Missão Prática | Nível 2 | Mundo 3

POLO RUA TEREZA - PETRÓPOLIS - RJ

Desenvolvimento Full Stack

Disciplina: Nível 2: Vamos Manter as Informações?

Número da Turma: RPG0015

Semestre Letivo: 3

Integrantes: Geovanne Alves Barcellos

Link Git: https://github.com/Geovanne28/Mundo-3-nivel-2-Estacio-Full-Stack

---

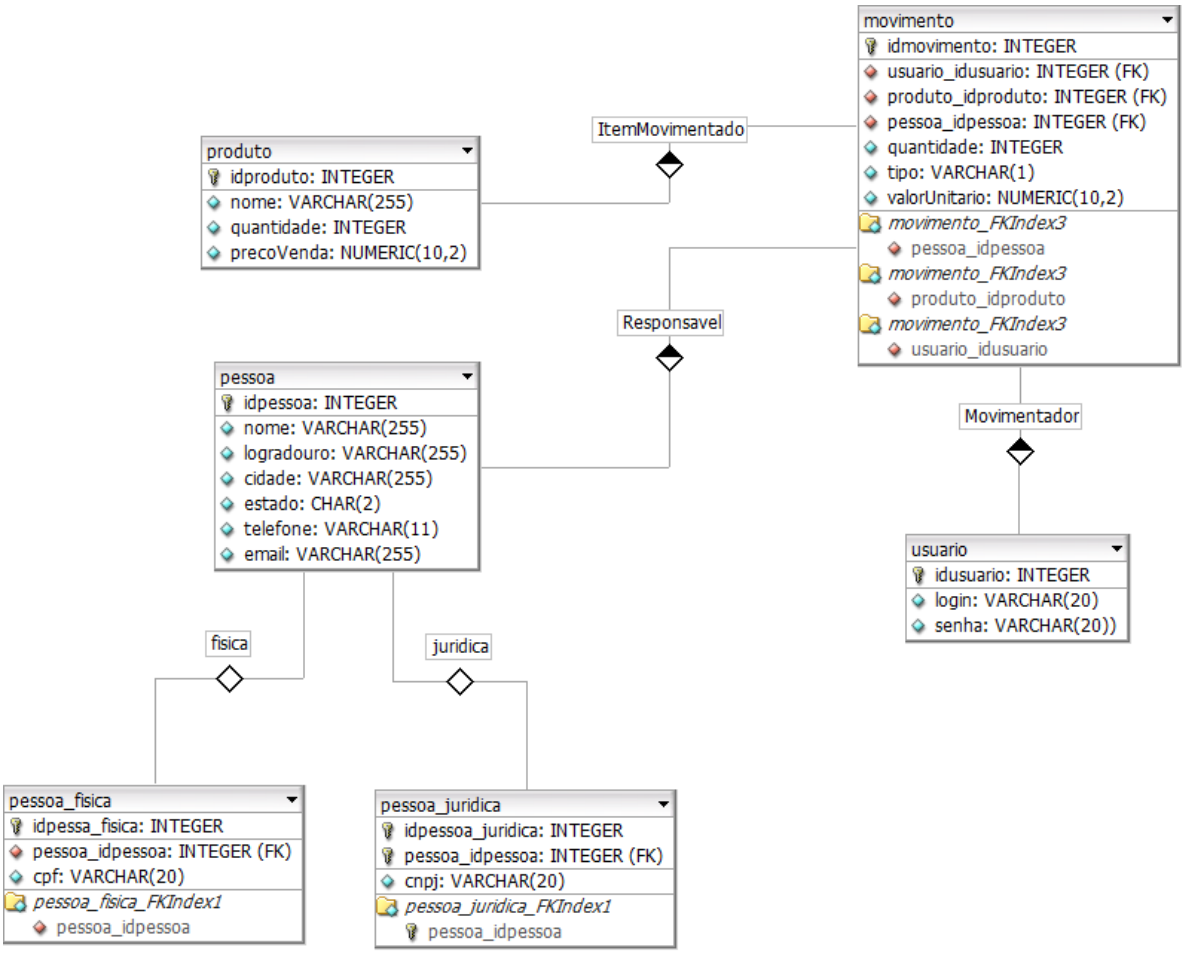
Objetivos da Prática:

- 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.

Título da Prática: 1° Procedimento | Criando o Banco de Dados

Códigos solicitados neste roteiro de aula:

- Modelagem:



- Script completo:

```
CREATE TABLE Movimento (
  idMovimento INTEGER IDENTITY NOT NULL PRIMARY KEY,
  Pessoa_idPessoa INTEGER NOT NULL CHECK (Pessoa_idPessoa > 0),
  Produto_idProduto INTEGER NOT NULL CHECK (Produto_idProduto > 0),
  Usuario_idUsuario INTEGER NOT NULL CHECK (Usuario_idUsuario > 0),
  quantidade INTEGER NULL CHECK (quantidade > 0),
  tipo CHAR(1) NULL,
  valorUnitario NUMERIC(10, 2) NULL
);

CREATE INDEX Movimento_FKIndex1 ON Movimento(Usuario_idUsuario);
CREATE INDEX Movimento_FKIndex2 ON Movimento(Produto_idProduto);
CREATE INDEX Movimento_FKIndex3 ON Movimento(Pessoa_idPessoa);


CREATE TABLE PessoaFisica (
  idPessoaFisica INTEGER IDENTITY NOT NULL PRIMARY KEY,
  Pessoa_idPessoa INTEGER NOT NULL CHECK (Pessoa_idPessoa > 0),
  cpf VARCHAR(20) NULL
);

CREATE INDEX PessaFisica_FKIndex1 ON PessoaFisica(Pessoa_idPessoa);


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


CREATE TABLE PessoaJuridica (
  idPessoaJuridica INTEGER IDENTITY NOT NULL PRIMARY KEY,
  Pessoa_idPessoa INTEGER NOT NULL CHECK (Pessoa_idPessoa > 0),
  cnpj VARCHAR(20) NULL
);

CREATE INDEX PessoaJuridica_FKIndex1 ON PessoaJuridica(Pessoa_idPessoa);


CREATE TABLE Produto (
  idProduto INTEGER IDENTITY NOT NULL PRIMARY KEY,
  nome VARCHAR(255) NULL,
  quantidade INTEGER NULL CHECK (quantidade > 0),
  precoVenda NUMERIC(10, 2) NULL -- Alterado o tipo de dados para NUMERIC(10, 2)
);


CREATE TABLE Usuario (
  idUsuario INTEGER IDENTITY NOT NULL PRIMARY KEY,
  login VARCHAR(20) NULL,
  senha VARCHAR(20) NULL
);
```

Análise e Conclusão

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

1X1 é quando uma linha de uma tabela se relaciona com apenas uma única linha de outra tabela e vice versa. 1XN é quando uma linha em uma tabela está relacionada a várias linhas em outra tabela, mas cada linha na segunda tabela está relacionada a apenas uma linha na primeira tabela. NxN é quando uma linha em uma tabela está relacionada a várias linhas em outra tabela e vise versa.

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

1XN

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

Ele oferece uma variedade de recursos que simplificam tarefas de gerenciamento, monitoramento e desenvolvimento com uma interface gráfica intuitiva que facilita o gerenciamento de bancos de dados.

Título da Prática: 2° Procedimento | Alimentando a Base

Consultas solicitadas

- Consultas:

```
-- Efetuar as seguintes consultas sobre os dados inseridos:

-- a - Dados completos de pessoas físicas.
SELECT  Pessoa,*,  PessoaFisica.cpf
FROM Pessoa
INNER JOIN PessoaFisica
    ON Pessoa.idPessoa = PessoaFisica.Pessoa_idPessoa
WHERE idpessoa = Pessoa_idPessoa

-- b - Dados completos de pessoas jurídicas.
SELECT Pessoa,*,  PessoaJuridica.cnpj
FROM Pessoa
INNER JOIN  PessoaJuridica
    ON Pessoa.idPessoa = PessoaJuridica.Pessoa_idPessoa
WHERE idPessoa = Pessoa_idPessoa


-- c - Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total
SELECT Produto.nome as produto,
    Pessoa.nome as fornecedor,
    Movimento.quantidade,
    Movimento.valorUnitario,
    (Movimento.quantidade * Movimento.valorUnitario) AS valor_total
FROM Movimento
INNER JOIN Pessoa
    ON Movimento.Pessoa_idPessoa = Pessoa.idPessoa
INNER JOIN Produto
    ON Movimento.Produto_idProduto = Produto.idProduto
WHERE Movimento.tipo = 'E';


-- d - Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total.
SELECT Produto.nome AS produto,
    Pessoa.nome AS comprador,
    Movimento.quantidade,
    Movimento.valorUnitario,
    (Movimento.quantidade * Movimento.valorUnitario) AS valor_total
FROM Movimento
INNER JOIN Pessoa
    ON Movimento.Pessoa_idPessoa = Pessoa.idPessoa
INNER JOIN Produto
    ON Movimento.Produto_idProduto = Produto.idProduto
WHERE Movimento.tipo = 'S';

-- e - Valor total das entradas agrupadas por produto.
SELECT SUM(Movimento.quantidade * Movimento.valorUnitario) as valor_total, Produto.nome as produto
FROM Movimento
INNER JOIN Produto ON Movimento.Produto_idProduto = Produto.idProduto
WHERE Movimento.tipo = 'E'
GROUP BY Produto.idProduto


-- e - Valor total das saídas agrupadas por produto.
SELECT SUM(Movimento.quantidade * Movimento.valorUnitario) as valor_total, Produto.nome as produto
FROM Movimento
INNER JOIN Produto ON Movimento.Produto_idProduto = Produto.idProduto
WHERE Movimento.tipo = 'S'
GROUP BY Produto.idProduto
```

```sql
-- g - Operadores que não efetuaram movimentações de entrada (compra).
SELECT Usuario.login AS operador
FROM Usuario
WHERE NOT EXISTS (
    SELECT 1
    FROM Movimento
    WHERE Movimento.Usuario_idUsuario = Usuario.idUsuario
    AND Movimento.tipo = 'E'
);


-- h - Valor total de entrada, agrupado por operador.
SELECT SUM(Movimento.quantidade * Movimento.valorUnitario) AS valor_total, Usuario.login AS operador
FROM Movimento
INNER JOIN Usuario
    ON Movimento.Usuario_idUsuario = Usuario.idUsuario
WHERE Movimento.tipo = 'E'
GROUP BY Usuario.idUsuario


-- i - Valor total de saída, agrupado por operador.
SELECT SUM(Movimento.quantidade * Movimento.valorUnitario) AS valor_total, Usuario.login AS operador
FROM Movimento
INNER JOIN Usuario
    ON Movimento.Usuario_idUsuario = Usuario.idUsuario
WHERE Movimento.tipo = 'S'
GROUP BY Usuario.idUsuario


-- j -Valor médio de venda por produto, utilizando média ponderada.
SELECT SUM(Movimento.quantidade * Movimento.valorUnitario)/SUM(Movimento.quantidade) AS media_ponderada, Produto.nome AS produto
FROM Movimento AS mov
INNER JOIN Produto AS pro ON Movimento.Produto_idProduto = Produto.idProduto
WHERE Movimento.tipo = 'S'
GROUP BY Produto.nome
```

Análise e Conclusão

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

A principal diferença entre eles é que o identity depende da tabela e o sequence é independente da tabela.

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

As chaves estrangeiras ajudam a evitar dados "órfãos" em que uma linha em uma tabela faz referência a uma linha inexistente em outra tabela. Permite crair relações 1x1, 1xN e NxN. Ajudam a garantir que operações como exclusão e updates não causem problemas de integridade referencial, garantindo a propagação das ações.

c) Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?

Algebra relacional: SELECT, WHERE, JOIN, UNION, EXCEPT, Cálculo relacional: Cálculo de Tupla e Cálculo de Domínio

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

é feito ultizando GROUP BY