Wellington Cesar Fonseca

DESENVOLVIMENTO FULL STACK

MISSÃO PRÁTICA | NÍVEL 2 | MUNDO 3

RPG0015 - VAMOS MANTER AS INFORMAÇÕES!

OBJETIVO

1. Identificar os requisitos de um sistema e transformá-los no modelo adequado.
2. Utilizar ferramentas de modelagem para bases de dados relacionais.
3. Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
4. Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
5. No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server.

1º PROCEDIMENTO | CRIANDO O BANCO DE DADOS

Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?

- **1x1** : Um relacionamento 1x1 indica que um registro em uma tabela está associado a exatamente um registro em outra tabela e vice-versa.
- **1xN** : Um relacionamento 1xN indica que um registro em uma tabela pode estar associado a muitos registros em outra tabela, mas cada registro na segunda tabela está associado a apenas um registro na primeira tabela.
- **NxN** : Um relacionamento NxN indica que vários registros em uma tabela podem estar associados a vários registros em outra tabela e vice-versa.

Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?

Em pesquisa nota-se três principais

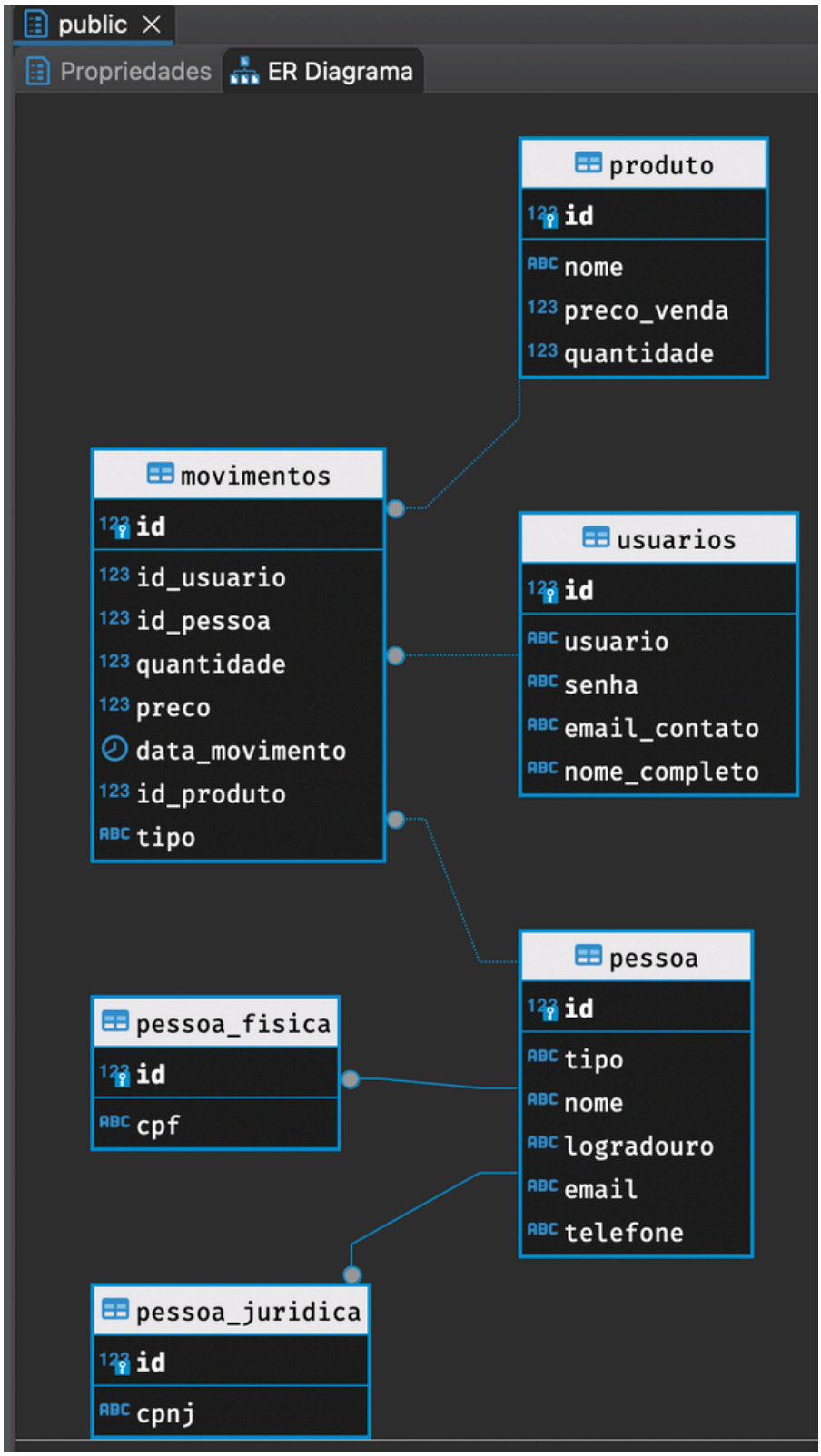
- **Single Table Inheritance (STI):** Neste, todas as heranças são representadas em uma única tabela. Uma coluna adicional é usada para discriminar os diferentes tipos.
- **Class Table Inheritance (CTI):** Este, cada herança tem sua tabela, mas as tabelas são relacionadas entre si.
- **Concrete Table Inheritance (CTI):** Cada tem sua própria tabela. As tabelas não compartilham uma estrutura comum de superclasse.

Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?

Não pude ter a experiência de usá-lo mas utilizando o dbeaver que trata-se do mesmo propósito há uma imensa vantagem pelo fato de você ter concentrado diversas funcionalidades, ganhando tempo em criação de tabela, consulta de dados, gerenciamento de usuários, validação de conexão, checagem de desempenho. Enfim, um client traz pra gente a vantagem de simplificar o acesso e gerenciamento.

Observação! Não consegui instalar no mac a ferramenta solicitada na missão e nem o sql server, os meus código apresentados e meu diagrama são: **Database Postgree** e **ER Diagram (DBeaver)**

```
CREATE TABLE public.usuarios (
 id bigserial NOT NULL,
 usuario varchar(100) NOT NULL,
 senha varchar(100) NOT NULL,
 email_contato varchar(100) NOT NULL,
 nome_completo varchar(100) NOT NULL,
 CONSTRAINT usuarios_pk PRIMARY KEY (id),
 CONSTRAINT usuarios_unique UNIQUE (usuario)
);

CREATE TABLE public.produto (
 id bigserial NOT NULL,
 nome varchar(100) NOT NULL,
 preco_venda float4 DEFAULT 0 NOT NULL,
 quantidade int4 DEFAULT 0 NOT NULL,
 CONSTRAINT produto_id PRIMARY KEY (id)
);

CREATE TABLE public.pessoa (
 id bigserial NOT NULL,
 tipo varchar(20) NOT NULL,
 nome varchar(100) NOT NULL,
 logradouro varchar(150) NOT NULL,
 email varchar(100) NOT NULL,
 telefone varchar(20) NOT NULL,
 CONSTRAINT pessoa_pk PRIMARY KEY (id)
);

CREATE TABLE public.pessoa_fisica (
 id bigserial NOT NULL,
 cpf varchar(20) NOT NULL,
 CONSTRAINT pessoa_fisica_pk PRIMARY KEY (id),
 CONSTRAINT pessoa_fisica_unique UNIQUE (cpf),
 CONSTRAINT pessoa_fisica_pessoa_fk FOREIGN KEY (id)
REFERENCES public.pessoa(id) ON DELETE CASCADE
);

CREATE TABLE public.pessoa_juridica (
 id bigserial NOT NULL,
 cpnj varchar(20) NOT NULL,
 CONSTRAINT pessoa_juridica_pk PRIMARY KEY (id),
 CONSTRAINT pessoa_juridica_unique UNIQUE (cpnj),
 CONSTRAINT pessoa_juridica_pessoa_fk FOREIGN KEY (id)
REFERENCES public.pessoa(id) ON DELETE CASCADE
);

CREATE TABLE public.movimentos (
 id bigserial NOT NULL,
 id_usuario bigserial NOT NULL,
 id_pessoa bigserial NOT NULL,
 quantidade int4 NOT NULL,
 preco float4 NOT NULL,
 data_movimento timestamp NOT NULL,
 id_produto bigserial NOT NULL,
 tipo varchar(20) NOT NULL,
 CONSTRAINT movimentos_pk PRIMARY KEY (id),
 CONSTRAINT movimentos_pessoa_fk FOREIGN KEY (id_pessoa)
REFERENCES public.pessoa(id),
 CONSTRAINT movimentos_produto_fk FOREIGN KEY (id_produto)
REFERENCES public.produto(id),
 CONSTRAINT movimentos_usuarios_fk FOREIGN KEY (id_usuario)
REFERENCES public.usuarios(id) ON DELETE CASCADE
);
```


public
Propriedades
ER Diagrama
produto
id
nome
preco_venda
quantidade
movimentos
id
id_usuario
id_pessoa
quantidade
preco
data_movimento
id_produto
tipo
usuarios
id
usuario
senha
email_contato
nome_completo
pessoa
id
tipo
nome
logradouro
email
telefone
pessoa_fisica
id
cpf
pessoa_juridica
id
cpnj