

# Persistência de Dados

utilizando PostgreSQL

**Vinicius Takeo Friedrich Kuwaki**

Universidade do Estado de Santa Catarina

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

Criando Banco de Dados

Criando Tabelas

Criando Sequências

Inserindo Dados

Buscando Dados

Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

Criando Banco de Dados

Criando Tabelas

Criando Sequências

Inserindo Dados

Buscando Dados

Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

# Instalação - Linux

- Acesse o link: <https://www.postgresql.org/download/>
- Escolha a opção que se adequa ao seu SO:
  - Linux
    - **Red Hat** family Linux (including **CentOS/Fedora/Scientific/Oracle** variants)
    - **Debian** GNU/Linux and derivatives
    - **Ubuntu** Linux and derivatives
    - **SUSE** and **openSUSE**
    - **Other** Linux
- Para ver as informações do SO execute o comando: `uname -a`
- Caso esteja usando uma distro baseada em Ubuntu, execute: `lsb_release -a`
- Siga os passos específicos para a sua distro;

# Instalação - Linux

- Feito isso, instale a ferramenta gráfica **pgadmin4**

## Included in distribution

Debian Includes PostgreSQL by default. To install PostgreSQL on Debian, use the `apt-get` (or other apt-driving) command:

```
apt-get install postgresql-12
```

The repository contains many different packages including third party addons. The most common and important packages are (substitute the version number as required):

postgresql-client-12	client libraries and client binaries
postgresql-12	core database server
postgresql-contrib-9.x	additional supplied modules (part of the postgresql-xx package in version 10 and later)
libpq-dev	libraries and headers for C language frontend development
postgresql-server-dev-12	libraries and headers for C language backend development
pgadmin4	pgAdmin 4 graphical administration utility

- Utilize o comando: `apt-get install postgresql-12 pgadmin4`
- Fique atento a versão do postgres, caso seja diferente da 12, utilize a correta:  
`apt-get install postgresql-XX pgadmin4`

# Instalação Windows e MacOS

- Acesse o link: <https://www.postgresql.org/download/>
- Instale utilizando o instalador apropriado;

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

**Criando Servidor Local**

Banco de Dados Relacional

Criando Banco de Dados

Criando Tabelas

Criando Sequências

Inserindo Dados

Buscando Dados

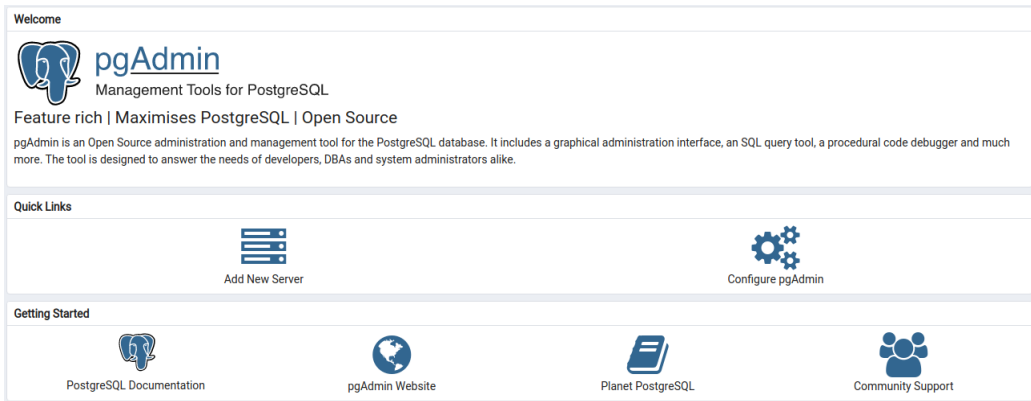
Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

# Criando um servidor local

- Abra o pgadmin4 (utilize o comando `pgadmin4` no linux);
- Selecione a opção **Add New Server**:





## Criando um servidor local

- Feito isso, escolha um nome para o servidor;
- No meu caso **takeofriedrich**:

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

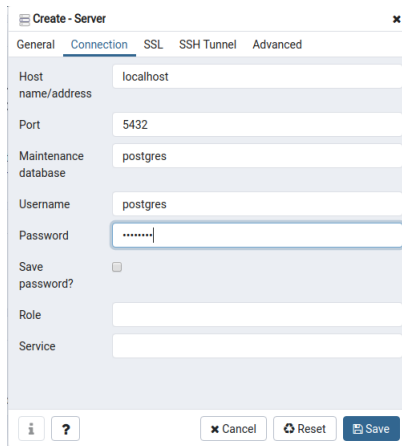
- Name:** takeofriedrich
- Server group:** Servers
- Background:** ☒
- Foreground:** ☒
- Connect now?:** ☒
- Comments:** (empty text area)

A red error message at the bottom states: "Either Host name, Address or Service must be specified."

Buttons at the bottom: Cancel, Reset, Save.

# Criando um servidor local

- Na aba Connection:
- No campo endereço (Hostname/address) digite **localhost**;
- Pois iremos utilizar o servidor Postgres instalado localmente;
- Deixe a porta padrão 5432;
- No campo password, digite **postgres**;
- Clique em save e está feita a configuração do servidor local;



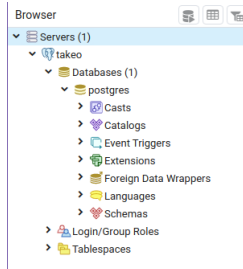
The screenshot shows a 'Create - Server' dialog box with the 'Connection' tab selected. The fields are filled as follows:

Field	Value
Host name/address	localhost
Port	5432
Maintenance database	postgres
Username	postgres
Password	.....
Save password?	<input type="checkbox"/>
Role	
Service	

At the bottom, there are buttons for 'Cancel', 'Reset', and 'Save'.

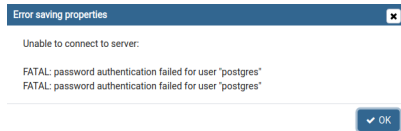
# Criando um servidor local

- Caso tudo tenha ocorrido certo, será possível ver o servidor criado:



## Criando um servidor local

- Caso tenha aparecido essa mensagem de erro, precisaremos alterar a senha pelo console;
- Vamos utilizar esse comando no terminal:  
*sudo -u user\_name psql;*
- Como o usuário padrão é postgres, digite esse comando: *sudo -u postgres psql;*
- Após isso será necessário utilizar um comando sql para alterar a senha: *ALTER USER postgres WITH PASSWORD 'nova senha';*
- É importante que a senha seja colocada dentro da String mesmo que ela seja somente numérica!



# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

**Banco de Dados Relacional**

Criando Banco de Dados

Criando Tabelas

Criando Sequências

Inserindo Dados

Buscando Dados

Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

- Conceitos básicos de banco de dados relacional;

Nome da relação

Atributos

Tuplas

ALUNO	Nome	SSN	FoneResidencia	Endereco	FoneEscritorio	Idade	MPG
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	<i>null</i>	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	<i>null</i>	18	2.89
	Dick Davidson	422-11-2320	<i>null</i>	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	<i>null</i>	19	3.25

- Primeiro precisamos falar de alguns conceitos básicos;
- Toda tabela é composta de **tuplas**;
- Tuplas são sequências ordenadas de elementos;
- Exemplos:
  - (Joinville, Santa Catarina, Brasil)
  - (Florianopolis, Santa Catarina, Brasil)
  - (Curitiba, Paraná, Brasil)
  - (Toronto, Ontário, Canadá)
- Todas as tuplas apresentadas são formadas por (Cidade, Estado, País);
- É importante lembrar que a ordem das informações na tupla **não muda!**

# Banco de Dados Relacional

- Mas como vamos diferenciar uma tupla da outra?
- Falando do ponto de vista computacional?
- Vamos usar o conceito de **chave primária** (primary key);
- Basicamente é um atributo da tupla que a diferencia das outras:
  - (1, Joinville, Santa Catarina, Brasil)
  - (2, Florianopolis, Santa Catarina, Brasil)
  - (3, Curitiba, Paraná, Brasil)
  - (4, Toronto, Ontário, Canadá)
- Agora caso quisermos buscar a cidade 3, teremos:  
(3, Curitiba, Paraná, Brasil);
- Dessa forma, agora podemos ter uma cidade chamada Joinville em outro país por exemplo:  
(5, Joinville, Haute-Marne, França)



# Banco de Dados Relacional

- Outro conceito importante a se falar é o de **chave estrangeira**;
- Imagine que agora vamos representar os aeroportos no banco de dados;
- Observe as tabelas a seguir:

id	cidade	estado	pais
1	Joinville	Santa Catarina	Brasil
2	Florianópolis	Santa Catarina	Brasil
3	Curitiba	Paraná	Brasil
4	Toronto	Ontário	Canadá

**Tabela 1:** Cidades

cidade	nome
Joinville	Lauro Carneiro
Florianopolis	Hercílio Luz
Curitiba	Afonso Pena
Curitiba	Bacacheri

**Tabela 2:** Aeroportos

# Banco de Dados Relacional

- Note que a tabela dos aeroportos tem um campo que representa a cidade;
- O jeito correto de fazer com que uma tupla dos aeroportos "aponte" para uma cidade é usar uma **chave estrangeira** (foreign key);
- Isto é, a chave estrangeira vai referenciar a tabela cidades;
- No caso ela vai referenciar a chave primária da tabela cidades (o campo id):

id	cidade	estado	pais
1	Joinville	Santa Catarina	Brasil
2	Florianópolis	Santa Catarina	Brasil
3	Curitiba	Paraná	Brasil
4	Toronto	Ontário	Canadá

**Tabela 3:** Cidades

cidade	nome
1	Lauro Carneiro
2	Hercílio Luz
3	Afonso Pena
3	Bacacheri

**Tabela 4:** Aeroportos

- Mas ainda não conseguimos diferenciar a tupla (3, Afonso Pena) da tupla (3, Bacacheri);
- Precisamos definir uma chave primária também, logo:

id	nome	cidade
1	Lauro Carneiro	1
2	Hercílio Luz	2
3	Afonso Pena	3
4	Bacacheri	3

**Tabela 5:** Aeroportos

- É comum representar as tabelas utilizando a seguinte notação;
  - 'nome da tabela'(*#*'chave primaria', ['campo 1', ..., 'campo N'], *&*'chave estrangeira');
  - *'chave estrangeira'* referencia *'nome da tabela referenciada'*
- As chaves primárias são representadas por *#*, enquanto as estrangeiras por *&*;
- Veja a representação das tabelas cidade e aeroporto:
  - cidade(*#*id, nome, estado, pais)
  - aeroporto(*#*id, nome, *&*id\_cidade)
    - id\_cidade referencia cidade

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

**Criando Banco de Dados**

Criando Tabelas

Criando Sequências

Inserindo Dados

Buscando Dados

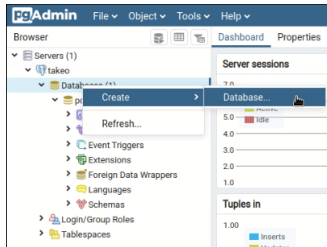
Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

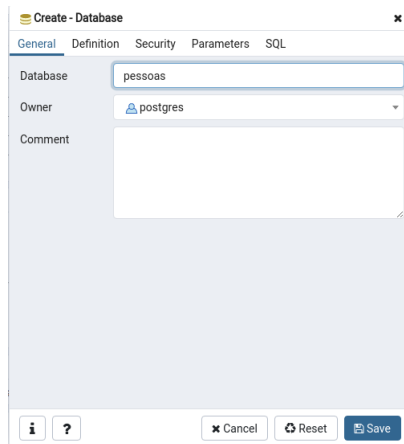
# Criando um Database

- Vamos criar um nova database (banco de dados) em nosso servidor local;
- Para isso, clique com botão direito do mouse sob a opção **Databases** do seu servidor;
- E selecione a opção **Create**;
- Depois selecione a opção **Database**:



## Criando um Database

- Feito isso, vamos criar uma tabela chamada **pessoas** para utilizarmos em nossos exemplos;
- Digite **pessoas** no campo **database**:
- Clique em salvar;



The screenshot shows a 'Create - Database' window with the following details:

- Title Bar:** Create - Database
- Tabs:** General (selected), Definition, Security, Parameters, SQL
- Database:** A text input field containing 'pessoas'.
- Owner:** A dropdown menu with 'postgres' selected.
- Comment:** A large empty text area.
- Bottom Bar:** Includes an information icon, a help icon, and three buttons: 'Cancel', 'Reset', and 'Save'.

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

Criando Banco de Dados

**Criando Tabelas**

Criando Sequências

Inserindo Dados

Buscando Dados

Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

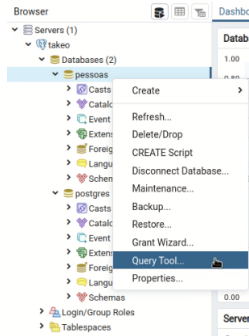


## Criando Tabelas

- Iremos apenas utilizar linhas de comando para criar as tabelas;
- Entretanto, saiba que é possível fazer isso também via interface gráfica do pgadmin!

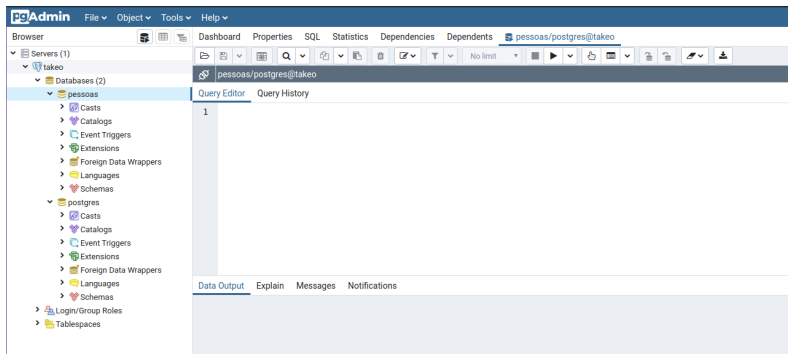
# Criando Tabelas

- Para criar uma tabela, primeiro clique com o botão esquerdo do mouse no banco de dados que deseja criá-la;
- Feito isso, clique com o botão direito do mouse sobre ela e selecione a opção **Query Tool**;
- Utilizaremos a ferramenta de query (query tool) para entrar com queries no banco de dados (executar comandos);



# Criando Tabelas

- Iremos digitar e executar os comandos sql no **Query Editor**;
- Os resultados serão exibidos no **Data Output**



## Criando Tabelas

- O comando para criar tabelas é:

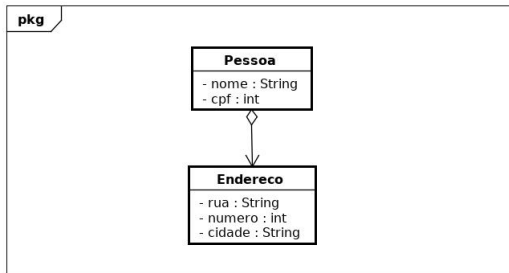
```
CREATE TABLE nomeTabela(  
    nomeCampo1 tipoDado,  
    nomeCampo2 tipoDado,  
    ...  
    nomeCampoN tipoDado,  
    PRIMARY KEY(nomeCampo),  
    FOREIGN KEY(nomeCampo) REFERENCES nomeTabelaReferenciada  
);
```

- Você pode criar uma chave primária composta por vários campos, se necessário;
- Você também pode ter mais de uma chave estrangeira, se necessário;

- Acesse o link da documentação:  
<https://www.postgresql.org/docs/12/index.html> para ver os tipos de dados disponíveis para a especificação dos campos das tabelas;
- Em resumo, utilizaremos principalmente os tipos:
  - **varchar(tam)**: string de tam caracteres;
  - **integer**: inteiro;
  - **boolean**: variável booleana;

## Criando Tabelas

- Vamos aplicar os conceitos em cima do UML a seguir:



- Veja exemplos de instâncias dessas classes, no formato de tabelas:

## Criando Tabelas

id	nome	cpf	telefone
1	Lucas	1111	12345
2	Julia	2222	54321
3	Leticia	3333	15243

**Tabela 6:** Pessoa

id	rua	numero	cidade	pessoa
1	Ministro Calógeras	11	Joinville	1
2	João Colin	22	Joinville	2
3	Marquês de Olinda	33	Joinville	3

**Tabela 7:** Endereco

## Criando Tabelas

- Vamos criar a tabela de pessoas primeiro;
- Na orientação a objetos, a classe Pessoa possui uma composição com a classe Endereço, isto é, os objetos apenas existem juntos;
- Como esse relacionamento é direcional partindo da Pessoa, apenas ela possui uma referência ao Endereço;
- No mundo relacional, faremos o contrário, cada Endereço irá possuir uma Pessoa;
- Nas disciplinas de Banco de Dados será visto mais a fundo o uso de chaves estrangeiras e como elas afetam a cardinalidade;
- No momento vamos colocar a chave estrangeira na tabela de endereço;



## Criando Tabelas

- Teremos apenas os atributos da Pessoa nessa tabela:
- Utilizaremos um integer para compor a chave primária *id* dessa tabela;
- O comando para criar a tabela será:

```
create table pessoa(  
    id int,  
    nome varchar(50),  
    cpf int,  
    telefone int,  
    primary key (id)  
);
```

## Criando Tabelas

- Agora vamos fazer a tabela de endereços;
- A chave estrangeira irá ficar nela, fazendo referencia a tabela de pessoa;

```
create table endereco(  
    id int,  
    rua varchar(50),  
    numero integer,  
    cidade varchar(50),  
    id_pessoa integer,  
    primary key (id),  
    foreign key (id_pessoa) references pessoa  
);
```

- Entre com os comandos sql no **Query Tool**, separados por ponto e virgula (;) e execute-os com F5;

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

Criando Banco de Dados

Criando Tabelas

**Criando Sequências**

Inserindo Dados

Buscando Dados

Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

## Criando Sequências

- Como banco de dados geralmente são sistemas de armazenamento compartilhados, gerar sua própria chave primária localmente pode trazer conflitos ao inserir novas tuplas;
- Como garantir que a chave primária que você gerou é única?
- Para isso vamos utilizar sequências;
- Toda vez que formos adicionar uma nova tupla, vamos incrementar a sequencia e usa-la como chave primária;
- Utilize `create sequencia nomeSequencia` para criar novas sequencias no postgres;

```
create sequence id_pessoa;  
create sequence id_endereco;
```

- Para pegar o próximo valor da sequencia utilize: `select nextval('nomeSequencia')`

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

Criando Banco de Dados

Criando Tabelas

Criando Sequências

**Inserindo Dados**

Buscando Dados

Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

## Inserindo Dados

- Agora que criamos as tabelas podemos inserir os dados nelas;
- Para isso utilizaremos o comando:

```
INSERT INTO nomeTabela ( campo1,...,campoN ) values ( valor1,...,valorN );
```

- Caso você insira todos os valores da tupla, você pode omitir a declaração dos campos:

```
INSERT INTO nomeTabela values ( valor1,...,valorN );
```

## Inserindo Dados

- Vamos inserir o Lucas em nosso banco de dados;
- Utilizaremos 1 para o id, posteriormente faremos utilizando sequencias:
  - nome: Lucas;
  - cpf: 1111;
  - telefone: 12345,

```
insert into pessoa values (1, 'Lucas', 1111, 12345);
```

## Inserindo Dados

- Agora vamos inserir um endereço em nosso banco de dados, esse endereço será correspondente ao do Lucas, por isso no campo id\_pessoa, vamos colocar 1;
- Definiremos o id desse endereço como 1 também.
  - rua: João Colin;
  - numero: 11;
  - cidade: Joinville;
  - id\_pessoa: 1;

```
insert into endereco values (1,'Ministro Calogeras',11,'Joinville',1);
```



# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

Criando Banco de Dados

Criando Tabelas

Criando Sequências

Inserindo Dados

**Buscando Dados**

Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

- Agora para buscar os dados do banco de dados, vamos utilizar o comando select;
- Ele é composto de três partes:
  - SELECT
  - FROM
  - WHERE
- Na parte SELECT você indica os campos que serão retornados;
- Cada busca no banco de retorna uma tabela, usando o SELECT você pode "filtrar" os campos desejados e/ou funções (não vem ao caso no momento);
- Caso queira todos os campos, utilize o \* (asterisco);

## Buscando Dados

- Na cláusula FROM você indica a tabela que será utilizada no "filtro";
- Na parte WHERE você determina uma condição de filtragem;
- Exemplos:
  - `SELECT * FROM pessoa;`
    - Vai retornar todas as tuplas da tabela pessoa com todos os seus campos;
  - `SELECT * FROM pessoa WHERE cpf = 1111;`
    - Vai retorna a(s) tupla(s) cuja(s) pessoa(s) possui(em) cpf 1111;
  - `SELECT id,nome FROM pessoa;`
    - Retorna apenas o id e nome de todas as pessoas da tabela;
  - `SELECT count(*) FROM pessoa;`
    - Retorna quantas tuplas existem na tabela pessoa;
  - `SELECT max(numero) FROM endereco WHERE rua = 'Ministro Calógeras';`
    - Retorna o maior numero da que existe na rua Ministro Calógeras;

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

Criando Banco de Dados

Criando Tabelas

Criando Sequências

Inserindo Dados

Buscando Dados

**Atualizando Dados**

Removendo Dados

Aterando Tabelas e Colunas

## Atualizando Dados

- O comando de atualização precisa especificar quais campos serão atualizados, quais serão os novos valores e quais registros sofrerão essa atualização;
- Também possui três partes:
  - UPDATE
  - SET
  - WHERE
- Na parte UPDATE você determina a tabela ao qual será aplicada a mudança;
- Na parte SET você indica o campo que será alterado e o valor;
- Já a parte WHERE restringe o alcance das alterações apenas as tuplas que satisfaçam a condição;
- Exemplos:
  - UPDATE pessoa SET telefone = '99999' WHERE nome = 'Julia';
    - Atualiza o telefone da Julia;
  - UPDATE endereco SET numero += 10 WHERE id = 1;
    - Incrementa 10 no número da casa do endereço de id 1;

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

Criando Banco de Dados

Criando Tabelas

Criando Sequências

Inserindo Dados

Buscando Dados

Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas

## Removendo Dados

- Já o comando de remover dados precisa ser especificada a condição de remoção;
- Ela possui duas partes:
  - DELETE FROM
  - WHERE
- Na parte DELETE FROM você determina a tabela ao qual será aplicada a mudança;
- Já a parte WHERE tem a mesma função do WHERE no comando de atualização;
- Exemplos:
  - DELETE pessoa WHERE nome = 'Julia';
    - Deleta a Julia do banco de dados;
  - DELETE endereco WHERE id = 1;
    - Deleta o endereço de id 1;

# Seções

Introdução

Instalando Postgres

Linux

Windows e MacOS

Criando Servidor Local

Banco de Dados Relacional

Criando Banco de Dados

Criando Tabelas

Criando Sequências

Inserindo Dados

Buscando Dados

Atualizando Dados

Removendo Dados

Aterando Tabelas e Colunas



## Removendo Tabelas

- Existe também o comando de remover tabelas;
- `DROP TABLE nomeTabela;`

## Alterando Tabelas



- É possível também alterar as tabelas já existentes;
- Utilizando o comando `ALTER TABLE`;
- É possível realizar várias alterações na tabela;

## Alterando Tabelas - Inserindo Colunas

- É possível através do comando `ADD COLUMN nomeCampo tipoDado`;
- Veja exemplos:
  - `ALTER TABLE pessoa ADD COLUMN rg integer`;
    - Adiciona uma coluna chamada rg, do tipo inteiro na tabela pessoa;

## Alterando Tabelas - Removendo Colunas

- Para remover uma coluna, utiliza-se o comando **DROP COLUMN** **nomeCampo** **tipoDado** **CASCADE**;
  - ALTER TABLE pessoa DROP COLUMN rg integer;
    - Remove uma coluna chamada rg, do tipo inteiro na tabela pessoa e seta null nos campos de outras tabelas que referenciavam essa coluna;

-  KUWAKI, V. T. F. Modelo de slides udesc lattex. In: . [S.l.]: Disponível em: <<https://github.com/takeofriedrich/slidesUdescLattex>>. Acesso em: 24 jan. 2020.
-  POSTGRES. Postgresql docs. In: . [S.l.]: Disponível em: <<https://www.postgresql.org/docs/>>. Acesso em: 8 mai. 2020.

Duvidas:  
Vinicius Takeo Friedrich Kuwaki  
vtkwki@gmail.com  
[github.com/takeofriedrich](https://github.com/takeofriedrich)



**UDESC**  
UNIVERSIDADE  
DO ESTADO DE  
SANTA CATARINA