



Estácio

Estácio - Mundo 3 - Missão Nível 2

Faculdade Estácio - Polo Centro – Belo Horizonte – MG

Curso: Desenvolvimento Full Stack.

Disciplina: Nível 2: Vamos Manter as Informações !

RPG0015.

Semestre Letivo: 3

Integrante: André Luiz Ferreira da Silva

Repositório: <https://github.com/Andre-Luiz22/m3-n2>

IDE: Sql Server Management Studio.

Título da Prática

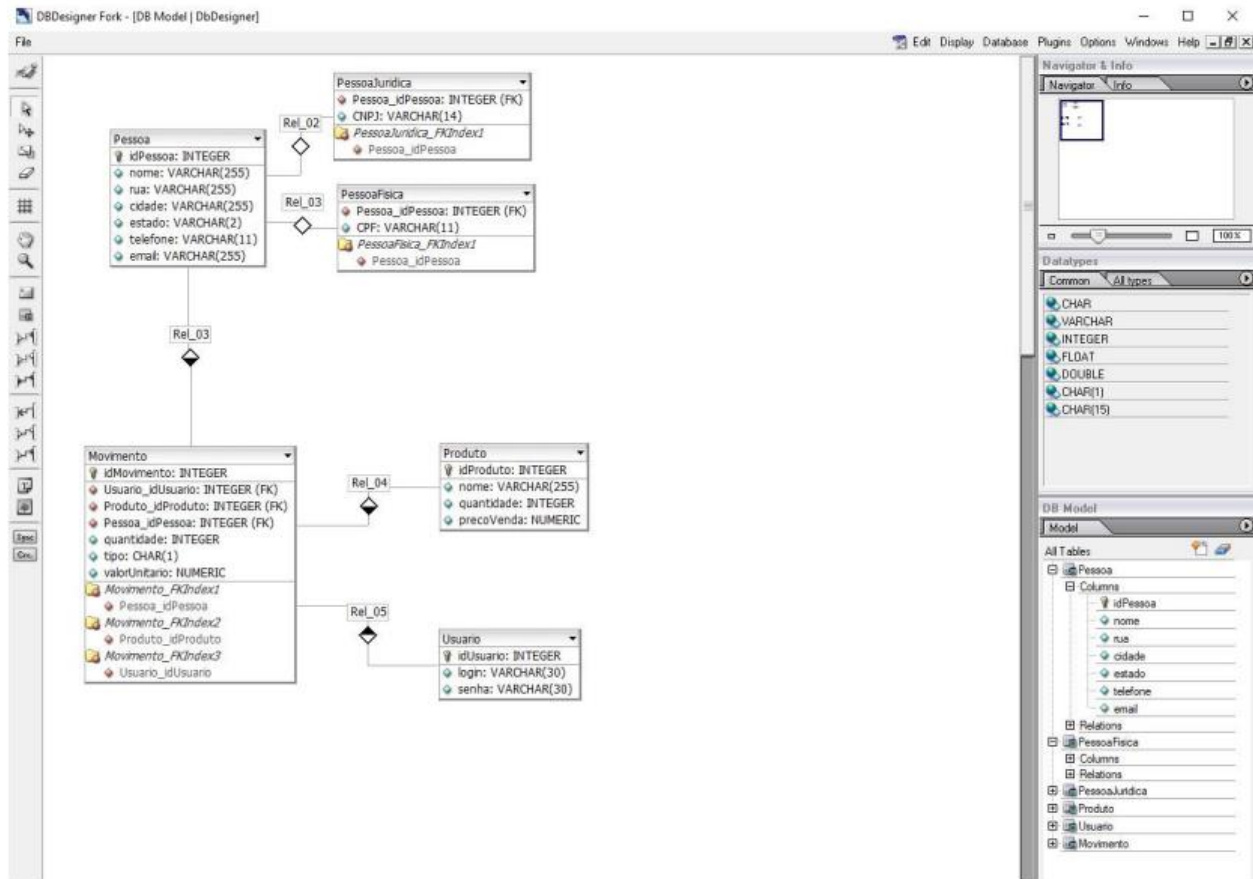
Vamos manter as informações!

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

Objetivos da Prática

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 de sintaxe SQL, na plataforma SQL Server.

Arquivos do Projeto



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;

Resultado

-   loja
 -   Diagramas de Banco de Dados
 -   Tabelas
 -   Tabelas do Sistema
 -   FileTables
 -   Tabelas Externas
 -   Tabelas de Grafo
 -   dbo.Movimento
 -   dbo.Pessoa
 -   dbo.PessoaFisica
 -   dbo.PessoaJuridica
 -   dbo.Produto
 -   dbo.Usuario
 -   Exibições
 -   Recursos Externos
 -   Sinônimos
 -   Programação
 -   Repositório de Consultas
 -   Service Broker
 -   Armazenamento
 -   Segurança

Análise e Conclusão:

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

As diferentes cardinalidades em um banco de dados relacional são implementadas através do uso de chaves estrangeiras e da modelagem correta das relações entre tabelas.

1. 1 para 1 (1:1):
 - Neste caso, cada registro em uma tabela está associado a exatamente um registro em outra tabela e vice-versa.
2. 1 para muitos (1:N):
 - Aqui, cada registro em uma tabela pode estar associado a muitos registros em outra tabela, mas cada registro nesta segunda tabela está associado a no máximo um registro na primeira tabela. A implementação envolve colocar uma chave estrangeira na tabela do "muitos" que referencia a chave primária na tabela do "um".
3. Muitos para muitos (N:N):
 - Em uma relação muitos para muitos, cada registro em uma tabela pode estar associado a muitos registros em outra tabela, e vice-versa. Isso é implementado através do uso de uma tabela de junção, também conhecida como tabela de associação ou tabela intermediária. Esta tabela de junção contém chaves estrangeiras que referenciam as chaves primárias das duas tabelas relacionadas.

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

Para representar o conceito de herança em bancos de dados relacionais, geralmente é utilizado um modelo conhecido como herança de tabela única (single-table inheritance) ou herança de várias tabelas (class table inheritance).

- 1) Herança de Tabela Única (Single-Table Inheritance):
 - Neste modelo, todas as classes (ou tipos) herdam de uma única tabela. A tabela contém colunas para todos os atributos de todas as classes e uma coluna adicional para indicar o tipo de cada registro. Essa abordagem pode

levar a registros com muitos valores nulos, pois cada registro deve acomodar todos os atributos possíveis de todas as classes.

2) Herança de Múltiplas Tabelas (Class Table Inheritance):

- Neste modelo, cada classe tem sua própria tabela no banco de dados. As tabelas filhas contêm apenas os atributos específicos daquela classe, além de uma chave estrangeira que referencia a tabela pai. Isso resulta em uma estrutura mais normalizada e evita o armazenamento de valores nulos, mas pode exigir junções (joins) para recuperar todos os dados de uma hierarquia de herança.

A escolha entre os dois modelos depende das necessidades específicas do sistema e das preferências de design. A herança de tabela única pode ser mais simples de implementar e pode ser preferível para hierarquias de herança simples com poucas classes e atributos. Por outro lado, a herança de várias tabelas oferece uma estrutura mais normalizada e pode ser mais adequada para hierarquias de herança complexas ou grandes volumes de dados.

3) 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 (SSMS) é uma ferramenta robusta fornecida pela Microsoft para gerenciar bancos de dados SQL Server. Ele oferece uma ampla gama de recursos que podem melhorar significativamente a produtividade nas tarefas relacionadas ao gerenciamento do banco de dados. Possui interface gráfica amigável, editor de consultas avançado, designer de tabelas e diagramas, gerenciamento de segurança, automatização de tarefas e muitos outros recursos.