Objetivos:

I. Modelo físico;
II. Restrição de domínio;
III. Restrição de valor nulo;
IV. Restrição de valor único;
V. Conversão de lógico para físico;
VI. Estrutura do SGBD PostgreSQL;
VII. Instalar o PostgreSQL e pgAdmin;
VIII. Acesso ao SGBD usando pgAdmin;
IX. Acesso ao SGBD usando psql

I. Modelo físico

O modelo físico também é chamado de modelo de implementação. No modelo físico fazemos a implementação física do modelo de banco de dados levando-se em consideração as limitações impostas pelo modelo de SGBD. Alguns exemplos de SGBDs relacionais são PostgreSQL, MySQL, SQL Server e Oracle.

Ainda a nível de modelo relacional devemos especificar as seguintes restrições do atributo - em inglês, constraints:

- Restrição de chave primária;
- Restrição de domínio;
- Restrição de valor nulo;
- Restrição de valor único.

II. Restrição de domínio

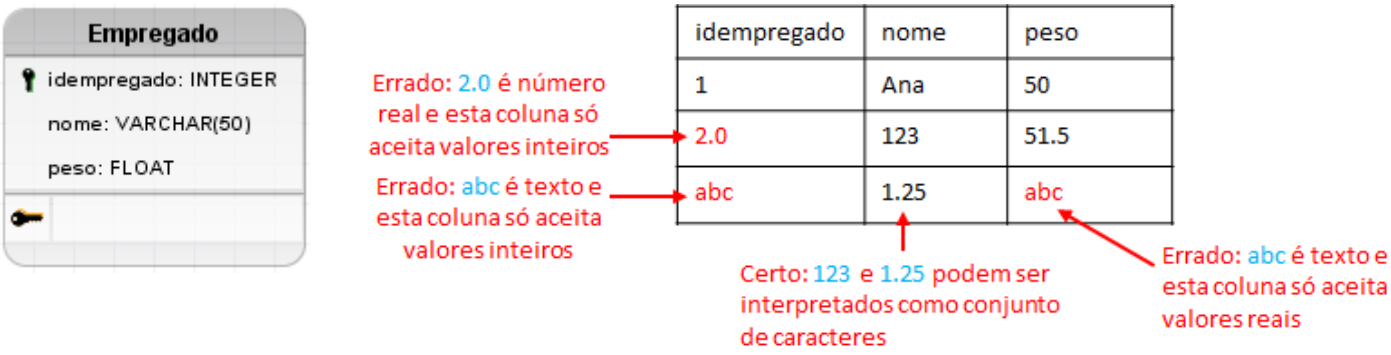
Determina que o valor de cada atributo deve ser um valor do domínio especificado. No modelo físico do SGBD este domínio recebe o nome de tipo de dado ou tipo de campo. A seguir tem-se os tipos de dados comuns a maioria dos SGBDs:

- Integer: aceita valores do conjunto dos números inteiros;
- Real, float e double: aceita valores do conjunto dos números reais;
- Date: aceita valores no formato de datas. Geralmente, especificadas no formato YYYY-MM-DD;
- Datetime ou timestamp: aceita valores no formato de data e horário. Geralmente, especificadas no formato YYYY-MM-DD HH:MM:SS;
- Boolean: aceita somente os valores booleanos True ou False;
- CHAR: aceita qualquer tipo de caractere, ou seja, é o tipo de dado para textos. Geralmente, vem associado a um comprimento, por exemplo, CHAR(5) significa que a célula ocupará sempre o espaço de 5 caracteres. Como exemplo:

- O texto “Maria” ocupará 5 caracteres;
- O texto “Ana” ocupará 5 caracteres. Como foram alocados 5 caracteres e “Ana” possui 3 caracteres, então o espaço de 2 caracteres não será usado;
- O texto “Renato” ocupará 5 caracteres. Como “Renato” possui 6 caracteres, então será armazenado somente os 5 primeiros caracteres, ou seja, “Renat”.

- VARCHAR: também chamado de “caractere variável”. É semelhante ao tipo CHAR, porém ao definir VARCHAR(5) significa que ele será redimensionado de acordo com o conteúdo até o limite de 5 caracteres. Como exemplo:
  - O texto “Maria” ocupará 5 caracteres;
  - O texto “Ana” ocupará 3 caracteres. Os demais caracteres não serão alocados, pois precisou-se apenas de 3 caracteres;
  - O texto “Renato” ocupará 5 caracteres. Como “Renato” possui 6 caracteres, então será armazenado até o limite especificado de 5 caracteres, ou seja, “Renat”.

No exemplo a seguir a tabela Empregado possui a definição dos tipos de dados das colunas.



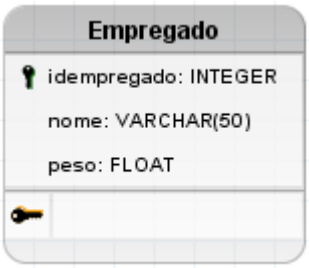
III. Restrição de valor nulo

O valor nulo significa ausência de conteúdo. É comum as pessoas pensarem que:

- NULL é zero, porém zero é um valor; ou
- NULL é um texto com zero caracteres, porém um texto com zero caracteres é um texto.

No exemplo a seguir considere que:

- A coluna nome não aceita valores nulos;
- A coluna peso aceita valores nulos.



Certo: vazio significa zero caracteres numa coluna que aceita texto

idempregado	nome	peso
1	Ana	50
2		51.5
3	NULL	NULL

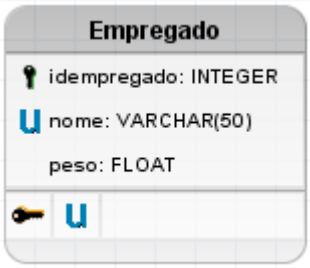
Erro: NULL a coluna nome não aceita nulos

Certo: a coluna peso aceita valores nulos

Na prática NULL é um conteúdo, assim com, 10 e Maria são conteúdos. A diferença é que o valor NULL significa que o conteúdo da célula deverá ser desconsiderado.

IV. Restrição de valor único

A restrição de valor único ou valores exclusivos não aceita valores repetidos. No exemplo a seguir o atributo nome foi definido para receber valores únicos.



idempregado	nome	peso
1	Ana	50
2	Pedro	51.5
3	Pedro	50

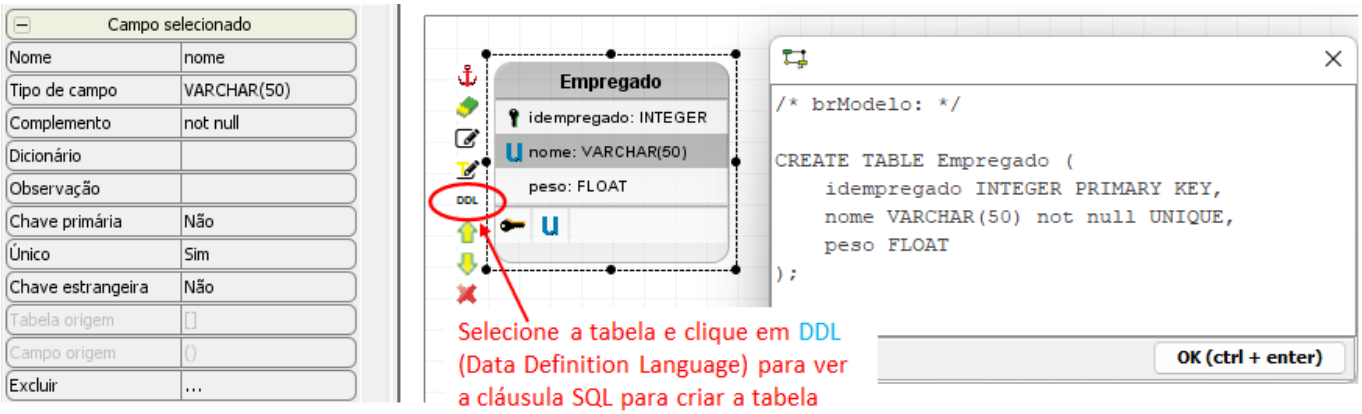
Erro: a coluna nome não aceita valores repetidos

Certo: a coluna peso aceita valores repetidos

V. Conversão de lógico para físico

Aqui utilizaremos o SGBD PostgreSQL, então teremos de converter os diagramas do modelo lógico para cláusulas SQL (Structured Query Language) do PostgreSQL.

O exemplo a seguir mostra o código SQL para criar a tabela Empregado no software brModelo.



Comando SQL para criar tabela:

Como toda a comunicação com o SGBD se dá através de cláusulas SQL, então teremos de aprender a usar esses comandos para fazer uso do SGBD.

Na linguagem SQL as instruções para o SGBD são dadas por cláusulas estruturadas. A seguir tem-se a estrutura de uma cláusula para criar uma tabela:

```
create table if not exists nomeDaTabela (
    nomeDaColuna tipoDeDado null/not null primary key,
    nomeDaColuna tipoDeDado null/not null unique,
    nomeDaColuna tipoDeDado null/not null
);
```

Explicando a cláusula create table:

- O termo if not exists é opcional, dará erro se tentarmos criar uma tabela que já existe, mas ao colocar o termo if not exists dará apenas uma mensagem dizendo que a tabela já existe, ou seja, é boa prática usar o termo if not exists. No exemplo a seguir é possível ver a diferença do uso do termo if not exists:

```
CREATE TABLE empregado (
    idempregado INTEGER PRIMARY KEY,
    nome VARCHAR(50) NOT NULL UNIQUE,
    peso FLOAT
);

ERROR:  relation "empregado" already exists
SQL state: 42P07
```

Mensagem de erro a cláusula não foi concluída com sucesso no SGBD

```
CREATE TABLE if not exists empregado (
    idempregado INTEGER PRIMARY KEY,
    nome VARCHAR(50) NOT NULL UNIQUE,
    peso FLOAT
);

NOTICE:  relation "empregado" already exists,
skipping CREATE TABLE

Query returned successfully in 17 msec.
```

Mensagem de aviso. Apesar de a cláusula não fazer aquilo que gostaríamos, ela não deu erro e foi concluída com sucesso no SGBD

- nomeDaTabela precisa ser uma única palavra, sem espaços, cedilhas, acentuação e pontuação, e não pode começar por um dígito numérico. Exemplos de nomes incorretos de tabela:
  - 1empregado: começa por um dígito numérico;
  - cadastro empregado: possui um caractere de espaço;
  - cadastro.empregado: possui um caractere de pontuação;
  - matrícula: possui um acento;
  - presença: possui cedilha.
- É obrigatório o par de parênteses após o nome da tabela. Dentro dos parênteses são especificadas as colunas da tabela, não é possível criar uma tabela sem colunas. As colunas são chamadas de atributos da tabela. Cada coluna é especificada da seguinte forma:

- nomeDaColuna: precisa ser uma única palavra, sem espaços, cedilhas, acentuação e pontuação, e não pode começar por um dígito numérico, ou seja, as regras para nomear coluna é igual as regras para nomear as tabelas;
- Uma coluna possui as seguintes restrições (constraints, em inglês):
  - ✓ tipoDeDado: define os valores que a coluna aceita. Os tipos mais comuns são: integer, float, varchar, boolean e date;
  - ✓ null ou not null: define se a coluna aceita valores nulos. O valor nulo é um símbolo que indica que a célula não possui conteúdo, não confundir com a célula vazia. A restrição null ou not null é opcional, pois o padrão (default) é aceitar nulo;
  - ✓ **primary key**: define que a coluna é usada para identificar um registro (linha) da tabela. Não podem existir valores repetidos nas colunas que compõem a chave primária. Apesar de não se obrigatório, é boa prática que a tabela tenha chave primária. A tabela pode ter várias colunas compondo a chave primária, porém, é boa prática que a tabela tenha somente uma coluna compondo a chave primária;
  - ✓ **unique**: define que a coluna não aceita valores repetidos. É uma restrição não obrigatória. A tabela pode ter no máximo uma coluna que recebe valores únicos.

- Dentro dos parênteses usamos vírgulas , para separar a definição das colunas.
- As cláusulas são terminadas pelo ponto e vírgula.

Comando SQL para excluir tabela:

O comando a seguir exclui a tabela empregado:

```
drop table empregado;
```

Da mesma forma o comando pode ser redigido usando o temo if exits. Desta forma, não será exibida uma mensagem de erro se a tabela Empregado não existir no SGBD:

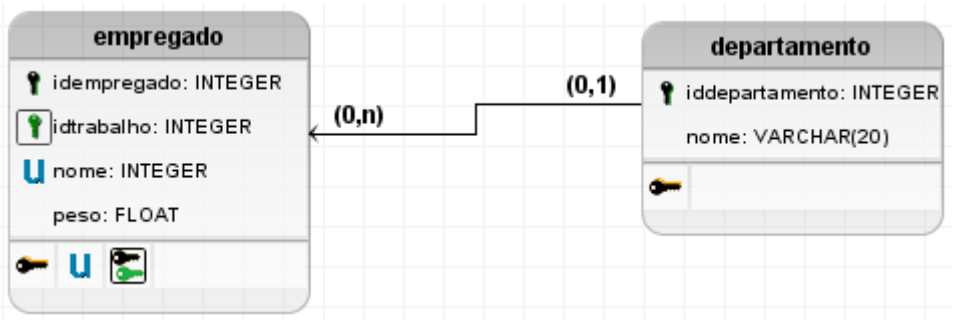
```
drop table if exists empregado;
```

Comando SQL para criar relacionamento entre tabelas:

Os relacionamentos entre as tabelas são criados na linguagem SQL através da referência de chave estrangeira.

O código a seguir possui as cláusulas SQL para criar as tabelas empregado e departamento. Observe a ordem que as tabelas são excluídas e criadas:

- As tabelas que possuem chave estrangeira precisam ser excluídas antes;
- As tabelas que possuem chave estrangeira precisam ser criadas por último.



É boa prática nomear as tabelas e colunas com letras minúsculas.

```
-- É boa prática primeiro excluir as tabelas
drop table if exists empregado;
drop table if exists departamento;

-- Primeiro tem de criar a tabela referenciada na chave estrangeira
CREATE TABLE if not exists departamento (
      iddepartamento INTEGER PRIMARY KEY,
      nome VARCHAR(20) NOT NULL
);

-- Depois tem de criar a tabela que possui a chave estrangeira
CREATE TABLE if not exists empregado (
      idempregado INTEGER PRIMARY KEY,
      idtrabalho INTEGER NOT NULL,
      nome VARCHAR(50) NOT NULL UNIQUE,
      peso FLOAT,
      CONSTRAINT pk_trabalho
      FOREIGN KEY (idtrabalho)
      REFERENCES departamento (iddepartamento)
);
```

No SQL do PostgreSQL os comentários são colocados à direita dos dois traços

O relacionamento entre as tabelas se dá pela chave estrangeira idtrabalho que referencia a coluna iddepartamento da tabela departamento. Na linguagem SQL a ligação é uma restrição (constraint, em inglês) da tabela. O comando SQL a seguir cria a restrição de chave estrangeira com o nome de `pk_trabalho`:

```
CONSTRAINT pk_trabalho
```

```
FOREIGN KEY (idtrabalho)
REFERENCES departamento (iddepartamento)
```

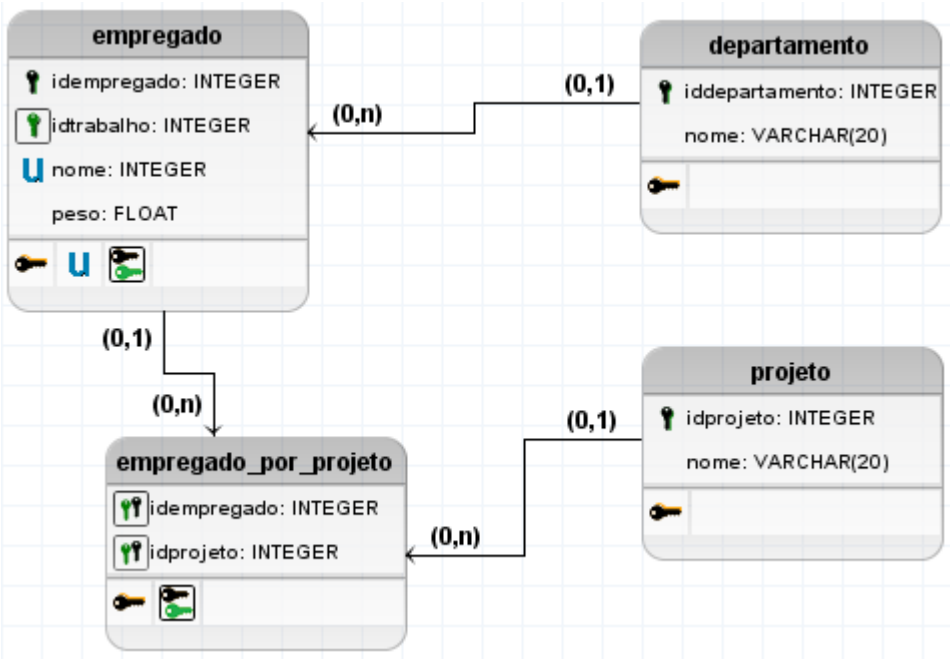
O SGBD se encarrega de dar um nome a restrição caso não seja especificado, como no exemplo a seguir:

```
FOREIGN KEY (idtrabalho)
REFERENCES departamento (iddepartamento)
```

A chave estrangeira pode ser definida também na definição da coluna idtrabalho:

```
CREATE TABLE if not exists empregado (
 idempregado INTEGER PRIMARY KEY,
 idtrabalho INTEGER NOT NULL REFERENCES departamento (iddepartamento),
 nome VARCHAR(50) NOT NULL UNIQUE,
 peso FLOAT
);
```

No exemplo a seguir a tabela empregado_por_projeto possui a chave primária composta. Além disso, a chave primária também é chave estrangeira.




```
-- Observe a ordem que as tabelas são excluídas
drop table if exists empregado;
drop table if exists departamento;
drop table if exists empregado_por_projeto;
drop table if exists empregado;
drop table if exists departamento;
```

```
drop table if exists projeto;

CREATE TABLE if not exists departamento (
      iddepartamento INTEGER PRIMARY KEY,
      nome VARCHAR(20) NOT NULL
);

CREATE TABLE if not exists projeto (
      idprojeto INTEGER PRIMARY KEY,
      nome VARCHAR(20) NOT NULL
);

CREATE TABLE if not exists empregado (
      idempregado INTEGER PRIMARY KEY,
      idtrabalho integer not null,
      nome VARCHAR(50) NOT NULL UNIQUE,
      peso FLOAT,
      FOREIGN KEY (idtrabalho)
      REFERENCES departamento (iddepartamento)
);

CREATE TABLE if not exists empregado_por_projeto (
      idempregado INTEGER,
      idprojeto INTEGER,
      PRIMARY KEY(idempregado,idprojeto),
      CONSTRAINT pk_empregado
      FOREIGN KEY (idempregado)
      REFERENCES empregado (idempregado),
      CONSTRAINT pk_projeto
      FOREIGN KEY (idprojeto)
      REFERENCES projeto (idprojeto)
);
```

A chave primária composta precisa ser criada como restrição da tabela: `PRIMARY KEY(idempregado,idprojeto)`.

Para mais detalhes de como criar tabelas no SGBD PostgreSQL consulte https://www.postgresql.org/docs/current/sql-createtable.html

VI. Estrutura do SGBD PostgreSQL

O SGBD não possui uma interface de acesso para o usuário, na verdade ele responde numa porta de rede do sistema operacional, ou seja, ele funciona como serviço.

> *Um serviço é um software que só pode ser acessado por um endereço de internet HTTP, ou seja, mesmo que o SGBD esteja instalado no próprio computador, será necessário usar o endereço http://localhost:5432 para acessar o SGBD.*

Curiosidades sobre domínios:

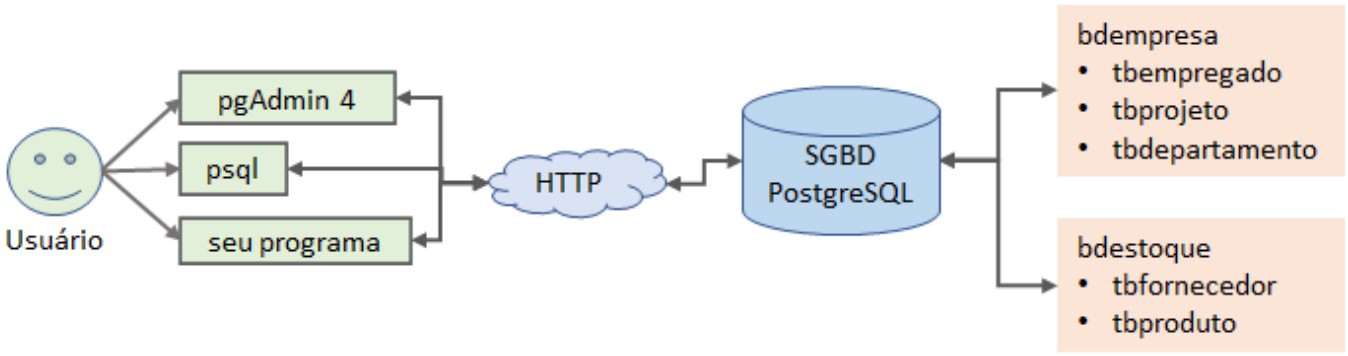
- google.com é um nome de domínio de rede que é mapeado para um endereço IP (Internet Protocol), ou seja, o que existe na prática é o endereço IP. O nome de domínio existe para facilitar a memorização do endereço;
- localhost é o nome de domínio para o próprio computador;
- localhost é o nome de domínio para o endereço IP 127.0.0.1;

Curiosidades sobre portas:

- O sistema operacional possui 65536 (2^{16}) portas de rede que são acessadas através do protocolo de internet HTTP, por exemplo, o SGBD PostgreSQL normalmente responde na http://localhost:5432. Em outras palavras, o software SGBD PostgreSQL será executado ao fazer uma requisição HTTP para localhost:5432;
- O seu computador pode ter até 65536 (2^{16}) softwares rodando em portas;
- Um software que responde numa porta é chamado de software que funciona “como serviço”;
- Um software que funciona como serviço só é executado quando existe a requisição, ou seja, nos outros momentos ele está “dormindo”.

Um software que funciona como serviço não tem uma interface gráfica, então precisaremos de algum software para fazer a comunicação com o SGBD PostgreSQL. A figura a seguir ilustra o caminho do usuário até o SGBD, podemos usar os softwares pgAdmin 4 (ambiente gráfico), psql (linha de comando) e as rotinas que codificaremos nos nossos programas para fazer a interface entre nós e o PostgreSQL.

Na representação bdempesa e bdestoque são BDs e tbempregado, tbprojeto, tbdepartamento, tbfornecedor e tbproduto são tabelas. Toda tabela precisa estar dentro de um BD. Os BDs são criados e acessados pelo SGBD, ou seja, não podemos manipular os BDs sem o SGBD.



VII. Instalar o PostgreSQL e pgAdmin

Para fazer a interface com o SGBD utilizaremos, preferencialmente, o software pgAdmin 4. Podemos baixar o PostgreSQL juntamente com o pgAdmin 4 e psql acessando https://www.enterprisedb.com/downloads/postgres-postgresql-downloads.

Antes de prosseguir é importante descobrir se o seu computador possui Windows de 32 ou 64 bits e fazer o download da versão correta do PostgreSQL. Acesse https://support.microsoft.com/pt-br/windows/qual-versão-do-sistema-operacional-microsoft-windows-estou-usando-628bec99-476a-2c13-5296-9dd081cdd808 para saber como descobrir a versão do seu Windows.

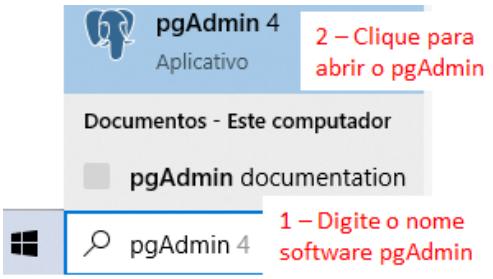
Durante o processo de instalação do SGBD PostgreSQL será criado um usuário administrador de nome postgres, recomenda-se que você coloque a senha 123 para evitar esquecer.

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
				64 bits	32 bits
15.1	postgresql.org	postgresql.org			Not supported
14.6	postgresql.org	postgresql.org			Not supported
13.9	postgresql.org	postgresql.org			Not supported
12.13	postgresql.org	postgresql.org			Not supported
11.18	postgresql.org	postgresql.org			Not supported
9.6.24*					

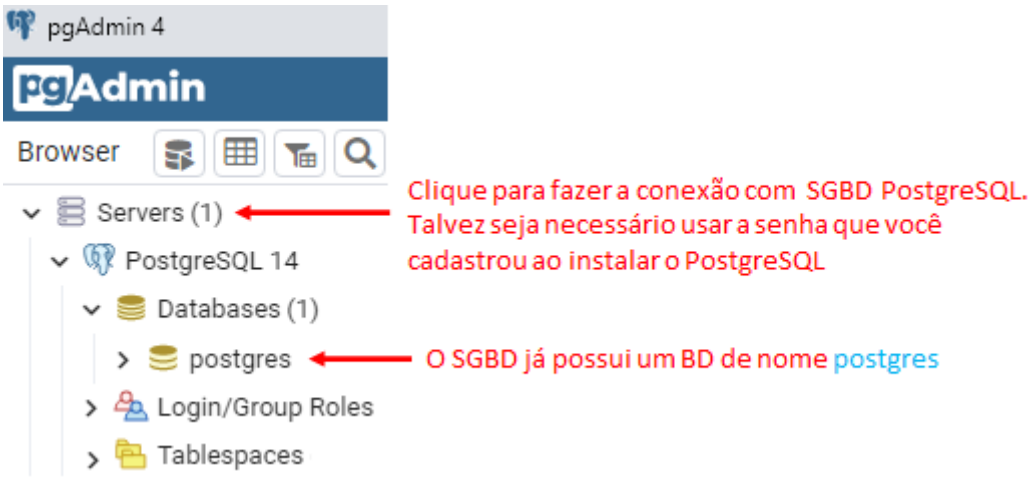
VIII. Acesso ao SGBD usando pgAdmin

Geralmente não é criado um ícone do pgAdmin, então é só usar o campo de pesquisa do Windows para localizar o pgAdmin 4.

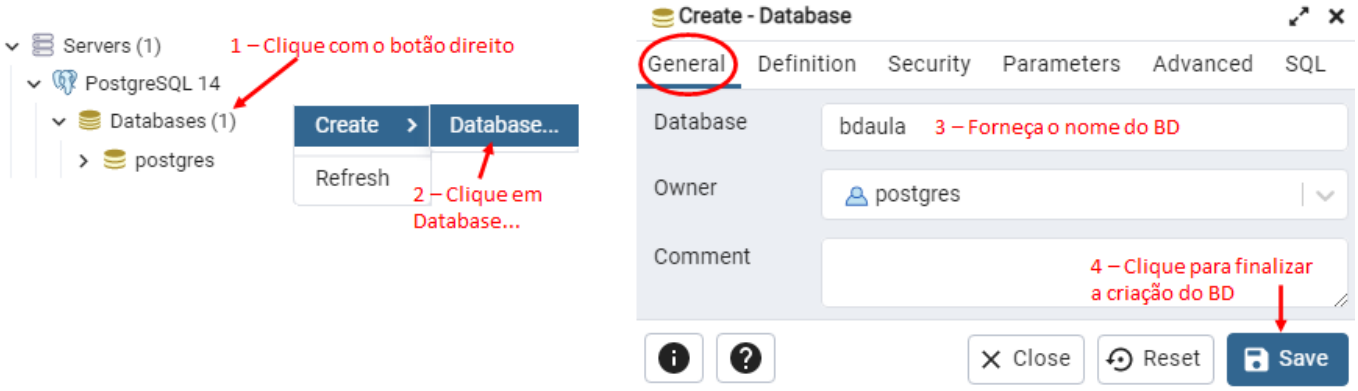
Demora-se cerca de 1 minuto para inicializar o pgAdmin 4.



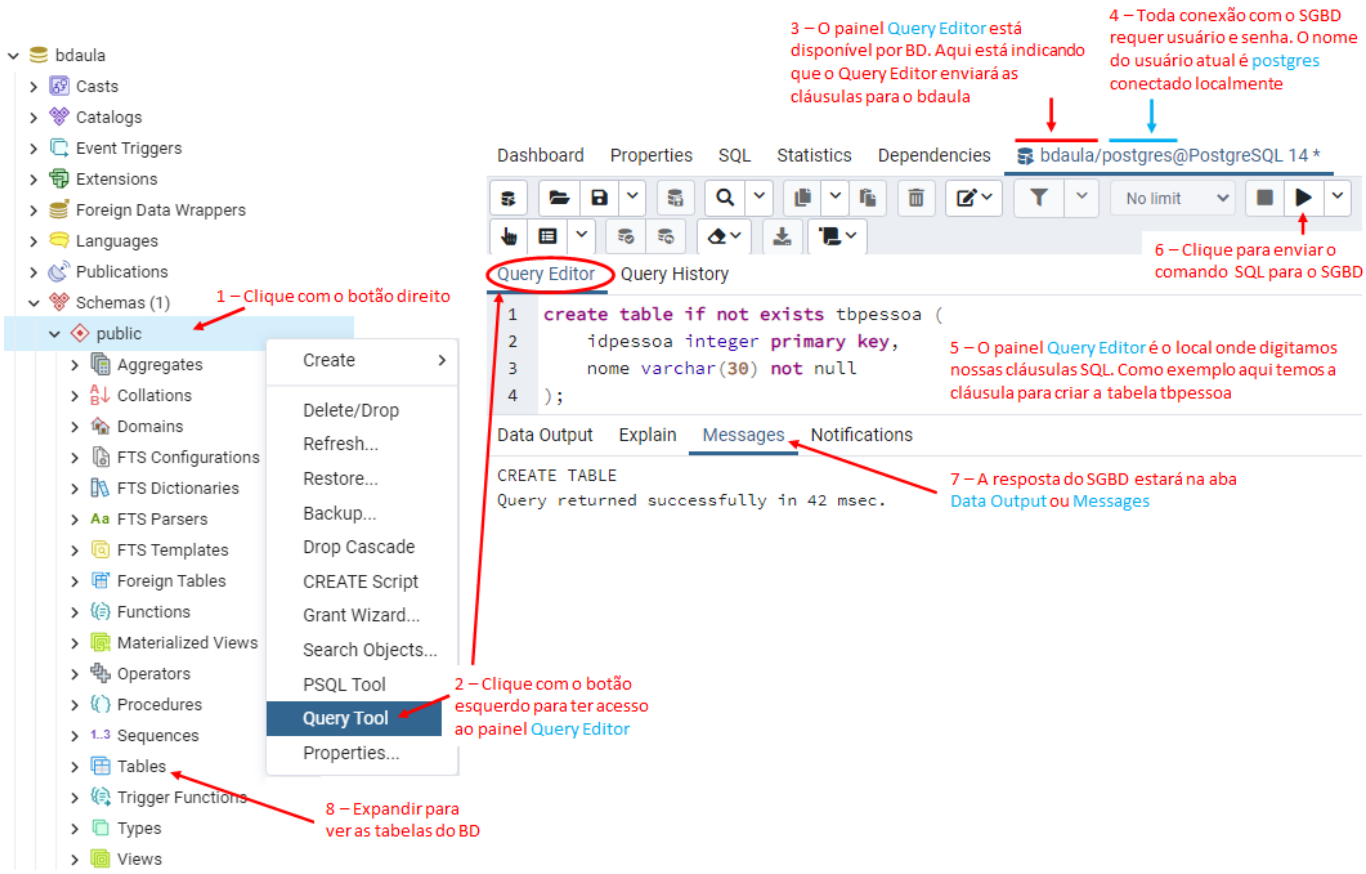
Após abrir o pgAdmin 4 você terá de fornecer a senha usada na instalação do PostgreSQL.



Apesar de já existir um BD de nome postgres. Siga os passos a seguir para criar o BD de nome bdaula.

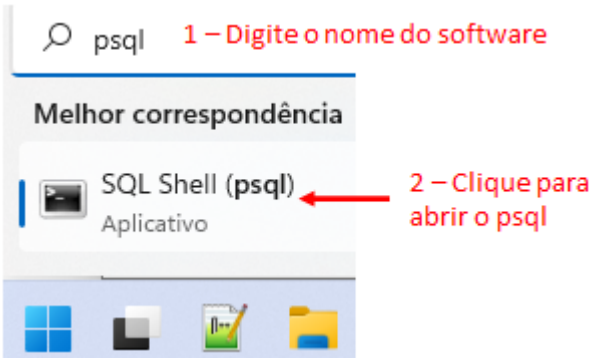


Toda a comunicação com o SGBD ocorre através de comandos SQL. A figura a seguir mostra os passos para acessarmos o ambiente para digitar e enviar os comandos SQL para o SGBD. Lembre-se que o pgAdmin é apenas um software que faz a interface com o SGBD, então as cláusulas SQL serão enviadas ao SBGD e a resposta será exibida pelo pgAdmin.

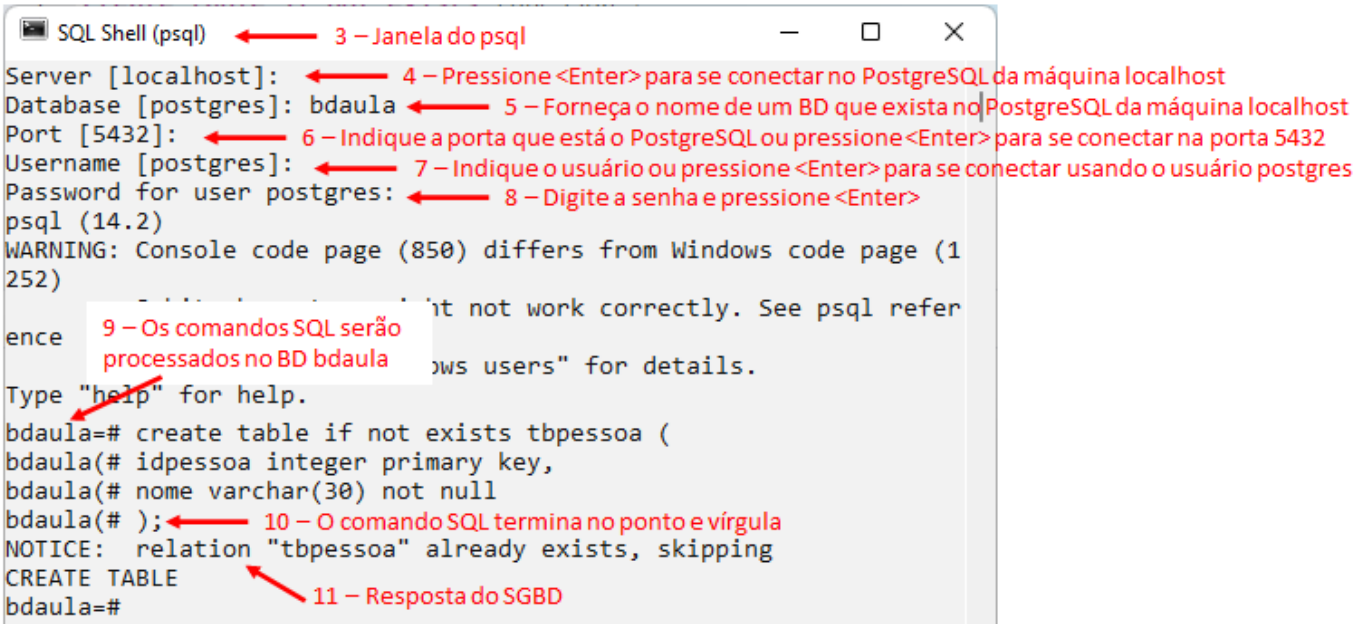


IX. Acesso ao SGBD usando psql

O ambiente de linha de comandos psql também não possui um ícone no desktop do computador, então precisamos usar o campo de pesquisa do Windows.

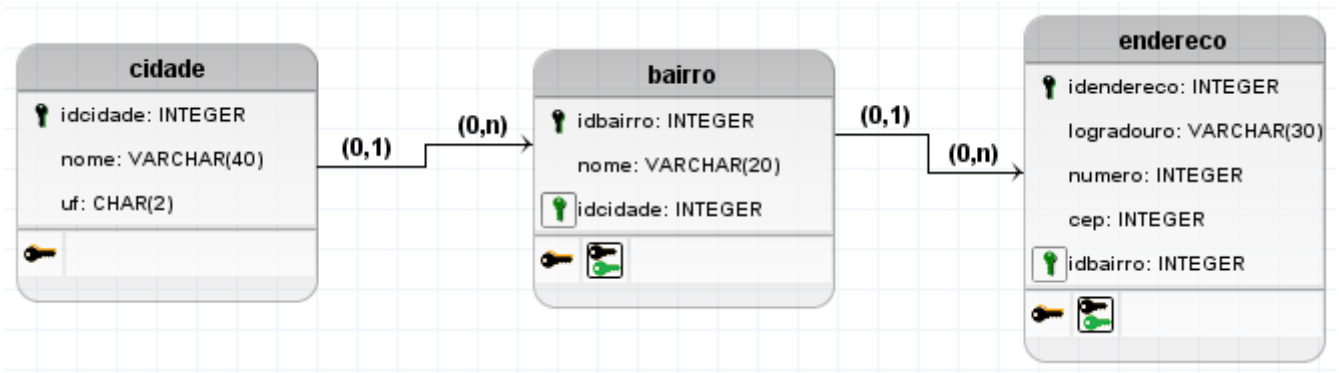


A seguir é mostrado como é feita a conexão no PostgreSQL pelo psql. Normalmente o SGBD PostgreSQL é instalado na porta 5432 do computador e durante a instalação foi criado um usuário de nome postgres com a senha que você colocou durante a instalação - lembre-se que foi sugerida a senha 123. Pelo ambiente do psql temos de fornecer todos os parâmetros de conexão ao BD. Pressione <Enter> para usar as sugestões indicados entre colchetes:



Exercícios

Exercício 1: Fazer as cláusulas SQL para excluir as tabelas a seguir.

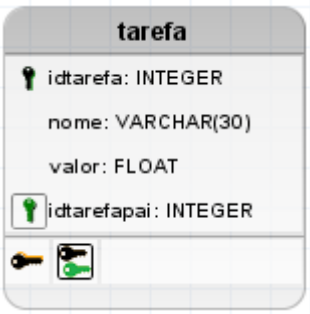


Exercício 2: Fazer as cláusulas SQL para criar as tabelas do Exercício 1. Considere os seguintes requisitos:

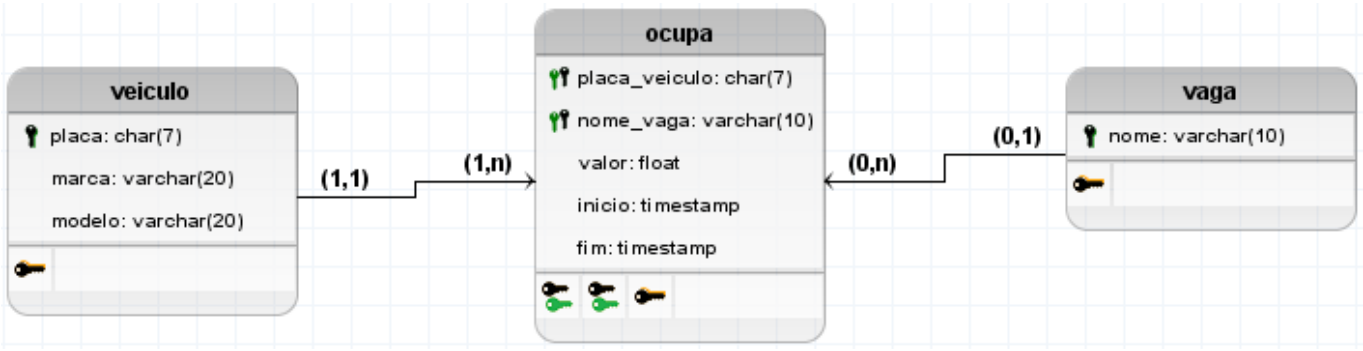
- Os atributos nome e UF são obrigatórios na tabela cidade;
- Os atributos idcidade e nome são obrigatórios na tabela bairro.

Exercício 3: Fazer a cláusula SQL para criar a tabela tarefa. Considere que:

- Uma tarefa pode ser formada por subtarefas.



Exercício 4: Fazer as cláusulas SQL para criar as tabelas do modelo lógico a seguir.



Exercício 5: Fazer as cláusulas SQL para criar as tabelas do modelo lógico a seguir. Considere que todo imóvel precisa ter um proprietário.