Nome do Campus: Polo Cruzeiro-Sp

Nome do Curso: Desenvolvimento FullStack

Nome da Disciplina: Vamos Manter as informações?

Número da Turma: 2025.1

Semestre Letivo: Primeiro Semestre

Nome do Aluno: Moniza de Oliveira Silva Santos Pelegrini

Matrícula: 202401190829

Título da Prática: Modelagem e implementação de um banco de dados simples, utilizando como base o SQL SERVER.

Objetivo da Prática: Identificar os requisitos de um sistema e transformá-los no modelo adequado; Utilizar ferramentas de modelagem para a base de dados relacionais; Explorar a sintaxe SQL na criação das estruturas do banco (DDL); Explorar a sintaxe SQL na criação das estruturas do banco (DML);

Segundo Procedimento: Alimentando a Base

Códigos utilizados neste roteiro

```
-- Criação da sequence para identificadores de pessoa
CREATE SEQUENCE seq_pessoa_id
    START WITH 1
    INCREMENT BY 1
    MINVALUE 1
    NO MAXVALUE
    CACHE 10;
GO
```

```
CREATE TABLE Produto (
    id_produto INT PRIMARY KEY IDENTITY(1,1),
    nome NVARCHAR(200) NOT NULL,
    quantidade_estoque INT NOT NULL DEFAULT 0 CHECK (quantidade_estoque >= 0),
    preco_venda DECIMAL(10, 2) NOT NULL CHECK (preco_venda >= 0)
);
GO
```

```
CREATE TABLE Usuario (
    id_usuario INT PRIMARY KEY IDENTITY(1,1),
    nome NVARCHAR(100) NOT NULL,
    login NVARCHAR(50) NOT NULL UNIQUE,
    senha NVARCHAR(255) NOT NULL,
    email NVARCHAR(100) NOT NULL UNIQUE
);
GO
```

```sql
CREATE TABLE Movimento (
    id_movimento INT PRIMARY KEY IDENTITY(1,1),
    id_usuario INT NOT NULL,
    id_pessoa INT NOT NULL,
    id_produto INT NOT NULL,
    quantidade INT NOT NULL CHECK (quantidade > 0),
    tipo CHAR(1) NOT NULL CHECK (tipo IN ('E', 'S')), -- 'E' para entrada, 'S' para saída
    valor_unitario DECIMAL(10, 2) NOT NULL CHECK (valor_unitario >= 0),
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario),
    FOREIGN KEY (id_pessoa) REFERENCES Pessoa(id_pessoa),
    FOREIGN KEY (id_produto) REFERENCES Produto(id_produto)
);
```

```sql
-- Inserção do CPF na tabela PessoaFisica
INSERT INTO PessoaFisica (id_pessoa, cpf)
VALUES
(7, '11111111111');
GO

-- Consulta para verificar os dados inseridos na tabela Pessoa
SELECT * FROM Pessoa WHERE id_pessoa = 7;
GO

-- Consulta para verificar os dados inseridos na tabela PessoaFisica
SELECT * FROM PessoaFisica WHERE id_pessoa = 7;
GO
```

```sql
-- Inserção de dados comuns na tabela Pessoa
INSERT INTO Pessoa (id_pessoa, tipo, nome_razao_social, endereco, telefone, email)
VALUES
(7, 'F', 'Joao', 'Rua 12, casa 3, Quitanda', 'Riacho do Sul', 'PA', '1111-1111', 'joao@ri
GO
```

```sql
-- Inserção de dados na tabela Movimento
INSERT INTO Movimento (id_usuario, id_pessoa, id_produto, quantidade, tipo, valor_unitario)
VALUES
(1, 7, 1, 20, 'S', 4.00), -- Saída de 20 unidades do produto 1 pelo usuário 1 e pessoa 7
(1, 7, 3, 15, 'S', 2.00), -- Saída de 15 unidades do produto 3 pelo usuário 1 e pessoa 7
(2, 7, 3, 10, 'S', 3.00), -- Saída de 10 unidades do produto 3 pelo usuário 2 e pessoa 7
(1, 15, 3, 15, 'E', 5.00), -- Entrada de 15 unidades do produto 3 pelo usuário 1 e pessoa 15
(1, 15, 4, 20, 'E', 4.00); -- Entrada de 20 unidades do produto 4 pelo usuário 1 e pessoa 15
GO
```

```sql
-- Inserção de dados na tabela Produto
INSERT INTO Produto (nome, quantidade_estoque, preco_venda)
VALUES
('Banana', 100, 5.00),
('Laranja', 500, 2.00),
('Manga', 800, 4.00);
GO
```

```sql
-- Inserção de dados na tabela Usuario
INSERT INTO Usuario (nome, login, senha, email)
VALUES
('Operador 1', 'op1', 'op1', 'op1@example.com'),
('Operador 2', 'op2', 'op2', 'op2@example.com');
GO
```

```sql
-- Obter o próximo valor da sequence
DECLARE @nextId INT;
SET @nextId = NEXT VALUE FOR seq_pessoa_id;

SELECT @nextId AS 'ProximoID';
GO
```

```sql
-- Inserção de dados na tabela Produto
INSERT INTO Produto (nome, quantidade_estoque, preco_venda)
VALUES
('Banana', 100, 5.00),
('Laranja', 500, 2.00),
('Manga', 800, 4.00);
GO

-- Consulta para verificar os dados inseridos
SELECT * FROM Produto;
GO
```

133 %

Resultados | Mensagens

	id_produto	nome	quantidade_estoque	preco_venda
1	1	Banana	100	5.00
2	2	Laranja	500	2.00
3	3	Manga	800	4.00

```sql
-- Inserção de dados na tabela Usuario
INSERT INTO Usuario (nome, login, senha, email)
VALUES
('Operador 1', 'op1', 'op1', 'op1@example.com'),
('Operador 2', 'op2', 'op2', 'op2@example.com');
GO

-- Consulta para verificar os dados inseridos
SELECT * FROM Usuario;
GO
```

133 %

Resultados | Mensagens

	id_usuario	nome	login	senha	email
1	1	Operador 1	op1	op1	op1@example.com
2	2	Operador 2	op2	op2	op2@example.com

```sql
-- Consulta: Dados completos de pessoas físicas
SELECT
    p.id_pessoa,
    p.nome_razao_social AS nome,
    p.endereco,
    p.telefone,
    p.email,
    pf.cpf
FROM
    Pessoa p
JOIN
    PessoaFisica pf ON p.id_pessoa = pf.id_pessoa
WHERE
    p.tipo = 'F';
GO
```

```sql
-- Consulta: Dados completos de pessoas jurídicas
SELECT
    p.id_pessoa,
    p.nome_razao_social AS razao_social,
    p.endereco,
    p.telefone,
    p.email,
    pj.cnpj
FROM
    Pessoa p
JOIN
    PessoaJuridica pj ON p.id_pessoa = pj.id_pessoa
WHERE
    p.tipo = 'J';
GO
```

```sql
-- Consulta: Movimentações de entrada
SELECT
    m.id_movimento,
    pr.nome AS produto,
    pe.nome_razao_social AS fornecedor,
    m.quantidade,
    m.valor_unitario,
    (m.quantidade * m.valor_unitario) AS valor_total
FROM
    Movimento m
JOIN
    Produto pr ON m.id_produto = pr.id_produto
JOIN
    Pessoa pe ON m.id_pessoa = pe.id_pessoa
WHERE
    m.tipo = 'E';
GO
```

```sql
-- Consulta: Valor total das entradas agrupadas por produto
SELECT
    pr.nome AS produto,
    SUM(m.quantidade * m.valor_unitario) AS valor_total_entrada
FROM
    Movimento m
JOIN
    Produto pr ON m.id_produto = pr.id_produto
WHERE
    m.tipo = 'E'
GROUP BY
    pr.nome;
GO
```

```sql
-- Consulta: Operadores que não efetuaram movimentações de entrada
SELECT
    u.id_usuario,
    u.nome AS operador
FROM
    Usuario u
LEFT JOIN
    Movimento m ON u.id_usuario = m.id_usuario AND m.tipo = 'E'
WHERE
    m.id_movimento IS NULL;
GO
```

```sql
-- Consulta: Valor total de entrada, agrupado por operador
SELECT
    u.nome AS operador,
    SUM(m.quantidade * m.valor_unitario) AS valor_total_entrada
FROM
    Movimento m
JOIN
    Usuario u ON m.id_usuario = u.id_usuario
WHERE
    m.tipo = 'E'
GROUP BY
    u.nome;
GO
```

```sql
-- Consulta: Valor total de saída, agrupado por operador
SELECT
    u.nome AS operador,
    SUM(m.quantidade * m.valor_unitario) AS valor_total_saida
FROM
    Movimento m
JOIN
    Usuario u ON m.id_usuario = u.id_usuario
WHERE
    m.tipo = 'S'
GROUP BY
    u.nome;
GO
```

```sql
-- Consulta: Valor médio de venda por produto, utilizando média ponderada
SELECT
    pr.nome AS produto,
    SUM(m.quantidade * m.valor_unitario) / SUM(m.quantidade) AS media_ponderada_venda
FROM
    Movimento m
JOIN
    Produto pr ON m.id_produto = pr.id_produto
WHERE
    m.tipo = 'S'
GROUP BY
    pr.nome;
GO
```

Análise e Conclusão

1. **Quais são as diferenças entre o uso de SEQUENCE e IDENTITY no SQL Server?**
   As principais diferenças entre SEQUENCE e IDENTITY estão na flexibilidade e controle na geração de valores únicos.
   - O IDENTITY é associado diretamente a uma coluna específica de uma tabela e gera valores automaticamente a cada nova inserção. Sua configuração é simples, mas limitada a um único uso por tabela.
   - Já o SEQUENCE é um objeto independente da tabela, podendo ser reutilizado por múltiplas colunas e tabelas. Ele permite maior

controle sobre o incremento, reinício e uso compartilhado dos valores.

2. **Qual é a importância das chaves estrangeiras para a consistência do banco de dados?**
   As chaves estrangeiras (foreign keys) são essenciais para garantir a integridade referencial em um banco de dados relacional. Elas estabelecem vínculos entre tabelas, assegurando que valores inseridos em uma tabela correspondam a registros válidos em outra. Dessa forma, evitam dados órfãos e mantêm a coerência das relações entre as entidades.

3. **Quais operadores SQL pertencem à álgebra relacional e quais ao cálculo relacional?**
   A álgebra relacional e o cálculo relacional são modelos teóricos usados para expressar consultas em bancos relacionais.

   - A **álgebra relacional** utiliza operadores como: **seleção (SELECT), projeção (PROJECT), junção (JOIN), união (UNION), interseção (INTERSECT), diferença (EXCEPT)** e **produto cartesiano (CROSS JOIN)**.

   - O **cálculo relacional**, por sua vez, baseia-se em expressões lógicas para descrever *o que* deve ser retornado, não *como*. É mais declarativo e próximo da forma como escrevemos consultas em SQL.
     Muitos comandos SQL têm correspondência direta com operadores da álgebra relacional, embora o SQL também inclua funcionalidades além do escopo teórico desses modelos.

4. **Como é realizado o agrupamento em consultas SQL e qual requisito é obrigatório?**
   O agrupamento em SQL é feito por meio da cláusula GROUP BY, que organiza os dados em grupos com base nos valores de uma ou mais colunas.
   Essa funcionalidade é indispensável quando se aplicam funções de agregação, como COUNT(), SUM(), AVG(), MIN() e MAX(), permitindo calcular resultados específicos para cada grupo.
   Requisito obrigatório: todas as colunas no SELECT que não estão envolvidas em funções de agregação devem obrigatoriamente estar presentes na cláusula GROUP BY.