Estácio

UNIVERSIDADE ESTACIO DE SÁ

POLO TAMBIÁ

DESENVOLVIMENTO FULL STACK

2024.1 FULL STACK

NIVEL 2: VAMOS MANTER OA INFORMAÇÕES!
MUNDO: 3

FRANCINALDO SOUZA BERNARDINO

Centro Universitário Estácio de Sá – Paraíba

Polo; Joao Pessoa - Tambiá

Curso: Desenvolvimento Full Stack

Disciplina: RPG0015 **Nível 2:**Vamos manter as informações! **Mundo:** 3

Número da Turma: 9003

Semestre Letivo: 3º

Aluno: Francinaldo Souza Bernardino

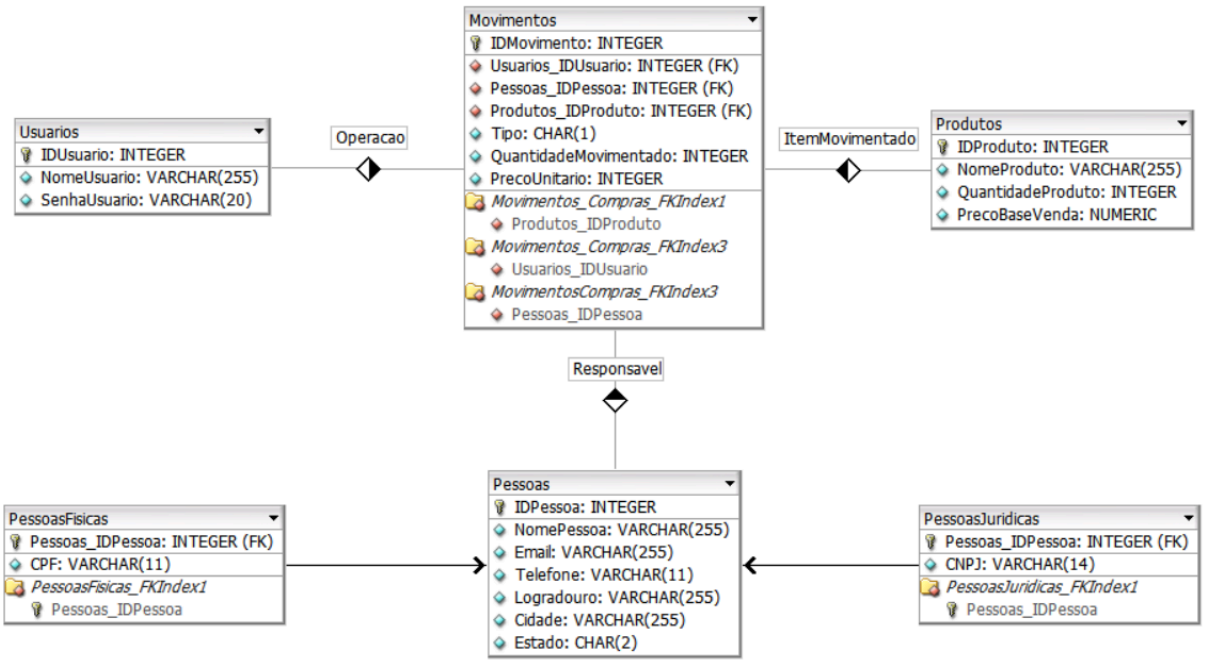
Repositorio Git: https://github.com/Anubyhs/N2M3

- **Objetivo da prática**

Utilizar as ferramentas DBDesigner e Microsoft SQL Server Management Studio (SSMS) para implementar um banco de dados relacional simples. O objetivo é identificar os requisitos do sistema, construir um diagrama entidade-relacionamento (DER), criar as estruturas do banco de dados usando a linguagem SQL (DDL - Data Definition Language / Linguagem de Definição de Dados) e realizar consultas e manipulações de dados (DML - Data Manipulation Language / Linguagem de Manipulação de Dados).

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

Com o DBDesigner foi modelado o seguinte diagrama entidade-relacionamento:



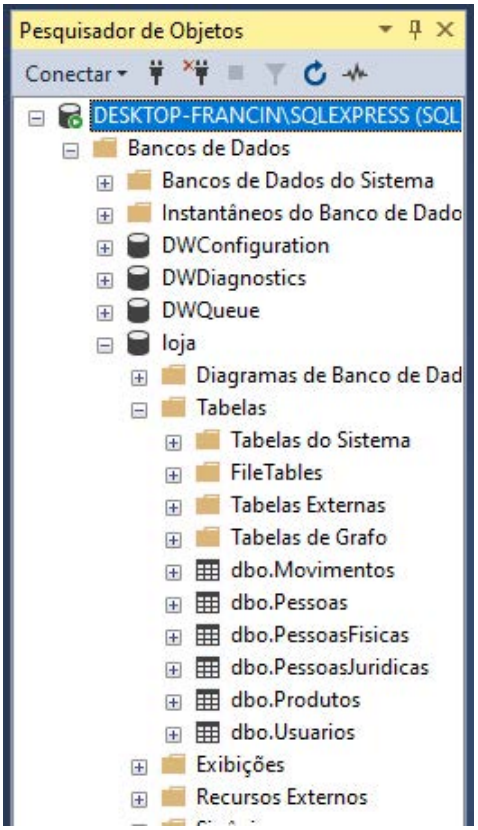
Com o SQL Server o seguinte script para a criação do banco de dados (DDL):

```
--Criando o banco de dados loja
CREATE DATABASE loja;
GO

--Usando o banco de dados loja
USE loja;
GO

--Criando uma sequence para ID de Pessoa
CREATE SEQUENCE Sequencia_IDPessoa
START WITH 1
INCREMENT BY 1;
GO

--Criando tabela Pessoas
CREATE TABLE Pessoas (
	IDPessoa INTEGER NOT NULL DEFAULT NEXT VALUE FOR
Sequencia_IDPessoa,
	NomePessoa VARCHAR(255) NOT NULL,
	Email VARCHAR(255) NOT NULL,
	Telefone VARCHAR(11) NOT NULL,
```

```
        Logradouro VARCHAR(255) NOT NULL,
        Cidade VARCHAR(255) NOT NULL,
        Estado CHAR(2) NOT NULL,
        PRIMARY KEY(IDPessoa)
);
GO

--Criando tabela PessoasFisicas
CREATE TABLE PessoasFisicas (
        Pessoas_IDPessoa INTEGER NOT NULL,
        CPF VARCHAR(11) NOT NULL UNIQUE,
        PRIMARY KEY(Pessoas_IDPessoa),
        FOREIGN KEY(Pessoas_IDPessoa) REFERENCES Pessoas(IDPessoa)
);
GO

--Criando tabela PessoasJuridicas
CREATE TABLE PessoasJuridicas (
        Pessoas_IDPessoa INTEGER NOT NULL,
        CNPJ VARCHAR(14) NOT NULL UNIQUE,
        PRIMARY KEY(Pessoas_IDPessoa),
        FOREIGN KEY(Pessoas_IDPessoa) REFERENCES Pessoas(IDPessoa)
);
GO

--Criando tabela Usuarios
CREATE TABLE Usuarios (
        IDUsuario INTEGER NOT NULL IDENTITY,
        NomeUsuario VARCHAR(255) NOT NULL,
        SenhaUsuario VARCHAR(20) NOT NULL,
        PRIMARY KEY(IDUsuario)
);
GO

--Criando tabela Produtos
CREATE TABLE Produtos (
        IDProduto INTEGER NOT NULL IDENTITY,
        NomeProduto VARCHAR(255) NOT NULL,
        QuantidadeProduto INTEGER NOT NULL,
        PrecoVendaBase NUMERIC(6,2) NOT NULL,
        PRIMARY KEY(IDProduto)
);
GO

--Criando tabela Movimentos
CREATE TABLE Movimentos (
        IDMovimento INTEGER NOT NULL IDENTITY,
        Usuarios_IDUsuario INTEGER NOT NULL,
```

```
        Pessoas_IDPessoa INTEGER NOT NULL,
        Produtos_IDProduto INTEGER NOT NULL,
        Tipo CHAR(1) NOT NULL,
        QuantidadeMovimentado INTEGER NOT NULL,
        PrecoUnitario NUMERIC(6,2) NOT NULL,
        PRIMARY KEY (IDMovimento),
        FOREIGN KEY (Usuarios_IDUsuario) REFERENCES Usuarios(IDUsuario),
        FOREIGN KEY (Pessoas_IDPessoa) REFERENCES Pessoas(IDPessoa),
        FOREIGN KEY (Produtos_IDProduto) REFERENCES Produtos(IDProduto)
);
GO
```

Tendo como resultado a seguinte estrutura no Pesquisador de Objetos do SQL Server:



- **Análise e conclusão | Procedimento 1**

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

Em um banco de dados relacionais, as diferentes cardinalidades definem o tipo de diferentes tabelas (como planilhas) em um banco de dados se conectam usando "chaves" (como identificações únicas). Existem três tipos principais de conexões, chamadas de cardinalidades:

Um para Um (1X1): Cada item em uma tabela está ligado a apenas um item em outra (como uma pessoa e seu perfil único).
Um para Muitos (1XN): Um item em uma tabela pode estar ligado a vários itens em outra (como um usuário e seus vários pedidos).

Muitos para Muitos (NXN): Vários itens em uma tabela podem estar ligados a vários itens em outra. Isso geralmente precisa de uma tabela extra para fazer a ligação (como alunos e cursos através de uma lista de inscrições).
Essas conexões ajudam a organizar e relacionar as informações no banco de dados.

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

Existem duas formas principais de organizar informações de uma categoria geral e suas subcategorias em um banco de dados (como planilhas):

Tudo em uma planilha: Junta todas as características da categoria geral e das subcategorias em uma única planilha. O problema é que se as subcategorias tiverem muitas características próprias, essa planilha pode ter muitas colunas vazias, tornando-se ineficiente.

Planilhas separadas: Cria uma planilha para a categoria geral (com informações comuns a todas) e planilhas separadas para cada subcategoria (com informações específicas). Essas planilhas são conectadas por uma "identificação única" em comum. Essa forma é mais organizada e evita colunas vazias.

C) 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) aumenta a produtividade ao organizar tudo em um só lugar, facilitando encontrar e acessar objetos do banco de dados.
Acelerar a escrita de código, oferecendo ajuda inteligente para escrever consultas (IntelliSense) e identificar erros.
Simplificar tarefas complexas, pois possui muitas ferramentas visuais e um bom assistente para backups, configurações e outras interações necessárias.
Ajudar a encontrar problemas, permitindo monitorar o desempenho e depurar os erros no código do banco de dados.
Automatizar tarefas rotineiras, de forma a permitir o agendamento de tarefas para serem executadas automaticamente.
Em poucas palavras, o SSMS torna o gerenciamento do SQL Server mais fácil, rápido e eficiente.

- **2º Procedimento | Alimentando a Base**

O script abaixo alimenta a base de dados (DML):

```
USE loja;
GO

--Inserindo usuarios
INSERT INTO Usuarios (NomeUsuario, SenhaUsuario)
VALUES ('op1', 'op1'),
        ('op2', 'op2'),
        ('op3', 'op3'),
        ('op4', 'op4');
GO

--Inserindo produtos
INSERT INTO Produtos (NomeProduto, QuantidadeProduto, PrecoVendaBase)
VALUES ('Banana', 100, 5.49),
        ('Laranja', 200, 2.99),
        ('Manga', 300, 4.49),
        ('Mortadela', 50, 8.50),
        ('Queijo Coalho', 75, 29.99);
GO

--Inserindo pessoas
INSERT INTO Pessoas (IDPessoa, NomePessoa, Email, Telefone, Logradouro, Cidade,
Estado)
VALUES (NEXT VALUE FOR Sequencia_IDPessoa, ''Francinaldo Souza',
'lee.frankcinaldo@imail.com', '12458956237', 'Rua da areia, 105', 'Joao Pessoa', 'PB'),
        (NEXT VALUE FOR Sequencia_IDPessoa, 'Wilson junior', 'wilson.ar@sesmt.com',
'11955544555', 'Rua dos coelhos, 300', 'Coelhos', 'PE'),
(NEXT VALUE FOR Sequencia_IDPessoa, 'Feira Sao Jose', 'van_coteminas@email.com',
                '45123974455', 'BR 230, KM 85', 'Joao Pessoa', 'PB'),
        (NEXT VALUE FOR Sequencia_IDPessoa, 'Panificadora Oya',
'pan_oya@email.com', '98653245893', 'Av cidade da jurema, S/N', 'Arruanda', 'AM');
GO

--Inserindo CPF de pessoas fisicas
INSERT INTO PessoasFisicas (Pessoas_IDPessoa, CPF)
VALUES (1, '12121313910'),
        (2, '54335499206');
GO

--Inserindo CNPJ de pessoas juridicas
INSERT INTO PessoasJuridicas (Pessoas_IDPessoa, CNPJ)
VALUES (3, '55884466000001'),
        (4, '12568905100001');
GO
```

```
--Inserindo movimentacao
INSERT INTO Movimentos (Usuarios_IDUsuario, Pessoas_IDPessoa, Produtos_IDProduto,
Tipo, QuantidadeMovimentado, PrecoUnitario)
VALUES (4, 3, 2, 'E', 27, 2.99),
        (4, 1, 5, 'S', 3, 29.99),
        (4, 3, 1, 'E', 15, 5.49),
        (1, 2, 4, 'S', 5, 8.50),
        (1, 1, 3, 'S', 2, 4.49),
        (1, 4, 5, 'E', 10, 29.99),
        (2, 2, 2, 'S', 15, 2.99);
GO
```

Para consultas aos dados o script a seguir (DML):

```
USE loja;
GO

--Dados completos de pessoas fisicas
SELECT p.*, pf.CPF
FROM Pessoas p
INNER JOIN PessoasFisicas pf ON p.IDPessoa = pf.Pessoas_IDPessoa
ORDER BY p.IDPessoa ASC;
GO

--Dados completos de pessoas juridicas
SELECT p.*, pj.CNPJ
FROM Pessoas p
INNER JOIN PessoasJuridicas pj ON p.IDPessoa = pj.Pessoas_IDPessoa
ORDER BY p.IDPessoa ASC;
GO

--Movimentos de entrada (E)
SELECT m.IDMovimento,
        p.NomePessoa AS Fornecedor,
        prod.NomeProduto,
        m.QuantidadeMovimentado,
        m.PrecoUnitario,
        (m.QuantidadeMovimentado * m.PrecoUnitario) AS ValorTotal
FROM Movimentos m
INNER JOIN Pessoas p ON m.Pessoas_IDPessoa = p.IDPessoa
INNER JOIN PessoasJuridicas pj ON p.IDPessoa = pj.Pessoas_IDPessoa
INNER JOIN Produtos prod ON m.Produtos_IDProduto = prod.IDProduto
WHERE m.Tipo = 'E'
ORDER BY m.IDMovimento ASC;
GO
```

```sql
--Movimentos de saida (S)
SELECT m.IDMovimento,
        p.NomePessoa AS Comprador,
        prod.NomeProduto,
        m.QuantidadeMovimentado,
        m.PrecoUnitario,
        (m.QuantidadeMovimentado * m.PrecoUnitario) AS ValorTotal
FROM Movimentos m
INNER JOIN Pessoas p ON m.Pessoas_IDPessoa = p.IDPessoa
INNER JOIN PessoasFisicas pf ON p.IDPessoa = pf.Pessoas_IDPessoa
INNER JOIN Produtos prod ON m.Produtos_IDProduto = prod.IDProduto
WHERE m.Tipo = 'S'
ORDER BY m.IDMovimento ASC;
GO

--Entrada total por produto (E)
SELECT prod.NomeProduto,
        SUM (m.QuantidadeMovimentado * m.PrecoUnitario) AS ValorTotalEntradas
FROM Movimentos m
INNER JOIN Produtos prod ON m.Produtos_IDProduto = prod.IDProduto
WHERE m.Tipo = 'E'
GROUP BY prod.NomeProduto
ORDER BY ValorTotalEntradas DESC;
GO

--Saida total por produto (S)
SELECT prod.NomeProduto,
        SUM (m.QuantidadeMovimentado * m.PrecoUnitario) AS ValorTotalSaidas
FROM Movimentos m
INNER JOIN Produtos prod ON m.Produtos_IDProduto = prod.IDProduto
WHERE m.Tipo = 'S'
GROUP BY prod.NomeProduto
ORDER BY ValorTotalSaidas DESC;
GO

--Usuarios sem entradas feitas (E)
SELECT u.IDUsuario, u.NomeUsuario
FROM Usuarios u
LEFT JOIN Movimentos m ON u.IDUsuario = m.Usuarios_IDUsuario AND m.Tipo = 'E'
WHERE m.IDMovimento IS NULL;
GO

--Total de entradas por usuario (E)
SELECT u.IDUsuario, u.NomeUsuario,
        SUM (m.QuantidadeMovimentado * m.PrecoUnitario) AS ValorTotalEntradas
FROM Movimentos m
INNER JOIN Usuarios u ON m.Usuarios_IDUsuario = u.IDUsuario
```

```
WHERE m.Tipo = 'E'
GROUP BY u.IDUsuario, u.NomeUsuario
ORDER BY ValorTotalEntradas DESC;
GO

--Total de saidas por usuario (S)
SELECT u.IDUsuario, u.NomeUsuario,
        SUM (m.QuantidadeMovimentado * m.PrecoUnitario) AS ValorTotalSaidas
FROM Movimentos m
INNER JOIN Usuarios u ON m.Usuarios_IDUsuario = u.IDUsuario
WHERE m.Tipo = 'S'
GROUP BY u.IDUsuario, u.NomeUsuario
ORDER BY  ValorTotalSaidas DESC;
GO

--Valor medio de venda (S)
SELECT p.IDProduto, p.NomeProduto,
        CAST (SUM (m.PrecoUnitario * m.QuantidadeMovimentado) / NULLIF (SUM
(m.QuantidadeMovimentado), 0) AS DECIMAL (10, 2)) AS MediaPonderadaVendas
FROM Movimentos m
INNER JOIN  Produtos p ON m.Produtos_IDProduto = p.IDProduto
WHERE m.Tipo = 'S'
GROUP BY p.IDProduto, p.NomeProduto
ORDER BY MediaPonderadaVendas DESC;
GO
```

Capturas de tela dos resultados das consultas:

- Consulta de pessoas físicas.

	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CPF
1	1	Francinaldo Souza	lee.frankcinaldo@imail.com	12458956237	Rua da areia, 105	Joao Pessoa	PB	12121313910
2	2	Wilson junior	wilson.ar@sesmt.com	11955544555	Rua dos coelhos, 300	Coelhos	PE	54335499206

- Consulta de pessoas jurídicas.

	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CNPJ
1	3	Feira Sao Jose	van_coteminas@email.com	45123974455	BR 230, KM 85	Joao Pessoa	PB	55884466000001
2	4	Panificadora Oya	pan_oya@email.com	98653245893	Av cidade da jurema, S/N	Arruanda	AM	12568905100001

- Consulta de entradas.

	IDMovimento	Fornecedor	NomeProduto	QuantidadeMovimentado	PrecoUnitario	ValorTotal
1	1	Feira Sao Jose	Laranja	27	2.99	80.73
2	3	Feira Sao Jose	Banana	15	5.49	82.35
3	6	Panificadora Oya	Queijo Coalho	10	29.99	299.90

- Consulta de saídas.

	IDMovimento	Comprador	NomeProduto	QuantidadeMovimentado	PrecoUnitario	ValorTotal
1	2	Francinaldo Souza	Queijo Coalho	3	29.99	89.97
2	4	Wilson junior	Mortadela	5	8.50	42.50
3	5	Francinaldo Souza	Manga	2	4.49	8.98
4	7	Wilson junior	Laranja	15	2.99	44.85

- Consulta do valor total de entrada por produto.

	NomeProduto	ValorTotalEntradas
1	Queijo Coalho	299.90
2	Banana	82.35
3	Laranja	80.73

- Consulta do valor total de saída por produto.

	NomeProduto	ValorTotalSaidas
1	Queijo Coalho	89.97
2	Laranja	44.85
3	Mortadela	42.50
4	Manga	8.98

- Consulta dos usuários que não fizeram entradas.

	IDUsuario	NomeUsuario
1	2	op2
2	3	op3

- Consulta do total de entradas por usuários.

	IDUsuario	NomeUsuario	ValorTotalEntradas
1	1	op1	299.90
2	4	op4	163.08

- Consulta do total de saídas por usuários.

	IDUsuario	NomeUsuario	ValorTotalSaidas
1	4	op4	89.97
2	1	op1	51.48
3	2	op2	44.85

- Consulta do valor médio de vendas por produto.

	IDProduto	NomeProduto	MediaPonderadaVendas
1	5	Queijo Coalho	29.99
2	4	Mortadela	8.50
3	3	Manga	4.49
4	2	Laranja	2.99

- **Análise e conclusão | Procedimento 2**

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

O uso de Identity é diretamente na definição de uma coluna de uma tabela, especificando que a coluna terá valores numéricos gerados de forma automática a cada linha inserida, restringindo e centralizando essa ação apenas para aquela coluna daquela tabela em que é declarado.

Sequence é um objeto separado das tabelas, que gera números sequenciais e que pode ser usado por diversas tabelas, conseguindo mais flexibilidade, permitindo que use a mesma sequência para diferentes contextos com controle do valor gerado, sem estar associado a uma tabela específica.

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

As chaves estrangeiras são importantes para garantir a integridade de referência no banco de dados, dando relacionamentos consistentes entre as tabelas, permitindo que valores em uma tabela sejam referenciados de forma correta para uma linha de outra tabela, impedindo registros órfãos ou inválidos.

Como exemplo, temos a tabela de **Movimentos** que contém chaves estrangeiras para as tabelas **Produtos**, **Pessoas** e **Usuarios**, sem essas referências, a base de dados poderia permitir a inserção de movimentações com produtos ou pessoas que não existem no banco de dados, afetando a integridade e consistência dos dados.

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

Alguns exemplos desses operadores do SQL, que pertencem à álgebra relacional, são os **SELECT**, **JOIN**, **UNION**, **INTERSECT** e **EXCEPT**.

Para os cálculos relacionais são, por exemplo, **AND**, **OR**, **NOT**, **EXISTS**, **ALL** e **ANY.**

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

O agrupamento em consultas é feito com a cláusula **GROUP BY**, que agrupa linhas com valores em comum em uma ou mais colunas, sendo possível realizar cálculos como somas, médias, contagens, entre outros.

O principal requisito obrigatório é que todas as colunas na cláusula **SELECT** precisam estar presentes na cláusula **GROUP BY** ou serem usadas em funções de agregação.