Campus – Estacio Tatui

Disciplina - Nível 2: Vamos Manter as Informações?

Integrantes – Clayton Prebelli Pires

Curso – Desenvolvimento Full Stack

Relatório de Prática: Modelagem e Implementação de Banco de Dados Comercial (RPG0015 - Vamos manter as informações!)

Título da Prática:

Modelagem e Implementação de Banco de Dados Comercial (RPG0015 - Vamos manter as informações!)

Objetivo da Prática:

O objetivo desta prática foi modelar e criar um banco de dados para um sistema comercial, utilizando SQL Server. O sistema gerencia usuários, produtos e movimentações de compra e venda, com diferenciação entre pessoas físicas e jurídicas. Durante a prática, foram realizadas as seguintes tarefas:

1. Modelagem do banco de dados, criação de tabelas e inserção de dados.
2. Utilização de sequências para geração de identificadores.
3. Execução de consultas para análise de movimentações de entrada e saída de produtos, valores e dados dos usuários.

Códigos solicitados

1. Criação do Banco de Dados e Estrutura das Tabelas

```
-- Criação do Banco de Dados

CREATE DATABASE loja;

GO


-- Usar o banco de dados
```

```
USE loja;
GO

-- Criação das tabelas
-- Tabela de Usuários
CREATE TABLE Usuarios (
  id_usuario INT IDENTITY PRIMARY KEY,
  nome VARCHAR(100),
  senha VARCHAR(100)
);
GO

-- Tabela de Produtos
CREATE TABLE Produtos (
  id_produto INT IDENTITY PRIMARY KEY,
  nome VARCHAR(100),
  quantidade INT,
  preco_venda DECIMAL(10, 2)
);
GO

-- Tabela de Pessoa
CREATE TABLE Pessoa (
  id_pessoa INT IDENTITY PRIMARY KEY,
  nome VARCHAR(100),
  endereco VARCHAR(200),
  telefone VARCHAR(20),
  tipo_pessoa VARCHAR(20) -- Pessoa Física ou Jurídica
```

```
);
GO

-- Tabela de Pessoa Física
CREATE TABLE PessoaFisica (
  id_pessoa INT PRIMARY KEY,
  cpf VARCHAR(14),
  FOREIGN KEY (id_pessoa) REFERENCES Pessoa(id_pessoa)
);
GO

-- Tabela de Pessoa Jurídica
CREATE TABLE PessoaJuridica (
  id_pessoa INT PRIMARY KEY,
  cnpj VARCHAR(18),
  FOREIGN KEY (id_pessoa) REFERENCES Pessoa(id_pessoa)
);
GO

-- Tabela de Movimentações (Compra e Venda)
CREATE TABLE Movimentacoes (
  id_movimentacao INT IDENTITY PRIMARY KEY,
  tipo_movimentacao CHAR(1),  -- 'E' para entrada (compra), 'S' para saída (venda)
  id_produto INT,
  id_usuario INT,
  id_pessoa INT,
  quantidade INT,
  preco_unitario DECIMAL(10, 2),
```

```
  valor_total DECIMAL(10, 2) AS (quantidade * preco_unitario) PERSISTED, -- Valor total (quantidade * preço unitário)
  data_movimentacao DATETIME DEFAULT GETDATE(),
  FOREIGN KEY (id_produto) REFERENCES Produtos(id_produto),
  FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario),
  FOREIGN KEY (id_pessoa) REFERENCES Pessoa(id_pessoa)
);
GO
```

2. Inserção de Dados

```
-- Inserir Usuários
INSERT INTO Usuarios (nome, senha) VALUES ('op1', 'op1');
INSERT INTO Usuarios (nome, senha) VALUES ('op2', 'op2');
GO

-- Inserir Produtos
INSERT INTO Produtos (nome, quantidade, preco_venda) VALUES ('Produto A', 100, 50.00);
INSERT INTO Produtos (nome, quantidade, preco_venda) VALUES ('Produto B', 200, 30.00);
INSERT INTO Produtos (nome, quantidade, preco_venda) VALUES ('Produto C', 150, 70.00);
GO

-- Inserir Pessoas Físicas
INSERT INTO Pessoa (nome, endereco, telefone, tipo_pessoa) VALUES ('João Silva', 'Rua A, 123', '1234567890', 'Física');
INSERT INTO PessoaFisica (id_pessoa, cpf) VALUES (1, '123.456.789-00');
GO
```

-- Inserir Pessoas Jurídicas

INSERT INTO Pessoa (nome, endereco, telefone, tipo_pessoa) VALUES ('Empresa X', 'Avenida B, 456', '9876543210', 'Jurídica');

INSERT INTO PessoaJuridica (id_pessoa, cnpj) VALUES (2, '12.345.678/0001-99');

GO

-- Inserir Movimentações (Compras e Vendas)

-- Movimentação de Compra

INSERT INTO Movimentacoes (tipo_movimentacao, id_produto, id_usuario, id_pessoa, quantidade, preco_unitario)

VALUES ('E', 1, 1, 2, 50, 40.00); -- Compra de 50 unidades do Produto A, pelo usuário op1 para a empresa X

-- Movimentação de Venda

INSERT INTO Movimentacoes (tipo_movimentacao, id_produto, id_usuario, id_pessoa, quantidade, preco_unitario)

VALUES ('S', 1, 2, 1, 20, 50.00); -- Venda de 20 unidades do Produto A, pelo usuário op2 para João Silva

GO

3 – Consultas

-- Dados completos de pessoas físicas

SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, pf.cpf

FROM Pessoa p

INNER JOIN PessoaFisica pf ON p.id_pessoa = pf.id_pessoa

WHERE p.tipo_pessoa = 'Física';

-- Dados completos de pessoas jurídicas

SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, pj.cnpj

FROM Pessoa p

INNER JOIN PessoaJuridica pj ON p.id_pessoa = pj.id_pessoa

WHERE p.tipo_pessoa = 'Jurídica';

```
-- Movimentações de entrada (compra)
SELECT m.tipo_movimentacao, p.nome AS produto, pj.nome AS fornecedor,
m.quantidade, m.preco_unitario, m.valor_total
FROM Movimentacoes m
INNER JOIN Produtos p ON m.id_produto = p.id_produto
INNER JOIN PessoaJuridica pj ON m.id_pessoa = pj.id_pessoa
WHERE m.tipo_movimentacao = 'E';

-- Movimentações de saída (venda)
SELECT m.tipo_movimentacao, p.nome AS produto, pf.nome AS comprador,
m.quantidade, m.preco_unitario, m.valor_total
FROM Movimentacoes m
INNER JOIN Produtos p ON m.id_produto = p.id_produto
INNER JOIN PessoaFisica pf ON m.id_pessoa = pf.id_pessoa
WHERE m.tipo_movimentacao = 'S';

-- Valor total das entradas agrupadas por produto
SELECT p.nome AS produto, SUM(m.valor_total) AS valor_total_entrada
FROM Movimentacoes m
INNER JOIN Produtos p ON m.id_produto = p.id_produto
WHERE m.tipo_movimentacao = 'E'
GROUP BY p.nome;

-- Valor total das saídas agrupadas por produto
SELECT p.nome AS produto, SUM(m.valor_total) AS valor_total_saida
FROM Movimentacoes m
INNER JOIN Produtos p ON m.id_produto = p.id_produto
WHERE m.tipo_movimentacao = 'S'
```

```
GROUP BY p.nome;

-- Operadores que não efetuaram movimentações de entrada (compra)
SELECT u.nome AS operador
FROM Usuarios u
WHERE NOT EXISTS (
  SELECT 1
  FROM Movimentacoes m
  WHERE m.id_usuario = u.id_usuario AND m.tipo_movimentacao = 'E'
);

-- Valor total de entrada, agrupado por operador
SELECT u.nome AS operador, SUM(m.valor_total) AS valor_total_entrada
FROM Movimentacoes m
INNER JOIN Usuarios u ON m.id_usuario = u.id_usuario
WHERE m.tipo_movimentacao = 'E'
GROUP BY u.nome;

-- Valor total de saída, agrupado por operador
SELECT u.nome AS operador, SUM(m.valor_total) AS valor_total_saida
FROM Movimentacoes m
INNER JOIN Usuarios u ON m.id_usuario = u.id_usuario
WHERE m.tipo_movimentacao = 'S'
GROUP BY u.nome;

-- Valor médio de venda por produto, utilizando média ponderada
SELECT p.nome AS produto,
    SUM(m.valor_total) / SUM(m.quantidade) AS valor_medio_venda
```

```
FROM Movimentacoes m

INNER JOIN Produtos p ON m.id_produto = p.id_produto

WHERE m.tipo_movimentacao = 'S'

GROUP BY p.nome;
```

Resultados da Execução

Após a execução dos códigos, os resultados esperados foram obtidos:

1. As tabelas foram criadas com sucesso.
2. Os dados de usuários, produtos, pessoas físicas e jurídicas foram inseridos corretamente.
3. As movimentações de compra e venda foram registradas conforme esperado.
4. As consultas para dados completos de pessoas físicas e jurídicas, movimentações de entrada e saída, e valores totais de movimentações foram executadas corretamente.

Análise e Conclusão

1. **Diferenças entre SEQUENCE e IDENTITY:**
    - **IDENTITY:** Garante que os valores dos identificadores sejam gerados automaticamente à medida que os dados são inseridos. Sua principal limitação é que não permite reuso ou controle manual do valor gerado.
    - **SEQUENCE:** Permite maior controle sobre a geração de valores, sendo possível reiniciar ou alterar o valor inicial da sequência, e pode ser chamada explicitamente durante inserções. É mais flexível que o IDENTITY.
2. **Importância das Chaves Estrangeiras:** As chaves estrangeiras são essenciais para garantir a integridade referencial do banco de dados, ou seja, para assegurar que os dados em tabelas relacionadas estejam consistentes. Elas evitam a inserção de registros inválidos e garantem que um registro não seja excluído se houver dependências em outras tabelas.
3. **Operadores do SQL pertencentes à Álgebra Relacional e Cálculo Relacional:**

- **Álgebra Relacional:** Inclui operadores como SELECT, JOIN, UNION, INTERSECT, DIFFERENCE e PRODUCT.
- **Cálculo Relacional:** Usa predicados lógicos, como WHERE e HAVING, em expressões para recuperar dados que atendam a certos critérios.

4. **Agrupamento em Consultas e Requisito Obrigatório:** O agrupamento em consultas é feito utilizando o operador GROUP BY. O requisito obrigatório para agrupar dados é que todos os campos não agregados na consulta devem ser mencionados na cláusula GROUP BY.