

Banco de Dados – IMD0401

Aula 15 – Structured Query Language

João Carlos Xavier Júnior

jcxavier@imd.ufrn.br

Structured Query Language

❑ Histórico:

- ❖ A linguagem SQL foi desenvolvida no início dos anos 1970 nos laboratórios da IBM em San Jose, dentro do projeto **System R**.
- ❖ O nome original da linguagem era **SEQUEL**, acrônimo para **Structured English Query Language** (Linguagem de Consulta Estruturada em Inglês).
- ❖ Mesmo tendo sido originalmente criada pela IBM, rapidamente surgiram vários "**dialetos**" produzidos por outros desenvolvedores. Essa expansão levou à necessidade de criar um **padrão para a linguagem**.

Structured Query Language

❑ Histórico:

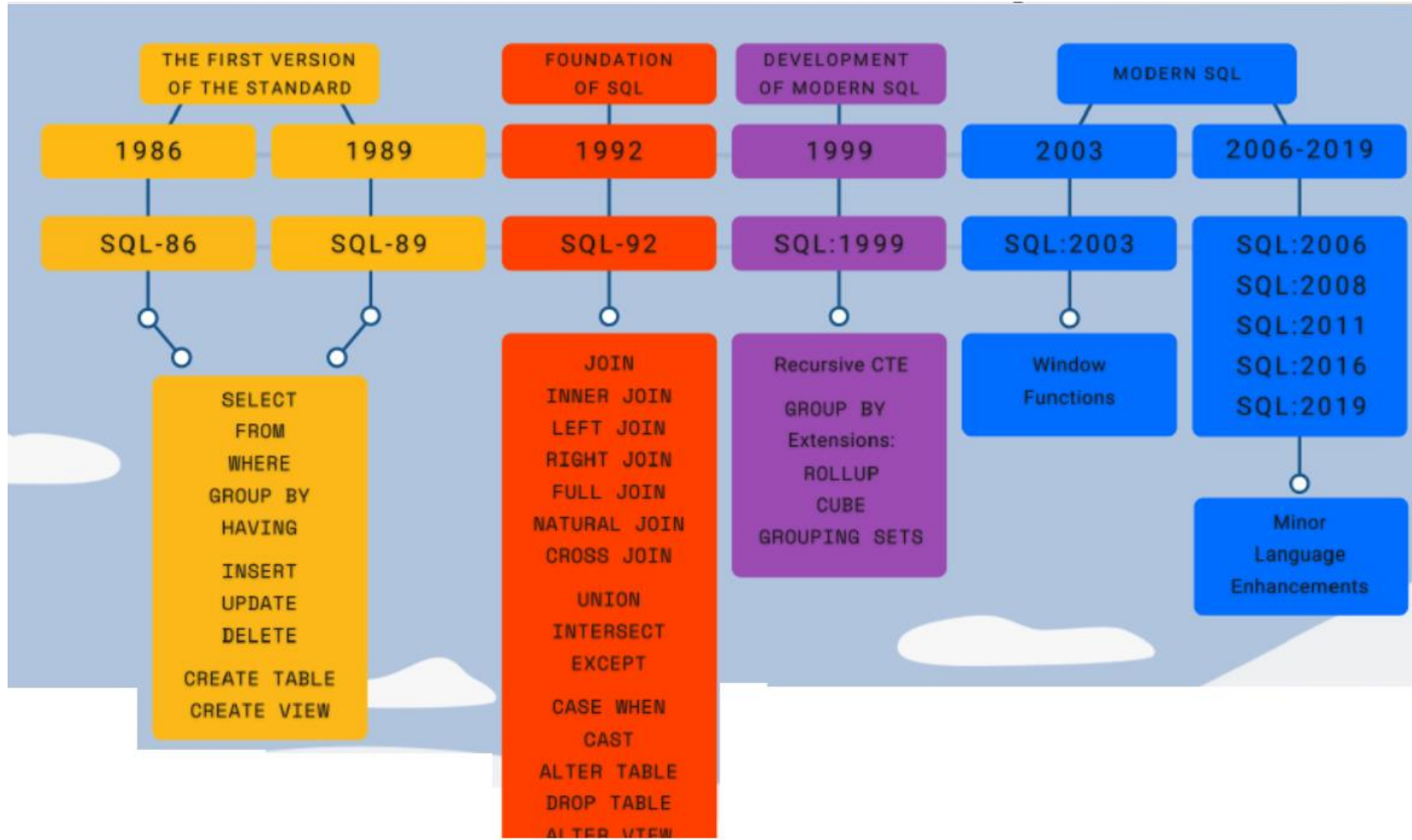
- ❖ Essa tarefa foi realizada pela American National Standards Institute (**ANSI**), em 1986, e pela International Organization for Standardization (**ISO**), em 1987.
- ❖ O SQL foi revisto em 1992, e essa versão foi dado o nome de **SQL-92**.
- ❖ Foi revisto novamente em 1999 e 2003 para se tornar **SQL:1999** (SQL3) e **SQL:2003**, respectivamente.
- ❖ O SQL:1999 usa queries recursivas e **gatilho** (triggers).

Structured Query Language

❑ Histórico:

- ❖ SQL:2003 introduz recursos relacionados ao XML (SQL / XML); Gerador de sequência, que permite sequências padronizadas.
- ❖ SQL:2006;
- ❖ SQL:2008;
- ❖ SQL:2011;
- ❖ SQL:2016
- ❖ SQL:2019 (Multi Dimensional Array type and operators)
- ❖ SQL:2023 (data type JSON).

Structured Query Language



<https://learnsql.com/blog/history-of-sql-standards/>

Structured Query Language - ANSI

□ DDL – Data Definition Language:

❖ Linguagem de Definição de Dados fornece comandos para:

- Definição de esquemas de relações;
- Exclusão de relações;
- Modificação nos esquemas de relações;
- Criação de índices.

□ DML – Data Manipulation Language:

❖ Linguagem de Manipulação de Dados fornece comandos para:

- Inserção (Insert), alteração (Update) e exclusão (Delete) de dados presentes em registros.

Structured Query Language - ANSI

□ DCL – Data Control Language:

❖ Linguagem de Controle de Dados, controla quem tem acesso para ver ou manipular dados dentro do banco de dados.

- GRANT
- REVOKE

□ DTL – Data Transaction Language:

❖ Linguagem de Transação de Dados, usado para o controle de transações no banco de dados.

- BEGIN WORK (ou START TRANSACTION)
- COMMIT
- ROLLBACK.

Structured Query Language - ANSI

□ DQL – Data Query Language:

- ❖ Linguagem de Consulta de Dados, permite ao usuário especificar uma consulta (query) como uma descrição do resultado desejado.
- ❖ O comando SELECT é composto de várias cláusulas e opções.
 - FROM
 - WHERE
 - GROUP BY
 - HAVING
 - ORDER BY
 - DISTINCT
 - UNION

Plataforma SQL

□ Plataforma a ser utilizada:

11th February 2021: [PostgreSQL 13.2, 12.6, 11.11, 10.16, 9.6.21, & 9.5.25 Released!](#)

Downloads

PostgreSQL Downloads

PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive if you want to build it yourself.

Packages and Installers

Select your operating system family:



Plataforma SQL

❑ Outra plataforma:



The world's most popular open source database



MYSQL.COM

DOWNLOADS

DOCUMENTATION

DEVELOPER ZONE

MySQL Database Service

with HeatWave for Real-time Analytics

<https://pt.wikipedia.org/wiki/MySQL>

DDL – Data Definition Language

SQL - DDL

❑ Criando um Banco:

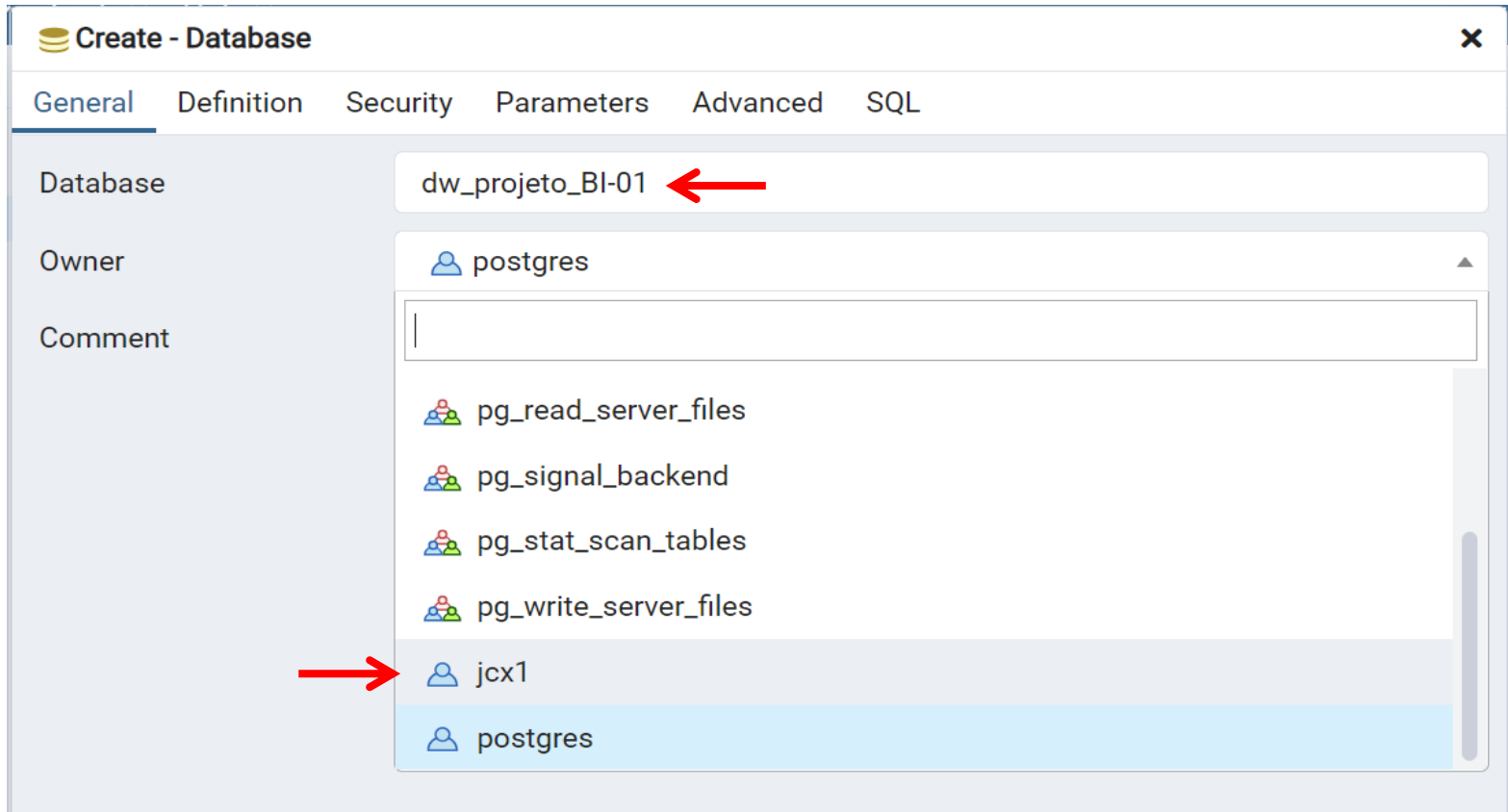
❖ Podemos criar bancos através do comando CREATE DATABASE.

❖ Sintaxe:

```
CREATE DATABASE nome_banco  
    WITH OWNER = jcx1  
    ENCODING = 'UTF8'  
    CONNECTION LIMIT = -1;
```

Banco de Dados - PostgreSQL

❑ Criando um Banco:

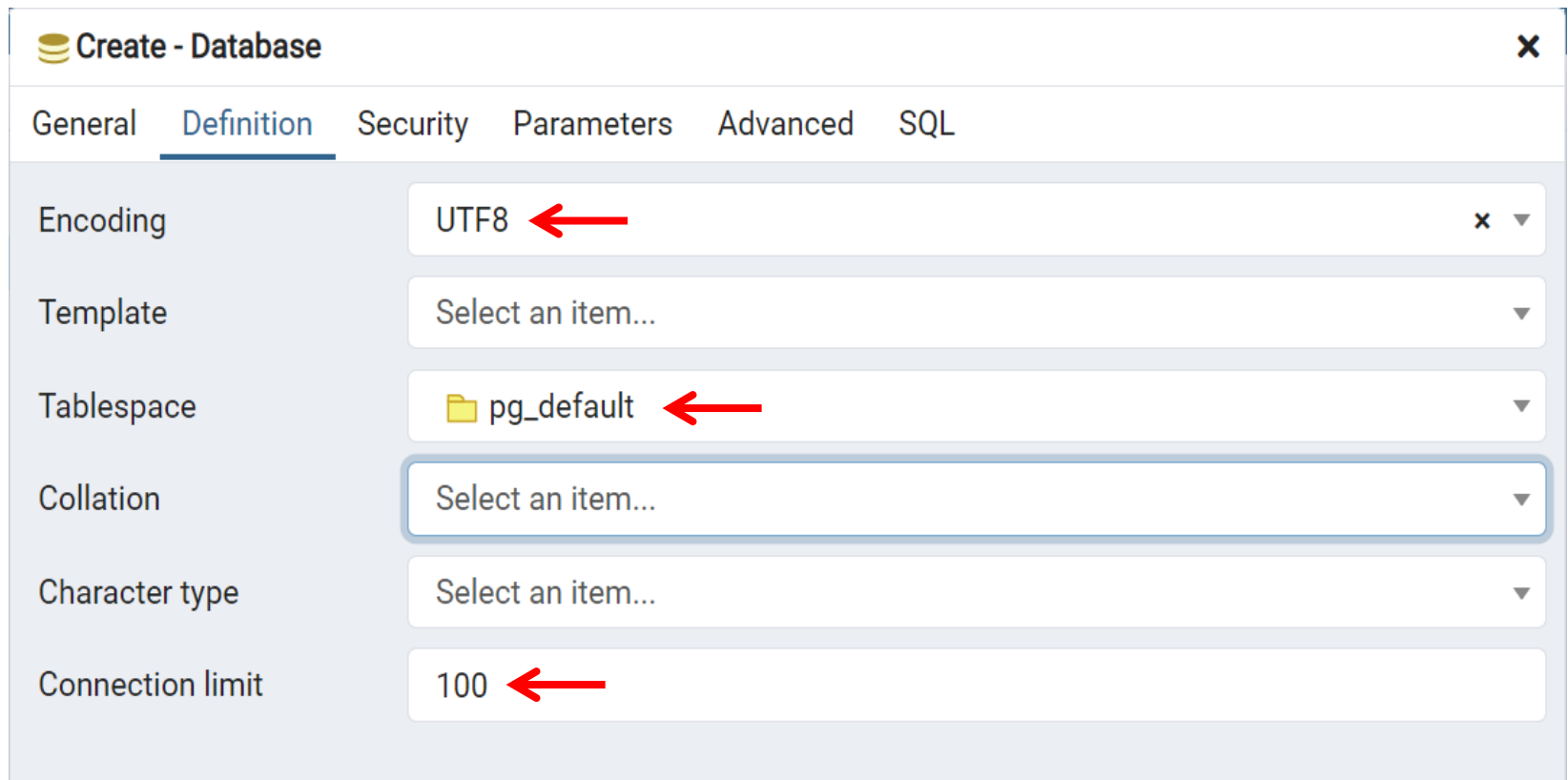


The screenshot shows the 'Create - Database' dialog box with the following fields and options:

- Database:** dw_projeto_BI-01 (indicated by a red arrow)
- Owner:** postgres
- Comment:** (empty text box)
- Roles:** A list of roles is displayed below the comment box. The roles are: pg_read_server_files, pg_signal_backend, pg_stat_scan_tables, pg_write_server_files, jcx1, and postgres. The role 'jcx1' is highlighted with a blue background and a red arrow pointing to it.

Banco de Dados - PostgreSQL

❑ Criando um Banco:



The screenshot shows the 'Create - Database' dialog box in PostgreSQL, with the 'Definition' tab selected. The dialog has a sidebar on the left with tabs: General, Definition (selected), Security, Parameters, Advanced, and SQL. The main area contains several configuration options, each with a red arrow pointing to its value:

- Encoding: UTF8
- Template: Select an item...
- Tablespace: pg_default
- Collation: Select an item...
- Character type: Select an item...
- Connection limit: 100

Banco de Dados - PostgreSQL

❑ Criando um Banco:

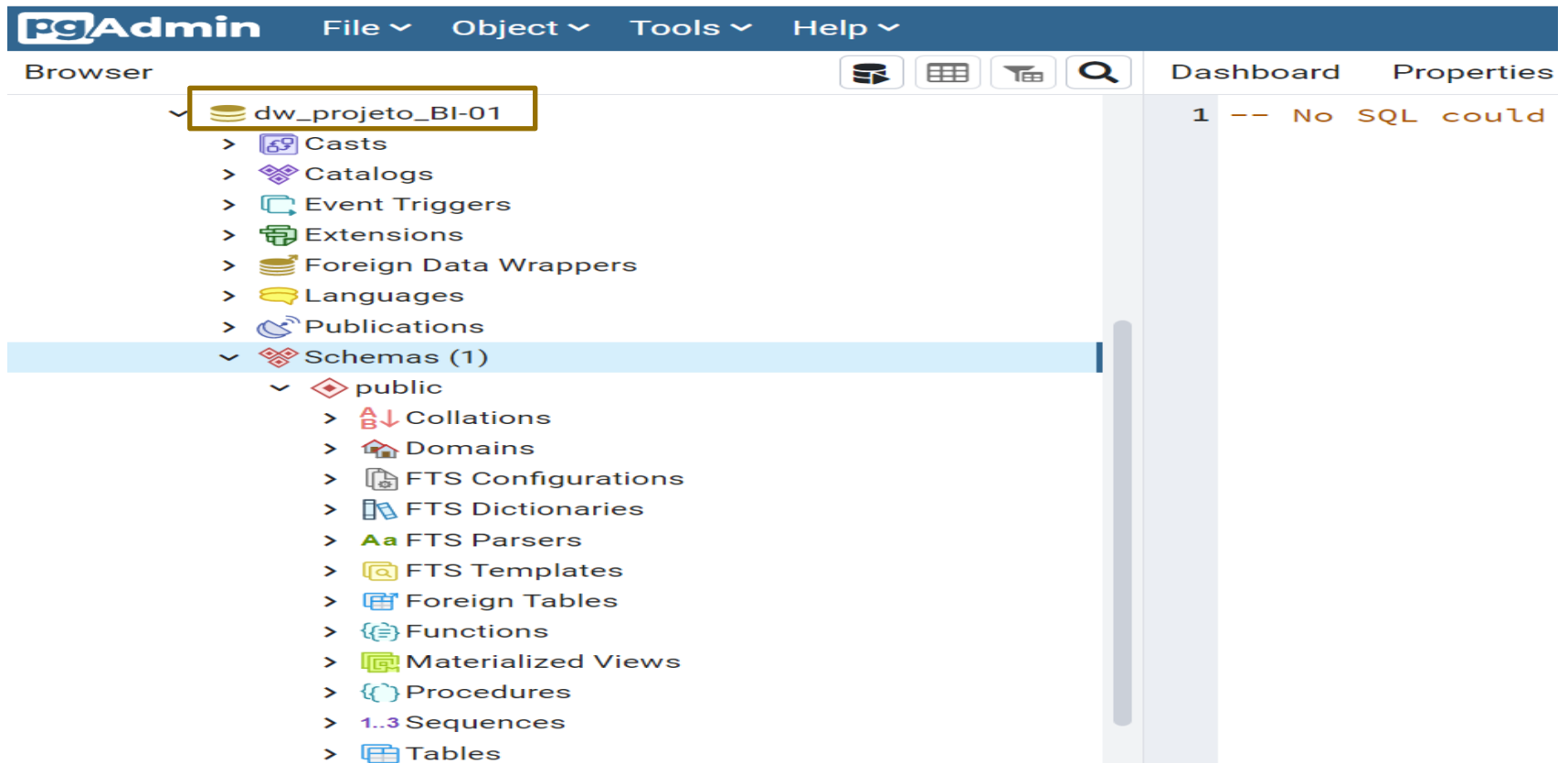
Create - Database

General Definition Security Parameters Advanced SQL

```
1 CREATE DATABASE "dw_projeto_BI-01"  
2     WITH  
3     OWNER = jcx1  
4     ENCODING = 'UTF8'  
5     TABLESPACE = pg_default  
6     CONNECTION LIMIT = 100;
```

Banco de Dados - PostgreSQL

❑ Criando um Banco:



SQL - DDL

❑ Criando uma Tabela:

❖ Uma relação é definida usando comando CREATE TABLE.

❖ Sintaxe:

```
CREATE TABLE Fornecedor (  
    idFornecedor INTEGER NOT NULL,  
    nome VARCHAR(45) NULL,  
    cidade VARCHAR(20) NULL,  
    estado VARCHAR(2) NULL,  
    PRIMARY KEY(idFornecedor)  
)
```

SQL - DDL

❑ Apagando Banco ou Tabela:

❖ A instrução DROP permite a exclusão de um banco de dados ou de uma tabela.

❖ Sintaxe:

```
DROP DATABASE nome_banco;
```

```
DROP TABLE nome_tabela;
```

SQL - DDL

❑ Alterando uma Tabela:

- ❖ A instrução ALTER é frequentemente utilizada para manipulação da estrutura de tabelas de um banco de dados.
- ❖ Permite renomear o nome de um campo;
- ❖ Adicionar um novo campo;
- ❖ Alterar o tipo de dado de um campo.
- ❖ Sintaxe:

```
ALTER TABLE nome_tabela ADD localizacao  
varchar(80);
```

SQL - DDL

❑ Alterando uma Tabela:

❖ Exemplos:

```
ALTER TABLE nome_tabela CHANGE COLUMN  
localizacao cidade varchar(80);
```

```
ALTER TABLE nome_tabela DROP COLUMN  
cidade;
```

SQL - DDL

□ Regras de Integridade:

❖ Também chamada de *constraint*.

❖ Tipos de regras:

- Atributo não-nulo (NOT NULL)
- Atributo único (UNIQUE)
- Checagem de domínio (CHECK)
- Chave primária (PRIMARY KEY)
- Chave estrangeira (FOREIGN KEY)

SQL - DDL

❑ NOT NULL:

❖ A constraint NOT NULL força um campo a não receber valores nulos.

❖ Exemplo:

```
CREATE TABLE Fornecedor (  
    idFornecedor INTEGER NOT NULL,  
    nome VARCHAR(45) NULL  
)
```

SQL - DDL

□ UNIQUE:

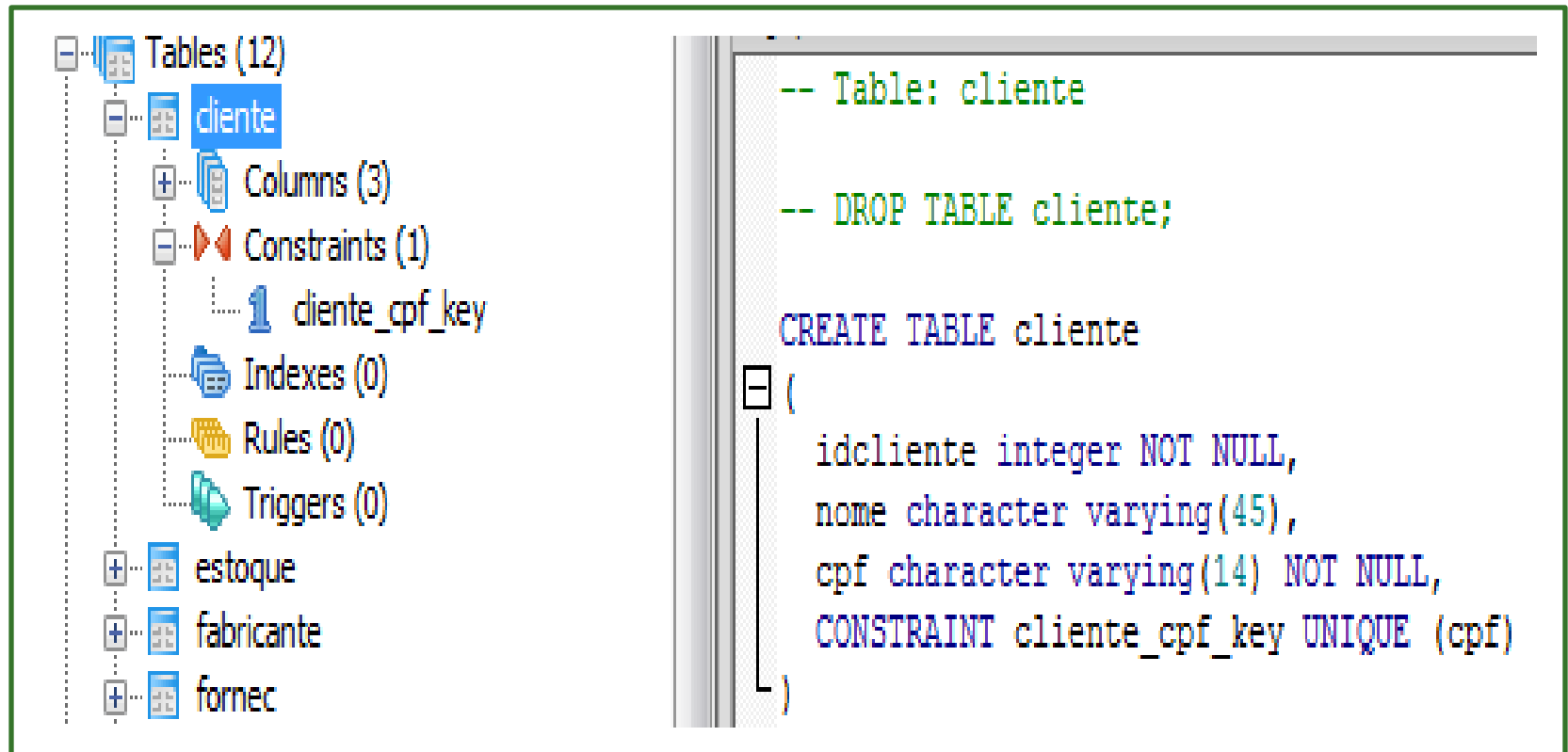
- ❖ A constraint UNIQUE serve para garantir a unicidade de cada registro de uma tabela.
- ❖ Lembrar o conceito de chave candidata.

❖ Exemplo:

```
CREATE TABLE Cliente (  
    idCliente INTEGER NOT NULL,  
    nome VARCHAR(45) NULL,  
    cpf VARCHAR(14) NOT NULL UNIQUE  
)
```

SQL - DDL

□ UNIQUE:



The screenshot displays a database management interface. On the left, a tree view under 'Tables (12)' shows the 'cliente' table selected. Below it, the 'cliente' table's structure is detailed: 'Columns (3)', 'Constraints (1)' (showing '1 cliente_cpf_key'), 'Indexes (0)', 'Rules (0)', and 'Triggers (0)'. Other tables listed are 'estoque', 'fabricante', and 'fornec'. On the right, a SQL editor window shows the following code:

```
-- Table: cliente  
  
-- DROP TABLE cliente;  
  
CREATE TABLE cliente  
(  
    idcliente integer NOT NULL,  
    nome character varying(45),  
    cpf character varying(14) NOT NULL,  
    CONSTRAINT cliente_cpf_key UNIQUE (cpf)  
)
```


SQL - DDL

❑ CHECK:

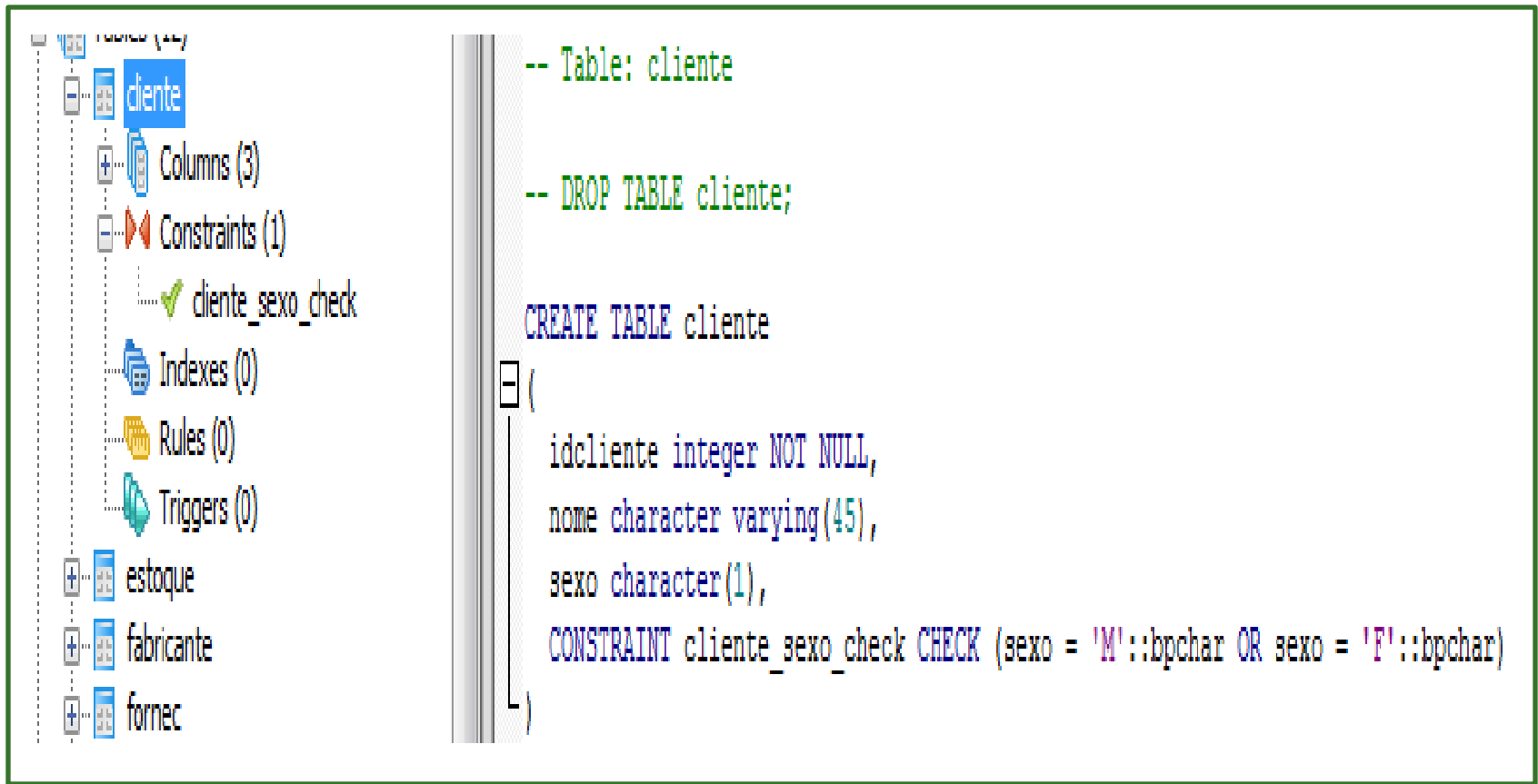
❖ A constraint CHECK estabelece condições para validar os valores de um atributo.

❖ Exemplo:

```
CREATE TABLE Cliente (  
    idCliente INTEGER NOT NULL,  
    nome VARCHAR(45) NULL,  
    sexo char(1) CHECK (sexo = 'M'  
                        or sexo = 'F'  
                        or sexo = 'I')  
)
```

SQL - DDL

❏ CHECK:



The screenshot displays a database management tool interface. On the left, a tree view shows the 'cliente' table selected, with its properties (Columns, Constraints, Indexes, Rules, Triggers) expanded. The 'cliente_sexo_check' constraint is highlighted. On the right, the SQL DDL code for the 'cliente' table is shown, including the table name, column definitions, and the check constraint.

```
-- Table: cliente  
-- DROP TABLE cliente;  
  
CREATE TABLE cliente  
(  
    idcliente integer NOT NULL,  
    nome character varying(45),  
    sexo character(1),  
    CONSTRAINT cliente_sexo_check CHECK (sexo = 'M'::bpchar OR sexo = 'F'::bpchar)  
)
```

SQL - DDL

□ PRIMARY KEY:

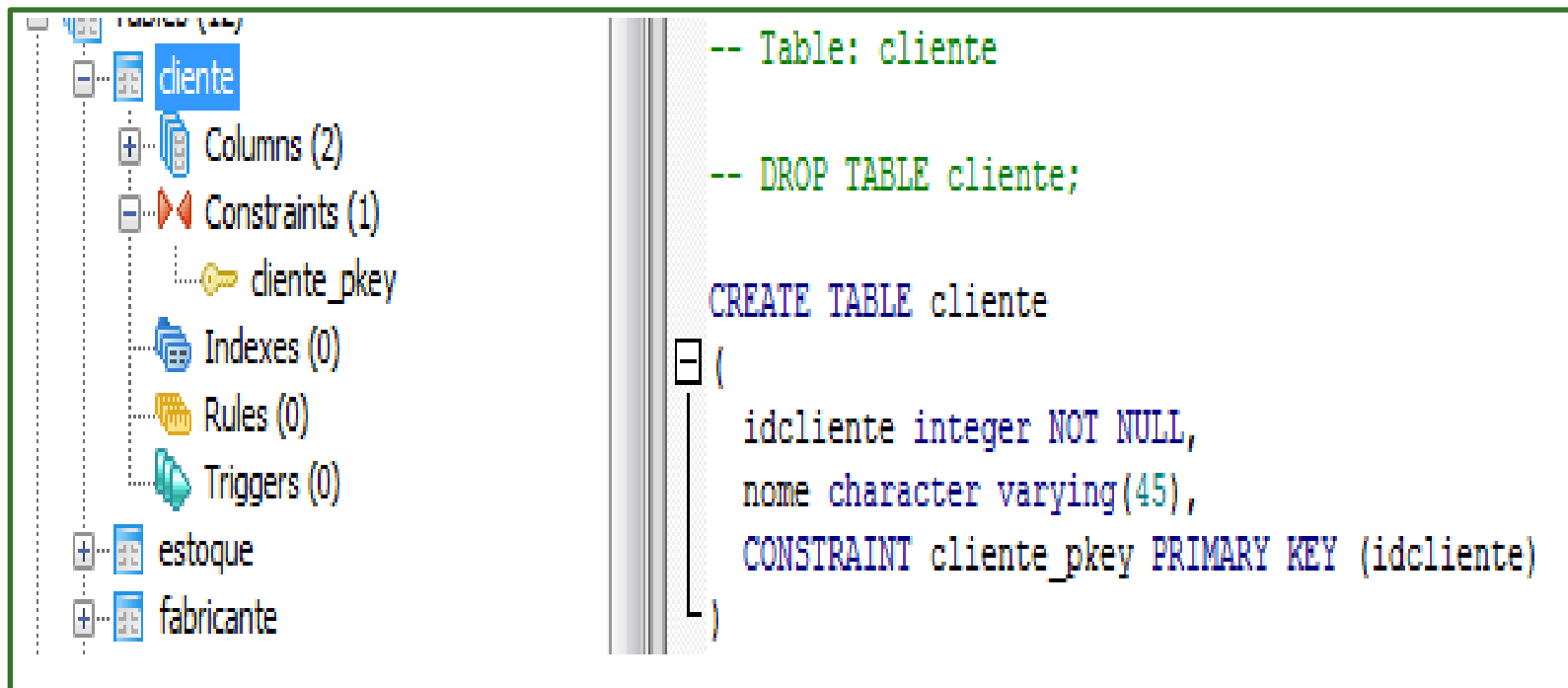
- ❖ A constraint PRIMARY KEY define a chave primária.
- ❖ Os atributos são obrigatoriamente **não nulos**.

❖ Exemplo:

```
CREATE TABLE Cliente (  
    idCliente INTEGER NOT NULL,  
    nome VARCHAR(45) NULL,  
    PRIMARY KEY(idCliente)  
)
```

SQL - DDL

PRIMARY KEY:



The screenshot displays a database management interface. On the left, a tree view shows the 'cliente' table selected, with its properties (Columns, Constraints, Indexes, Rules, Triggers) expanded. The 'Constraints' section shows a primary key constraint named 'cliente_pkey'. On the right, the SQL DDL code for the 'cliente' table is shown, including comments and the 'CREATE TABLE' statement.

```
-- Table: cliente  
-- DROP TABLE cliente;  
  
CREATE TABLE cliente  
(  
    idcliente integer NOT NULL,  
    nome character varying(45),  
    CONSTRAINT cliente_pkey PRIMARY KEY (idcliente)  
)
```

SQL - DDL

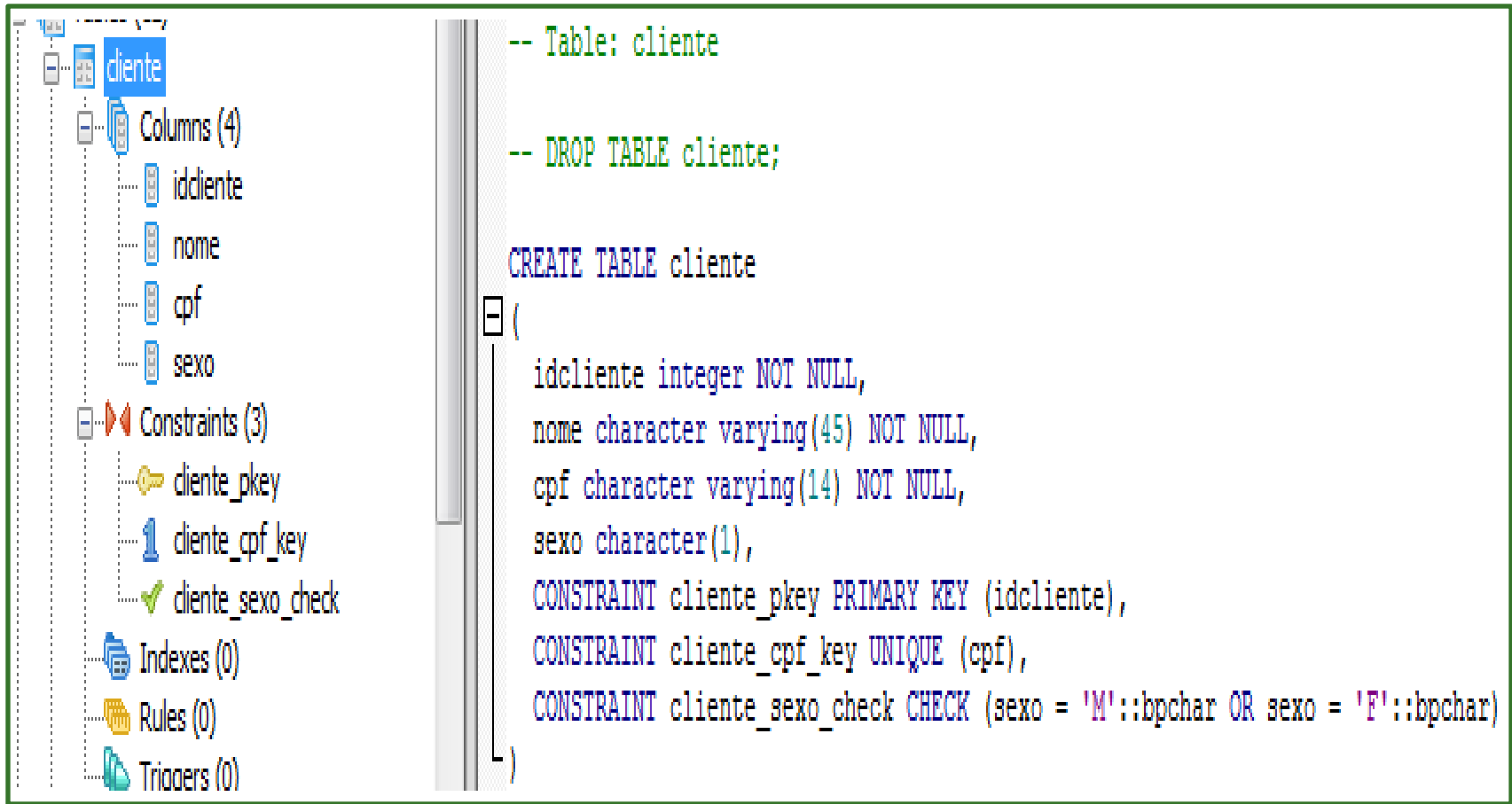
□ Todas as quatro constraints juntas:

❖ Exemplo:

```
CREATE TABLE Cliente (  
    idCliente INTEGER NOT NULL,  
    nome VARCHAR(45) NOT NULL,  
    cpf VARCHAR(14) NOT NULL UNIQUE,  
    sexo char(1) CHECK (sexo = 'M' or  
    sexo = 'F'),  
    PRIMARY KEY(idCliente)  
)
```

SQL - DDL

□ Todas as quatro constraints juntas:



The screenshot displays a database management tool interface. On the left, a tree view shows the 'cliente' table with its columns (idcliente, nome, cpf, sexo) and constraints (cliente_pkey, cliente_cpf_key, cliente_sexo_check). On the right, the SQL DDL code for the table is shown, including comments, a DROP statement, and the CREATE TABLE statement with all four constraints.

```
-- Table: cliente  
  
-- DROP TABLE cliente;  
  
CREATE TABLE cliente  
(  
    idcliente integer NOT NULL,  
    nome character varying(45) NOT NULL,  
    cpf character varying(14) NOT NULL,  
    sexo character(1),  
    CONSTRAINT cliente_pkey PRIMARY KEY (idcliente),  
    CONSTRAINT cliente_cpf_key UNIQUE (cpf),  
    CONSTRAINT cliente_sexo_check CHECK (sexo = 'M'::bpchar OR sexo = 'F'::bpchar)  
)
```

SQL - DDL

❑ Relação Cliente:

```
CREATE TABLE cliente
(
  idcliente serial NOT NULL,
  nome character varying(45) NOT NULL,
  cpf character varying(14) NOT NULL,
  genero character(1) NOT NULL,
  cidade character varying(15),
  CONSTRAINT cliente_pkey PRIMARY KEY (idcliente),
  CONSTRAINT cliente_cpf_key UNIQUE (cpf),
  CONSTRAINT cliente_genero_check CHECK (genero = 'M'::bpchar OR genero = 'F'::bpchar)
)
```

SQL - DDL

❑ FOREIGN KEY:

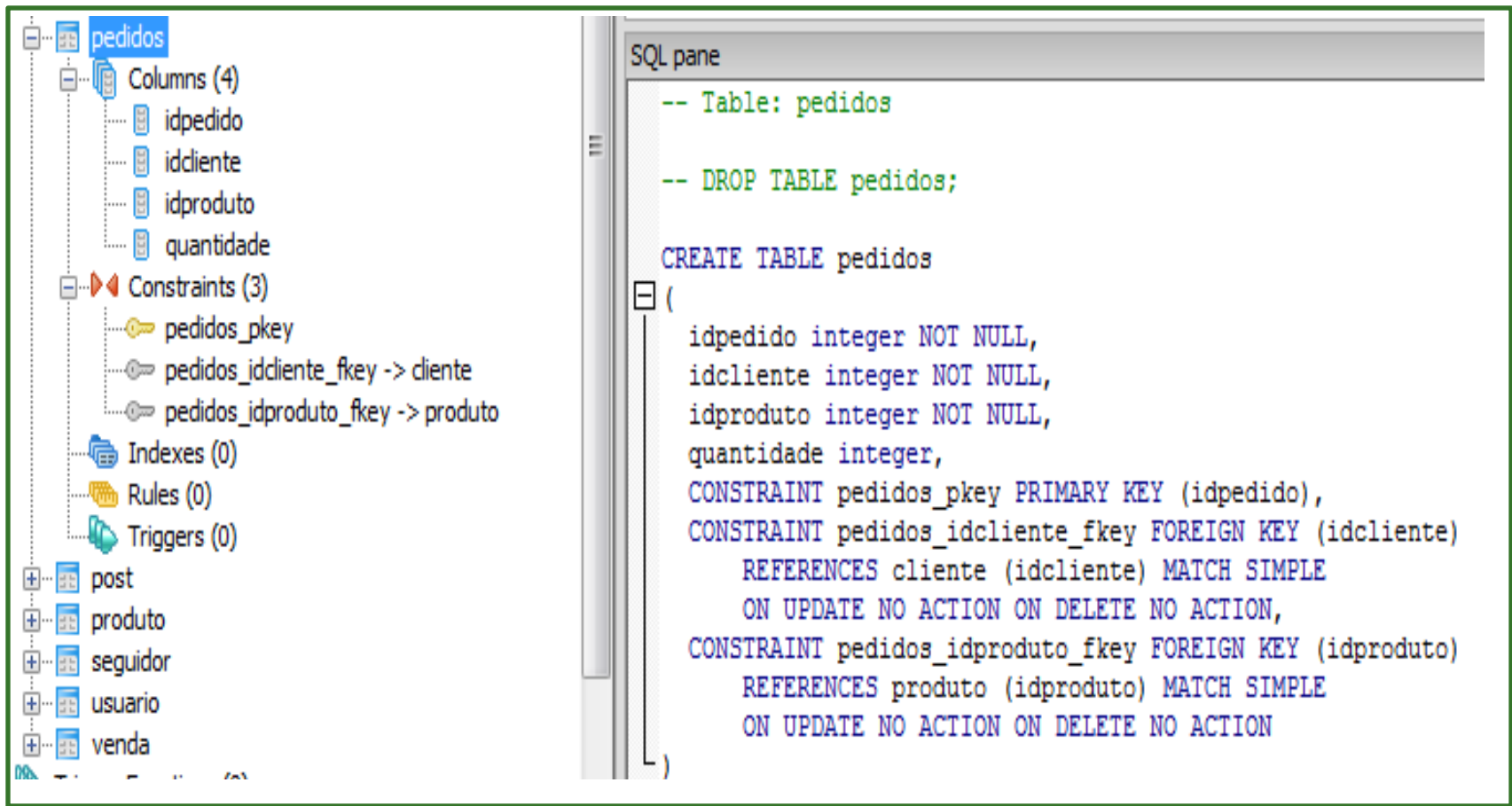
- ❖ Define a chave estrangeira.
- ❖ Garante a integridade referencial.

❖ Exemplo:

```
CREATE TABLE Pedidos (  
    idPedido SERIAL NOT NULL,  
    idCliente  INTEGER  NOT NULL REFERENCES  
    Cliente,  
    idProduto  INTEGER  NOT NULL REFERENCES  
    Produto,  
    quantidade INTEGER NULL,  
    PRIMARY KEY(idPedido)  
)
```


SQL - DDL

FOREIGN KEY:



The screenshot displays a database management interface. On the left, a tree view shows the 'pedidos' table structure. It includes four columns: 'idpedido', 'idcliente', 'idproduto', and 'quantidade'. There are three constraints: a primary key 'pedidos_pkey' on 'idpedido', and two foreign keys, 'pedidos_idcliente_fkey' and 'pedidos_idproduto_fkey', which reference the 'cliente' and 'produto' tables respectively. The right pane, titled 'SQL pane', shows the corresponding SQL DDL code for creating the 'pedidos' table.

```
-- Table: pedidos
-- DROP TABLE pedidos;

CREATE TABLE pedidos
(
    idpedido integer NOT NULL,
    idcliente integer NOT NULL,
    idproduto integer NOT NULL,
    quantidade integer,
    CONSTRAINT pedidos_pkey PRIMARY KEY (idpedido),
    CONSTRAINT pedidos_idcliente_fkey FOREIGN KEY (idcliente)
        REFERENCES cliente (idcliente) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT pedidos_idproduto_fkey FOREIGN KEY (idproduto)
        REFERENCES produto (idproduto) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

SQL - DDL

❑ FOREIGN KEY:

❖ Inclui regras de exclusão/atualização:

- On Delete;
- On Update.

❖ Para a ação de deletar (apagar), as opções mais comuns são:

- RESTRICT: evita a exclusão um registro que possui outras referências (pai/filhos).
- NO ACTION: checa se há referências entre pai e filhos, se houver, um erro é gerado.
- CASCADE: especifica que quando um registro referenciado (pai) é excluído, os registros filhos devem ser excluídos.
- SET NULL e SET DEFAULT.

SQL - DDL

❑ FOREIGN KEY:

❖ On Delete and On Update.

```
CREATE TABLE pedidos(  
    idpedido integer NOT NULL,  
    idcliente integer NOT NULL,  
    idproduto integer NOT NULL,  
    quantidade integer,  
    CONSTRAINT pedidos_pkey PRIMARY KEY (idpedido),  
    CONSTRAINT pedidos_idcliente_fkey FOREIGN KEY  
        (idcliente) REFERENCES cliente (idcliente)  
        MATCH SIMPLE ON UPDATE NO ACTION ON DELETE  
        NO ACTION,  
    CONSTRAINT pedidos_idproduto_fkey FOREIGN KEY  
        (idproduto) REFERENCES produto (idproduto)  
        MATCH SIMPLE ON UPDATE NO ACTION ON DELETE  
        NO ACTION  
)
```

Questões...



DML – Data Manipulation Language

SQL - DML

❑ Inclusão:

❖ O comando INSERT é usado para adicionar uma tupla a uma relação.

❖ Exemplo:

```
Insert into Cliente (idCliente, nome,  
cpf, genero, cidade) values (1, 'Roberto  
Carlos', '111.111.111-01', 'M',  
'Natal');
```

❖ Ou

```
Insert into Cliente values (2, 'Ana  
Maria', '222.222.222-02', 'F',  
'Parnamirim');
```

SQL - DML

❑ Inclusão:

❖ Podemos omitir uma ou mais colunas da relação destino. Toda tupla inserida terá um **valor nulo** em cada posição de coluna omitida.

❖ Exemplo:

```
Insert      into      Cliente      values      (3,  
'Francisco      dos      Santos',      '333.333.333-  
03',      'M');
```

SQL - DML

□ Insert:

Ciente

	idcliente [PK] integer	nome character varying(45)	cpf character varying(14)	sexo character(1)	cidade character varying
1	1	Roberto Carlos	111.111.111-01	M	Natal
2	2	Ana Maria	222.222.222-02	F	Parnamirim
3	3	Francisco dos Santos	333.333.333-03	M	
*					

SQL - DML

❑ Insert:

- ❖ Quais os códigos SQL abaixo apresentarão erros de violação de constraint?

```
Insert into Cliente values (4, 'Maria dos  
Anjos', '444.444.444-04', null, 'Mossoro');
```

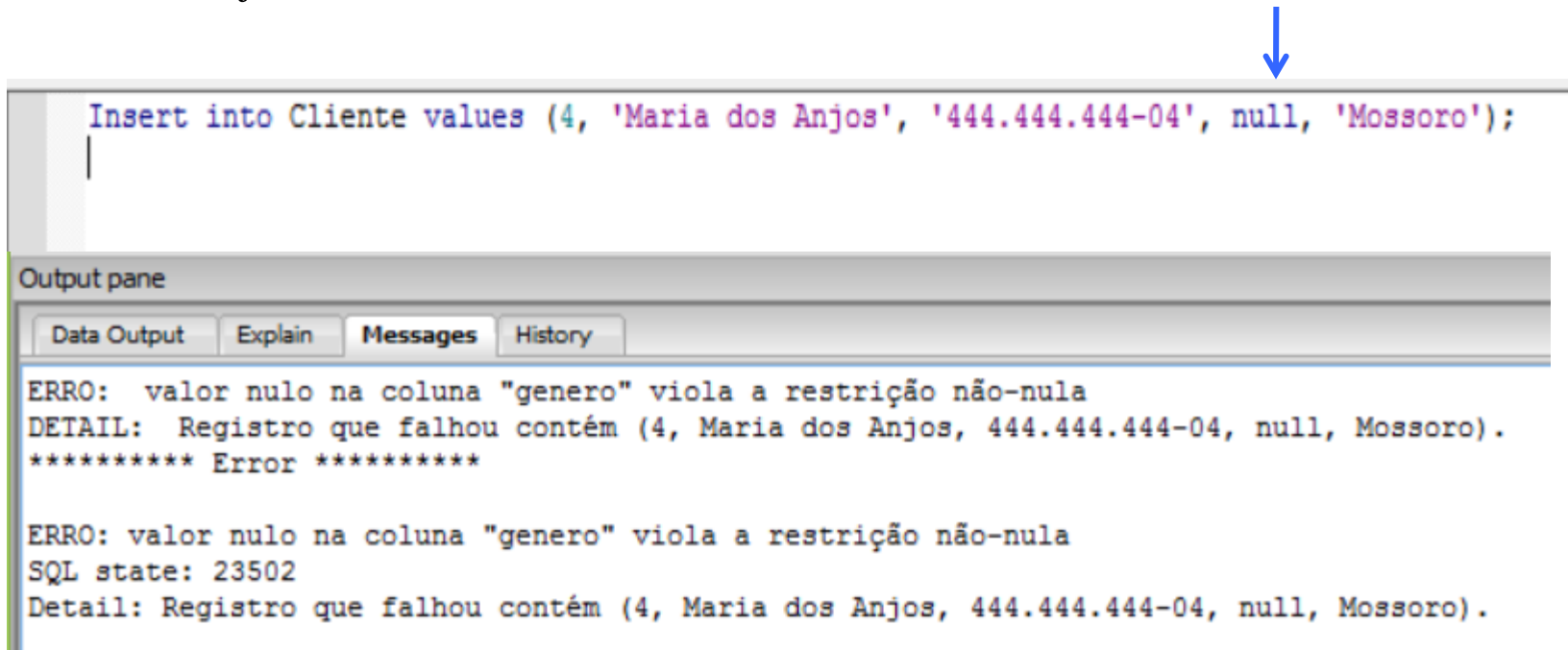
```
Insert into Cliente values (4, 'Maria dos  
Anjos', null, 'F', 'Mossoro');
```

```
Insert into Cliente values (4, null,  
'444.444.444-04', 'F', 'Mossoro');
```

SQL - DML

❑ Insert:

- ❖ Quais os códigos SQL abaixo apresentarão erros de violação de constraint?

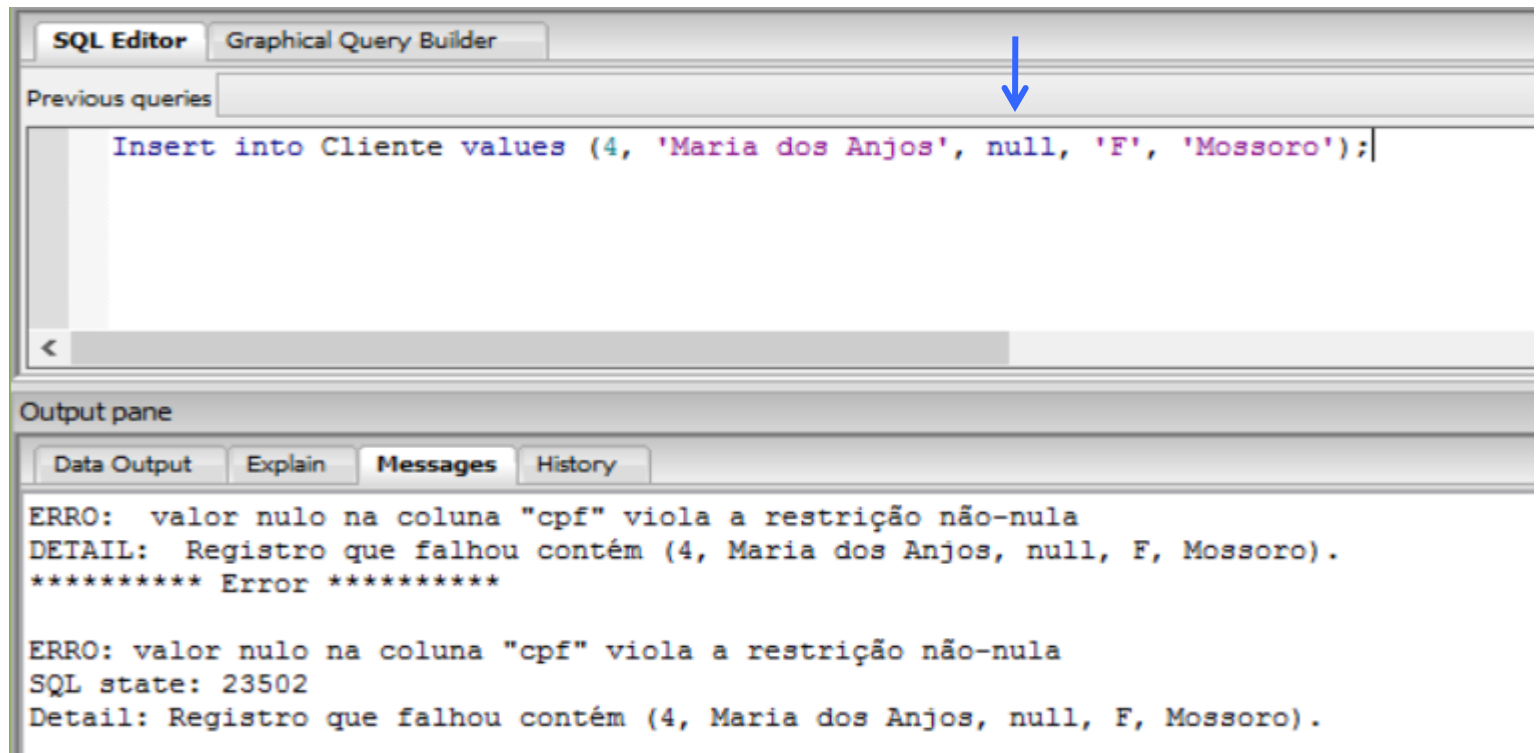


A screenshot of a SQL IDE interface. At the top, a blue arrow points down to a text area containing the SQL statement: `Insert into Cliente values (4, 'Maria dos Anjos', '444.444.444-04', null, 'Mossoro');`. Below the text area is an "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Messages" tab is selected, displaying two error messages. The first message is: `ERRO: valor nulo na coluna "genero" viola a restrição não-nula`, followed by `DETAIL: Registro que falhou contém (4, Maria dos Anjos, 444.444.444-04, null, Mossoro).` and `***** Error *****`. The second message is: `ERRO: valor nulo na coluna "genero" viola a restrição não-nula`, followed by `SQL state: 23502` and `Detail: Registro que falhou contém (4, Maria dos Anjos, 444.444.444-04, null, Mossoro).`

SQL - DML

❑ Insert:

- ❖ Quais os códigos SQL abaixo apresentarão erros de violação de constraint?



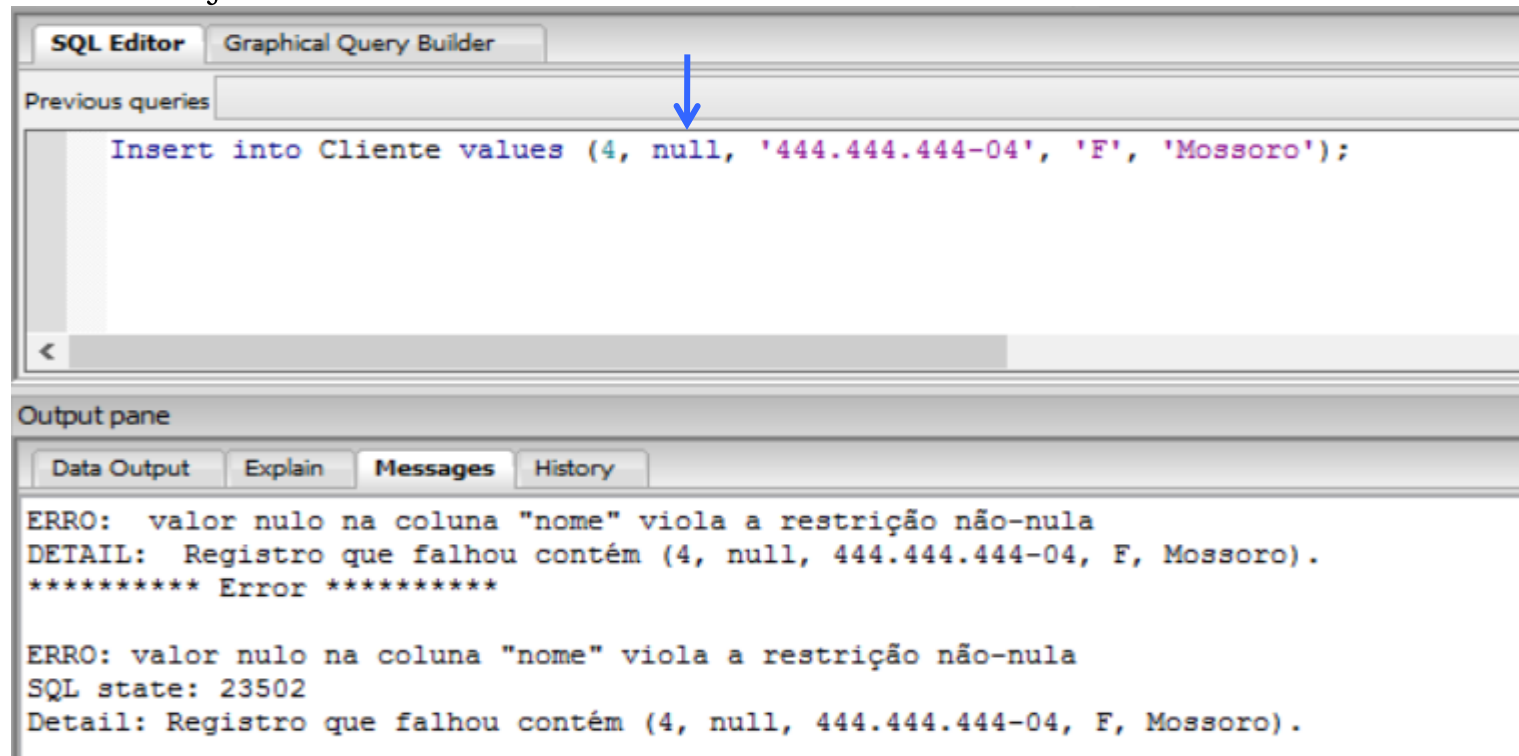
The screenshot shows a SQL Editor window with two tabs: "SQL Editor" and "Graphical Query Builder". The "SQL Editor" tab is active, and a blue arrow points to the "Previous queries" list. The SQL statement in the editor is: `Insert into Cliente values (4, 'Maria dos Anjos', null, 'F', 'Mossoro');`. Below the editor is the "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Messages" tab is selected, displaying two error messages:
ERRO: valor nulo na coluna "cpf" viola a restrição não-nula
DETAIL: Registro que falhou contém (4, Maria dos Anjos, null, F, Mossoro).
***** Error *****

ERRO: valor nulo na coluna "cpf" viola a restrição não-nula
SQL state: 23502
Detail: Registro que falhou contém (4, Maria dos Anjos, null, F, Mossoro).

SQL - DML

❑ Insert:

- ❖ Quais os códigos SQL abaixo apresentarão erros de violação de constraint?



The screenshot shows a SQL Editor window with two tabs: "SQL Editor" and "Graphical Query Builder". The "SQL Editor" tab is active, and a blue arrow points to the word "null" in the following SQL statement:

```
Insert into Cliente values (4, null, '444.444.444-04', 'F', 'Mossoro');
```

Below the editor is an "Output pane" with four tabs: "Data Output", "Explain", "Messages", and "History". The "Messages" tab is selected, displaying the following error message:

```
ERRO: valor nulo na coluna "nome" viola a restrição não-nula  
DETAIL: Registro que falhou contém (4, null, 444.444.444-04, F, Mossoro).  
***** Error *****  
  
ERRO: valor nulo na coluna "nome" viola a restrição não-nula  
SQL state: 23502  
Detail: Registro que falhou contém (4, null, 444.444.444-04, F, Mossoro).
```

SQL - DML

□ Insert:

Ciente

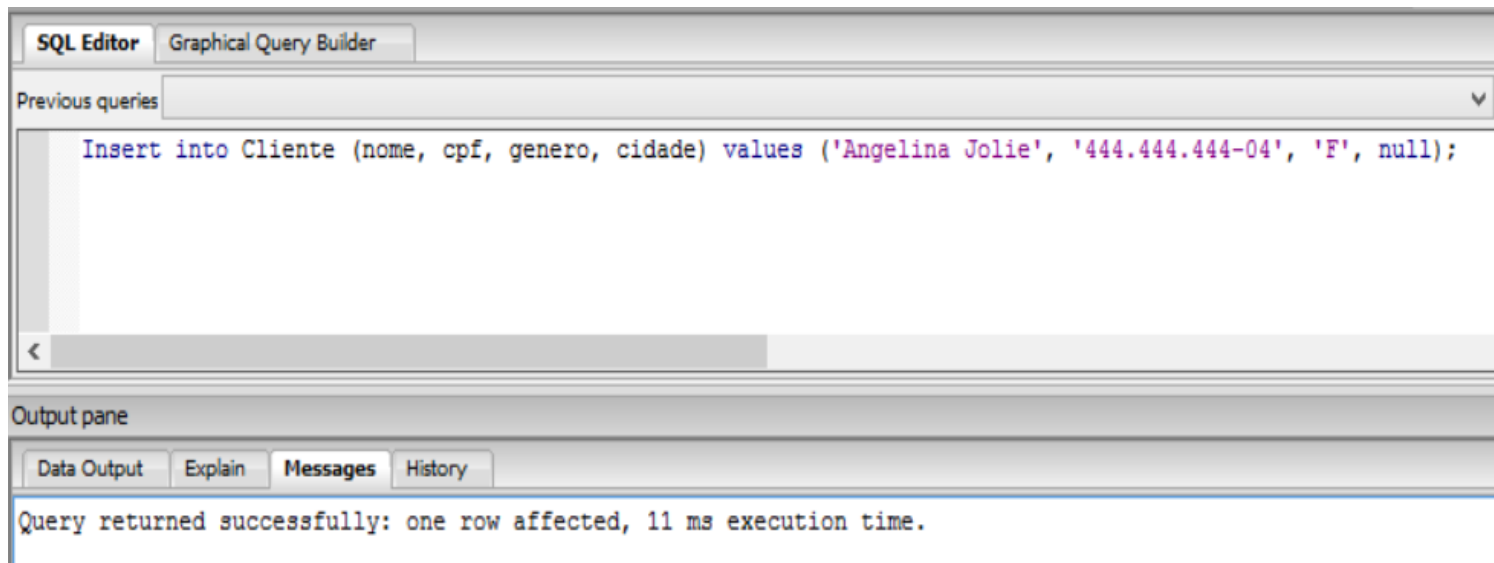
	idcliente [PK] integer	nome character varying(45)	cpf character varying(14)	sexo character(1)	cidade character varying
1	1	Roberto Carlos	111.111.111-01	M	Natal
2	2	Ana Maria	222.222.222-02	F	Parnamirim
3	3	Francisco dos Santos	333.333.333-03	M	
*					

SQL - DML

❑ Inclusão de campo **SERIAL**:

❖ Exemplo:

```
Insert into Cliente (nome, cpf, genero, cidade) values ('Angelina Jolie', '444.444.444-04', 'F', null);
```



SQL - DML

□ Insert:

Ciente

	idcliente [PK] serial	nome character varying(45)	cpf character varying(14)	genero character(1)	cidade character varyi
1	1	Roberto Carlos	111.111.111-01	M	Natal
2	2	Ana Maria	222.222.222-02	F	Parnamirim
3	3	Francisco dos Santos	333.333.333-03	M	
4	4	Angelina Jolie	444.444.444-04	F	
*					

SQL - DML

❑ Alteração:

❖ O comando UPDATE modifica o valor de atributos de uma ou mais tuplas.

❖ Exemplo:

```
Update Cliente set cidade = 'Natal';
```

	idcliente [PK] serial	nome character varying(45)	cpf character varying(14)	genero character(1)	cidade character varying(15)
1	1	Roberto Carlos	111.111.111-01	M	Natal
2	2	Ana Maria	222.222.222-02	F	Natal
3	3	Francisco dos Santos	333.333.333-03	M	Natal
*					

SQL - DML

❑ Alteração:

❖ O comando UPDATE modifica o valor de atributos de uma ou mais tuplas.

❖ Ou

```
Update Cliente set cidade = 'Parnamirim'
where cpf = '333.333.333-03';
```

	idcliente [PK] serial	nome character varying(45)	cpf character varying(14)	genero character(1)	cidade character varying(15)
1	1	Roberto Carlos	111.111.111-01	M	Natal
2	2	Ana Maria	222.222.222-02	F	Natal
3	3	Francisco dos Santos	<u>333.333.333-03</u>	M	<u>Parnamirim</u>
*					

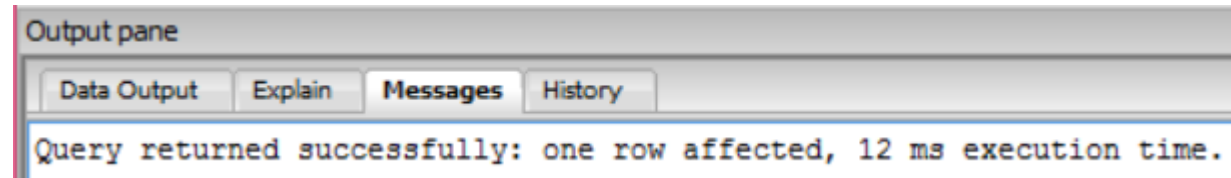
SQL - DML

❑ Exclusão:

❖ O comando DELETE remove tuplas de uma relação.

❖ Exemplo:

```
Delete from Cliente where cpf =  
'222.222.222-02';
```



	idcliente [PK] serial	nome character varying(45)	cpf character varying(14)	genero character(1)	cidade character varying(15)
1	1	Roberto Carlos	111.111.111-01	M	Natal
2	3	Francisco dos Santos	333.333.333-03	M	Parnamirim
*					

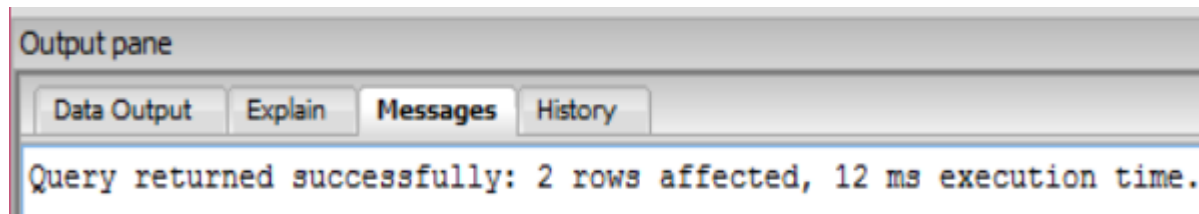
SQL - DML

❏ Exclusão:

❖ Implica em remover todas as tuplas da relação, ou seja, a relação permanece no BD como uma relação vazia.

❖ Exemplo:

`Delete from Cliente;`



	idcliente [PK] serial	nome character varying(45)	cpf character varying(14)	genero character(1)	cidade character varying(15)
*					

Questões...



Obrigado!!!

