Campus: 1197 - POLO CENTRO - SÃO LOURENÇO DA MATA - PE Curso: Desenvolvimento Full Stack - Graduação Tecnóloga Disciplina: RPG0014 - Iniciando o caminho pelo Java

Turma: 9001 Semestre: 2024.1 Matrícula: 2023.01.53256-6

Aluno: Gilvan Pereira de Oliveira

Repositório GitHub: [GilvanPOliveira/Loja (github.com)](https://github.com/GilvanPOliveira/Loja)

**Relatório discente de acompanhamento**

**1º Procedimento | Criando o Banco de Dados**

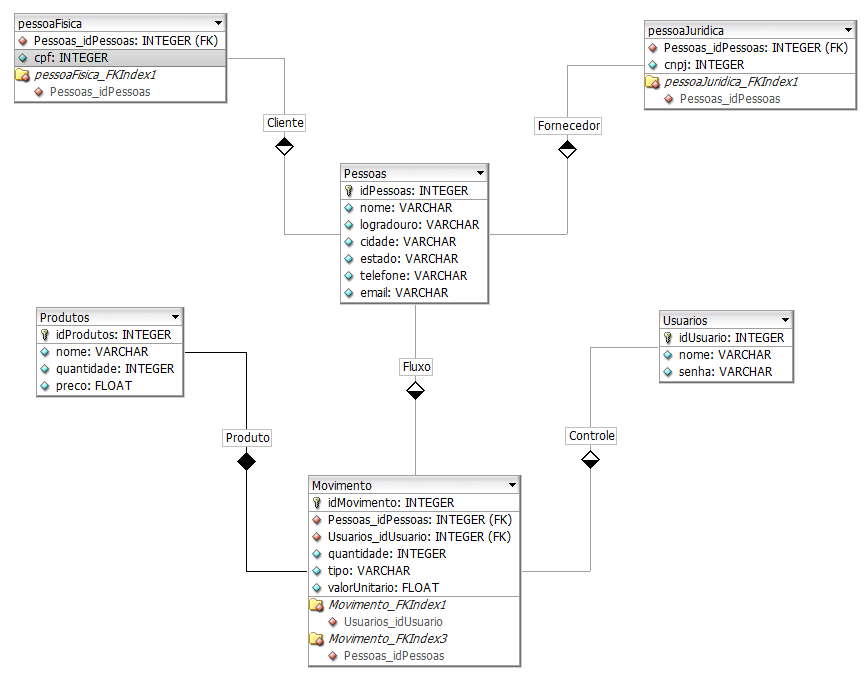
**Objetivo da prática:**

Desenvolver um banco de dados do zero, identificando os tipos de relacionamentos necessários por meio da modelagem e implementação, visando a criação de um sistema de vendas, englobando usuários, clientes, produtos e transações de compra e venda.

**Prática:**

* Utilizar o DBDesigner Fork para definir o modelo de dados do banco e seus relacionamentos;
* Utilizar o SQL Management Studio para criar o banco de dados.

**Códigos e resultados obtidos:**



Definindo um modelo de dados e criando o banco:

-- Criando o login "Loja" com senha "loja" e concedendo permissões necessárias

USE master;

GO

CREATE LOGIN Loja WITH PASSWORD = 'loja';

GO

ALTER SERVER ROLE sysadmin ADD MEMBER Loja;

GO

-- Criando o banco de dados

CREATE DATABASE Loja;

GO

USE Loja;

-- Criando as tabelas

-- Tabela Usuários

CREATE TABLE usuarios (

idUsuario INT PRIMARY KEY,

nome NVARCHAR(100),

senha NVARCHAR(50)

);

-- Tabela Pessoas

CREATE TABLE pessoas (

idPessoa INT PRIMARY KEY IDENTITY,

nome NVARCHAR(100),

logradouro NVARCHAR(200),

cidade NVARCHAR(50),

estado NVARCHAR(50),

telefone NVARCHAR(50),

email NVARCHAR(50)

);

-- Tabela pessoaFisica

CREATE TABLE pessoaFisica (

idPessoa INT PRIMARY KEY,

cpf NVARCHAR(14),

FOREIGN KEY (idPessoa) REFERENCES pessoas(idPessoa)

);

-- Tabela pessoaJuridica

CREATE TABLE pessoaJuridica (

idPessoa INT PRIMARY KEY,

cnpj NVARCHAR(18),

FOREIGN KEY (idPessoa) REFERENCES pessoas(idPessoa)

);

-- Tabela Produtos

CREATE TABLE produtos (

idProduto INT PRIMARY KEY,

nome NVARCHAR(100),

quantidade INT,

preco DECIMAL(10, 2)

);

-- Criação da tabela de Movimentos

CREATE TABLE movimento (

idMovimento INT PRIMARY KEY IDENTITY,

idUsuario INT,

idPessoa INT,

idProduto INT,

quantidade INT,

tipo VARCHAR(10), -- 'Entrada/Compra' ou 'Saida/Venda'

valorUnitario DECIMAL(10,2),

FOREIGN KEY (idUsuario) REFERENCES usuarios(idUsuario),

FOREIGN KEY (idProduto) REFERENCES produtos(idProduto),

FOREIGN KEY (idPessoa) REFERENCES pessoas(idPessoa)

);

-- Criando a sequência para geração dos identificadores de pessoa

CREATE SEQUENCE pessoaIDSeq

START WITH 1

INCREMENT BY 1;

**Conclusão:**

1. Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em

um banco de dados relacional?

Para 1x1, uma tabela contém uma chave estrangeira que referência a chave primária de outra tabela. Para 1xN, uma tabela contém uma chave estrangeira que referência a chave primária de outra tabela, permitindo que um registro se relacione com vários registros em outra tabela. E para NxN, é necessária uma tabela intermediária que contém pares de chaves estrangeiras para relacionar registros entre duas tabelas.

1. 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, usa-se o modelo de tabela única (ou tabela por classe). Uma tabela central contém atributos comuns a todas as entidades, enquanto tabelas secundárias (ou por subclasse) têm atributos específicos. Essas tabelas estão conectadas por chaves primárias e estrangeiras, garantindo integridade e coesão.

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

Gerenciando o banco de dados através de sua interface gráfica intuitiva, integrando outras ferramentas da Microsoft como visual Studio, possui um editor SQL robusto, dentre outras qualidades

**2º Procedimento | Alimentando a Base**

**Objetivo da prática:**

Utilizar o SQL Server Management Studio para operar em um banco de dados, abrangendo desde inserção de dados e criação de entidades até consultas mais complexas, alimentando tabelas e executando scripts.

**Prática:**

* Inserir dados básicos do sistema no banco de dados;
* Criar movimentações na base de dados;
* Efetuar consultas sobre os dados inseridos.

**Códigos e resultados obtidos:**

Inserindo os dados para alimentar o banco:

-- Inserção de usuários

INSERT INTO usuarios (idUsuario, nome, senha) VALUES

(1, 'op1', 'op1'),

(2, 'op2', 'op2');

-- Inserção de produtos

INSERT INTO produtos (idProduto, nome, quantidade, preco) VALUES

(1, 'Banana', 100, 5.00),

(3, 'Laranja', 500, 2.00),

(4, 'Manga', 800, 4.00);

-- Inserção de pessoas físicas

INSERT INTO pessoas (nome, logradouro, cidade, estado, telefone, email) VALUES

('Joao', 'Rua 12, casa 3, Quitanda', 'Riacho do Sul', 'PA', '1111-1111', 'joao@riacho.com');

INSERT INTO pessoaFisica (idPessoa, cpf) VALUES

(SCOPE\_IDENTITY(), '11111111111');

-- Inserção de pessoas jurídicas

INSERT INTO pessoas (nome, logradouro, cidade, estado, telefone, email) VALUES

('JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '1212-1212', 'jjc@riacho.com');

INSERT INTO pessoaJuridica (idPessoa, cnpj) VALUES

(SCOPE\_IDENTITY(), '22222222222222');

-- Inserção de movimentações

INSERT INTO movimento (idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario) VALUES

(1, 1, 1, 20, 'S', 4.00), -- Venda de 20 bananas pelo usuário 1 à pessoa física 1

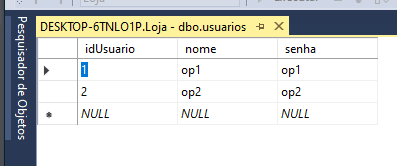
(1, 1, 3, 15, 'S', 2.00), -- Venda de 15 laranjas pelo usuário 1 à pessoa física 1

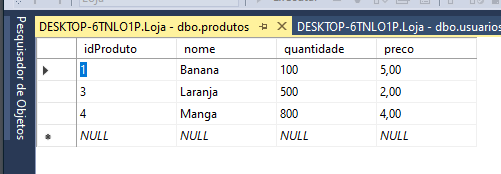
(2, 2, 3, 10, 'S', 3.00), -- Venda de 10 laranjas pelo usuário 2 à pessoa jurídica 2

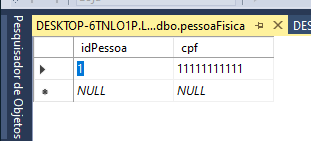
(1, 2, 3, 15, 'E', 5.00), -- Compra de 15 laranjas pelo usuário 1 da pessoa jurídica 2

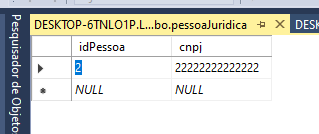
(1, 2, 4, 20, 'E', 4.00); -- Compra de 20 mangas pelo usuário 1 da pessoa jurídica 2

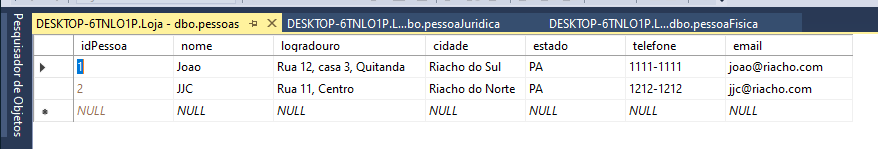
Resultados das consultas

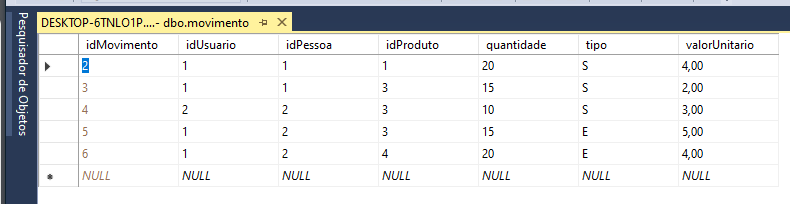
****

****

****

****

****

****

Consultas:

-- Consulta: Dados completos de pessoas físicas

SELECT \*

FROM pessoas

INNER JOIN pessoaFisica ON pessoas.idPessoa = pessoaFisica.idPessoa;

-- Consulta: Dados completos de pessoas jurídicas

SELECT \*

FROM pessoas

INNER JOIN pessoaJuridica ON pessoas.idPessoa = pessoaJuridica.idPessoa;

-- Consulta: Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total

SELECT p.nome AS produto, pf.nome AS fornecedor, m.quantidade, m.valorUnitario, (m.quantidade \* m.valorUnitario) AS valor\_total

FROM movimento m

JOIN produtos p ON m.idProduto = p.idProduto

JOIN pessoas pf ON m.idPessoa = pf.idPessoa

WHERE m.tipo = 'E';

-- Consulta: Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total

SELECT p.nome AS produto, pc.nome AS comprador, m.quantidade, m.valorUnitario, (m.quantidade \* m.valorUnitario) AS valor\_total

FROM movimento m

JOIN produtos p ON m.idProduto = p.idProduto

JOIN pessoas pc ON m.idPessoa = pc.idPessoa

WHERE m.tipo = 'S';

-- Consulta: Valor total das entradas agrupadas por produto

SELECT p.nome AS produto, SUM(m.quantidade \* m.valorUnitario) AS valor\_total\_entrada

FROM movimento m

JOIN produtos p ON m.idProduto = p.idProduto

WHERE m.tipo = 'E'

GROUP BY p.nome;

-- Consulta: Valor total das saídas agrupadas por produto

SELECT p.nome AS produto, SUM(m.quantidade \* m.valorUnitario) AS valor\_total\_saida

FROM movimento m

JOIN produtos p ON m.idProduto = p.idProduto

WHERE m.tipo = 'S'

GROUP BY p.nome;

-- Consulta: Operadores que não efetuaram movimentações de entrada (compra)

SELECT u.nome AS operador

FROM usuarios u

LEFT JOIN movimento m ON u.idUsuario = m.idUsuario

WHERE m.idMovimento IS NULL;

-- Consulta: Valor total de entrada, agrupado por operador

SELECT u.nome AS operador, SUM(m.quantidade \* m.valorUnitario) AS valor\_total\_entrada

FROM movimento m

JOIN usuarios u ON m.idUsuario = u.idUsuario

WHERE m.tipo = 'E'

GROUP BY u.nome;

-- Consulta: Valor total de saída, agrupado por operador

SELECT u.nome AS operador, SUM(m.quantidade \* m.valorUnitario) AS valor\_total\_saida

FROM movimento m

JOIN usuarios u ON m.idUsuario = u.idUsuario

WHERE m.tipo = 'S'

GROUP BY u.nome;

-- Consulta: Valor médio de venda por produto, utilizando média ponderada

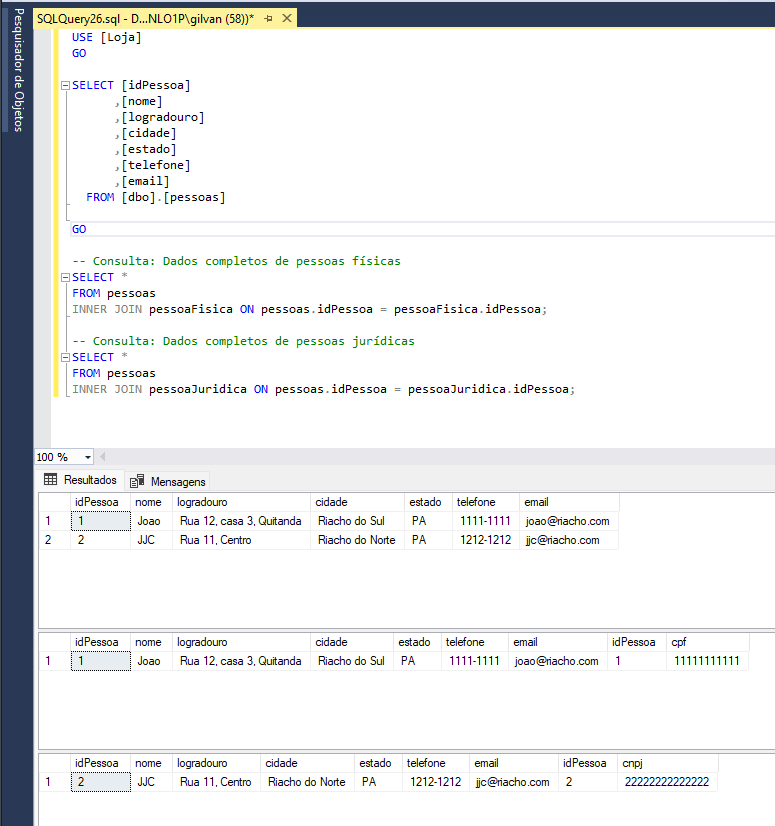
SELECT p.nome AS produto, SUM(m.quantidade \* m.valorUnitario) / SUM(m.quantidade) AS valor\_medio\_venda

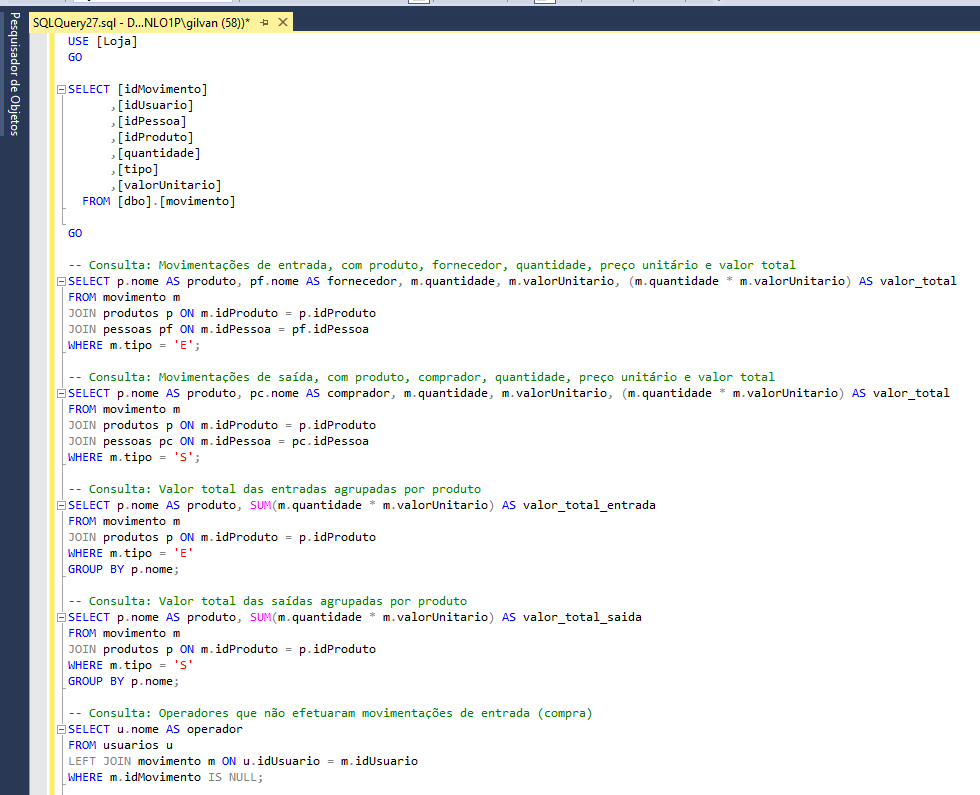
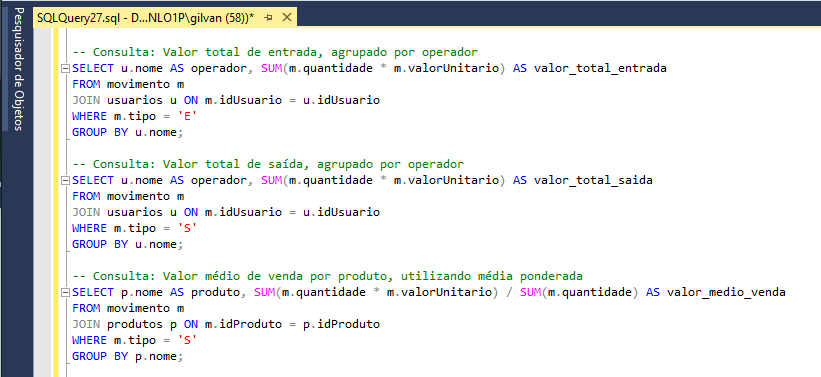
FROM movimento m

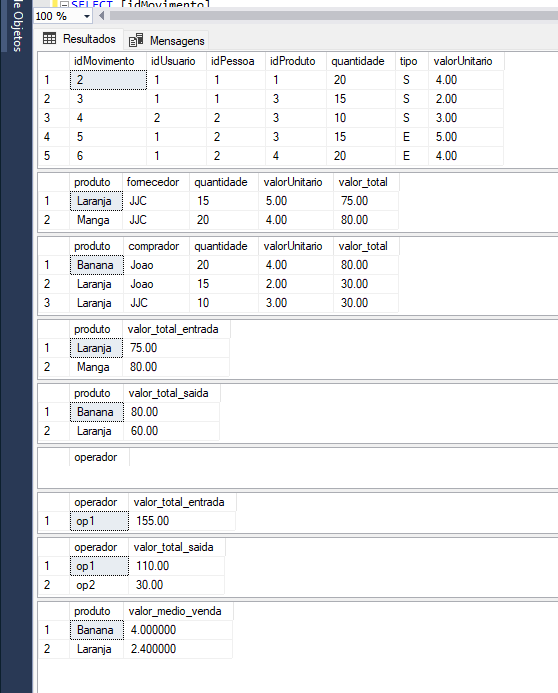
JOIN produtos p ON m.idProduto = p.idProduto

WHERE m.tipo = 'S'

GROUP BY p.nome;

****

**  
**

****

**Conclusão:**

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

**Sequence** é usada para gerar valores únicos automaticamente em uma coluna, enquanto **identity** é um atributo de coluna usados para gerar automaticamente valores únicos para nova linha inserida.

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

Chaves estrangeiras garantem a integridade referencial entre tabelas, mantendo a consistência dos dados, evitando referências a registros inexistentes.

1. Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?

Operadores como **SELECT**, **PROJECT**, **JOIN**, **UNION**, **INTERSECT**, **DIFFERENCE** pertencem à álgebra relacional, enquanto operadores como **FORALL**, **EXISTS**, **IN** são definidos no cálculo relacional.

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

O agrupamento em consultas é feito usando a cláusula **GROUP** **BY**, onde os resultados são agrupados com base nos valores de uma ou mais colunas. É obrigatório incluir uma função de agregação, como **SUM**, **COUNT**, **AVG**, etc., em colunas que não estão incluídas na cláusula **GROUP** **BY**.