Estácio

Campus: Polo Taguatinga Sul
Curso: **Desenvolvimento Full Stack**
Disciplina: Vamos Manter as Informações? (RPG0015)
Turma: 9001
Semestre: 3º Semestre
Integrantes: Yan Silva Sales

Criando Banco de dados

Objetivo da Pratica:

1. Identificar os requisitos de um sistema e transformá-los no modelo adequado.
2. Utilizar ferramentas de modelagem para bases de dados relacionais.
3. Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
4. Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
5. 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.

Códigos:

```sql
-- Criação do banco de dados

CREATE DATABASE SistemaComercial;

USE SistemaComercial;

-- Sequence para geração de IDs para Pessoa
CREATE SEQUENCE seq_pessoa
    AS INT
    START WITH 1
    INCREMENT BY 1;

-- Tabela de Usuários
CREATE TABLE Usuarios (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Nome VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    Senha VARCHAR(255), -- Deverá ser armazenada com hash
    Tipo VARCHAR(50)
);

-- Tabela de Pessoas
CREATE TABLE Pessoas (
    ID INT DEFAULT (NEXT VALUE FOR seq_pessoa) PRIMARY KEY,
    Nome VARCHAR(100),
    Endereço VARCHAR(255),
    Telefone VARCHAR(20),
    Email VARCHAR(100)
);

-- Tabela de Pessoa Física
CREATE TABLE Pessoa_Fisica (
    PessoaID INT PRIMARY KEY,
    CPF VARCHAR(11) UNIQUE,
    FOREIGN KEY (PessoaID) REFERENCES Pessoas(ID)
);
```

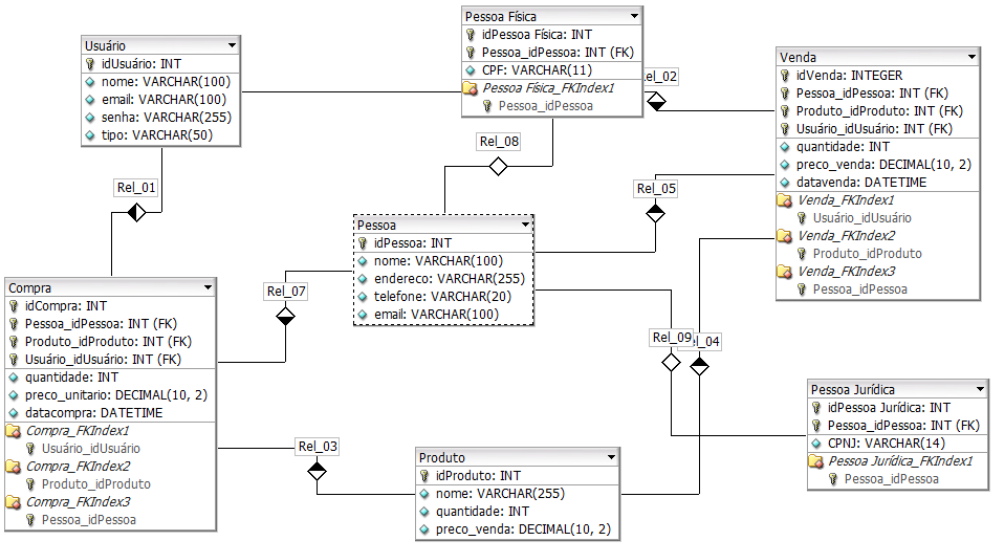
```sql
-- Tabela de Pessoa Jurídica
CREATE TABLE Pessoa_Juridica (
    PessoaID INT PRIMARY KEY,
    CNPJ VARCHAR(14) UNIQUE,
    FOREIGN KEY (PessoaID) REFERENCES Pessoas(ID)
);

-- Tabela de Produtos
CREATE TABLE Produtos (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Nome VARCHAR(100),
    Quantidade INT,
    PreçoVenda DECIMAL(10, 2)
);

-- Tabela de Compras
CREATE TABLE Compras (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    ProdutoID INT,
    OperadorID INT,
    PessoaJuridicaID INT,
    Quantidade INT,
    PreçoUnitario DECIMAL(10, 2),
    DataCompra DATETIME,
    FOREIGN KEY (ProdutoID) REFERENCES Produtos(ID),
    FOREIGN KEY (OperadorID) REFERENCES Usuarios(ID),
    FOREIGN KEY (PessoaJuridicaID) REFERENCES Pessoa_Juridica(PessoaID)
);

-- Tabela de Vendas
CREATE TABLE Vendas (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    ProdutoID INT,
    OperadorID INT,
    PessoaFisicaID INT,
    Quantidade INT,
    PreçoUnitario DECIMAL(10, 2),
    DataVenda DATETIME,
    FOREIGN KEY (ProdutoID) REFERENCES Produtos(ID),
    FOREIGN KEY (OperadorID) REFERENCES Usuarios(ID),
    FOREIGN KEY (PessoaFisicaID) REFERENCES Pessoa_Fisica(PessoaID)
);
```

Resultados:



Conclusão

Com base nas atividades anteriores, foi possível analisar como as diferentes cardinalidades são implementadas em um banco de dados relacional, bem como a melhor forma de representar conceitos de herança. Também discutimos como o SQL Server Management Studio (SSMS) contribui para a melhoria da produtividade nas tarefas relacionadas ao gerenciamento de banco de dados.

Implementação de Cardinalidades

Em bancos de dados relacionais, a cardinalidade define como as tabelas estão relacionadas entre si. No caso do modelo desenvolvido, observamos os seguintes padrões:

- **Cardinalidade 1x1 (Um para Um)**: Essa configuração é usada quando para cada registro em uma tabela existe no máximo um registro correspondente em outra tabela. A implementação comum é por meio de chaves primárias compartilhadas ou chaves estrangeiras únicas. No modelo, vimos essa configuração entre "Pessoas" e "Pessoa_Física" ou "Pessoa_Jurídica".
- **Cardinalidade 1xN (Um para Muitos)**: Aqui, um registro em uma tabela pode estar associado a vários registros em outra tabela. Essa relação é implementada por meio de chaves estrangeiras. No exemplo, observamos essa cardinalidade entre "Produtos" e "Compras", e entre "Produtos" e "Vendas".

- **Cardinalidade NxN (Muitos para Muitos)**: Essa cardinalidade permite que vários registros de uma tabela estejam associados a vários registros de outra tabela. Isso geralmente é implementado com uma tabela de junção. Embora não tenhamos um exemplo explícito desta cardinalidade, descrevemos como ela seria usada para situações de relacionamento entre duas tabelas com muitos para muitos.

Representação de Herança

No contexto de bancos de dados relacionais, a herança é representada de forma diferente do que em linguagens de programação orientadas a objetos. No modelo, a herança é simulada com relacionamentos 1x1 entre uma tabela base e suas tabelas derivadas. O uso de uma chave primária compartilhada permite simular a herança, como visto entre "Pessoas" e suas subclasses "Pessoa_Física" e "Pessoa_Jurídica".

SQL Server Management Studio (SSMS)

O SQL Server Management Studio (SSMS) é uma ferramenta abrangente para gerenciamento de bancos de dados. Ele melhora a produtividade ao oferecer uma interface gráfica intuitiva, um editor SQL robusto e ferramentas para gerenciamento, otimização e monitoramento de banco de dados. Os recursos do SSMS, como o Object Explorer, editor SQL com destaque de sintaxe, ferramentas de análise de desempenho e backup/restauração, facilitam a gestão de banco de dados e a resolução de problemas.

Considerações Finais

Este estudo demonstrou como diferentes cardinalidades são implementadas em bancos de dados relacionais e como a herança pode ser simulada com relacionamentos 1x1. Também exploramos a importância do SQL Server Management Studio para aumentar a produtividade no gerenciamento de bancos de dados. O modelo criado oferece um exemplo prático de como estruturar e manter um banco de dados relacional com essas características.