Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

2º Procedimento | Alimentando a Base

Aluno: Caio Viana Castelo Branco

Matrícula: 202307465925

Disciplina: RPG0015 - Vamos manter as informações!

Curso: Desenvolvimento Full Stack

Turma: 9001

Semestre Letivo: 2025.1

Campus: Parangaba

1. Objetivos 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);
- No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server.

2. Códigos utilizados na prática

- Inserindo usuários:

```
INSERT INTO Usuario (login, senha, nome)
VALUES
('op1', 'op1', 'Operador 1'),
('op2', 'op2', 'Operador 2'),
('admin', 'admin123', 'Administrador');
```

- Inserindo produtos:

```
INSERT INTO Produto (nome, quantidade, preco_venda)
VALUES
  ('Notebook Dell i7', 15, 4500.00),
  ('Mouse Sem Fio', 50, 120.00),
  ('Teclado Mecânico', 30, 250.00),
  ('Monitor 24"', 20, 1200.00),
  ('Headphone Bluetooth', 40, 350.00);
```

- Inserindo pessoas físicas e jurídicas:

```
-- Pessoas Físicas

DECLARE @id_pf1 INT = NEXT VALUE FOR seq_pessoa_id;
INSERT INTO Pessoa (id_pessoa, tipo, nome, endereco, telefone)
VALUES (@id_pf1, 'F', 'João Silva', 'Rua A, 123', '(11) 9999-8888');

INSERT INTO PessoaFisica (id_pessoa, cpf)
VALUES (@id_pf1, '123.456.789-00');

DECLARE @id_pf2 INT = NEXT VALUE FOR seq_pessoa_id;
INSERT INTO Pessoa (id_pessoa, tipo, nome, endereco, telefone)
VALUES (@id_pf2, 'F', 'Maria Souza', 'Av. B, 456', '(21) 98888-7777');

INSERT INTO PessoaFisica (id_pessoa, cpf)
VALUES (@id_pf2, '987.654.321-00');

-- Pessoas Jurídicas
DECLARE @id_pj1 INT = NEXT VALUE FOR seq_pessoa_id;
INSERT INTO Pessoa (id_pessoa, tipo, nome, endereco, telefone)
VALUES (@id_pj1, 'J', 'Tech Ltda', 'Av. Comercial, 1000', '(11) 3030-4040');

INSERT INTO PessoaJuridica (id_pessoa, cnpj)
VALUES (@id_pj1, '12.345.678/0001-90');

DECLARE @id_pj2 INT = NEXT VALUE FOR seq_pessoa_id;
INSERT INTO Pessoa (id_pessoa, tipo, nome, endereco, telefone)
VALUES (@id_pj2, 'J', 'Eletro Distribuidora', 'Rua Industrial, 500', '(11) 5050-6060');

INSERT INTO PessoaJuridica (id_pessoa, cnpj)
VALUES (@id_pj2, '98.765.432/0001-10');
```

- Criar movimentações

```
-- Movimentação de Entrada

INSERT INTO Movimento (tipo, id_usuario, id_produto, id_pessoa, quantidade,
preco_unitario)
VALUES ('C', 1, 1, 3, 10, 4000.00), -- Compra de Notebooks
('C', 2, 2, 4, 50, 90.00), -- Compra de Mouses
('C', 1, 3, 3, 30, 200.00); -- Compra de Teclados

-- Movimentação de Saída
INSERT INTO Movimento (tipo, id_usuario, id_produto, id_pessoa, quantidade,
preco_unitario)
VALUES ('V', 1, 1, 1, 2, 4500.00), -- Venda de Notebook
('V', 2, 2, 2, 5, 120.00), -- Venda de Mouse
('V', 1, 4, 1, 1, 1200.00); -- Venda de Monitor
```

- Consultas solicitadas no item 4

```
-- a. Dados completos de pessoas físicas
SELECT p.*, pf.cpf
FROM Pessoa p
JOIN PessoaFisica pf
ON p.id_pessoa = pf.id_pessoa;

-- b. Dados completos de pessoas jurídicas
SELECT p.*, pj.cnpj
FROM Pessoa p
JOIN PessoaJuridica pj
ON p.id_pessoa = pj.id_pessoa;

-- c. Movimentações de entrada
SELECT m.id_movimento, p.nome AS produto, pj.nome AS fornecedor,
m.quantidade, m.preco_unitario, (m.quantidade * m.preco_unitario) AS valor_total
FROM Movimento m
JOIN Produto p
ON m.id_produto = p.id_produto
JOIN Pessoa pj
ON m.id_pessoa = pj.id_pessoa
WHERE m.tipo = 'C';



-- d. Movimentações de saída
SELECT m.id_movimento, p.nome AS produto, pf.nome AS cliente, m.quantidade,
```

```
m.preco_unitario, (m.quantidade * m.preco_unitario) AS valor_total
FROM Movimento m
JOIN Produto p
ON m.id_produto = p.id_produto
JOIN Pessoa pf
ON m.id_pessoa = pf.id_pessoa
WHERE m.tipo = 'V';

-- e. Valor total das entradas por produto
SELECT p.nome AS produto, SUM(m.quantidade * m.preco_unitario) AS
total_entradas
FROM Movimento m
JOIN Produto p
ON m.id_produto = p.id_produto
WHERE m.tipo = 'C'
GROUP BY p.nome;

-- f. Valor total das saídas por produto
SELECT p.nome AS produto, SUM(m.quantidade * m.preco_unitario) AS total_saidas
FROM Movimento m
JOIN Produto p
ON m.id_produto = p.id_produto
WHERE m.tipo = 'V'
GROUP BY p.nome;

-- g. Operadores que não efetuaram compras
SELECT u.*
FROM Usuario u
WHERE u.id_usuario NOT IN ( SELECT DISTINCT id_usuario FROM Movimento
WHERE tipo = 'C' );

-- h. Valor total de entrada por operador
SELECT u.nome AS operador, SUM(m.quantidade * m.preco_unitario) AS
total_entradas
FROM Movimento m
JOIN Usuario u
ON m.id_usuario = u.id_usuario
WHERE m.tipo = 'C'
GROUP BY u.nome;

-- i. Valor total de saída por operador
SELECT u.nome AS operador, SUM(m.quantidade * m.preco_unitario) AS
total_saidas
FROM Movimento m
JOIN Usuario u
ON m.id_usuario = u.id_usuario
WHERE m.tipo = 'V' GROUP BY u.nome;
```

```
-- j. Valor médio de venda por produto (média ponderada)
SELECT
p.nome AS produto,
SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_ponderada
FROM Movimento m
JOIN Produto p
ON m.id_produto = p.id_produto
WHERE m.tipo = 'V' GROUP BY p.nome;
```

3. Análise e Conclusão

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

- IDENTITY é um atributo usado em colunas (geralmente em SQL Server) para autoincrementar valores numéricos inteiros (como PKs). É controlado diretamente pelo banco e não pode ser reutilizado se uma inserção falhar (há "lacunas").
- SEQUENCE (padrão SQL, presente em Oracle, PostgreSQL, etc.) é um objeto independente que gera sequências numéricas e pode ser usado em múltiplas colunas/tabelas. Oferece mais flexibilidade (ex.: definir incremento, ciclo, reinício) e é menos restrito que o IDENTITY.

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

Chaves estrangeiras (FKs) garantem integridade referencial:

- Evitam registros "órfãos".
- Forçam que operações de DELETE/UPDATE sigam regras (como CASCADE, RESTRICT ou SET NULL).

- Sem FKs, o banco pode ter dados inconsistentes (ex.: IDs que não existem na tabela referenciada), prejudicando consultas e aplicações.

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

A álgebra relacional usa operadores como SELECT, PROJECT, JOIN, UNION e outros para manipular relações de forma procedural. Já o cálculo relacional é baseado em lógica de predicados, sendo mais declarativo. O SQL combina os dois: operações como JOIN vêm da álgebra, enquanto subconsultas com IN ou EXISTS derivam do cálculo.

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

O agrupamento (GROUP BY) organiza linhas com valores iguais em colunas específicas, geralmente usado com funções de agregação (SUM, COUNT, etc.). O requisito obrigatório é que todas as colunas no SELECT devem estar no GROUP BY ou ser funções de agregação.

Exemplo:

SELECT departamento, COUNT(*) FROM funcionarios GROUP BY departamento