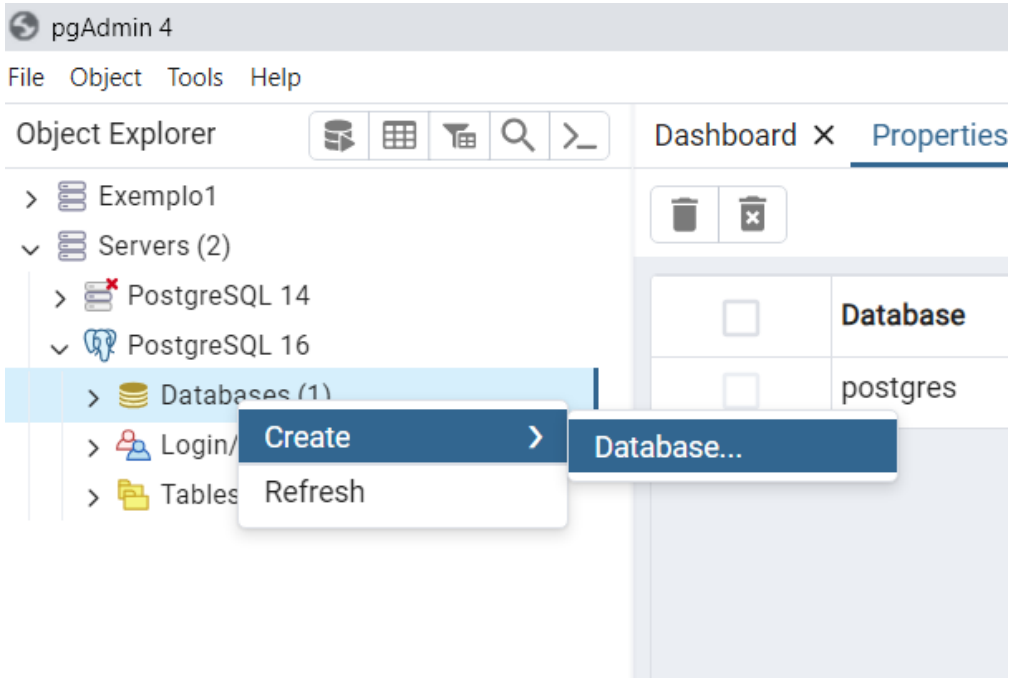
Na realidade é a isso que você se conecta: no PostgreSQL uma conexão está sempre vinculada a um banco de dados dentro de uma instância, o que acontece logo no início, logo após a autenticação do usuário.

O objetivo de um esquema

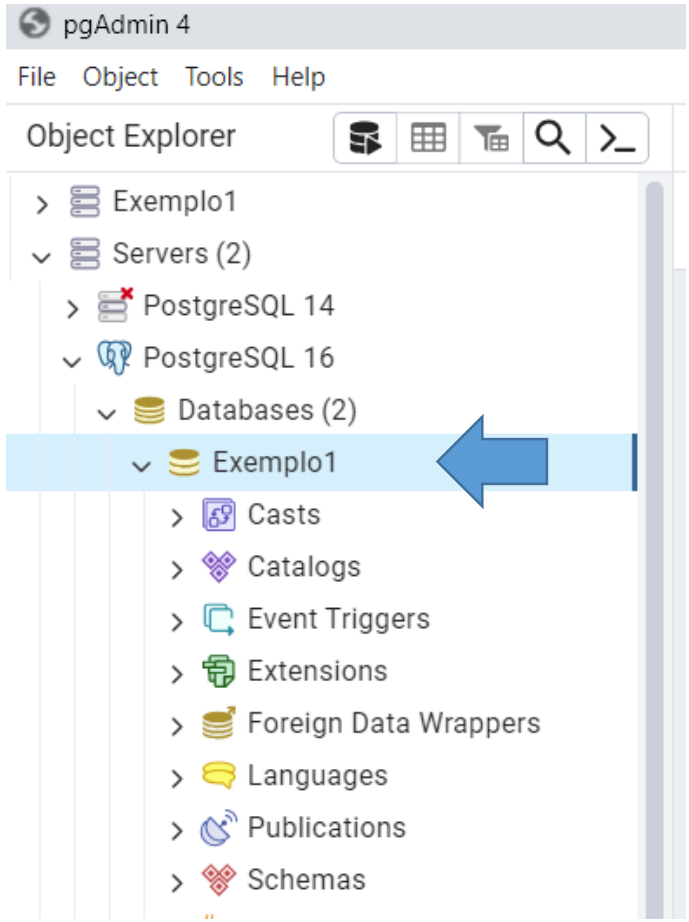
Uma “instância” é basicamente o que você inicia ao implantar o PostgreSQL.

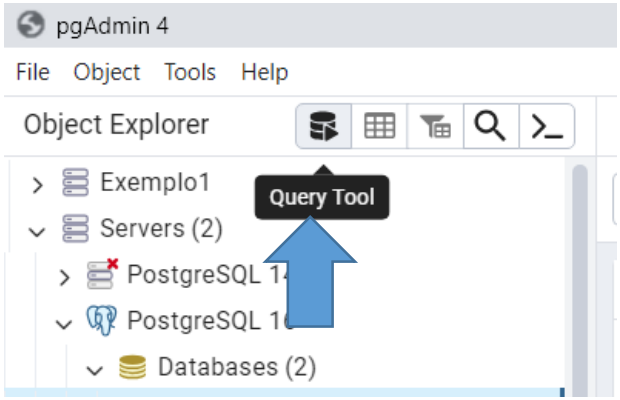
A próxima camada é um banco de dados.

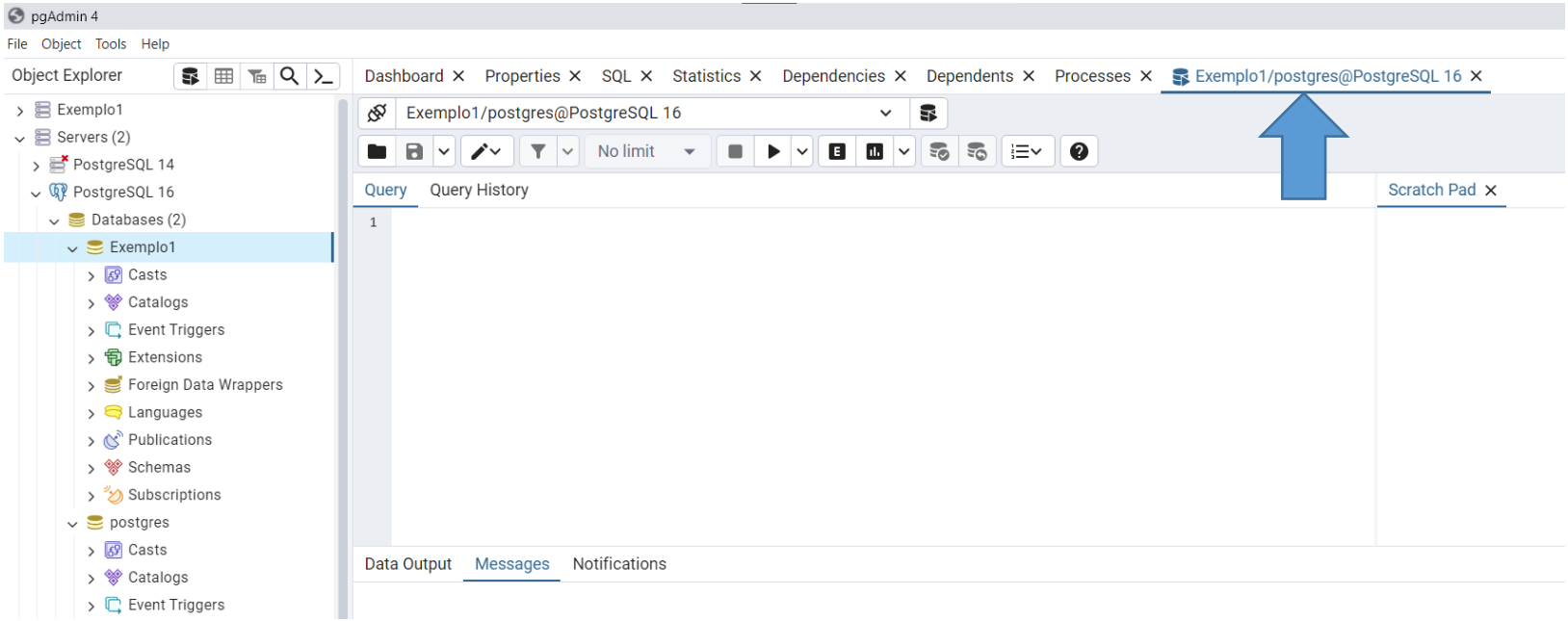
```
CREATE DATABASE "Exemplo1"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LOCALE_PROVIDER = 'libc'
    CONNECTION LIMIT = -1
    IS_TEMPLATE = False;
```



Resultado:


pgAdmin 4
File Object Tools Help
Object Explorer
Exemplo1
Servers (2)
PostgreSQL 14
PostgreSQL 16
Databases (2)
Exemplo1
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas


pgAdmin 4
File Object Tools Help
Object Explorer
Query Tool
Exemplo1
Servers (2)
PostgreSQL 14
PostgreSQL 16
Databases (2)


pgAdmin 4
File Object Tools Help
Object Explorer
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
Exemplo1/postgres@PostgreSQL 16
Exemplo1/postgres@PostgreSQL 16
No limit
Query
Query History
Scratch Pad
1
Data Output
Messages
Notifications
Exemplo1
Servers (2)
PostgreSQL 14
PostgreSQL 16
Databases (2)
Exemplo1
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
postgres
Casts
Catalogs
Event Triggers

Tabelas de grupo de esquemas

O que é importante é a próxima camada abaixo, entre bancos de dados e tabelas: Esquemas.

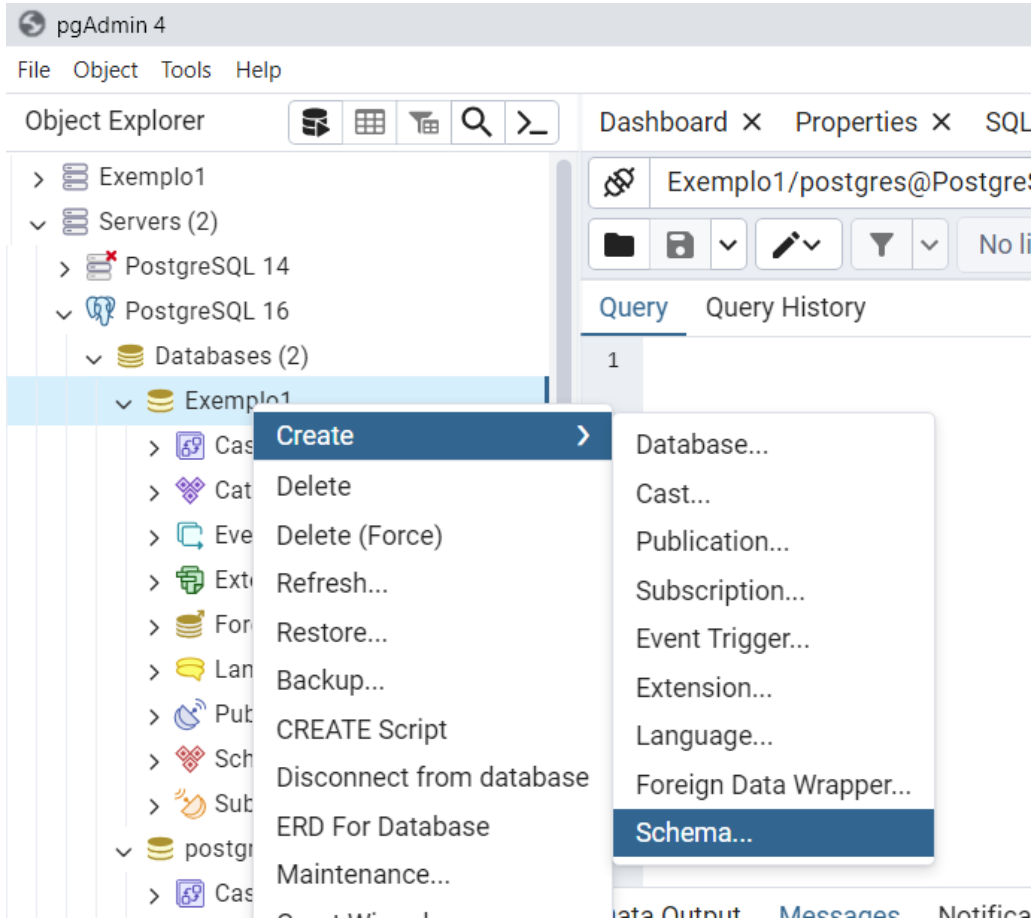
Basicamente, os esquemas são uma forma de agrupar tabelas.

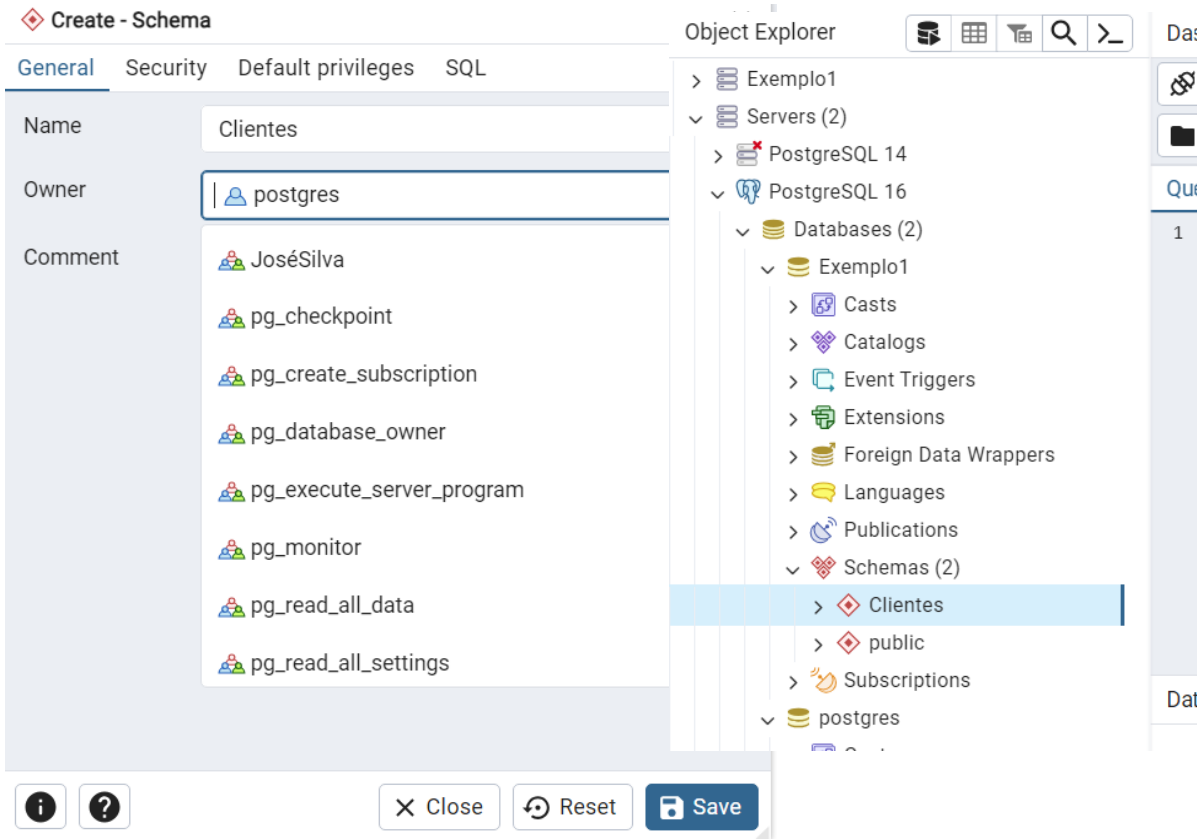
Vamos supor que haja uma estrutura de dados bastante grande: ter 500 tabelas em um só lugar é certamente mais difícil de gerenciar e entender do que ter 10 grupos contendo 50 tabelas cada.

É simplesmente como organizar imagens: você não colocaria todas elas na mesma pasta, mas sim agrupá-las por ano, local, etc. A mesma lógica pode ser aplicada às tabelas.

Criando schemas

```
CREATE SCHEMA Clientes
    AUTHORIZATION postgres;
```



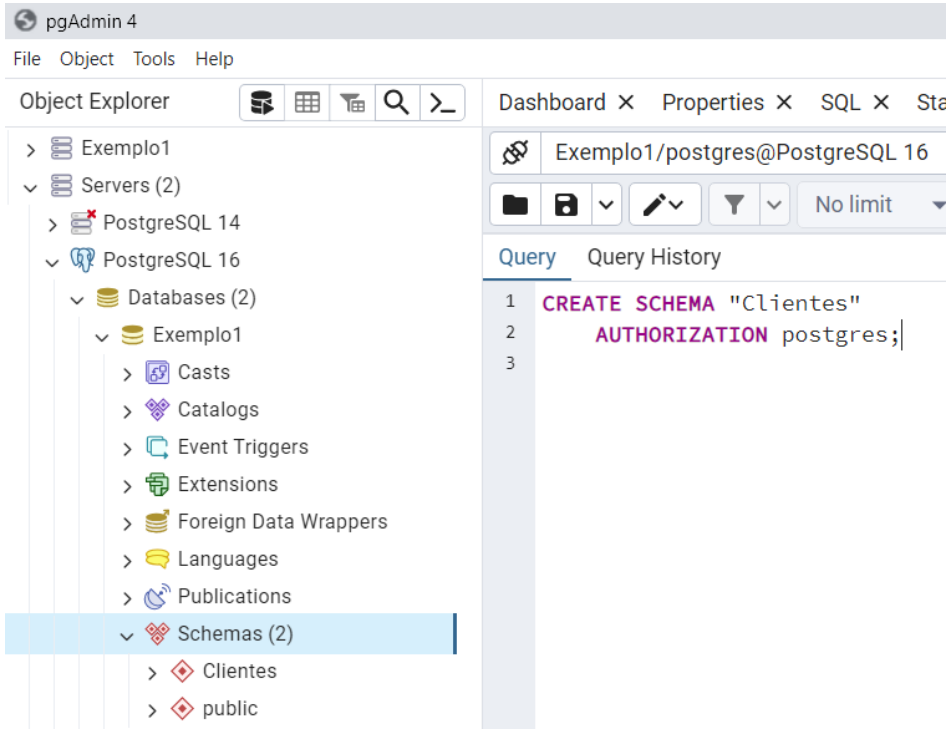


Use o comando CREATE TABLE para criar uma nova tabela.

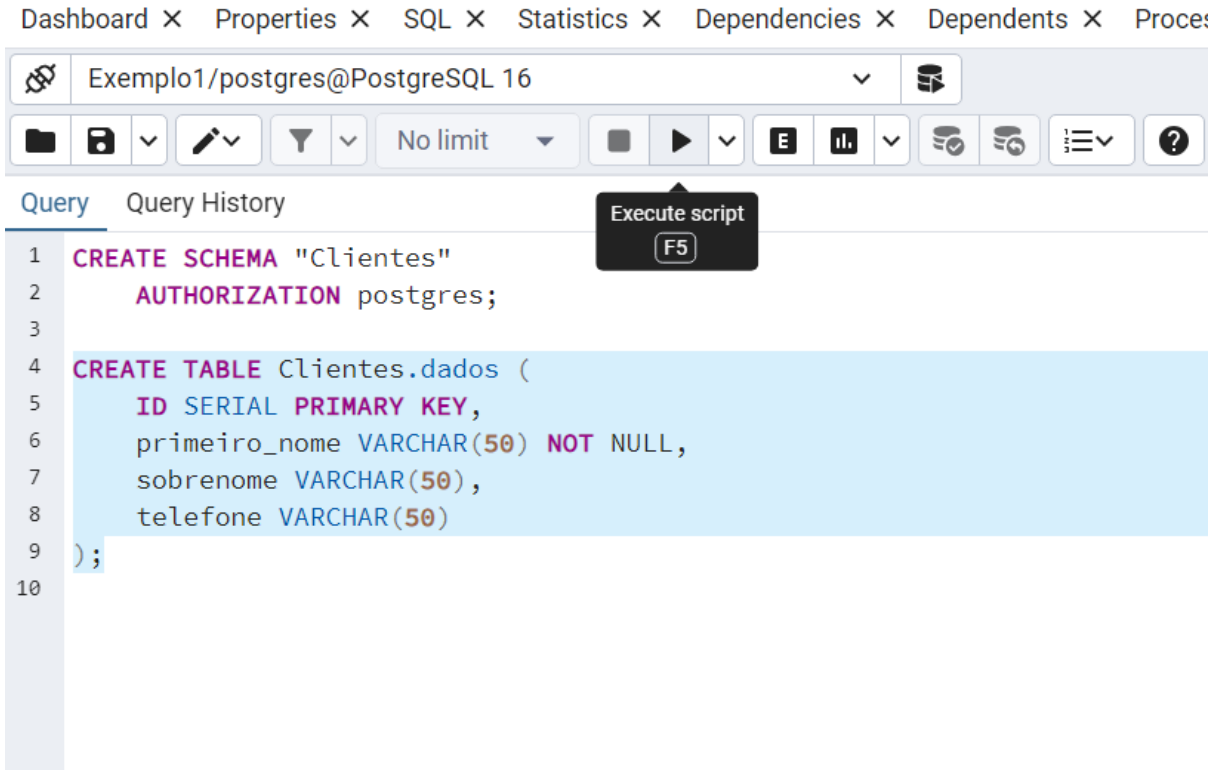
A sintaxe básica é a seguinte:

```
CREATE TABLE nome_da_tabela (
        nome_da_coluna1 tipo_de_dado_da_coluna1,
        nome_da_coluna2 tipo_de_dado_da_coluna2,
        ...
        nome_da_colunaN tipo_de_dado_da_colunaN
);
```

Criando tabelas nesse esquema (schema)




```
CREATE TABLE Clientes.dados (
        ID SERIAL PRIMARY KEY,
        primeiro_nome VARCHAR(50) NOT NULL,
        sobrenome VARCHAR(50),
        telefone VARCHAR(50)
);
```



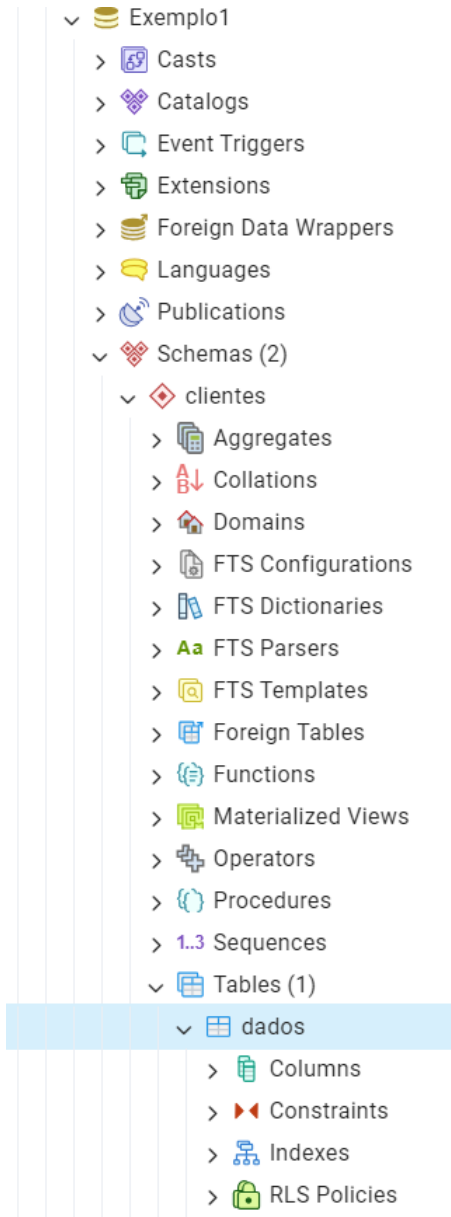
Query | Query History

```
1  CREATE SCHEMA Clientes
2      AUTHORIZATION postgres;
3
4  CREATE TABLE Clientes.dados (
5      ID SERIAL PRIMARY KEY,
6      primeiro_nome VARCHAR(50) NOT NULL,
7      sobrenome VARCHAR(50),
8      telefone VARCHAR(50)
9  );
10
```

Data Output | Messages | Notifications

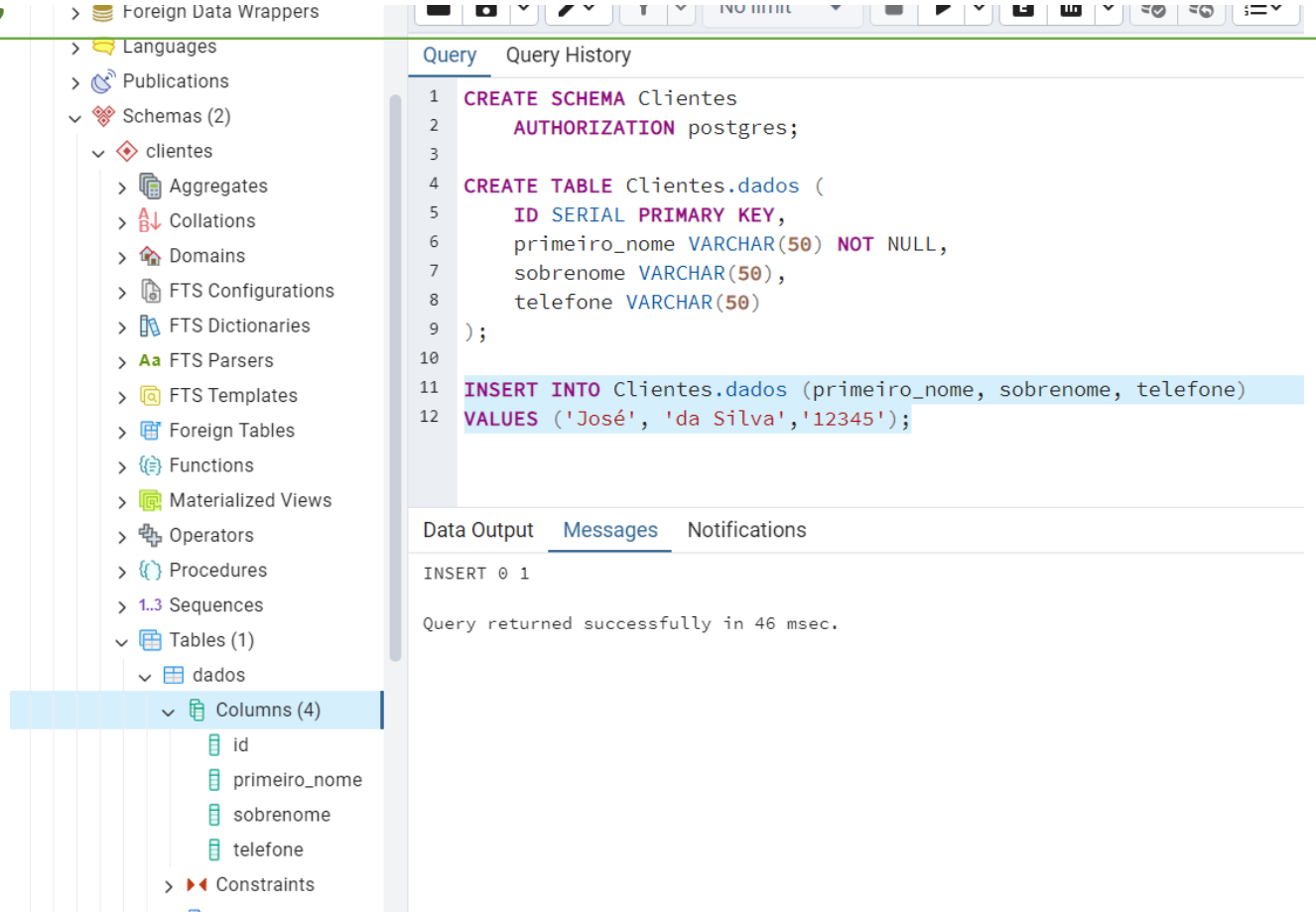
```
CREATE TABLE

Query returned successfully in 160 msec.
```

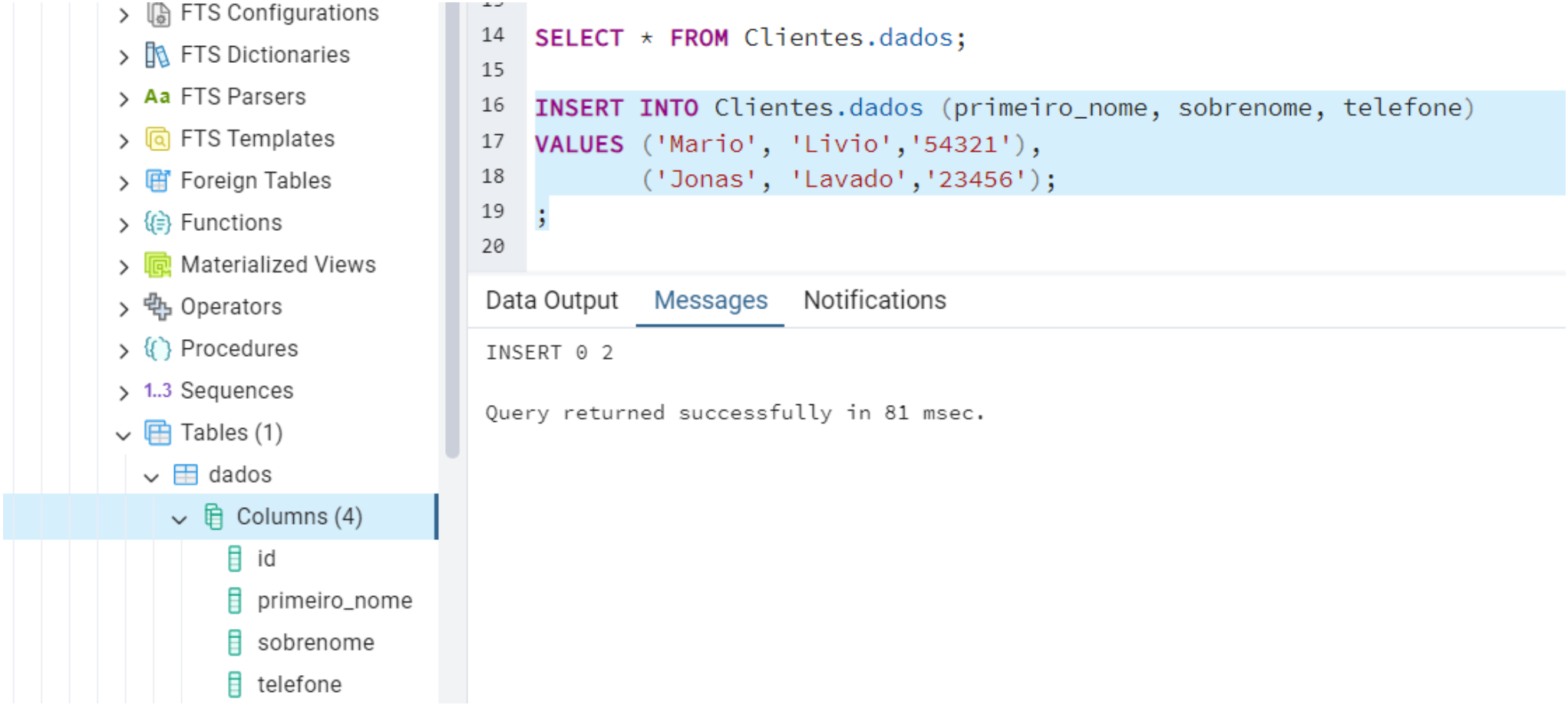


Inserindo dados:

INSERT INTO Clientes.dados (primeiro_nome, sobrenome, telefone)
VALUES ('José', 'da Silva','12345');



```
INSERT INTO Clientes.dados (primeiro_nome, sobrenome, telefone)
VALUES ('Mario', 'Livio','54321'),
       ('Jonas', 'Lavado','23456');
;
```



Lendo dados de uma tabela no PostgreSQL

SELECT

Para realizar a leitura de dados em uma tabela no PostgreSQL, você pode utilizar o comando SELECT. A sintaxe básica do comando SELECT é a seguinte:

```
SELECT coluna1, coluna2, ..., colunaN
FROM nome_da_tabela
WHERE condição;
```

Visualizando...

```
10
11  INSERT INTO Clientes.dados (primeiro_nome, sobrenome, telefone)
12  VALUES ('José', 'da Silva','12345');
13
14  SELECT * FROM Clientes.dados;
15
16  INSERT INTO Clientes.dados (primeiro_nome, sobrenome, telefone)
17  VALUES ('Mario', 'Livio','54321'),
18         ('Jonas', 'Lavado','23456');
19  ;
20
```

Data Output | Messages | Notifications

	id [PK] integer	primeiro_nome character varying (50)	sobrenome character varying (50)	telefone character varying (50)
1	1	José	da Silva	12345
2	2	Mario	Livio	54321
3	3	Jonas	Lavado	23456

Apagando dados de uma tabela no PostgreSQL

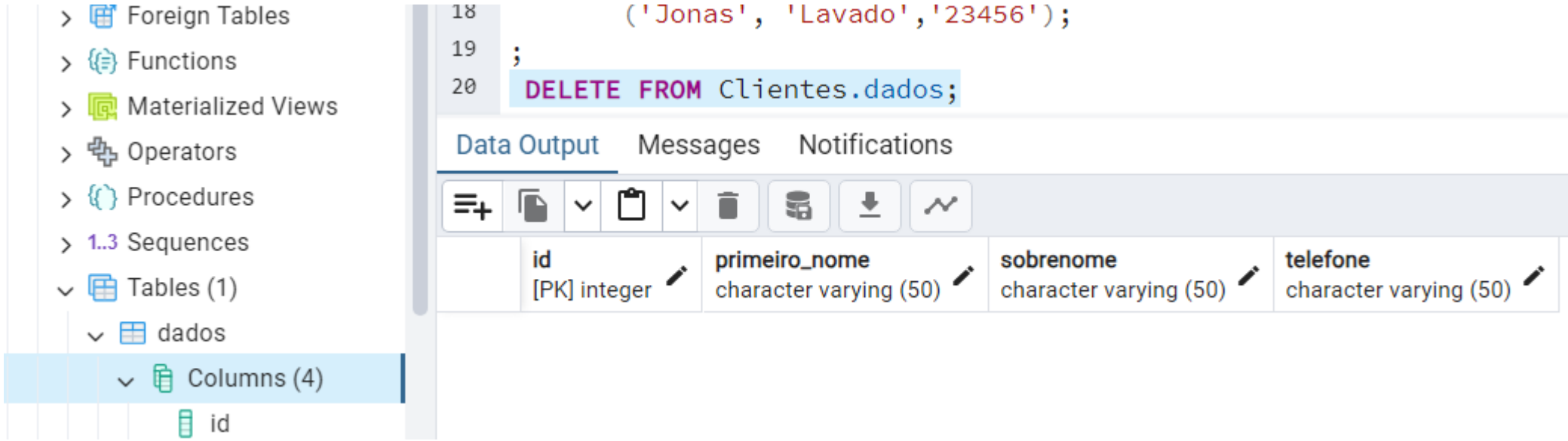
DELETE

Para apagar dados em uma tabela no PostgreSQL, você pode utilizar o comando DELETE. A sintaxe básica do comando DELETE é a seguinte:

```
DELETE FROM nome_da_tabela
WHERE condição;
```

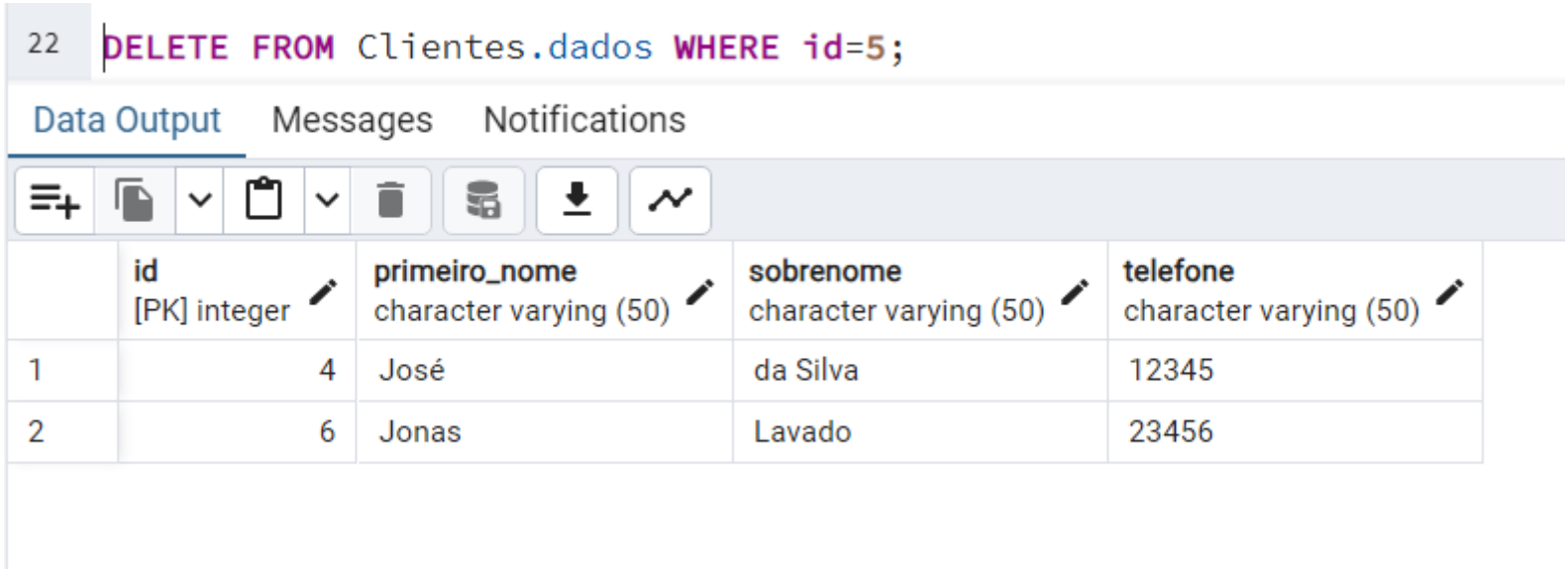
Apagando dados

- DELETE FROM Clientes.dados;



Apagando apenas uma linha:

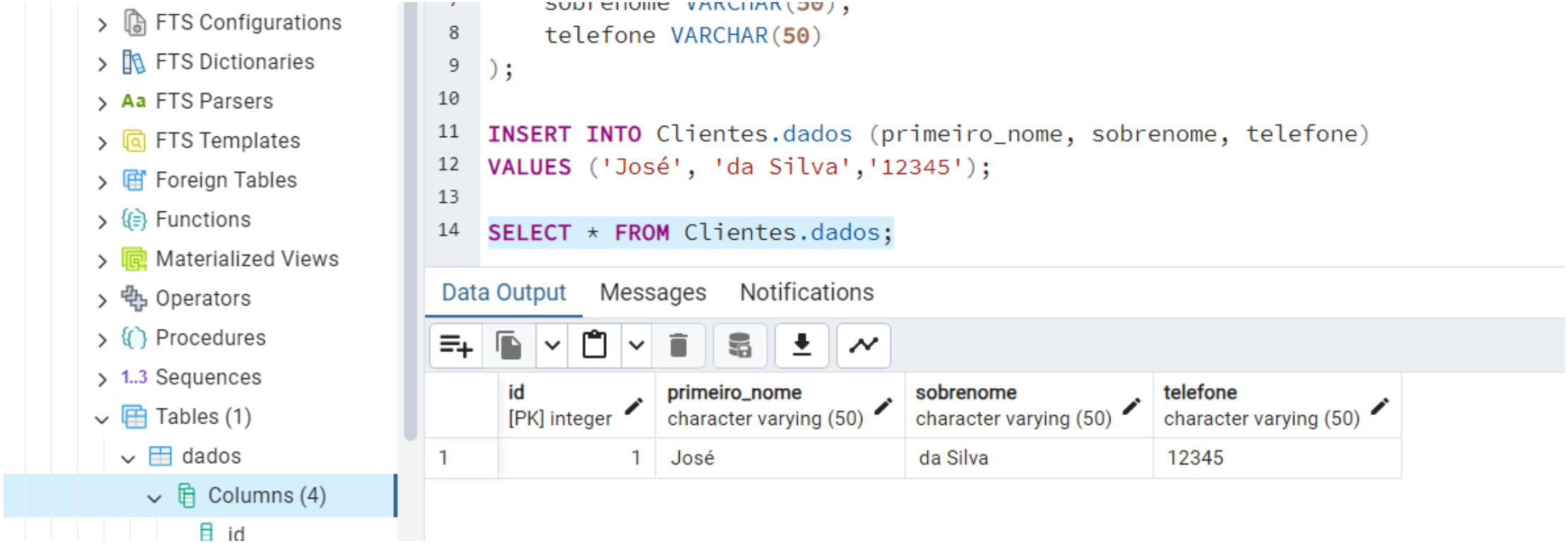
```
DELETE FROM Clientes.dados WHERE id=5;
```



	id [PK] integer	primeiro_nome character varying (50)	sobrenome character varying (50)	telefone character varying (50)
1	4	José	da Silva	12345
2	6	Jonas	Lavado	23456

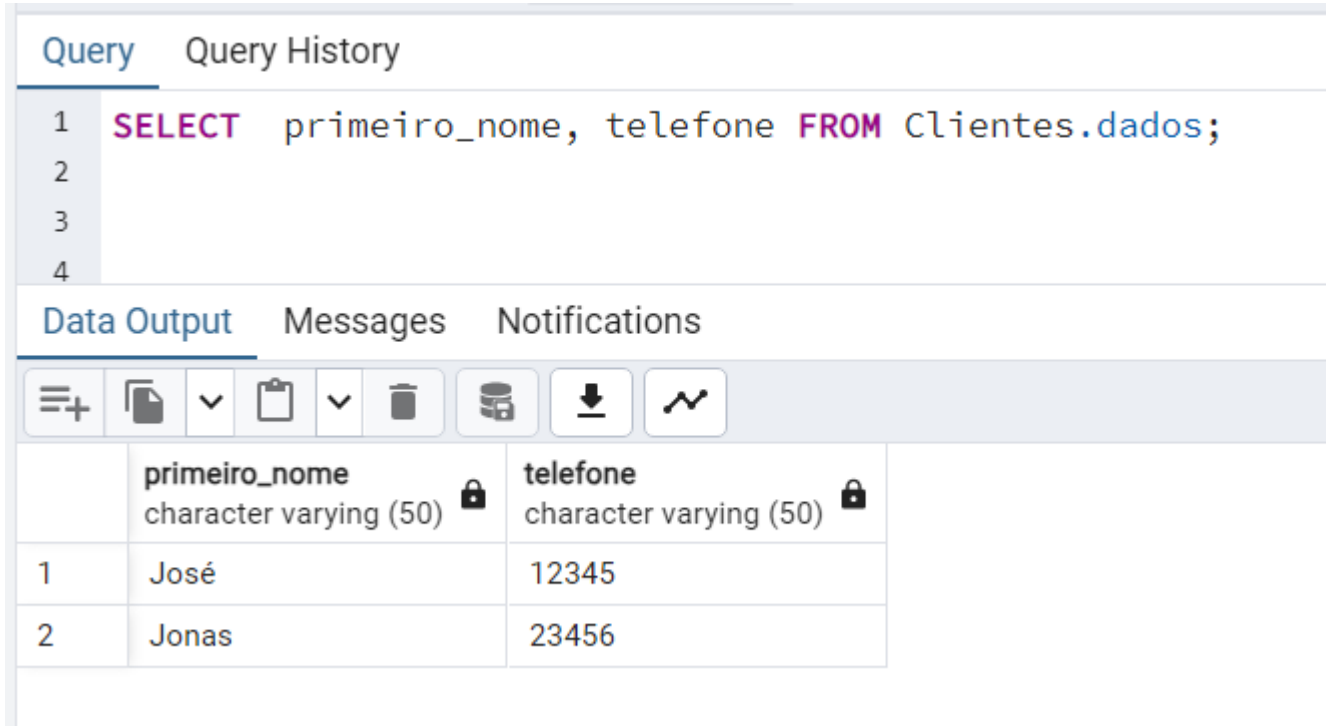
Preenchendo uma tabela com linhas

Selecionando (visualizando) dados



Visualizando apenas alguns campos

```
SELECT  primeiro_nome, telefone FROM Clientes.dados;
```



Mais seleções...

Query | Query History

```
3  SELECT * FROM Clientes.dados
4   WHERE primeiro_nome= 'José' AND sobrenome = 'da Silva';
5
```

Data Output | Messages | Notifications

	id [PK] integer	primeiro_nome character varying (50)	sobrenome character varying (50)	telefone character varying (50)
1	4	José	da Silva	12345

Ordenando...

```
SELECT * FROM Clientes.dados
 ORDER BY primeiro_nome;
```

Query Query History

```
4  WHERE primeiro_nome= 'José' AND sobrenome = 'da Silva';
5
6  SELECT * FROM Clientes.dados
7   ORDER BY primeiro_nome;
```

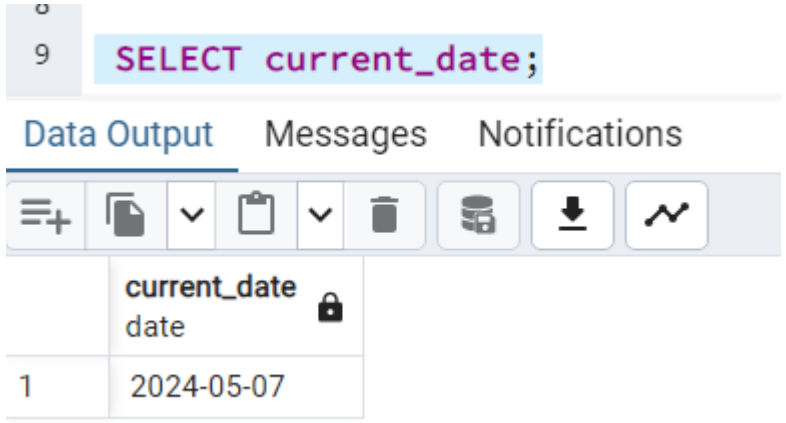
Data Output Messages Notifications

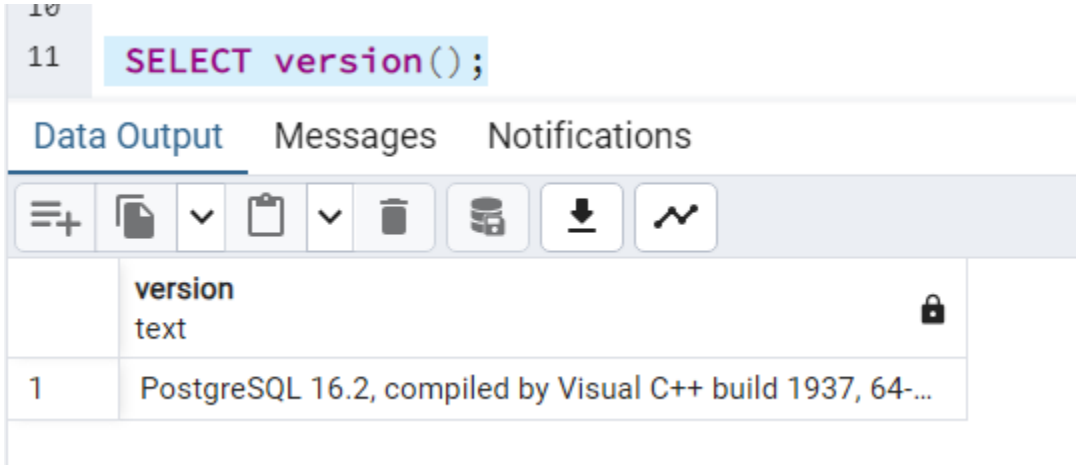
	id [PK] integer	primeiro_nome character varying (50)	sobrenome character varying (50)	telefone character varying (50)
1	6	Jonas	Louvado	23456
2	4	José	da Silva	12345

Selecionado variáveis internas

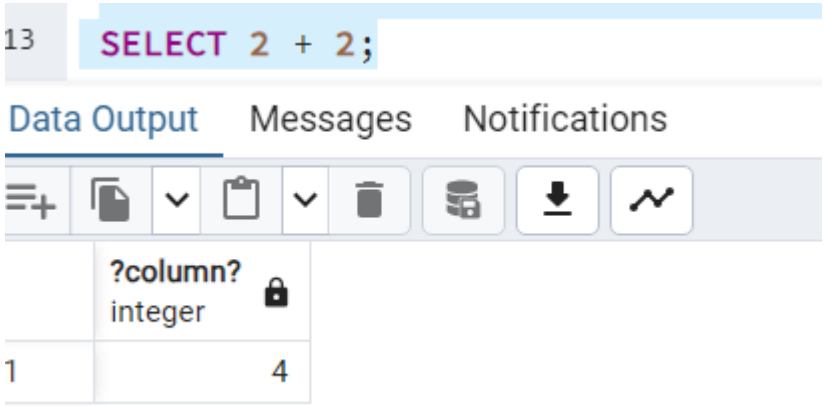
```
SELECT current_date;
```

```
SELECT version();
```

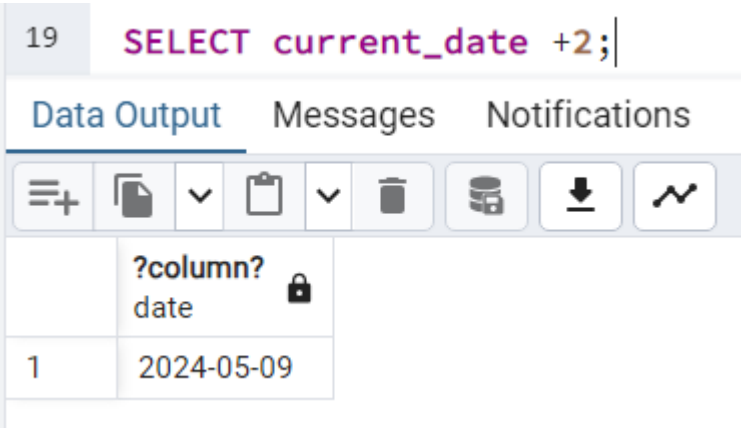




SELECT 2 + 2;



SELECT current_date +2;



Atualizando dados de uma tabela no PostgreSQL

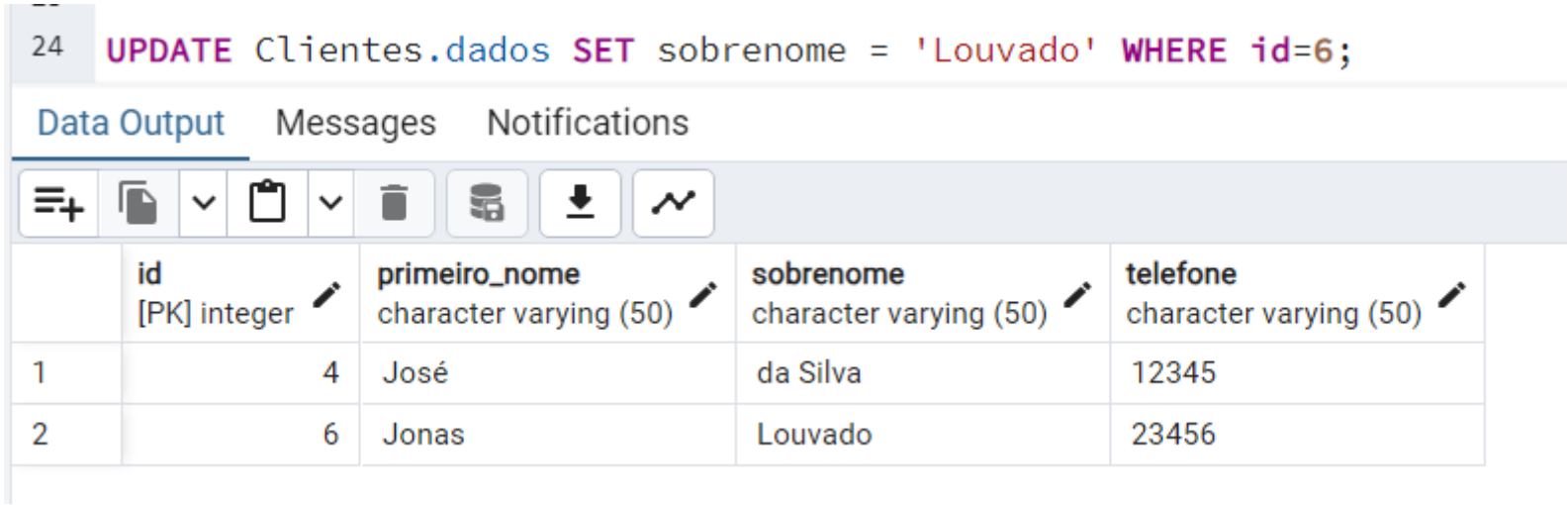
UPDATE

Para modificar dados em uma tabela no PostgreSQL, você pode utilizar o comando UPDATE. A sintaxe básica do comando UPDATE é a seguinte:

```
UPDATE nome_da_tabela
SET coluna1 = valor1, coluna2 = valor2, ..., colunaN = valorN
WHERE condição;
```

Atualizando (alterando) dados.

```
UPDATE Clientes.dados SET sobrenome = 'Louvado' WHERE id=6;
```



	id [PK] integer	primeiro_nome character varying (50)	sobrenome character varying (50)	telefone character varying (50)
1	4	José	da Silva	12345
2	6	Jonas	Louvado	23456

Referencia:

- https://www.postgresql.org/files/documentation/pdf/15/postgresql-15-A4.pdf#page=45&zoom=100,96,96

FIM