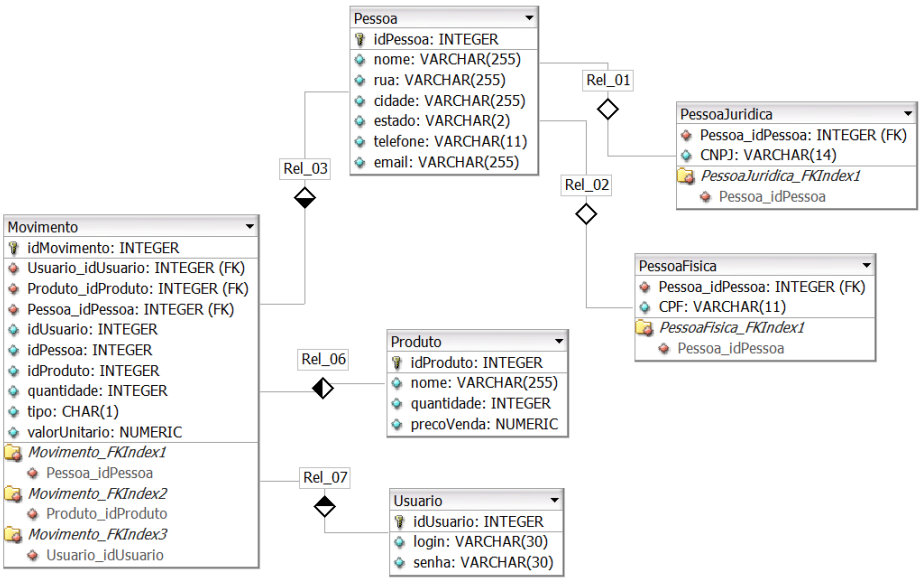
Faculdade Estácio - Polo Centro - Canela - RS
Curso: Desenvolvimento Full Stack
Disciplina: RPG0015 - Vamos manter as informações?
Turma: 9001 - Semestre Letivo: 2025.1 - 3º semestre
Integrante: Pedro Henrique Marques Medeiros Pinho
Matrícula: 202402031831
IDE: Sql Server Management Studio
Repositório Git: https://github.com/PedroPinho23/Vamos-Manter-as-Informa-es.git

Vamos manter as informações?

Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

Objetivos da Prática

- Identificar os requisitos de um sistema e transformá-los no modelo adequado.
- Utilizar ferramentas de modelagem para bases de dados relacionais.
- Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
- Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
- 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.



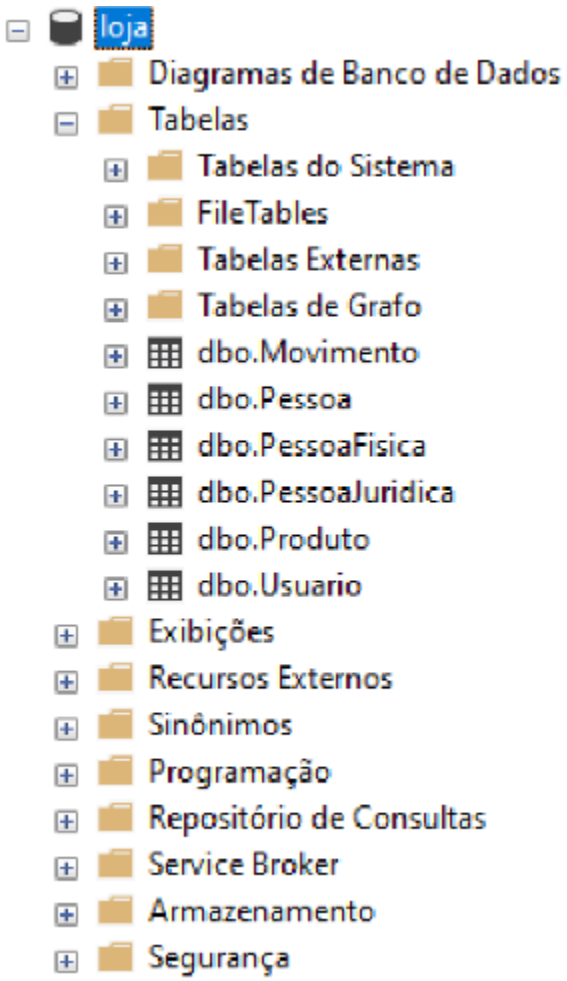
Códigos Utilizados:

```
CREATE DATABASE loja;

USE loja;

CREATE TABLE
        Pessoa (
        idPessoa INTEGER NOT NULL IDENTITY (1, 1) PRIMARY KEY,
        nome VARCHAR(255) NOT NULL,
        rua VARCHAR(255) NOT NULL,
        cidade VARCHAR(255) NOT NULL,
        estado CHAR(2) NOT NULL,
        telefone VARCHAR(11) NOT NULL,
        email VARCHAR(255) NOT NULL
```

```
	);

CREATE TABLE
	PessoaJuridica (
	idPessoa INTEGER NOT NULL PRIMARY KEY,
	cnpj VARCHAR(14) NOT NULL,
	CONSTRAINT fk_PessoaJuridica_Pessoa FOREIGN KEY (idPessoa)
REFERENCES dbo.Pessoa (idPessoa)
	);

CREATE TABLE
	PessoaFisica (
	idPessoa INTEGER NOT NULL PRIMARY KEY,
	cpf VARCHAR(11) NOT NULL,
	constraint fk_PessoaFisica_Pessoa foreign key (idPessoa) references Pessoa
(idPessoa)
	);

CREATE TABLE
	Produto (
	idProduto INTEGER NOT NULL IDENTITY (1, 1) PRIMARY KEY,
	nome VARCHAR(255) NOT NULL,
	quantidade INTEGER NOT NULL,
	precoVenda NUMERIC NOT NULL
	);

CREATE TABLE
	Usuario (
	idUsuario INTEGER NOT NULL IDENTITY (1, 1) PRIMARY KEY,
	login VARCHAR(50) NOT NULL,
	senha VARCHAR(50) NOT NULL
	);

CREATE TABLE
	Movimento (
	idMovimento INTEGER NOT NULL IDENTITY (1, 1) PRIMARY KEY,
	idUsuario INTEGER NOT NULL,
	idPessoa INTEGER NOT NULL,
	idProduto INTEGER NOT NULL,
	quantidade INTEGER NOT NULL,
```

```
	tipo CHAR(1) NOT NULL,
	valorUnitario FLOAT NOT NULL,
	constraint fk_Movimento_Produto foreign key (idProduto) references
dbo.Produto (idProduto),
	constraint fk_Movimento_Usuario foreign key (idUsuario) references dbo.Usuario
(idUsuario),
	constraint fk_Movimento_Pessoa foreign key (idPessoa) references dbo.Pessoa
(idPessoa)
	);

CREATE SEQUENCE dbo.CodigoPessoa
	START WITH 1
	INCREMENT BY 1 ;
```

Resultados:



Conclusão:

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

Em um banco de dados relacional, as cardinalidades são implementadas através de chaves primárias e estrangeiras. Na relação **1x1**, coloca-se uma chave estrangeira na segunda tabela, referenciando a chave primária da primeira. Para **1xN**, a chave estrangeira é adicionada à tabela do lado "N", referenciando a tabela do lado "1". Já na relação **NxN**, cria-se uma tabela intermediária, chamada de tabela de junção, que contém as chaves estrangeiras de ambas as tabelas envolvidas. Essas implementações garantem a integridade e o correto mapeamento das relações entre as tabelas no banco de dados.

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

Para representar herança em bancos de dados relacionais, existem três maneiras principais. No Single Table Inheritance (STI), todas as classes ficam em uma única tabela, com uma coluna extra para identificar o tipo de cada registro. No Class Table Inheritance (CTI), cada classe tem sua própria tabela, com uma chave estrangeira para a superclasse. E no Concrete Table Inheritance (CTI), cada classe tem uma tabela separada, sem chaves estrangeiras, mas com dados duplicados da superclasse.

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

O SQL Server Management Studio ajuda a melhorar a produtividade nas tarefas de gerenciamento de banco de dados oferecendo uma interface gráfica intuitiva, onde é possível gerenciar e configurar bancos de dados, realizar consultas, e criar scripts de maneira mais fácil. Ele também oferece recursos como o IntelliSense, que autocompleta comandos SQL, além de permitir a execução de tarefas administrativas como backups, restaurações e monitoramento de performance. Com ferramentas de análise e relatórios, o SSMS facilita o diagnóstico e otimização do banco de dados, tornando o gerenciamento mais eficiente.