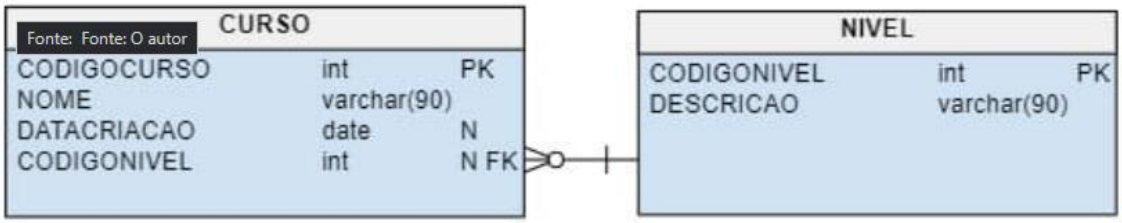
Criando Database: Comandos para criação e alteração de tabelas



-- Comando para criar um database. (“--" significa comentário)

```
CREATE DATABASE BDESTUDO;
```

-- Comando para remover um *database*.

```
DROP DATABASE BDESTUDO;
```

Antes de prosseguirmos com a criação de tabelas, principal objetivo deste módulo, é preciso compreender que todo *database* criado no PostgreSQL possui um *schema* padrão denominado *public*, onde as tabelas a serem criadas no *database* serão armazenadas. Assim, se não especificarmos a qual *schema* do *database* pertence uma tabela que estamos criando, esta será armazenada no *schema public*. Para especificarmos um *schema* diferente do *public*, antes de criar uma tabela, devemos criar o respectivo *schema*, com o comando CREATE SCHEMA.

-- Comando para criar um schema.

```
CREATE SCHEMA esquema;
```

CRIANDO TABELAS

Sintaxe **básica** do comando CREATE TABLE:

```
CREATE TABLE NOMETABELA (
COLUNA1 - TIPODEDADOS [RESTRIÇÃO],
COLUNAN - TIPODEDADOS [RESTRIÇÃO],
PRIMARY KEY (COLUNA),
FOREIGN KEY (COLUNA) REFERENCES NOMETABELA (COLUNA)
CONSTRAINT RESTRIÇÃO);
```

TIPOS DE DADOS

bigint	valores inteiros compreendidos entre -9.223.372.036.854.775.808 e 9.223.372.036.854.775.807
char(comprimento)	útil para sequências de caracteres de tamanho fixo. O parâmetro comprimento determina o valor da sequência. Esse tipo de dado preenche a coluna com espaços em branco até completar o total de caracteres definidos, caso a totalidade do tamanho do campo não esteja preenchida
date	data de calendário no formato AAAA-MM-DD
decimal	determina a precisão do valor de casas decimais
double	precisão do valor de até 15 casas decimais
int ou integer	valores inteiros compreendidos entre -2.147.483.648 e 2.147.483.647
money	valores monetários compreendidos entre –92.233.720.368.547.758.08 e 92.233.720.368.547.758.07
numeric	precisão do valor de casas decimais
real	precisão do valor de até seis casas decimais
serial	gera valor único inteiro sequencial para um novo registro entre 1 e 2.147.483.647
smallint	representa valores compreendidos entre 32.768 e 32.767
time	representa horário no intervalo de tempo entre 00:00:00 e 24:00:00
varchar(comprimento)	útil para sequência de dados de caracteres com comprimento variável. Não armazena espaços em branco não utilizados para compor *string* (em branco) em seu lado direito

```
-- Comando para Criar a tabela nivel;
CREATE TABLE NIVEL (
        CODIGONIVEL INT NOT NULL,
        DECRIÇÃO VARCHAR(90) NOT NULL,
        CONSTRAINT CHAVEPNIVEL PRIMARY KEY (CODIGONIVEL)
);
-- Comando para criar a tabela curso;
CREATE TABLE CURSO (
        CODIGOCURSO int NOT NULL,
        NOME varchar(90) NOT NULL UNIQUE,
        DATACRIACAO date NULL,
```

```
    CODIGONIVEL int NULL,
    CONSTRAINT CHAVEPCURSO PRIMARY KEY (CODIGOCURSO),
    FOREIGN KEY (CODIGONIVEL) REFERENCES NIVEL (CODIGONIVEL)
);
```

Gerenciamento de *scripts* na prática

Acesse o diretório onde é salvo os databases do postgreSql:

C:\Program Files\PostgreSQL\15\data\base

Lá os databases são numerados com um **OID**.

Após a execução do comando **CREATE DATABASE TESTEBANCO;** foi criada a pasta <número qualque atribuído pelo postgreSQl>

Ainda, o PostgreSQL mantém informações sobre todos os *databases* em um *database* especial denominado catálogo, cujas tabelas possuem nomes iniciados com o prefixo PG_. Por exemplo, informações sobre os *databases* existentes em um servidor são armazenadas na tabela PG_DATABASE. Assim, caso você queira identificar o nome correspondente ao OID do PostgreSQL, basta executar o comando a seguir:

```
SELECT OID, DATNAME FROM PG_DATABASE;
```

ALTERAÇÃO DE TABELA

Suponha que surgiu a necessidade de modelar a informação sobre a data de primeiro reconhecimento do curso. Podemos, então, alterar a estrutura da tabela CURSO, adicionando uma coluna opcional denominada DTRECONH. O comando ALTER TABLE é útil no contexto dessa tarefa.

Sintaxe **básica** do comando ALTER TABLE:

ALTER TABLE <NOMEDATABELA> AND <COLUNA><TIPODEDADOS>;

Alteração da tabela CURSO:

```
ALTER TABLE CURSO ADD DTRECONH DATE;
```

Para remover uma coluna de uma tabela:

```
ALTER TABLE <NOMEDATABELA> DROP <COLUNA> ;
```

```
-- Comando para alterar a tabela CURSO, removendo a coluna DTRECONH
ALTER TABLE CURSO DROP DTRECONH ;
```

Remoção de tabela

Remoção de tabelas do Banco de dados:

DROP TABLE <NOMETABELA>;

```
-- Comando para remover a tabela CURSO
DROP TABLE CURSO;
```

CRIAÇÃO E ALTERAÇÃO DE TABELAS RELACIONADAS

O relacionamento entre as tabelas NIVEL e CURSO foi declarado no bloco CREATE TABLE da tabela CURSO. No entanto, nós poderíamos ter optado por criar as tabelas NIVEL e CURSO sem relacionamento, para, em seguida, alterar a tabela CURSO, adicionando a restrição de chave estrangeira.

Na hipótese dessa estratégia, o *script* SQL ficaria do seguinte modo:

-- Comando para criar a tabela nivel;

CREATE TABLE NIVEL(

CODIGONIVEL int NOT NULL,

DESCRICAO varchar(90) NOT NULL,

CONSTRAINT CHAVEPNIVEL PRIMARY KEY (CODIGONIVEL)

);

--Comando para criar a tabela curso;

CREATE TABLE CURSO (

CODIGOCURSO int NOT NULL,

NOME varchar(90) NOT NULL UNIQUE,

DATACRIACAO date NULL,

CODIGONIVEL int NULL,

CONSTRAINT CHAVECURSO PRIMARY KEY (CODIGOCURSO)

);

--Comando para alterar a tabela curso, adicionando chave estrangeira;

ALTER TABLE CURSO ADD FOREIGN KEY (CODIGONIVEL) REFERENCES NIVEL;

Note que, no *script* anterior, as tabelas foram criadas sem chave estrangeira (linhas 1 a 12). Na linha 14, o comando ALTER TABLE modifica a estrutura da tabela CURSO, implementando a restrição de chave estrangeira que representa o relacionamento entre CURSO e NIVEL.

Cuidados ao manipular tabelas relacionadas

Em algumas situações, mesmo que o comando para alteração ou exclusão esteja correto sob o ponto de vista sintático, o SGBD sempre prioriza a integridade dos dados e pode inibir sua execução caso o resultado tenha potencial para gerar inconsistência nos dados.

EX:

```
-- Comando para remover a tabela NIVEL
DROP TABLE NIVEL;
```

O SGBD não removerá a tabela NIVEL e retornará uma mensagem de erro, informando que há um objeto (tabela CURSO) que depende da tabela NIVEL.

Perceba que se o SGBD removesse a tabela NIVEL, a tabela CURSO ficaria inconsistente, visto que sua chave estrangeira (CODIGONIVEL) faz referência à chave primária da tabela NIVEL. Assim, antes de remover uma tabela do banco de dados, é necessário avaliar todos os relacionamentos desta.

E se caso quiséssemos remover a tabela nível, o que fazer? Teríamos que remover todas as dependências para em seguida podermos remover a tabela?

Não, o sbgd possou um recurso que remove de forma automática:

Trata-se da remoção em cascata

```
-- Comando para remover a tabela NIVEL - remoção em cascata
DROP TABLE NIVEL CASCADE;
```

Internamente, o comando altera a estrutura da tabela CURSO, removendo a restrição de chave estrangeira da coluna CODIGONIVEL. Em seguida, a tabela NIVEL é removida do banco de dados.