Campus:
Norte Shopping – Av. Pres. Vargas 2560, Rio de Janeiro, RJ, 20210-031

Curso: Desenvolvimento Full Stack

Disciplina: RPG0015 – Vamos Manter as Informações?

Turma: 9001

Semestre Letivo: 2024.3

Aluno: Gabriel Bernardo Carneiro **Matrícula:** 202308953541

Link do repositório GitHub:
https://github.com/GbDev1907/missao2Mundo3.git

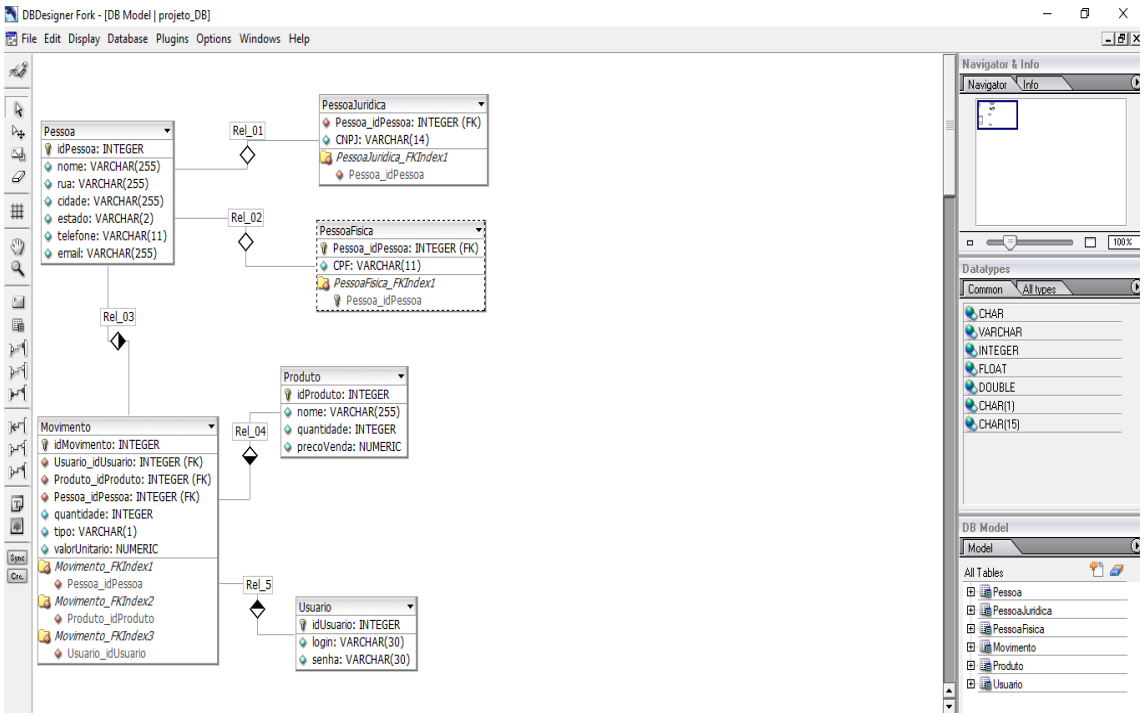
Título da Prática

RPG0015 - Vamos manter as informações!

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
2. adequado.
3. Utilizar ferramentas de modelagem para bases de dados relacionais.
4. Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
5. Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
6. No final do exercício, o aluno terá vivenciado a experiência de modelar a base
7. de dados para um sistema simples, além de implementá-la, através da
8. sintaxe SQL, na plataforma do SQL Server.



1º Procedimento | Criando o Banco de Dados

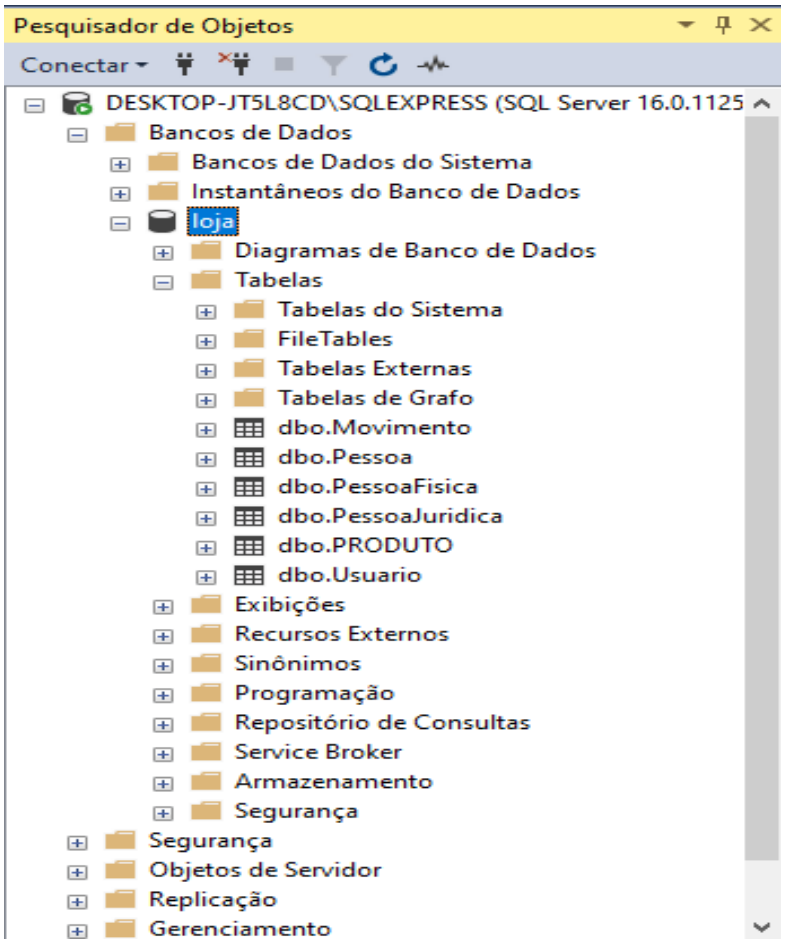
Códigos solicitado:

```
USE loja;


CREATE TABLE Pessoa (
idPessoa INTEGER NOT NULL IDENTITY(1,1)PRIMARY KEY,
nome VARCHAR(255) NOT NULL,
rua VARCHAR(255) NOT NULL,
cidade VARCHAR(255) NOT NULL,
estado CHAR(2) NOT NULL,
telefone VARCHAR(11) NOT NULL,
email VARCHAR(255) NOT NULL
);


CREATE TABLE PessoaJuridica (
idPessoa INTEGER NOT NULL PRIMARY KEY,
CNPJ VARCHAR(14) NOT NULL,
CONSTRAINT fk_PessoaJuridica_Pessoa
```

```
        FOREIGN KEY (idPessoa)
        REFERENCES dbo.Pessoa(idPessoa)
);

CREATE TABLE PessoaFisica (
idPessoa INTEGER NOT NULL PRIMARY KEY,
CPF VARCHAR(11) NOT NULL,
CONSTRAINT fk_PessoaFisica_Pessoa
        FOREIGN KEY (idPessoa)
        REFERENCES Pessoa(idPessoa)
);

CREATE TABLE PRODUTO (
idProduto INTEGER NOT NULL IDENTITY(1,1) PRIMARY KEY,
nome VARCHAR(255) NOT NULL,
quantidade INTEGER NOT NULL,
precoVenda NUMERIC(10, 2) NOT NULL
);

CREATE TABLE Usuario (
idUsuario INTEGER NOT NULL IDENTITY (1,1) PRIMARY KEY,
login VARCHAR(50) NOT NULL,
senha VARCHAR(50) NOT NULL
);

CREATE TABLE Movimento (
idMovimento INTEGER NOT NULL IDENTITY (1,1) PRIMARY KEY,
idUsuario INTEGER NOT NULL,
idPessoa INTEGER NOT NULL,
idProduto INTEGER NOT NULL,
quantidade INTEGER NOT NULL,
tipo CHAR(1) NOT NULL,
valorUnitario FLOAT NOT NULL,
```

```
CONSTRAINT fk_Movimento_Produto FOREIGN KEY (idProduto) REFERENCES
dbo.Produto(idProduto),

CONSTRAINT fk_Movimento_Usuario FOREIGN KEY (idUsuario) REFERENCES
dbo.Usuario(idUsuario),

CONSTRAINT fk_Movimento_Pessoa FOREIGN KEY (idPessoa) REFERENCES
dbo.Pessoa(idPessoa)

);


CREATE SEQUENCE dbo.CodigoPessoa

START WITH 1

INCREMENT BY 1;
```

Resultado:



Análise e Conclusão:

1) **Como são implementadas as diferentes cardinalidades, basicamente 1x1, 1XN ou NXN, em um banco de dados relacional?**

Relação 1x1: Duas tabelas com uma chave primária em uma delas que também é uma chave estrangeira na outra.

Relação 1xN: Uma tabela com uma chave primária que é referenciada por uma chave estrangeira em outra tabela.

Relação NxN: Duas tabelas com uma tabela intermediária (ou de junção) que contém chaves estrangeiras referenciando ambas as tabelas.

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

- Tabela por subclasse (Subclass Table):

Cada classe tem uma tabela seperada, onde a tabela da subclasse contém uma chave primária que é também chave estrangeira da superclasse. Vantagem: Permite armazenamento de atributos específicos de cada subclasse

- Tabela por Superclasse (Superclass Table):

Uma única tabela para todas as classes, com um campo discriminador para identificar a subclasse. Vantagem: Estrutura simples, mas pode ter muitos campos nulos.

- Tabela por Composição (Class Table):

A superclasse tem uma tabela e cada subclasse tem sua própria tabela, sem chaves estrangeiras. Vantagem: Estrutura separada e fácil adição de novos tipos.

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

O SSMS melhora a produtividade ao fornecer uma interface intuitiva, assistentes úteis, um editor de consultas avançado e ferramentas de monitoramento, permitindo que os administradores de banco de dados realizem tarefas de forma mais eficiente e eficaz.

2º Procedimento | Alimentando a Base

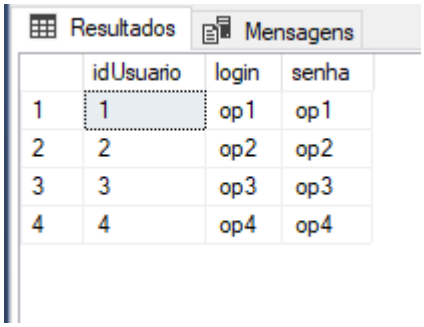
Códigos solicitado:

```
use loja


INSERT INTO Usuario (login, senha)
VALUES
('op1','op1'),
('op2','op2'),
('op3','op3'),
('op4','op4');


SELECT * FROM Usuario;


INSERT INTO Produto (nome, quantidade, precoVenda)
VALUES
('banana', 100, 5.0),
('laranja', 500, 2.0),
('Manga', 800, 4.0),
('Pera', 300, 6.0),
('Abacaxi', 300, 3.0);


SELECT * FROM Produto;


INSERT INTO Pessoa (nome, rua, cidade, estado, telefone, email)
VALUES
('Joao', 'Rua 12,casa3,Quintanda', 'Riacho do Sul', 'PA','11111111', 'joao@riacho.com'),
('JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '12121212', 'jjc@riacho.com');


SELECT * FROM Pessoa;


INSERT INTO PessoaFisica (idPessoa, cpf)
```

```
VALUES
(1, '11111111111');

SELECT * FROM PessoaFisica;

INSERT INTO PessoaJuridica (idPessoa, cnpj)
VALUES
(2, '22222222222');

SELECT * FROM PessoaJuridica;

INSERT INTO Movimento (idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario)
VALUES
(1, 1, 1, 10, 'E', 3.0),
(1, 1, 2, 10, 'E', 1.0),
(1, 1, 3, 10, 'E', 3.0),
(2, 2, 1, 3, 'S', 5.0),
(2, 2, 2, 1, 'S', 2.0),
(2, 2, 3, 8, 'S', 4.0),
(3, 2, 1, 5, 'E', 4.75),
(3, 2, 2, 7, 'E', 1.32),
(3, 1, 1, 9, 'S', 5.0),
(3, 1, 2, 3, 'S', 2.0),
(4,2, 1, 8, 'S', 5.0),
(4, 2, 2, 6, 'S', 2.0);

SELECT * FROM Movimento;
```

Resultados:

Usuario



Resultados | Mensagens

	idUsuario	login	senha
1	1	op1	op1
2	2	op2	op2
3	3	op3	op3
4	4	op4	op4

Produto

Resultados | Mensagens

	idProduto	nome	quantidade	precoVenda
1	1	banana	100	5.00
2	2	laranja	500	2.00
3	3	Manga	800	4.00
4	4	Pera	300	6.00
5	5	Abacaxi	300	3.00

Pessoa

Resultados | Mensagens

	idPessoa	nome	rua	cidade	estado	telefone	email
1	1	Joao	Rua 12,casa3,Quintanda	Riacho do Sul	PA	11111111	joao@riacho.com
2	2	JJC	Rua 11, Centro	Riacho do Norte	PA	12121212	jjc@riacho.com

PessoaFisica

	idPessoa	CPF
1	1	11111111111

PessoaJuridica

Resultados | Mensagens

	idPessoa	CNPJ
1	2	22222222222

Movimento

Resultados | Mensagens

	idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valorUnitario
1	1	1	1	1	10	E	3
2	2	1	1	2	10	E	1
3	3	1	1	3	10	E	3
4	4	2	2	1	3	S	5
5	5	2	2	2	1	S	2
6	6	2	2	3	8	S	4
7	7	3	2	1	5	E	4,75
8	8	3	2	2	7	E	1,32
9	9	3	1	1	9	S	5
10	10	3	1	2	3	S	2
11	11	4	2	1	8	S	5
12	12	4	2	2	6	S	2

Análise e conclusão:

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

No SQL, tanto SEQUENCE quanto IDENTITY são usados para gerar valores únicos, geralmente em colunas de chave primária, mas têm algumas diferenças principais:

1. Definição:
- IDENTITY: É uma propriedade de coluna que gera automaticamente um número sequencial ao inserir uma nova linha. O valor é incrementado automaticamente conforme as inserções.
- SEQUENCE: É um objetivo separado que gera números sequenciais independente de tabelas. Pode ser utilizado em múltiplas tabelas e pode ser controlado de forma mais flexível.

2. Flexibilidade:
- IDENTITY: Está ligado a uma única coluna e tabela. As opções de controle são limitadas.
- SEQUENCE: Pode ser reutilizado em diferentes tabelas e pode ter incrementos e valores iniciais personalizados, além de permitir chamadas explícitas em comando SQL.

3. Reinicialização:
- IDENTITY: Não pode ser reinicializado facilmente; requer a truncagem da tabela ou uma operação mais complexa.
- SEQUENCE: Pode ser redefinido a qualquer momento com o comando ALTER SEQUENCE.

4. Performance:
- IDENTITY: Geralmente tem melhor desempenho em inserções, pois o valor é gerado internamente.
- SEQUENCE: Pode ser mais eficiente em cenários de alta concorrência, pois pode gerar números fora da transação.

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

As chaves estrangeiras são fundamentais para a consistência do banco de dados, pois garantem a integridade referencial entre tabelas. Elas asseguram que um valor em uma coluna de uma tabela (chave estrangeira) corresponda a um valor existente em outra tabela (chave primária). Isso evita a criação de registros órfãos e assegura que as relações entre os dados sejam mantidas, permitindo a operações de inserção, atualização e exclusão de forma segura. Sem chaves estrangeiras, a integridade dos dados pode ser comprometida, resultando em inconsistências e dificultando a confiabilidade das informações armazenadas.

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

No cálculo relacional, a abordagem é mais declarativa, e os operadores incluem:

1. **Seleção (σ):** Filtra linhas com base em uma condição.
2. **Projeção (π):** Seleciona colunas específicas de uma tabela.
3. **União (∪):** Combina resultados de duas tabelas com a mesma estrutura.
4. **Interseção (∩):** Retorna registros comuns a duas tabelas.
5. **Diferença (-):** Retorna registros de uma tabela que não estão na outra.
6. **Produto Cartesiano (×):** Combina todas as linhas de duas tabelas.

No cálculo relacional, a abordagem é mais declarativa, e os operadores incluem:

Condições de consulta: Define o que se deseja buscar, sem especificar como.

Tuplas e Domínios: Permitem expressar condições sobre os dados em forma de variáveis.

Enquanto a álgebra relacional foca em operações sobre conjuntos, o cálculo relacional enfatiza a descrição dos resultados desejados.

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

O agrupamento em consultas SQL é realizado com a cláusula GROUP BY, que agrupa linhas com valores iguais em colunas especificadas. Um requisito obrigatório é que todas as colunas na cláusula SELECT que não são funções de agregação devem estar incluídas no GROUP BY. Isso garante que os resultados sejam consistentes e bem definidos. Por exemplo, ao usar SELECT coluna1, COUNT(*) FROM tabela GROUP BY coluna1, coluna1 deve aparecer em ambas as cláusulas.