



# Estácio

Estácio - Unidade São Pedro

RJ 140 Km 2, 512 loja 1, São Pedro da Aldeia - RJ, 28941-182

Desenvolvimento Full Stack

Classe: Missão Prática | Nível 2 | Mundo 3

3º Semestre

Marvin de Almeida Costa

## **Título da Prática: 1º Procedimento | Criando o Banco de Dados**

### **Objetivo da Prática:**

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
6. dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na
7. plataforma do SQL Server.

### **Todos os códigos solicitados neste roteiro de aula:**

```
CREATE TABLE Pessoa (  
    idPessoa INT IDENTITY(1,1) PRIMARY KEY,  
    nome NVARCHAR(255),  
    logradouro NVARCHAR(255),  
    cidade NVARCHAR(255),  
    estado CHAR(2),  
    telefone NVARCHAR(255),  
    email NVARCHAR(255)  
);
```

```
INSERT INTO Pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES ('nome a', 'logradouro a', 'cidade a','es', 'telefone a', 'email a');
```

```
SELECT * FROM Pessoa;
```

```
CREATE TABLE Produto (  
    idProduto INT IDENTITY(1,1) PRIMARY KEY,  
    nome NVARCHAR(255),  
    quantidade INT,  
    precoVenda NUMERIC  
);
```

```
INSERT INTO Produto (nome, quantidade, precoVenda) VALUES ('nome produto', 50, 5000);
```

```
SELECT * FROM Produto;
```

```
CREATE TABLE Usuario (  
    idUsuario INT IDENTITY(1,1) PRIMARY KEY,  
    login NVARCHAR(255),  
    senha NVARCHAR(255)  
);
```

```
INSERT INTO Usuario (login, senha) VALUES ('login u', 'senhau');
```

```
SELECT * FROM Usuario;
```

```
CREATE TABLE PessoaFisica (  
    cpf NVARCHAR(11) PRIMARY KEY,  
    idPessoa INT FOREIGN KEY REFERENCES Pessoa(idPessoa)  
);
```

```
CREATE TABLE PessoaJuridica (  
    cnpj NVARCHAR(14) PRIMARY KEY,  
    idPessoa INT FOREIGN KEY REFERENCES Pessoa(idPessoa)  
);
```

```
INSERT INTO PessoaFisica (cpf, idPessoa) VALUES ('454516456', 1);
```

```
SELECT * FROM PessoaFisica;
```

```
INSERT INTO PessoaJuridica (cnpj, idPessoa) VALUES ('454516213456', 1);
```

```
SELECT * FROM PessoaJuridica;
```

```

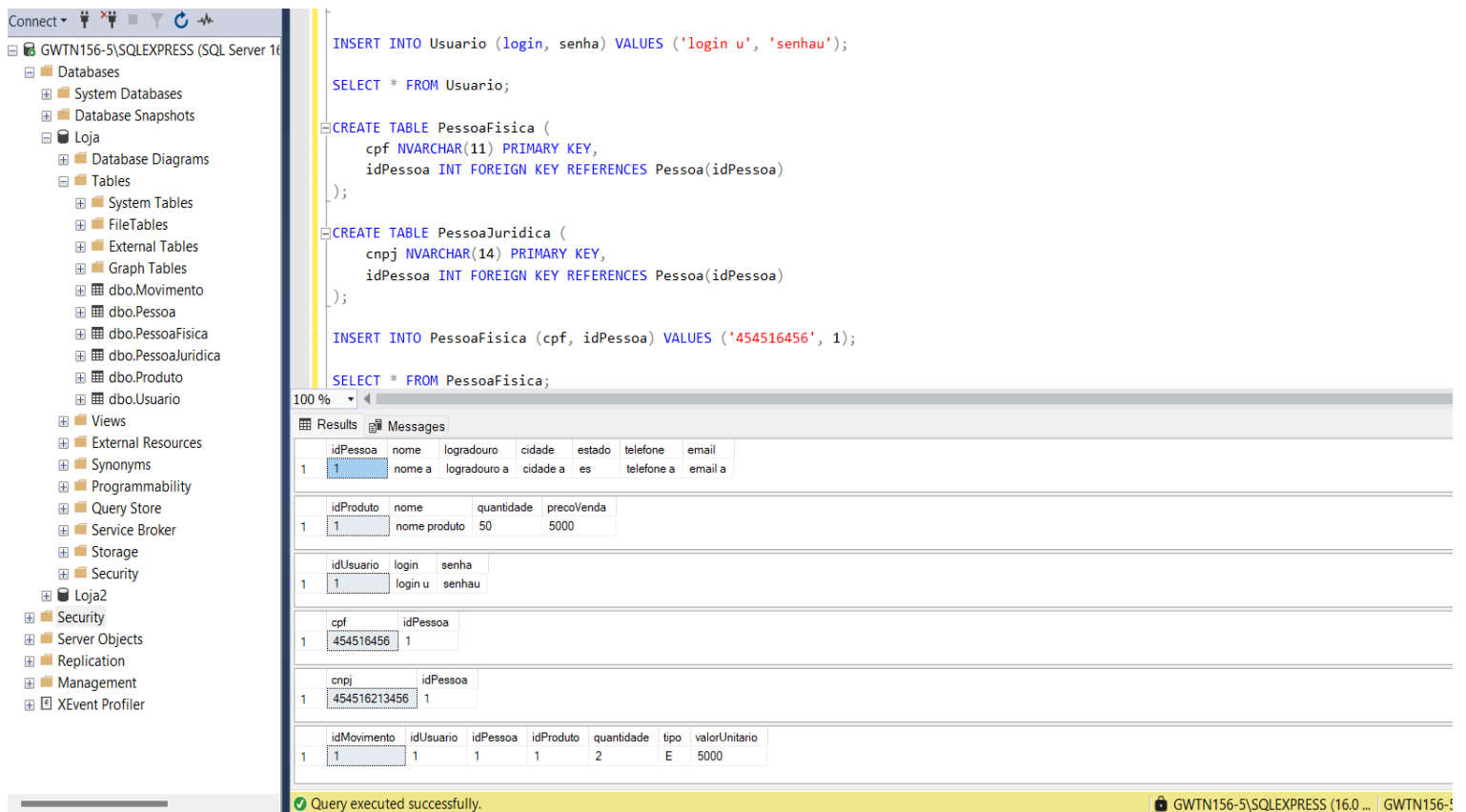
CREATE TABLE Movimento (
    idMovimento INT IDENTITY(1,1) PRIMARY KEY,
    idUsuario INT FOREIGN KEY REFERENCES Usuario(idUsuario),
    idPessoa INT FOREIGN KEY REFERENCES Pessoa(idPessoa),
    idProduto INT FOREIGN KEY REFERENCES Produto(idProduto),
    quantidade INT,
    tipo CHAR(1),
    valorUnitario FLOAT
);

INSERT INTO Movimento (idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario)
VALUES (1, 1, 1, 2, 'E', 5000);

SELECT * FROM Movimento;

```

**Os resultados da execução dos códigos também devem ser apresentados;**



The screenshot displays the SQL Server Enterprise Manager interface. The left-hand pane shows the 'Databases' tree with 'Loja' selected. The right-hand pane shows the SQL script and the results of the queries. The results are displayed in a grid format, showing the data for the tables created in the script.

**Results:**

idPessoa	nome	logradouro	cidade	estado	telefone	email
1	nome a	logradouro a	cidade a	es	telefone a	email a

idProduto	nome	quantidade	precoVenda
1	nome produto	50	5000

idUsuario	login	senha
1	login u	senhau

cpf	idPessoa
454516456	1

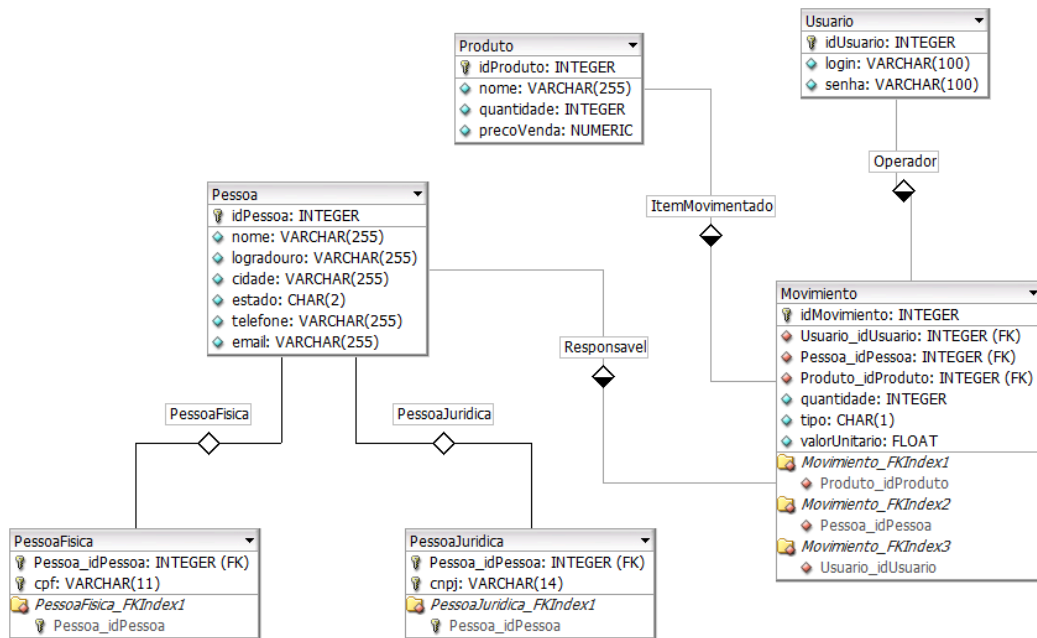
  

cnpj	idPessoa
454516213456	1

idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valorUnitario
1	1	1	1	2	E	5000

Query executed successfully.



## Análise e Conclusão:

**Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?**

As cardinalidades em bancos de dados relacionais descrevem o número de instâncias de uma entidade que podem estar associadas a uma instância de outra entidade. As cardinalidades mais comuns são:

1x1 (Um para Um)

1xN (Um para Muitos)

NxN (Muitos para Muitos)

1x1 (Um para Um)

Em um relacionamento 1x1, cada registro em uma tabela está associado a exatamente um registro em outra tabela. Isso é comum quando você tem duas tabelas que representam conceitos distintos, mas onde cada instância de um conceito está diretamente associada a apenas uma instância do outro.

1xN (Um para Muitos)

No relacionamento 1xN, uma instância de uma entidade pode estar associada a várias instâncias de outra entidade. Por exemplo, um autor pode escrever vários livros, mas cada livro é escrito por apenas um autor.

NxN (Muitos para Muitos)

O relacionamento NxN ocorre quando várias instâncias de uma entidade podem estar associadas a várias instâncias de outra entidade. Por exemplo, os alunos podem se inscrever em vários cursos, e cada curso pode ter vários alunos.

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

Para representar o uso de herança em bancos de dados relacionais, você deve utilizar um tipo especial de relacionamento conhecido como "herança" ou "substituição". A herança é uma característica que permite criar relações entre tabelas onde uma tabela pode ter atributos comuns com outra(s) tabela(s), mas também pode ter seus próprios atributos específicos.

#### **- Tipos de Herança**

Existem dois tipos principais de herança em bancos de dados relacionais:

Herança Completa (Single Table Inheritance): Neste modelo, cada registro em uma tabela filha pertence a apenas uma tabela pai. É semelhante à herança em linguagens de programação orientadas a objetos, onde cada objeto pode herdar de apenas uma classe.

Herança Parcial (Table per Type Inheritance): Neste modelo, registros podem pertencer a mais de uma tabela pai. Isso é útil quando os registros têm diferentes conjuntos de atributos dependendo do contexto.

### **Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?**

O SQL Server Management Studio (SSMS) é uma ferramenta poderosa que oferece diversas funcionalidades para melhorar a produtividade em tarefas relacionadas ao gerenciamento de bancos de dados no Microsoft SQL Server. Aqui estão algumas maneiras pelas quais o SSMS contribui para essa melhoria:

#### **1. Interface Gráfica Usável**

Explorador de Objetos: Permite navegar facilmente pelos objetos do banco de dados, como tabelas, procedimentos armazenados, funções e mais, através de uma interface gráfica intuitiva.

## 2. Automação de Tarefas Comuns

Scripts Gerados: O SSMS pode gerar scripts para criar ou modificar objetos do banco de dados, economizando tempo na escrita manual desses scripts.

## 3. Monitoramento e Desempenho

Tela de Monitoramento: Oferece uma visão geral do desempenho do servidor, incluindo CPU, memória, espaço em disco e atividades de I/O, ajudando a identificar gargalos de desempenho rapidamente.

Rastreamento de Consultas: Permite rastrear consultas em tempo real para entender melhor como elas estão sendo executadas e otimizá-las.

## 4. Segurança e Controle de Acesso

Gerenciamento de Usuários e Funções: Facilita a criação e gestão de usuários e funções de segurança, garantindo que apenas os usuários autorizados tenham acesso aos recursos do banco de dados.

## 5. Integração com Outras Ferramentas

Azure Integration: O SSMS suporta a integração com serviços Azure, permitindo gerenciar facilmente bancos de dados hospedados na plataforma Azure.

## 6. Personalização e Extensibilidade

Plugins e Extensões: O SSMS suporta plugins e extensões desenvolvidas pela comunidade, expandindo suas capacidades e adaptando-se às necessidades específicas dos usuários.

## 7. Depuração e Teste

Ambiente de Depuração Integrado: Permite depurar código SQL diretamente dentro do SSMS, facilitando a identificação e correção de erros.