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

SQL Server.

Eros Santos de Vasconcelos/ 202307120545

Polo Iputinga
Vamos Manter as Informações? – 9001 – 3° semestre

Objetivo da Prática

O objetivo desta prática é guiar o desenvolvimento de um banco de dados relacional utilizando o **SQL Server**. A prática envolve a identificação dos requisitos de um sistema de cadastro de usuários, pessoas físicas e jurídicas, produtos, além de transações de compra e venda. Usando a ferramenta **DBDesigner Fork** para modelagem e o **SQL Server Management Studio (SSMS)** para implementação, a atividade visa simular a estrutura de um sistema simples e a execução de operações com dados. Ao final, espera-se que o banco de dados seja criado e testado, consolidando o conhecimento de modelagem e **SQL (DDL e DML)**.

1º Procedimento | Criando o Banco de Dados

Códigos do 1º procedimento:

1. **Criação do Banco de Dados:**

```
CREATE LOGIN loja WITH PASSWORD = 'loja';
CREATE USER loja FOR LOGIN loja;
ALTER SERVER ROLE sysadmin ADD MEMBER loja;

CREATE DATABASE SistemaLoja;

USE SistemaLoja;
```

2. **Criação das Tabelas:**

- **Tabela Usuarios:**

```
CREATE TABLE Usuarios (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Nome VARCHAR(100),
    Login VARCHAR(50),
    Senha VARCHAR(50)

);
```

- **Tabela Pessoas:**

```
CREATE TABLE Pessoas (
    ID_Pessoa INT IDENTITY(1,1) PRIMARY KEY,
    Nome VARCHAR(100),
    Endereco VARCHAR(100),
    Telefone VARCHAR(15),
    Email VARCHAR(50)
);
```

- **Tabelas PessoasFisicas e PessoasJuridicas:**

```
CREATE TABLE PessoasFisicas (
    ID_Pessoa INT PRIMARY KEY,
    CPF VARCHAR(11),
    FOREIGN KEY (ID_Pessoa) REFERENCES Pessoas(ID_Pessoa)
);

CREATE TABLE PessoasJuridicas (
    ID_Pessoa INT PRIMARY KEY,
    CNPJ VARCHAR(14),
    FOREIGN KEY (ID_Pessoa) REFERENCES Pessoas(ID_Pessoa)
);
```

- **Tabela Produtos:**

```
CREATE TABLE Produtos (
    ID_Produto INT IDENTITY(1,1) PRIMARY KEY,
    Nome VARCHAR(100),
    Quantidade INT,
    PrecoVenda DECIMAL(10, 2)
);
```

- **Tabelas de Movimentos (Compras e Vendas):**

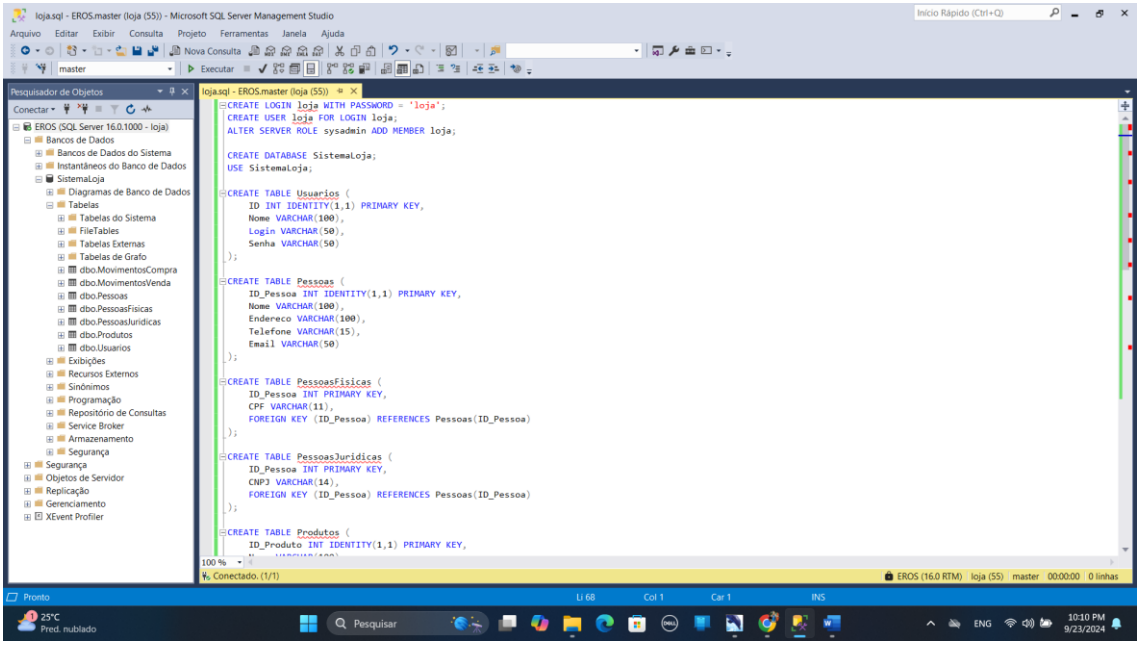
```
CREATE TABLE MovimentosCompra (
    ID_Compra INT IDENTITY(1,1) PRIMARY KEY,
    ID_Usuario INT,
    ID_Produto INT,
    ID_PessoaJuridica INT,
    Quantidade INT,
    PrecoUnitario DECIMAL(10, 2),
    FOREIGN KEY (ID_Usuario) REFERENCES Usuarios(ID),
    FOREIGN KEY (ID_Produto) REFERENCES Produtos(ID_Produto),
    FOREIGN KEY (ID_PessoaJuridica) REFERENCES PessoasJuridicas(ID_Pessoa)
);

CREATE TABLE MovimentosVenda (
    ID_Venda INT IDENTITY(1,1) PRIMARY KEY,
    ID_Usuario INT,
    ID_Produto INT,
    ID_PessoaFisica INT,
    Quantidade INT,
    PrecoVenda DECIMAL(10, 2),
    FOREIGN KEY (ID_Usuario) REFERENCES Usuarios(ID),
    FOREIGN KEY (ID_Produto) REFERENCES Produtos(ID_Produto),
    FOREIGN KEY (ID_PessoaFisica) REFERENCES PessoasFisicas(ID_Pessoa)
);
```

3. **Geração de Identificadores com Sequence:**

```
CREATE SEQUENCE Seq_Pessoa AS INT START WITH 1 INCREMENT BY 1;
```

Resultado da execução:

Os comandos foram executados com sucesso no SQL Server Management Studio. As tabelas foram criadas, e o banco de dados foi configurado corretamente para gerenciar usuários, pessoas, produtos e movimentações de compra e venda.



a) **Como são implementadas as diferentes cardinalidades, basicamente 1x1, 1xN ou NxN, em um banco de dados relacional?**

1x1 é quando um registro de uma tabela só pode se relacionar com um registro de outra tabela, e vice-versa. Cada pessoa só pode ter um CPF, e esse CPF pertence a uma única

pessoa. Então ligamos as duas tabelas usando uma chave estrangeira que faz referência à chave primária da outra tabela.

A pessoa com ID 1 na tabela Pessoas vai se ligar ao CPF na tabela PessoasFisicas também com o ID 1.

1xN é quando um registro de uma tabela pode se relacionar com vários registros de outra. Onde um usuário pode fazer várias compras. Então, um único usuário na tabela Usuarios pode estar relacionado a várias linhas na tabela MovimentosCompra.

Onde você adiciona uma chave estrangeira na tabela de "muitos" (MovimentosCompra), para apontar para a tabela do "um" (Usuarios).

NxN é quando muitos registros de uma tabela podem se relacionar com muitos registros de outra. Um exemplo seria estudante e cursos – um estudante pode se matricular em vários cursos, e um curso pode ter vários estudantes. Para fazer isso, criamos uma tabela intermediária que liga essas duas tabelas, contendo as chaves de ambas.

b) Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?

No banco de dados, herança é representada criando uma tabela geral para os atributos comuns e outras tabelas para as características específicas. Por exemplo, para armazenar informações sobre pessoas, temos uma tabela Pessoas com informações comuns como nome e endereço. Depois, criamos tabelas separadas para PessoasFisicas e PessoasJuridicas que armazenam o CPF ou CNPJ.

Para conectar essas tabelas, usamos um relacionamento 1x1. A tabela PessoasFisicas, por exemplo, terá uma chave que faz referência a tabela Pessoas. Assim, cada registro na tabela PessoasFisicas está ligado a um registro em Pessoas.

c) **Como o SQL Server Management Studio melhora a produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?**

O SSMS tem uma interface gráfica que permite você criar tabelas, consultar dados e gerenciar o banco de dados sem precisar digitar tudo manualmente.

Ele proporciona organização visual mostrando todos os objetos do banco de dados (tabelas, consultas, funções) de forma organizada. Assim, fica mais fácil encontrar o que você precisa.

Ele agiliza o processo completando automaticamente comandos e verificando erros enquanto você digita, evitando erros.

2º Procedimento | Alimentando a Base

Códigos do 2º procedimento:

1. **Inserindo Usuários:**

```
INSERT INTO Usuarios (Nome, Login, Senha) VALUES
('Operador 1', 'op1', 'op1'),
('Operador 2', 'op2', 'op2');
```

2. **Inserindo Produtos:**

```
INSERT INTO Produtos (Nome, Quantidade, PrecoVenda) VALUES
('Produto A', 100, 10.00),
('Produto B', 200, 15.50),
('Produto C', 150, 7.25);
```

3. **Inserindo Pessoas Físicas e Jurídicas:**

```
INSERT INTO Pessoas (Nome, Endereco, Telefone, Email)
VALUES ('João da Silva', 'Rua A, 123', '123456789', 'joao@example.com');

DECLARE @ID_PessoaFisica INT = SCOPE_IDENTITY();

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

INSERT INTO Pessoas (Nome, Endereco, Telefone, Email)
VALUES ('Empresa XYZ', 'Avenida C, 789', '456789123', 'contato@xyz.com');

DECLARE @ID_PessoaJuridica INT = SCOPE_IDENTITY();

INSERT INTO PessoasJuridicas (ID_Pessoa, CNPJ)
VALUES (@ID_PessoaJuridica, '12345678000195');
```

4. Criando Movimentações:

```
INSERT INTO MovimentosCompra (ID_Usuario, ID_Produto, ID_PessoaJuridica,
Quantidade, PrecoUnitario) VALUES
(11, 16, 26, 10, 10.00),
(11, 17, 26, 5, 15.50);

INSERT INTO MovimentosVenda (ID_Usuario, ID_Produto, ID_PessoaFisica, Quantidade,
PrecoVenda) VALUES
(11, 16, 25, 2, 10.00),
(11, 17, 25, 1, 15.50);
```

5. Consultando os dados inseridos:

```
SELECT P.ID_Pessoa, P.Nome, P.Endereco, P.Telefone, P.Email, PF.CPF
FROM Pessoas P
JOIN PessoasFisicas PF ON P.ID_Pessoa = PF.ID_Pessoa;

SELECT P.ID_Pessoa, P.Nome, P.Endereco, P.Telefone, P.Email, PJ.CNPJ
FROM Pessoas P
JOIN PessoasJuridicas PJ ON P.ID_Pessoa = PJ.ID_Pessoa;

SELECT
    P.Nome AS Fornecedor,
    Prod.Nome AS Produto,
    MC.Quantidade,
    MC.PrecoUnitario,
    MC.Quantidade * MC.PrecoUnitario AS ValorTotal
FROM MovimentosCompra MC
JOIN Produtos Prod ON MC.ID_Produto = Prod.ID_Produto
JOIN PessoasJuridicas PJ ON MC.ID_PessoaJuridica = PJ.ID_Pessoa
JOIN Pessoas P ON PJ.ID_Pessoa = P.ID_Pessoa;

SELECT
    P.Nome AS Comprador,
    Prod.Nome AS Produto,
    MV.Quantidade,
    MV.PrecoVenda,
    MV.Quantidade * MV.PrecoVenda AS ValorTotal
FROM MovimentosVenda MV
JOIN Produtos Prod ON MV.ID_Produto = Prod.ID_Produto
JOIN PessoasFisicas PF ON MV.ID_PessoaFisica = PF.ID_Pessoa
JOIN Pessoas P ON PF.ID_Pessoa = P.ID_Pessoa;

SELECT
    Prod.Nome AS Produto,
    SUM(MC.Quantidade * MC.PrecoUnitario) AS ValorTotalEntradas
FROM MovimentosCompra MC
JOIN Produtos Prod ON MC.ID_Produto = Prod.ID_Produto
GROUP BY Prod.Nome;

SELECT U.*
FROM Usuarios U
WHERE NOT EXISTS (
    SELECT 1
    FROM MovimentosCompra MC
    WHERE MC.ID_Usuario = U.ID
);

SELECT
```

```
    U.Nome AS Operador,
    SUM(MC.Quantidade * MC.PrecoUnitario) AS ValorTotalEntradas
FROM MovimentosCompra MC
JOIN Usuarios U ON MC.ID_Usuario = U.ID
GROUP BY U.Nome;

SELECT
    U.Nome AS Operador,
    SUM(MV.Quantidade * MV.PrecoVenda) AS ValorTotalSaidas
FROM MovimentosVenda MV
JOIN Usuarios U ON MV.ID_Usuario = U.ID
GROUP BY U.Nome;

SELECT
    Prod.Nome AS Produto,
    SUM(MV.PrecoVenda * MV.Quantidade) / SUM(MV.Quantidade) AS ValorMedioVenda
FROM MovimentosVenda MV
JOIN Produtos Prod ON MV.ID_Produto = Prod.ID_Produto
GROUP BY Prod.Nome;
```

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

Diferenças:

- **Identity**: É uma propriedade de coluna que gera valores únicos automaticamente para cada nova linha. É gerenciado pelo SQL Server e não pode ser alterado diretamente após a criação.
- **Sequence**: É um objeto separado que pode ser utilizado para gerar números únicos em várias tabelas. Oferece maior flexibilidade, permitindo controle sobre o valor inicial, incremento, e possibilidade de serem chamadas em diferentes contextos.

b. Qual a importância das chaves estrangeiras para a consistência do banco?

As chaves estrangeiras são fundamentais para garantir a integridade referencial entre as tabelas. Elas asseguram que os valores de uma coluna em uma tabela (chave estrangeira) correspondam a valores existentes em outra tabela (chave primária). Isso previne dados órfãos e mantém a consistência do banco de dados, evitando inserções e atualizações que poderiam criar inconsistências.

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

Álgebra Relacional:

- Seleção (σ)
- Projeção (π)

- União (∪)
- Diferença (-)
- Produto Cartesiano (×)
- Junção (⋈)

Cálculo Relacional:

- O cálculo relacional é baseado em expressões que especificam as condições que os dados devem satisfazer, e não em como os dados devem ser recuperados. Ele é dividido em:
    - Cálculo relacional de tuplas
    - Cálculo relacional de domínios

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

O agrupamento em consultas SQL é feito utilizando a cláusula GROUP BY, que agrupa os resultados com base em uma ou mais colunas especificadas. O requisito obrigatório é que todas as colunas na cláusula SELECT que não estão dentro de uma função de agregação (como SUM, COUNT, AVG) devem estar listadas na cláusula GROUP BY.

Conclusão

A prática de alimentação da base de dados foi fundamental para compreender a estrutura e o funcionamento de um banco de dados relacional. A inserção de dados, juntamente com o uso de chaves estrangeiras e sequences, permitiu entender a importância da integridade referencial e da geração automática de identificadores. As consultas realizadas posteriormente ilustraram como extrair informações significativas de dados armazenados, utilizando conceitos de álgebra e cálculo relacionais.