Implementação de Banco de Dados para Gerenciamento de Movimentos de Compra e Venda

Missão Prática | Mundo 3 | Nivel 2

- **Aluno:** Lucas de Oliveira dos Santos
- **Matrícula:** 202303688466
- **Campus:** Monte Castelo
- **Curso:** Desenvolvimento Full-Stack
- **Disciplina:** Nível 2: Vamos Manter as Informações?
- **Turma:** 01.23
- **Semestre Letivo:** 3º Semestre
- Link do Repositório GitHub

Objetivo da Prática

Esta prática tem como objetivo a criação de um banco de dados relacional para o gerenciamento de movimentos de compra e venda dentro de uma plataforma de negociações, utilizando SQL Server Management Studio e aplicando conceitos de modelagem UML e SQL.

👉 1º Procedimento – Criando o Banco de Dados

Definição do Modelo de Dados

A estrutura do banco de dados é projetada seguindo o diagrama de classe UML, definindo as entidades `Usuario`, `Pessoa`, `PessoaFisica`, `PessoaJuridica`, `Produto`, `MovimentoCompra`, e `MovimentoVenda`, cada uma com seus atributos e relações, exemplificando a organização e relacionamentos entre as tabelas.

Criação da Base de Dados

```
CREATE  TABLE  Usuario (
idOperador INT  PRIMARY KEY,
nome VARCHAR(255),
senha VARCHAR(255)
);

CREATE  TABLE  Pessoa (
idPessoa INT  PRIMARY KEY,
nome VARCHAR(255),
logradouro VARCHAR(255),
cidade VARCHAR(255),
estado CHAR(2),
telefone VARCHAR(11),
email VARCHAR(255)
);

CREATE  TABLE  PessoaFisica (
idPessoaFisica INT  PRIMARY KEY,
cpf VARCHAR(11),
FOREIGN KEY (idPessoaFisica) REFERENCES Pessoa(idPessoa)
);

CREATE  TABLE  PessoaJuridica (
idPessoaJuridica INT  PRIMARY KEY,
cnpj VARCHAR(14),
FOREIGN KEY (idPessoaJuridica) REFERENCES Pessoa(idPessoa)
);

CREATE  TABLE  Produto (
idProduto INT  PRIMARY KEY,
nome VARCHAR(255),
quantidade INT,
precoVenda NUMERIC
);

CREATE  TABLE  PessoaFisica (
idPessoaFisica INT  PRIMARY KEY,
cpf VARCHAR(11),
FOREIGN KEY (idPessoaFisica) REFERENCES Pessoa(idPessoa)
);

CREATE  TABLE  MovimentoVenda (
idMovimentoVenda INT  PRIMARY KEY,
quantidade INT,
precoUnitario NUMERIC,
idProduto INT,
idComprador INT,
idOperador INT,
FOREIGN KEY (idProduto) REFERENCES Produto(idProduto),
FOREIGN KEY (idComprador) REFERENCES PessoaFisica(idPessoaFisica),
FOREIGN KEY (idOperador) REFERENCES Usuario(idOperador)
```

```
);

CREATE  TABLE  MovimentoCompra (
idMovimentoCompra INT  PRIMARY KEY,
quantidade INT,
precoUnitario NUMERIC,
idProduto INT,
idFornecedor INT,
idOperador INT,
FOREIGN KEY (idProduto) REFERENCES Produto(idProduto),
FOREIGN KEY (idFornecedor) REFERENCES PessoaJuridica(idPessoaJuridica),
FOREIGN KEY (idOperador) REFERENCES Usuario(idOperador)
);

CREATE  SEQUENCE  PessoaIdSequence
START  WITH  1
INCREMENT BY  1;


## Análise e Conclusão

### Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um ba

Cardinalidades em bancos de dados relacionais são implementadas da seguinte forma:

- **1X1 (Um para Um):** Quando uma unidade está ligada a, no máximo, outra unidade, é formado

- **1XN (Um para Muitos):** Na dinâmica de um para muitos, acontece quando uma instância pode

- **NxN (Muitos para Muitos):** Uma conexão muitos para muitos é expressa por meio de uma tabe

### Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos

AA representação da herança em bancos de dados relacionais é comumente realizada por meio de u

### Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relaci

O SQL Server Management Studio (SSMS) eleva a eficiência ao disponibilizar uma interface gráfi

## 👉 2º Procedimento – Alimentando a Base

### Alimentação Inicial das Tabelas

Incluindo dados nas tabelas:

**Inserção de Usuários:**

```sql
INSERT INTO Usuario (nome, senha) VALUES ('op1', 'op1'), ('op2', 'op2');
```

Inserção de Produtos:
```

```
INSERT INTO Produto (idProduto, nome, quantidade, precoVenda) VALUES (1, 'Banana', 100, 5.00);
INSERT INTO Produto (idProduto, nome, quantidade, precoVenda) VALUES (2, 'Laranja', 500, 2.00)
INSERT INTO Produto (idProduto, nome, quantidade, precoVenda) VALUES (3, 'Manga', 800, 4.00);
```

Análise e Conclusão

a. Quais as diferenças no uso de sequence e identity?

As principais diferenças entre `SEQUENCE` e `IDENTITY` são:

SEQUENCE: Trata-se de um recurso desenvolvido e controlado pelo sistema de banco de dados, responsável por produzir uma sequência numérica contínua, independente de uma tabela em particular. Sua utilidade abrange diversas tabelas e permanece inalterada mesmo após a remoção de registros.

- **Flexibilidade:** Uma `SEQUENCE` é um elemento independente que produz uma sequência de números em ordem, sem estar vinculado a uma tabela específica.
- **Reutilização:** Pode ser usada por múltiplas tabelas e colunas.
- **Controle:** Oferece um controle avançado no processo de geração de números, permitindo a definição do valor inicial, do incremento, do valor mínimo e máximo, e a opção de reciclar a sequência conforme necessário. **IDENTITY**: É uma propriedade de coluna específica de uma tabela que gera automaticamente valores numéricos sequenciais. É restrito a uma coluna em uma tabela e é geralmente reiniciado quando os registros são deletados e a tabela é recriada.
- **Simplicidade:** A propriedade `IDENTITY` é usada para gerar automaticamente valores numéricos sequenciais diretamente em uma coluna de uma tabela.
- **Especificidade:** Está diretamente ligada a uma coluna específica em uma tabela.
- **Facilidade de uso:** É mais fácil de configurar, pois requer menos parâmetros.

b. Qual a importância das chaves estrangerias para a consistência do banco?

Chaves estrangeiras são essenciais para:

- **Integridade Referencial:** A utilização de chaves estrangeiras é fundamental para assegurar a integridade referencial entre tabelas, garantindo a preservação das relações entre elas.
- **Prevenção de Orfãos:** Impedem a existência de registros "órfãos" em tabelas que dependem de outras para manterem sua integridade e significado.

- **Consistência de Dados:** Asseguram que apenas dados válidos sejam inseridos na tabela que possui a chave estrangeira.

c.

Na **Álgebra Relacional**, operadores como `SELECT`, `PROJECT`, `JOIN`, `UNION`, `INTERSECT`, e `DIFFERENCE` são usados para manipular e consultar dados em bancos de dados relacionais.

- **Diferença:** Retorna diferenças entre duas consultas.
- **Produto Cartesiano:** Combina todas as linhas de duas tabelas.
- **Seleção:** Filtragem de linhas.
- **Projeção:** Filtragem de colunas.
- **União:** Combina resultados de duas consultas.
- **Junção:** Combina linhas baseadas em condições de junção.
- **Predicados:** Expressões que retornam verdadeiro ou falso.
- **Quantificadores Universais e Existenciais:** Usados para expressar consultas com condições "para todos" ou "existe". No **Cálculo Relacional**, utiliza-se uma coleção de operadores lógicos como `AND`, `OR`, `NOT`, e `EXISTS`, que permitem a formulação de consultas baseadas em predicados e condições.

d. Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

A consulta em SQL emprega a funcionalidade GROUP BY para agrupar registros que compartilham os mesmos valores em colunas específicas.

- **Requisito Obrigatório**: Quando utilizado, todas as colunas listadas na cláusula `SELECT` que não estão incluídas nas funções agregadas (`COUNT`, `MAX`, `MIN`, `SUM`, `AVG`) devem estar presentes na cláusula `GROUP BY`.