**Universidade Estácio de Sá**

**Polo Estácio EAD - Batatais / SP**

**Curso:** Desenvolvimento Full Stack  
**Disciplina:** Vamos manter as informacoes  
**Missão Prática - Nível 2  
Turma:** RPG0014  
**Semestre Letivo:** 2024.3  
 **Integrante:** Daiana Maira de Oliveira Lascala

**Título**

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 da sintaxe SQL, na plataforma do SQL Server.

**Descrição dos Códigos:**

**Desenho no DBDesigner Fork**

**A screenshot of a computer

Description automatically generated**

**Codigo no SQL Server:**

CREATE TABLE Pessoa (

idpessoa INTEGER NOT NULL IDENTITY ,

tipo CHAR(1) NOT NULL CHECK (tipo IN ('F', 'J')),

nomePessoa VARCHAR(100) NOT NULL ,

PRIMARY KEY(idpessoa));

GO

CREATE TABLE Produto (

idProduto INTEGER NOT NULL IDENTITY ,

nomeProduto VARCHAR(100) NOT NULL ,

quantidadeProduto INTEGER NOT NULL ,

precoProduto NUMERIC NOT NULL ,

PRIMARY KEY(idProduto));

GO

CREATE TABLE Usuario (

idUsuario INTEGER NOT NULL IDENTITY ,

nomeUsuario VARCHAR(100) NOT NULL ,

senhaUsuario VARCHAR(20) NOT NULL ,

emailUsuario VARCHAR(80) NOT NULL ,

tipoUsuario BIT NOT NULL CHECK (tipoUsuario IN (0, 1)) DEFAULT 0 ,

PRIMARY KEY(idUsuario));

GO

CREATE TABLE pessoaJuridica (

idpessoaJuridica INTEGER NOT NULL IDENTITY ,

Pessoa\_idpessoa INTEGER NOT NULL ,

cnpjJuridica CHAR(14) NOT NULL ,

razaoSocialJuridica VARCHAR(200) NOT NULL ,

logradouroJuridica VARCHAR(200) ,

cidadeJuridica VARCHAR(50) NOT NULL ,

estadoJuridica CHAR(2) NOT NULL ,

cepJuridica CHAR(8) NOT NULL ,

PRIMARY KEY(idpessoaJuridica, Pessoa\_idpessoa) ,

FOREIGN KEY(Pessoa\_idpessoa)

REFERENCES Pessoa(idpessoa));

GO

CREATE INDEX pessoaJuridica\_FKIndex1 ON pessoaJuridica (Pessoa\_idpessoa);

GO

CREATE INDEX IFK\_fkPessoaJuridica ON pessoaJuridica (Pessoa\_idpessoa);

GO

CREATE TABLE pessoaFisica (

idpessoaFisica INTEGER NOT NULL IDENTITY ,

Pessoa\_idpessoa INTEGER NOT NULL ,

cpfFisica CHAR(11) NOT NULL ,

dataNascimentoFisica DATE NOT NULL ,

logradouroFisica VARCHAR(200) ,

cidadeFisica VARCHAR(50) NOT NULL ,

estadoFisica CHAR(2) NOT NULL ,

cepFisica CHAR(8) NOT NULL ,

PRIMARY KEY(idpessoaFisica, Pessoa\_idpessoa) ,

FOREIGN KEY(Pessoa\_idpessoa)

REFERENCES Pessoa(idpessoa));

GO

CREATE INDEX pessoaFisica\_FKIndex1 ON pessoaFisica (Pessoa\_idpessoa);

GO

CREATE INDEX IFK\_fkPessoaFisica ON pessoaFisica (Pessoa\_idpessoa);

GO

CREATE TABLE movimentoCompra (

idMovimentoCompra INTEGER NOT NULL IDENTITY ,

pessoaJuridica\_Pessoa\_idpessoa INTEGER NOT NULL ,

pessoaJuridica\_idpessoaJuridica INTEGER NOT NULL ,

Produto\_idProduto INTEGER NOT NULL ,

Usuario\_idUsuario INTEGER NOT NULL ,

quantidadeCompra INT NOT NULL ,

precoUnitarioCompra DECIMAL NOT NULL ,

dataMovimentoCompra DATETIME NOT NULL DEFAULT GETDATE() ,

PRIMARY KEY(idMovimentoCompra) ,

FOREIGN KEY(Usuario\_idUsuario)

REFERENCES Usuario(idUsuario),

FOREIGN KEY(Produto\_idProduto)

REFERENCES Produto(idProduto),

FOREIGN KEY(pessoaJuridica\_idpessoaJuridica, pessoaJuridica\_Pessoa\_idpessoa)

REFERENCES pessoaJuridica(idpessoaJuridica, Pessoa\_idpessoa));

GO

CREATE INDEX MovimentoCompra\_FKIndex1 ON movimentoCompra (Usuario\_idUsuario);

GO

CREATE INDEX MovimentoCompra\_FKIndex3 ON movimentoCompra (Produto\_idProduto);

GO

CREATE INDEX movimentoCompra\_FKIndex\_3 ON movimentoCompra (pessoaJuridica\_idpessoaJuridica, pessoaJuridica\_Pessoa\_idpessoa);

GO

CREATE INDEX IFK\_fkUsuarioMovimentoCompra ON movimentoCompra (Usuario\_idUsuario);

GO

CREATE INDEX IFK\_fkProdutoMovimentoCompra ON movimentoCompra (Produto\_idProduto);

GO

CREATE INDEX IFK\_fkJuridicaMovimentoCompra ON movimentoCompra (pessoaJuridica\_idpessoaJuridica, pessoaJuridica\_Pessoa\_idpessoa);

GO

CREATE TABLE movimentoVenda (

idmovimentoVenda INTEGER NOT NULL IDENTITY ,

Usuario\_idUsuario INTEGER NOT NULL ,

pessoaFisica\_Pessoa\_idpessoa INTEGER NOT NULL ,

pessoaFisica\_idpessoaFisica INTEGER NOT NULL ,

Produto\_idProduto INTEGER NOT NULL ,

quantidadeVenda INTEGER NOT NULL ,

precoUnitarioVenda DECIMAL NOT NULL ,

dataMovimentoVenda DATETIME NOT NULL DEFAULT GETDATE() ,

PRIMARY KEY(idmovimentoVenda) ,

FOREIGN KEY(Produto\_idProduto)

REFERENCES Produto(idProduto),

FOREIGN KEY(pessoaFisica\_idpessoaFisica, pessoaFisica\_Pessoa\_idpessoa)

REFERENCES pessoaFisica(idpessoaFisica, Pessoa\_idpessoa),

FOREIGN KEY(Usuario\_idUsuario)

REFERENCES Usuario(idUsuario));

GO

CREATE INDEX movimentoVenda\_FKIndex1 ON movimentoVenda (Produto\_idProduto);

GO

CREATE INDEX movimentoVenda\_FKIndex2 ON movimentoVenda (pessoaFisica\_idpessoaFisica, pessoaFisica\_Pessoa\_idpessoa);

GO

CREATE INDEX movimentoVenda\_FKIndex3 ON movimentoVenda (Usuario\_idUsuario);

GO

CREATE INDEX IFK\_fkProdutoMovimentoVenda ON movimentoVenda (Produto\_idProduto);

GO

CREATE INDEX IFK\_fkFisicaMovimentoVenda ON movimentoVenda (pessoaFisica\_idpessoaFisica, pessoaFisica\_Pessoa\_idpessoa);

GO

CREATE INDEX IFK\_fkUsuarioMovimentoVenda ON movimentoVenda (Usuario\_idUsuario);

GO

**Análise e Conclusão**

**1. Implementação das Diferentes Cardinalidades em Bancos de Dados Relacionais**

* 1x1 (Um para Um): Cada registro de uma tabela está relacionado com no máximo um registro de outra tabela. Exemplo: Tabelas Pessoa e PessoaFisica onde cada pessoa física tem uma única entrada na tabela PessoaFisica.
* 1xN (Um para Muitos): Um registro de uma tabela pode estar relacionado com múltiplos registros de outra tabela. Exemplo: Uma PessoaJuridica pode ter múltiplos MovimentosCompra.
* NxN (Muitos para Muitos): Múltiplos registros de uma tabela podem estar relacionados com múltiplos registros de outra tabela. Para implementar, geralmente utiliza-se uma tabela intermediária. Exemplo: Se uma tabela de Produtos pudesse estar relacionada com múltiplas Categorias e vice-versa, seria criada uma tabela ProdutoCategoria.

**2. Relacionamento para Representar Herança em Bancos de Dados Relacionais**

Para representar herança em bancos de dados relacionais, geralmente utiliza-se o Relacionamento 1x1 ou 1xN. No modelo utilizado nesta prática, a herança é implementada através de tabelas específicas (PessoaFisica e PessoaJuridica) que possuem uma chave estrangeira referenciando a tabela Pessoa. Cada registro na tabela Pessoa pode estar relacionado a apenas uma das tabelas especializadas, representando assim uma relação de herança.

**3. Melhoria da Produtividade com o SQL Server Management Studio (SSMS)**

O SQL Server Management Studio (SSMS) oferece diversas funcionalidades que aumentam a produtividade no gerenciamento de bancos de dados, tais como:

* **Interface Intuitiva:** Permite a criação, modificação e gerenciamento de objetos de banco de dados através de uma interface gráfica amigável.
* **Editor de Código Avançado:** Com recursos como auto-completar, realce de sintaxe, e sugestões de código que facilitam a escrita de scripts SQL.
* **Ferramentas de Depuração:** Permite depurar procedimentos armazenados e scripts SQL, identificando e corrigindo erros de forma eficiente.
* **Integração com Controle de Versão:** Facilita o versionamento de scripts e a colaboração em equipe.
* **Visualização de Dados e Estruturas:** Oferece formas visuais de visualizar dados, diagramas de banco de dados e dependências entre objetos.
* **Automação de Tarefas:** Através de scripts e tarefas agendadas, é possível automatizar operações recorrentes, como backups e manutenção.

Essas funcionalidades tornam o SSMS uma ferramenta poderosa para desenvolvedores e administradores de banco de dados, otimizando o fluxo de trabalho e reduzindo o tempo necessário para realizar tarefas complexas.

**Repositório no GitHub**

https://github.com/DaianaLascala/RPG0015\_VamosManterAsInformacoes-