

Criação de Esquema de Banco de Dados utilizando PostgreSQL

Vinicius Takeo Friedrich Kuwaki

Universidade do Estado de Santa Catarina

Seções

Exemplo

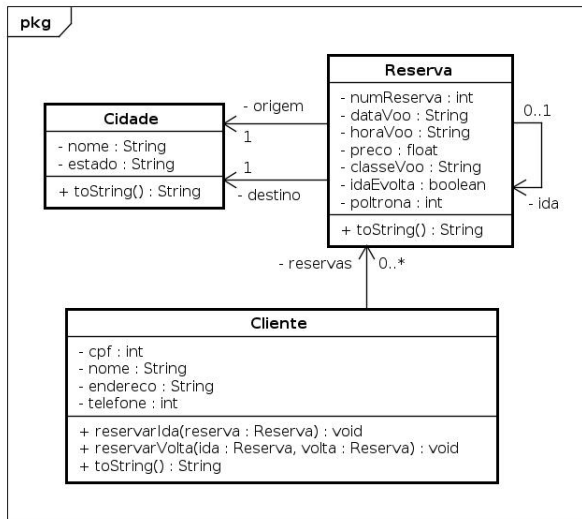
Resolução

Exercício

Exemplo

- A partir do do exercício da Aula Prática 3: Introdução a UML: Diagrama de Classes (diagrama no próximo slide);
- Transforme as classes do pacote de dados em tabelas relacionais para serem persistidas em um banco de dados SQL (veja o diagrama no slide a seguir);
- Utilize o PostgreSQL para isso;
- Consulte os slides do seguinte [link](#) para maiores explicações sobre a instalação do PostgreSQL e os comandos para criação das tabelas no banco;

Exemplo



Seções

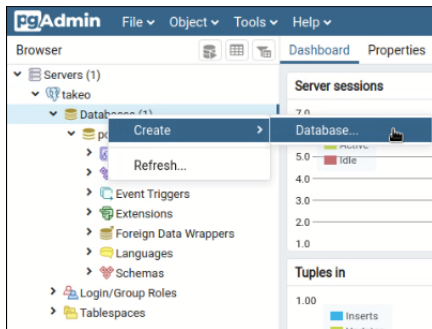
Exemplo

Resolução


Exercício

Resolução

- Primeiramente, vamos acessar o pgadmin4 para podermos criar o banco de dados;
- Vamos começar criando uma database chamada **reservas**:



Resolução

 Create - Database ✕

General

Definition

Security


Parameters

SQL



Database

reservas

Owner

 postgres ▾

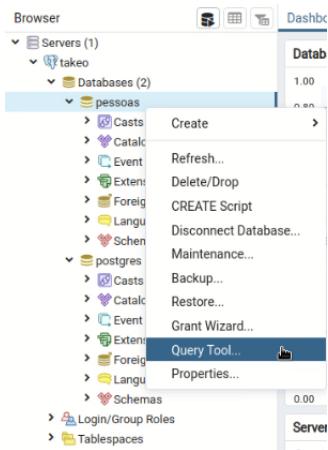
Comment

✕ Cancel ↺ Reset 💾 Save

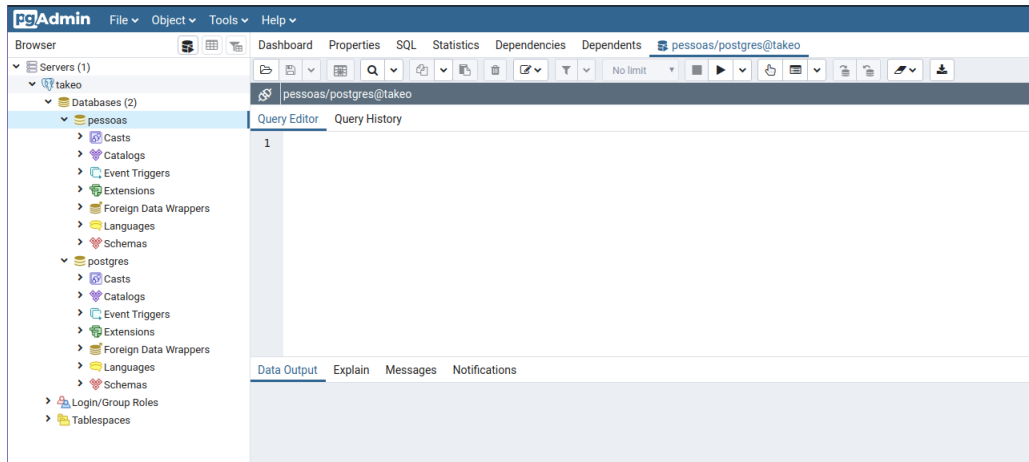
Resolução

- Para criar as tabelas utilizando comandos SQL iremos utilizar a ferramenta Query Tool;
- Para acessá-la clique com o botão direito sob a database **reservas**;



Resolução

- Execute os comandos SQL na janela do Query Editor.



Resolução

- Vamos começar definindo a tabela para os objetos do tipo Cidade;
- A classe Cidade possui dois atributos do tipo String: nome e estado;
- Utilizaremos o comando create table para isso;
- Vamos incluir um atributo id para ser a nossa chave primária.

```
create table cidade(  
    id int,  
);
```

- Vamos transforma os atributos do tipo String em campos do tipo varchar;
- Definindo um limite de 50 caracteres para essas Strings:

```
create table cidade(  
    id int,  
    nome varchar(50),  
    estado varchar(50),  
);
```

- Para definirmos o id como sendo a chave primária, vamos declará-lo como *primary key*:

```
create table cidade(  
    id int,  
    nome varchar(50),  
    estado varchar(50),  
    primary key (id)  
);
```

- Para definir automaticamente o valor do id para cada nova cidade inserida no banco, e garantir que esse id seja realmente único, vamos criar uma sequência chamada id_cidade:

```
create table cidade(  
    id int,  
    nome varchar(50),  
    estado varchar(50),  
    primary key (id)  
);  
  
create sequence id_cidade;
```

Resolução

- Agora, vamos criar a tabela para os objetos do tipo Cliente;
- Os tipos dos campos serão os mesmos que os usados na tabela de Cidade;
- Mas, vamos definir o endereço como um campo que pode conter 100 caracteres;
- A tabela também terá uma chave primária chamada id além dos atributos do objeto do tipo Cliente;

```
create table cliente(  
    id int,  
    cpf int,  
    nome varchar(50),  
    endereco varchar(100),  
    telefone int,  
    primary key (id)  
);
```

- Similarmente a tabela cidade, vamos definir outra sequência para a tabela Cliente

```
create table cliente(  
    id int,  
    cpf int,  
    nome varchar(50),  
    endereco varchar(100),  
    telefone int,  
    primary key (id)  
);  
  
create sequence id_cliente;
```

- Agora precisamos modelar a tabela da classe Reserva;
- Como a classe Reserva já possuía um atributo chamado numReserva, faremos com que esse seja o id;

```
create table reserva(  
    id int;  
)
```


- Para a data da Reserva, vamos utilizar um tipo de dados do postgresSQL chamado **date**:

```
create table reserva(  
    id int;  
    data date,  
)
```

- Para a hora da Reserva, vamos utilizar o tipo de dados do postgresSQL chamado **time**:

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
)
```

- O preço será um float:

```
create table reserva(  
  id int;  
  data date,  
  hora time,  
  preco float ,  
)
```

- A classe do voo vamos utilizar um varchar de 10 caracteres;

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
    preco float,  
    classeVoo varchar(10),  
)
```

Resolução

- Para o atributo idaEVolta, vamos defini-lo como um booleano;
- Caso o valor de idaEVolta seja **true**, significa que existe uma ida associada a reserva e, nesse caso, haverá uma reserva associada ao campo id_ida;
- Esse campo id_ida vai ser uma chave estrangeira que aponta para a própria tabela reserva;
- Mas, definiremos ela mais a frente;

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
    preco float,  
    classeVoo varchar(10),  
    idaEvolta boolean,  
    id_ida int,  
)
```

- Para os relacionamentos com a classe Cidade, que representam a origem e o destino, é necessário incluir a chave primária de Cidade como campos com chaves estrangeiras para referenciar a tabela Cidade;

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
    preco float,  
    classeVoo varchar(10),  
    idaEvolta boolean,  
    id_ida int,  
    id_origem int,  
    id_destino int,  
)
```

- A tabela Cliente também é referenciada para representar o relacionamento entre o Cliente e suas Reservas;
- Para isso, é necessário incluir o campo id_cliente como campo na tabela Reserva.

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
    preco float,  
    classeVoo varchar(10),  
    idaEvolta boolean,  
    id_ida int,  
    id_origem int,  
    id_destino int,  
    id_cliente int,  
)
```

- A chave primária é o id:

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
    preco float,  
    classeVoo varchar(10),  
    idaVolta boolean,  
    id_ida int,  
    id_origem int,  
    id_destino int,  
    id_cliente int,  
    primary key (id),  
)
```


- A chave estrangeira que referencia a tabela Reserva:

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
    preco float,  
    classeVoo varchar(10),  
    idaVolta boolean,  
    id_ida int,  
    id_origem int,  
    id_destino int,  
    id_cliente int,  
    primary key (id),  
    foreign key (id_ida) references reserva,  
)
```

- As chaves estrangeiras que referenciam a tabela Cidade (origem e destino):

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
    preco float,  
    classeVoo varchar(10),  
    idaVolta boolean,  
    id_ida int,  
    id_origem int,  
    id_destino int,  
    id_cliente int,  
    primary key (id),  
    foreign key (id_ida) references reserva,  
    foreign key (id_origem) references cidade,  
    foreign key (id_destino) references cidade,  
)
```

- Por fim, a chave estrangeira que referencia a tabela Cliente:

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
    preco float,  
    classeVoo varchar(10),  
    idaVolta boolean,  
    id_ida int,  
    id_origem int,  
    id_destino int,  
    id_cliente int,  
    primary key (id),  
    foreign key (id_ida) references reserva,  
    foreign key (id_origem) references cidade,  
    foreign key (id_destino) references cidade,  
    foreign key (id_cliente) references cliente  
)
```

- A tabela das reservas também precisa de uma sequência para controlar os id:

```
create table reserva(  
    id int;  
    data date,  
    hora time,  
    preco float,  
    classeVoo varchar(10),  
    idaEvolta boolean,  
    id_ida int,  
    id_origem int,  
    id_cliente int,  
    primary key (id),  
    foreign key (id_ida) references reserva,  
    foreign key (id_origem) references cidade,  
    foreign key (id_destino) references cidade,  
    foreign key (id_cliente) references cliente  
)  
  
create sequence id_reserva;
```

- Agora vamos inserir algumas tuplas nas tabelas;
- Para podemos obter os id's, utilizaremos a função **select nextval()**, passando como parâmetro a sequência que desejamos;
- Vamos inserir algumas cidades;
- Para isso, utilizaremos o comando **insert into cidade()** passando id, nome e estado como parâmetro:

```
insert into cidade(select nextval('id_cidade'), 'Joinville', 'SC')  
insert into cidade(select nextval('id_cidade'), 'Florianopolis', 'SC')  
insert into cidade(select nextval('id_cidade'), 'Curitiba', 'PR')
```

- Para listas as cidades no banco de dados, utilize **select * from cidade;**

Resolução

- Agora vamos incluir um cliente no banco;
- O comando é o mesmo, só os parâmetros serão diferentes;
- Passaremos: cpf, nome, endereço e telefone.
- Para obter os ids, a sequência utilizada será a `id_cliente`:

```
insert into cliente(select nextval('id_cliente'),11111,'Joao','Rua João Colin',213123);
```

- Para listar os clientes no banco de dados, utilize **select * from cliente;**

- Por fim, vamos adicionar duas reservas;
- Como acabamos de criar o banco de dados e inserimos algumas tuplas, temos os ids 1, 2 e 3 de cidades e 1 para o cliente;
- Então podemos utilizar o **insert into reserva**;
- Vamos utilizar a sequência id_reserva para o id da reserva;
- Vamos criar o comando pouco a pouco, pois precisaremos com alguns detalhes;

```
insert into reserva(select nextval('id_reserva'),)
```

- Para a data vamos inserir no formato 'ANO-MES-DIA':

```
insert into reserva(select nextval('id_reserva'), '2020-06-12')
```


- Para a hora vamos inserir no formato 'HORA:MINUTO:SEGUNDO':

```
insert into reserva(select nextval('id_reserva'), '2020-06-12', '11:00:00')
```

- Para o valor vamos colocar 100;
- E o tipo de voo vamos colocar Comercial;

```
insert into reserva(select nextval('id_reserva'), '2020-06-12', '11:00:00', 100.00, 'Comercial')
```

- Para idaEvolta vamos definir false, pois essa sera a ida;
- Precisaremos colocar o id_ida como null:

```
insert into reserva(select nextval('id_reserva'), '2020-06-12', '11:00:00', 100.00, 'Comercial', false, null)
```

- Por fim, vamos definir a cidade de origem como a 1 e de destino como 2:

```
insert into reserva(select nextval('id_reserva'), '2020-06-12', '11:00:00', 100.00, 'Comercial', false, null, 1, 2)
```

- E o cliente será o 1:

```
insert into reserva(select nextval('id_reserva'), '2020-06-12', '11:00:00', 100.00, 'Comercial', false, null, 1, 2, 1)
```

- Agora vamos definir uma volta;
- O valor de idaEvolta deverá ser true;
- E o id_ida vamos definir com 1;
- Precisamos inverter as cidades;
- Se uma aplicação for criada para utilizar o banco de dados, é ela quem tem que administrar esse "problema".

```
insert into reserva(select nextval('id_reserva'),'2020-06-12','17:00:00',100.00,'Comercial',true,1,2,1,1)
```

Seções

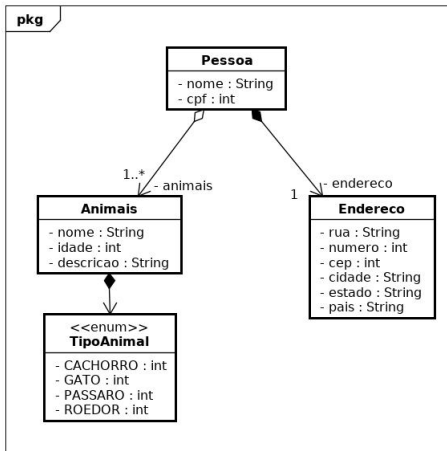
Exemplo


Resolução

Exercício

Exercício

- Para as classes abaixo, crie tabelas em um banco de dados para poder persistir objetos delas;



 KUWAKI, V. T. F. Modelo de slides udesc lattex. In: . [S.l.]: Disponível em: <<https://github.com/takeofriedrich/slidesUdescLattex>>. Acesso em: 5 jun. 2020.

Duvidas:
Vinicius Takeo Friedrich Kuwaki
vtkwki@gmail.com
github.com/takeofriedrich



UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA