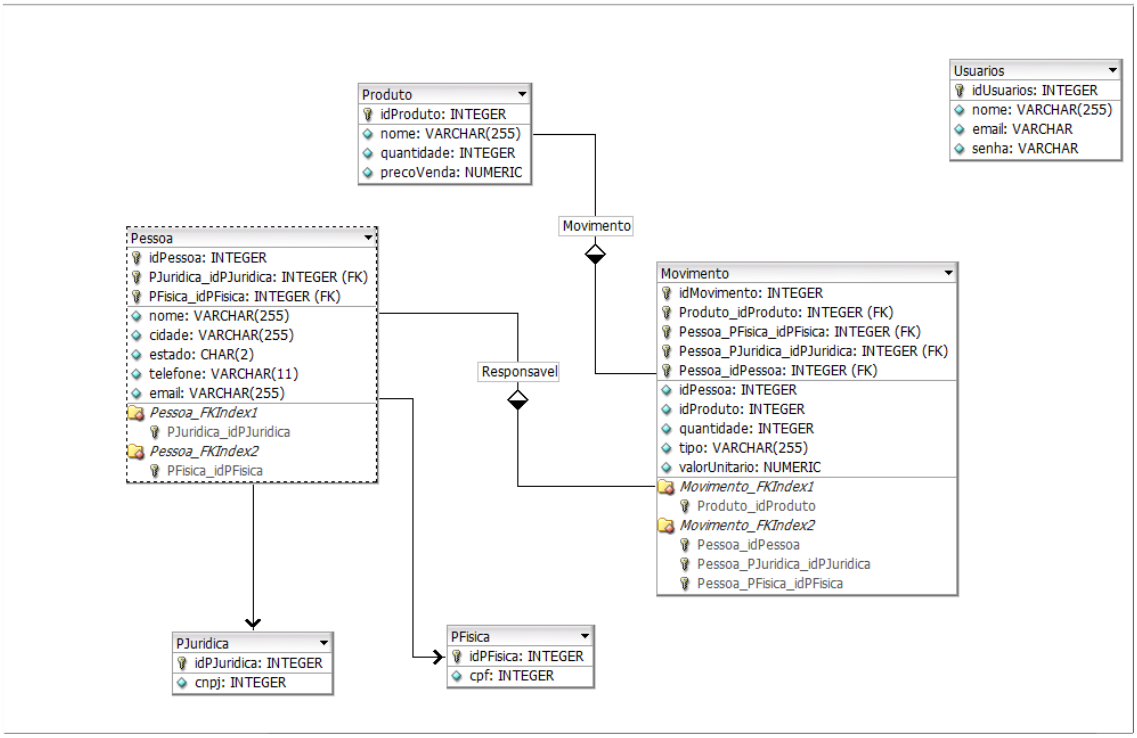
Centro Universitário Estácio de Ribeirão Preto – Polo São Dimas - Piracicaba
**Curso:** Desenvolvimento Full Stack **Disciplina:** Nível 2: Vamos manter as informações!
**Turma:** 9001 / **Semestre:** 2024.3 3º Semestre
**Aluno:** Bruno Boralli Prudente dos Anjos
**Repositório:** [https://github.com/bboralli/RPG0015-Vamos-manter-as-informaces.git](https://github.com/bboralli/RPG0015-Vamos-manter-as-informaces.git)

**Objetivo 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).

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

Modelo de dados:




Criação das tabelas:

```
CREATE DATABASE Loja;

CREATE TABLE Usuarios (
	idUsuario INT PRIMARY KEY,
	[login] VARCHAR(255),
	senha VARCHAR(255)
);
CREATE TABLE Pessoas (
	idPessoa INT PRIMARY KEY,
	nome VARCHAR(255),
cidade VARCHAR(255),
estado CHAR(2),
telefone VARCHAR(11),
email VARCHAR(255)
);
CREATE TABLE PessoasFisicas (
	idPFisica INT PRIMARY KEY DEFAULT NEXT VALUE FOR seqId,
	cpf VARCHAR(14),
	FOREIGN KEY (idPFisica) REFERENCES Pessoas(idPessoa)
);
CREATE TABLE PessoasJuridicas (
	idPJuridica INT PRIMARY KEY DEFAULT NEXT VALUE FOR seqId,
	cnpj VARCHAR(14),
	FOREIGN KEY (idPjuridica) REFERENCES Pessoas(idPessoa)
);
CREATE TABLE Produtos (
```

```
	idProduto INT PRIMARY KEY,
	nome VARCHAR(255),
	quantidade INT,
	precoVenda NUMERIC(10, 2)
);
CREATE TABLE Movimentos (
	idMovimentos INT PRIMARY KEY,
	idUsuario INT,
idPessoa INT,
idProduto INT,
Quantidade INT,
Tipo CHAR(1) CHECK (Tipo IN ('E', 'S')),
	PrecoUnitario NUMERIC(10, 2),
FOREIGN KEY (idUsuario) REFERENCES Usuarios(idUsuario),
	FOREIGN KEY (idProduto) REFERENCES Produtos(idProduto),
	FOREIGN KEY (idPessoa) REFERENCES Pessoas(idPessoa)
);
CREATE SEQUENCE seqId
	AS INT
	START WITH 1
	INCREMENT BY 1;
```

a) Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?
São implementadas por meio de relacionamentos em tabelas, usando chaves estrangeiras e chaves primárias.

b) Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?
O relacionamento de tabela por Subclasse é uma abordagem eficaz para representar herança em bancos de dados relacionais, permitindo que você modele adequadamente as diferentes classes e subclasses, mantendo a integridade dos dados e facilitando consultas específicas.

c) Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?
Edição de Consultas SQL, um editor de consultas com realce de sintaxe, preenchimento automatico de depuração.
Designer de Tabelas e Diagramas, permite criar, modificar e vizualizar o esquemas de banco de dados.
Scripting, permite gerar scripts SQL automaticamente, criando tabelas e geração de dados testes.
Administração de Segurança, oferece ferramentas para gerenciar segurança, criaçao de usuarios e atribuições de permissão.

2º Procedimento – Alimentando a Base

Alimentando a base de dados:

```
INSERT INTO Usuarios (idUsuario, [login], senha)
VALUES (1, 'op1', 'op1');
INSERT INTO Usuarios (idUsuario, [login], senha)
VALUES (2, 'op2', 'op2');
INSERT INTO Produtos (idProduto, nome, quantidade, precoVenda)
VALUES (
1,
'Banana',
'100',
'5.00'
);
INSERT INTO Produtos (idProduto, nome, quantidade, precoVenda)
VALUES (
3,
'Laranja',
'500',
'2.00'
);
INSERT INTO Produtos (idProduto, nome, quantidade, precoVenda)
VALUES (
4,
'Manga',
'800',
'4.00'
);
DECLARE @ProximoID INT;
SET @ProximoID = NEXT VALUE FOR seqID;
INSERT INTO Pessoas (idPessoa, nome, cidade, estado, telefone, email)
VALUES (@ProximoID, 'João', 'Riacho do Sul', 'PA', '1111-1111', 'joao@riacho.com');
INSERT INTO PessoasFisicas (idPFisica, cpf)
VALUES (@ProximoID, '11111111111');
INSERT INTO Pessoas (idPessoa, nome, cidade, estado, telefone, email)
VALUES (@ProximoID, 'JJC', 'Riacho do Norte', 'PA', '1212-1212', 'jjc@riacho.com');
INSERT INTO PessoasJuridicas(idPJuridica, cnpj)
VALUES (@ProximoID, '222222222222');
INSERT INTO Movimentos (idMovimentos, idUsuario, idPessoa, idProduto, Quantidade,
Tipo, PrecoUnitario)
VALUES (1, 1, 7, 1, 20,'S', 4.00);
INSERT INTO Movimentos (idMovimentos, idUsuario, idPessoa, idProduto,
Quantidade,Tipo, PrecoUnitario)
VALUES (4, 1, 7, 3, 15,'S', 2.00);
INSERT INTO Movimentos (idMovimentos, idUsuario, idPessoa, idProduto,
Quantidade,Tipo, PrecoUnitario)
VALUES (5, 2, 7, 3, 10,'S', 3.00);
```

```
INSERT INTO Movimentos (idMovimentos, idUsuario, idPessoa, idProduto,
Quantidade,Tipo, PrecoUnitario)
VALUES (7, 1, 8, 3, 15,'E', 5.00);
INSERT INTO Movimentos (idMovimentos, idUsuario, idPessoa, idProduto,
Quantidade,Tipo, PrecoUnitario)
VALUES (8, 1, 8, 4, 20,'E', 4.00);
```

Consultas:
Dados completos de pessoas físicas:

```
SELECT P.*, PF.*
FROM PessoasFisicas PF
INNER JOIN Pessoas P ON PF.idPFisica = P.idPessoa;
```

| | idPessoa | nome | cidade | estado | telefone | email | idPFisica | cpf |
|---|---|---|---|---|---|---|---|---|
| 1 | 7 | João | Riacho do Sul | PA | 1111-1111 | joao@riacho.com | 7 | 11111111111 |

Dados completos de pessoas jurídicas:

```
SELECT P.*, PJ.*
FROM PessoasJuridicas PJ
INNER JOIN Pessoas P ON PJ.idPJuridica = P.idPessoa;
```

| | idPessoa | nome | cidade | estado | telefone | email | idPJuridica | cnpj |
|---|---|---|---|---|---|---|---|---|
| 1 | 8 | JJC | Riacho do Norte | PA | 1212-1212 | jjc@riacho.com | 8 | 222222222222 |

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

```
SELECT
        M.idMovimentos,
        P.nome AS Produto,
        F.nome AS Fornecedor,
        M.Quantidade,
        M.PrecoUnitario,
        M.Quantidade * M.PrecoUnitario AS ValorTotal
FROM
        Movimentos M
JOIN
        Produtos P ON M.idProduto = P.idProduto
JOIN
        Pessoas F ON M.idPessoa = F.idPessoa
WHERE
        M.Tipo = 'E';
```

| | idMovimentos | Produto | Fornecedor | Quantidade | PrecoUnitario | ValorTotal |
|---|---|---|---|---|---|---|
| 1 | 7 | Laranja | JJC | 15 | 5.00 | 75.00 |
| 2 | 8 | Manga | JJC | 20 | 4.00 | 80.00 |

Movimentações de saída, com produto, fornecedor, quantidade, preço unitário e valor total:

```
SELECT
	M.idMovimentos,
	P.nome AS Produto,
	C.nome AS Comprador,
	M.Quantidade,
	M.PrecoUnitario,
	M.Quantidade * M.PrecoUnitario AS ValorTotal
FROM
	Movimentos M
JOIN
	Produtos P ON M.idProduto = P.idProduto
JOIN
	Pessoas C ON M.idPessoa = C.idPessoa
WHERE
	M.Tipo = 'S';
```

| | idMovimentos | Produto | Comprador | Quantidade | PrecoUnitario | ValorTotal |
|---|---|---|---|---|---|---|
| 1 | 1 | Banana | João | 20 | 4.00 | 80.00 |
| 2 | 4 | Laranja | João | 15 | 2.00 | 30.00 |
| 3 | 5 | Laranja | João | 10 | 3.00 | 30.00 |

Valor total das entradas agrupadas por produto:

```
SELECT
	P.nome AS Produto,
	SUM(M.Quantidade * M.PrecoUnitario) AS ValorTotalEntradas
FROM
	Movimentos M
JOIN
	Produtos P ON M.idProduto = P.idProduto
WHERE
	M.Tipo = 'E'
GROUP BY
	P.nome;
```

| | Produto | ValorTotalEntradas |
|---|---|---|
| 1 | Laranja | 75.00 |
| 2 | Manga | 80.00 |

Valor total das saídas agrupadas por produto:

```
SELECT
```

```
        P.nome AS Produto,
        SUM(M.Quantidade * M.PrecoUnitario) AS ValorTotalSaidas
FROM
        Movimentos M
JOIN
        Produtos P ON M.idProduto = P.idProduto
WHERE
        M.Tipo = 'S' -- Movimentações de saída
GROUP BY
        P.nome;
```

| | Produto | ValorTotalSaidas |
|---|---|---|
| 1 | Banana | 80.00 |
| 2 | Laranja | 60.00 |

Operadores que não efetuaram movimentações de entrada:

```
SELECT
        U.[login] AS Operador
FROM
        Usuarios U
LEFT JOIN
        Movimentos M ON U.idUsuario = M.idUsuario AND M.Tipo = 'E'
WHERE
        M.idMovimentos IS NULL;
```

| | Operador |
|---|---|
| 1 | op2 |

Valor total de entrada, agrupado por operador:

```
SELECT
        U.[login] AS Operador,
        SUM(M.Quantidade * M.PrecoUnitario) AS ValorTotalEntrada
FROM
        Usuarios U
JOIN
        Movimentos M ON U.idUsuario = M.idUsuario AND M.Tipo = 'E'
GROUP BY
        U.[login];
```

| | Operador | ValorTotalEntrada |
|---|---|---|
| 1 | op1 | 155.00 |

Valor total de saída, agrupado por operador:

```
SELECT
        U.[login] AS Operador,
```

```
        SUM(M.Quantidade * M.PrecoUnitario) AS ValorTotalSaida
FROM
        Usuarios U
JOIN
        Movimentos M ON U.idUsuario = M.idUsuario AND M.Tipo = 'S'
GROUP BY
        U.[login];
```

| | Operador | ValorTotalSaida |
|---|---|---|
| 1 | op1 | 110.00 |
| 2 | op2 | 30.00 |

Valor médio de venda por produto, utilizando média ponderada:

```
SELECT
        P.nome AS Produto,
        CAST(ROUND(SUM(M.Quantidade * M.PrecoUnitario) / SUM(M.Quantidade), 2)
AS DECIMAL(10, 2)) AS ValorMedioVenda
FROM
        Produtos P
JOIN
        Movimentos M ON P.idProduto = M.idProduto AND M.Tipo = 'S'
GROUP BY
        P.nome;
```

| | Produto | ValorMedioVenda |
|---|---|---|
| 1 | Banana | 4.00 |
| 2 | Laranja | 2.40 |

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

As sequências oferecem mais flexibilidade e controle na geração de valores sequenciais, enquanto a propriedade IDENTITY é mais conveniente e direta para a geração automática de identificadores primários em uma única tabela.

b) Qual a importância das chaves estrangerias para a consistência do banco?
As chaves estrangeiras são importantes na manutenção da consistência, integridade e qualidade dos dados em um banco de dados relacional. Elas ajudam a evitar problemas de integridade, garantir a precisão dos dados e facilitar a recuperação de informações relacionadas.

c) Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?
álgebra relacional:
eleção: Filtra as linhas de uma tabela com base em uma condição.
Projeção: Seleciona colunas específicas de uma tabela.

Junção: Combina registros de duas ou mais tabelas com base em uma condição.
União: Combina dois conjuntos de registros distintos, retornando um conjunto com todos os registros únicos.
Interseção: Retorna registros que estão presentes em ambos os conjuntos de registros.
Diferença: Retorna registros presentes em um conjunto, mas não no outro.
Produto Cartesiano: Combina todos os registros de uma tabela com todos os registros de outra tabela.
Divisão: Retorna registros de uma tabela que correspondem a todos os valores de outra tabela.
cálculo relacional:
Seleção: Expressa condições para selecionar tuplas da relação.
Projeção: Especifica as colunas a serem projetadas na consulta.
Quantificação Existencial: Verifica se existe pelo menos uma tupla que satisfaça a condição.
Quantificação Universal: Verifica se todas as tuplas satisfazem a condição.
Junção: Combina duas ou mais relações com base em uma condição.
Negacão: Retorna tuplas que não satisfazem uma determinada condição.

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

O agrupamento em consultas de banco de dados é realizado por meio da cláusula GROUP BY. Essa cláusula é usada para organizar os resultados de uma consulta em grupos com base nos valores de uma ou mais colunas específicas. Cada grupo representa um conjunto de registros que compartilham valores semelhantes nas colunas de agrupamento. O requisito obrigatório para usar a cláusula GROUP BY é que todas as colunas listadas na cláusula SELECT devem ser colunas de agrupamento ou funções de agregação. As funções de agregação, como SUM, COUNT, AVG, entre outras, são frequentemente usadas para calcular valores resumidos dentro de cada grupo.

Conclusão:

Neste trabalho, exploramos a criação e gerenciamento de um banco de dados no SQL Server Management Studio para um sistema de controle de loja. Aprendemos a modelar entidades, estabelecer relacionamentos, usar sequências para gerar identificadores únicos e realizar operações de inserção, consulta e exclusão de dados. Também abordamos conceitos como herança e consultas complexas para obter insights valiosos. Essa prática nos proporcionou uma compreensão sólida da administração de bancos de dados e sua aplicação em cenários do mundo real.