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

```
-- Criando o banco de dados
CREATE DATABASE Loja;
GO

-- Usando o banco de dados
USE Loja;
GO

-- Criando a sequence para os IDs de Pessoa
CREATE SEQUENCE Seq_PessoaID
```

```
    START WITH 1
    INCREMENT BY 1;
GO

-- Criando a tabela de Usuários
CREATE TABLE Usuarios (
    UsuarioID INT IDENTITY(1,1) PRIMARY KEY,
    Nome NVARCHAR(100) NOT NULL,
    Senha NVARCHAR(100) NOT NULL
);

-- Criando a tabela de Pessoas
CREATE TABLE Pessoas (
    PessoaID INT PRIMARY KEY DEFAULT (NEXT VALUE FOR Seq_PessoaID),
    Nome NVARCHAR(100) NOT NULL,
    Endereco NVARCHAR(255),
    Telefone NVARCHAR(20)
);

-- Criando a tabela de Pessoas Físicas
CREATE TABLE PessoasFisicas (
    PessoaID INT PRIMARY KEY,
    CPF CHAR(11) UNIQUE NOT NULL,
    FOREIGN KEY (PessoaID) REFERENCES Pessoas(PessoaID) ON DELETE CASCADE
);

-- Criando a tabela de Pessoas Jurídicas
CREATE TABLE PessoasJuridicas (
    PessoaID INT PRIMARY KEY,
    CNPJ CHAR(14) UNIQUE NOT NULL,
    FOREIGN KEY (PessoaID) REFERENCES Pessoas(PessoaID) ON DELETE CASCADE
);

-- Criando a tabela de Produtos
CREATE TABLE Produtos (
    ProdutoID INT IDENTITY(1,1) PRIMARY KEY,
    Nome NVARCHAR(100) NOT NULL,
    Quantidade INT NOT NULL,
    PrecoVenda DECIMAL(10,2) NOT NULL
);
```

```
-- Criando a tabela de Movimentações
CREATE TABLE Movimentacoes (
    MovimentacaoID INT IDENTITY(1,1) PRIMARY KEY,
    Tipo CHAR(1) CHECK (Tipo IN ('E', 'S')) NOT NULL,
    UsuarioID INT NOT NULL,
    ProdutoID INT NOT NULL,
    PessoaID INT NOT NULL,
    Quantidade INT NOT NULL,
    PrecoUnitario DECIMAL(10,2) NOT NULL,
    DataMovimentacao DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (UsuarioID) REFERENCES Usuarios(UsuarioID),
    FOREIGN KEY (ProdutoID) REFERENCES Produtos(ProdutoID),
    FOREIGN KEY (PessoaID) REFERENCES Pessoas(PessoaID)
);
```

2. Inserção de Dados

```
-- Usando o banco de dados

USE LojaDB;

GO


-- Inserindo usuários

INSERT INTO Usuarios (Nome, Senha) VALUES ('op1', 'op1');

INSERT INTO Usuarios (Nome, Senha) VALUES ('op2', 'op2');


-- Inserindo produtos

INSERT INTO Produtos (Nome, Quantidade, PrecoVenda) VALUES ('Produto A',
50, 10.00);

INSERT INTO Produtos (Nome, Quantidade, PrecoVenda) VALUES ('Produto B',
30, 20.50);

INSERT INTO Produtos (Nome, Quantidade, PrecoVenda) VALUES ('Produto C',
100, 5.75);


-- Inserindo pessoas físicas
```

```
DECLARE @PessoaID INT = NEXT VALUE FOR Seq_PessoaID;

INSERT INTO Pessoas (PessoaID, Nome, Endereco, Telefone) VALUES (@PessoaID,
'João Silva', 'Rua 1, 100', '11987654321');

INSERT INTO PessoasFisicas (PessoaID, CPF) VALUES (@PessoaID,
'12345678901');


SET @PessoaID = NEXT VALUE FOR Seq_PessoaID;

INSERT INTO Pessoas (PessoaID, Nome, Endereco, Telefone) VALUES (@PessoaID,
'Maria Souza', 'Rua 2, 200', '11976543210');

INSERT INTO PessoasFisicas (PessoaID, CPF) VALUES (@PessoaID,
'98765432100');


-- Inserindo pessoas jurídicas

SET @PessoaID = NEXT VALUE FOR Seq_PessoaID;

INSERT INTO Pessoas (PessoaID, Nome, Endereco, Telefone) VALUES (@PessoaID,
'Empresa XYZ', 'Av. Industrial, 500', '1133324455');

INSERT INTO PessoasJuridicas (PessoaID, CNPJ) VALUES (@PessoaID,
'11222333444455');


SET @PessoaID = NEXT VALUE FOR Seq_PessoaID;

INSERT INTO Pessoas (PessoaID, Nome, Endereco, Telefone) VALUES (@PessoaID,
'Empresa ABC', 'Av. Central, 600', '1144455566');

INSERT INTO PessoasJuridicas (PessoaID, CNPJ) VALUES (@PessoaID,
'55667788990011');


-- Inserindo movimentações de entrada (compra de produtos)

INSERT INTO Movimentacoes (Tipo, UsuarioID, ProdutoID, PessoaID, Quantidade,
PrecoUnitario)

VALUES ('E', 1, 1, 3, 20, 9.50);
```

```
INSERT INTO Movimentacoes (Tipo, UsuarioID, ProdutoID, PessoaID, Quantidade, PrecoUnitario)

VALUES ('E', 2, 2, 4, 10, 19.00);


-- Inserindo movimentações de saída (venda de produtos)

INSERT INTO Movimentacoes (Tipo, UsuarioID, ProdutoID, PessoaID, Quantidade, PrecoUnitario)

VALUES ('S', 1, 1, 1, 5, 10.00);


INSERT INTO Movimentacoes (Tipo, UsuarioID, ProdutoID, PessoaID, Quantidade, PrecoUnitario)

VALUES ('S', 2, 2, 2, 3, 20.50);
```

3 – Consultas

```
-- Usando o banco de dados

USE LojaDB;

GO


-- 1. Dados completos de pessoas físicas

SELECT P.*, PF.CPF

FROM Pessoas P

JOIN PessoasFisicas PF ON P.PessoaID = PF.PessoaID;


-- 2. Dados completos de pessoas jurídicas

SELECT P.*, PJ.CNPJ

FROM Pessoas P

JOIN PessoasJuridicas PJ ON P.PessoaID = PJ.PessoaID;


-- 3. Movimentações de entrada (compras)
```

```
SELECT M.MovimentacaoID, M.DataMovimentacao, U.Nome AS Operador, P.Nome AS Produto, Pe.Nome AS Fornecedor, M.Quantidade, M.PrecoUnitario, (M.Quantidade * M.PrecoUnitario) AS ValorTotal

FROM Movimentacoes M

JOIN Usuarios U ON M.UsuarioID = U.UsuarioID

JOIN Produtos P ON M.ProdutoID = P.ProdutoID

JOIN Pessoas Pe ON M.PessoaID = Pe.PessoaID

WHERE M.Tipo = 'E';


-- 4. Movimentações de saída (vendas)

SELECT M.MovimentacaoID, M.DataMovimentacao, U.Nome AS Operador, P.Nome AS Produto, Pe.Nome AS Comprador, M.Quantidade, M.PrecoUnitario, (M.Quantidade * M.PrecoUnitario) AS ValorTotal

FROM Movimentacoes M

JOIN Usuarios U ON M.UsuarioID = U.UsuarioID

JOIN Produtos P ON M.ProdutoID = P.ProdutoID

JOIN Pessoas Pe ON M.PessoaID = Pe.PessoaID

WHERE M.Tipo = 'S';


-- 5. Valor total das entradas agrupadas por produto

SELECT P.Nome AS Produto, SUM(M.Quantidade * M.PrecoUnitario) AS ValorTotal

FROM Movimentacoes M

JOIN Produtos P ON M.ProdutoID = P.ProdutoID

WHERE M.Tipo = 'E'

GROUP BY P.Nome;


-- 6. Valor total das saídas agrupadas por produto

SELECT P.Nome AS Produto, SUM(M.Quantidade * M.PrecoUnitario) AS ValorTotal

FROM Movimentacoes M
```

```
JOIN Produtos P ON M.ProdutoID = P.ProdutoID

WHERE M.Tipo = 'S'

GROUP BY P.Nome;


-- 7. Operadores que não efetuaram movimentações de entrada

SELECT U.Nome

FROM Usuarios U

WHERE U.UsuarioID NOT IN (SELECT DISTINCT UsuarioID FROM Movimentacoes WHERE
Tipo = 'E');


-- 8. Valor total de entrada agrupado por operador

SELECT U.Nome AS Operador, SUM(M.Quantidade * M.PrecoUnitario) AS ValorTotal

FROM Movimentacoes M

JOIN Usuarios U ON M.UsuarioID = U.UsuarioID

WHERE M.Tipo = 'E'

GROUP BY U.Nome;


-- 9. Valor total de saída agrupado por operador

SELECT U.Nome AS Operador, SUM(M.Quantidade * M.PrecoUnitario) AS ValorTotal

FROM Movimentacoes M

JOIN Usuarios U ON M.UsuarioID = U.UsuarioID

WHERE M.Tipo = 'S'

GROUP BY U.Nome;


-- 10. Valor médio de venda por produto (média ponderada)

SELECT P.Nome AS Produto, SUM(M.Quantidade * M.PrecoUnitario) /
NULLIF(SUM(M.Quantidade), 0) AS ValorMedioVenda

FROM Movimentacoes M

JOIN Produtos P ON M.ProdutoID = P.ProdutoID
```

```
WHERE M.Tipo = '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.