



# Estácio

**Título:** Missão Prática | Nível 2 | Mundo 3

- **Nome do Campus:** Nova América
- **Nome do Curso:** Desenvolvimento Full Stack
- **Disciplina:** RPG0015 - Vamos manter as informações!
- **Número da Turma:** 9001
- **Semestre:** 2024.1
- **Aluna:** Camilla Rodrigues Alves Gomes

## **Objetivos da Prática:**

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

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

## ***1º Procedimento / Criação das Entidades e Sistema de Persistência***

### **Códigos Solicitados:**

```
CREATE DATABASE Loja;
```

```
USE Loja;
```

```
CREATE SEQUENCE ordemPessoaId  
START WITH 1  
INCREMENT BY 1;
```

```
CREATE TABLE Pessoa (  
    idPessoa INTEGER NOT NULL CONSTRAINT PK_Pessoa PRIMARY KEY,  
    nome VARCHAR(255) NOT NULL,  
    logradouro VARCHAR(255),  
    cidade VARCHAR(255),  
    estado CHAR(2) NOT NULL,  
    telefone VARCHAR(11),  
    email VARCHAR(255)  
);  
GO
```

```
CREATE TABLE PessoaFisica (  
    idPessoa INTEGER NOT NULL CONSTRAINT PK_PessoaFisica PRIMARY KEY,  
    cpf VARCHAR(11) NOT NULL,  
    CONSTRAINT FK_PessoaFisica_Pessoa FOREIGN KEY (idPessoa) REFERENCES Pessoa (idPessoa)  
);  
GO
```

```
CREATE TABLE PessoaJuridica (  
    idPessoa INTEGER NOT NULL CONSTRAINT PK_PessoaJuridica PRIMARY KEY,  
    cnpj VARCHAR(14) NOT NULL,  
    CONSTRAINT FK_PessoaJuridica_Pessoa FOREIGN KEY (idPessoa) REFERENCES Pessoa  
(idPessoa)  
);  
GO
```

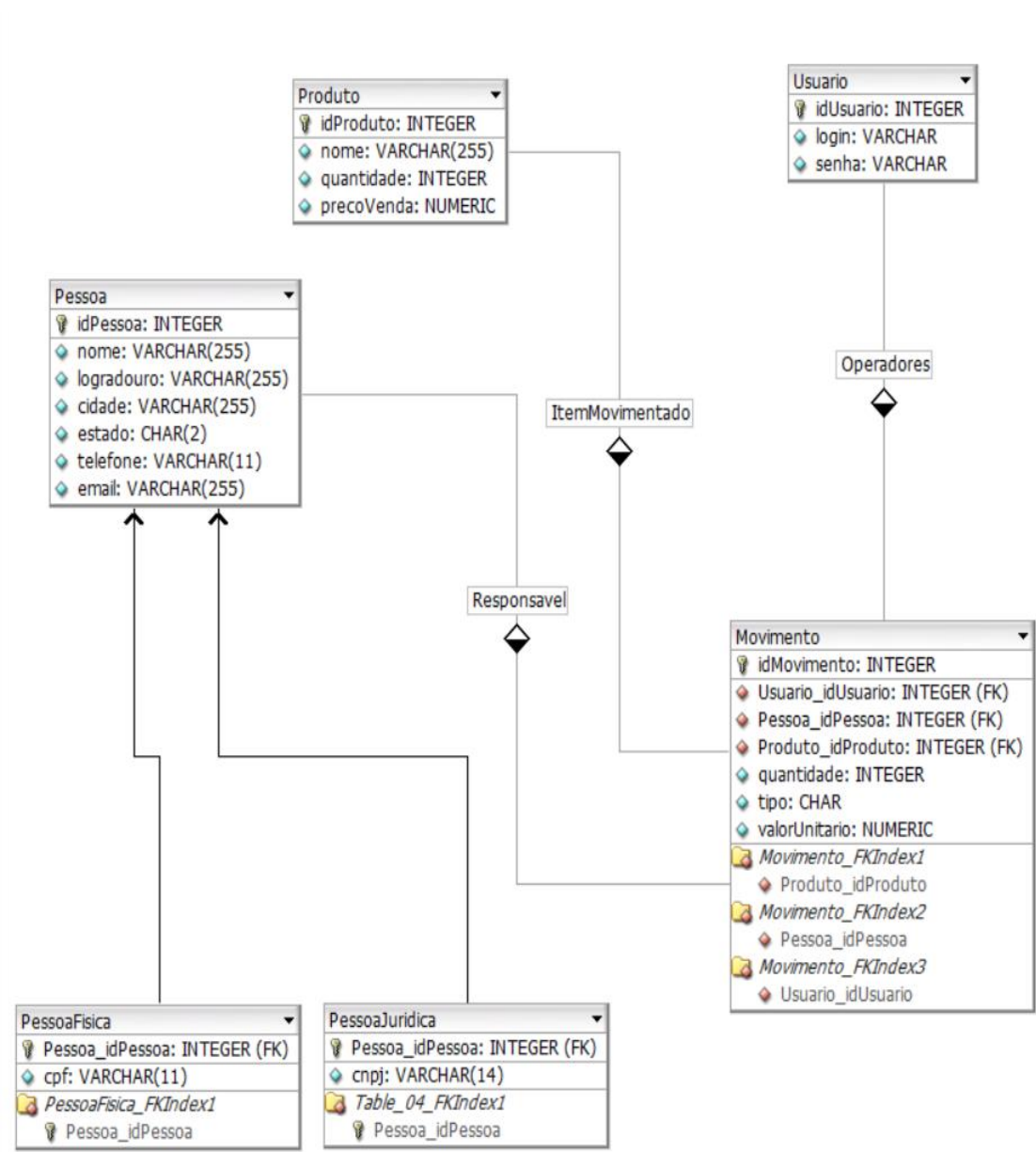
```
CREATE TABLE Usuario (  
    idUsuario INTEGER NOT NULL CONSTRAINT PK_Usuario PRIMARY KEY IDENTITY,  
    login VARCHAR(20) NOT NULL,  
    senha VARCHAR(20) NOT NULL  
);  
GO
```

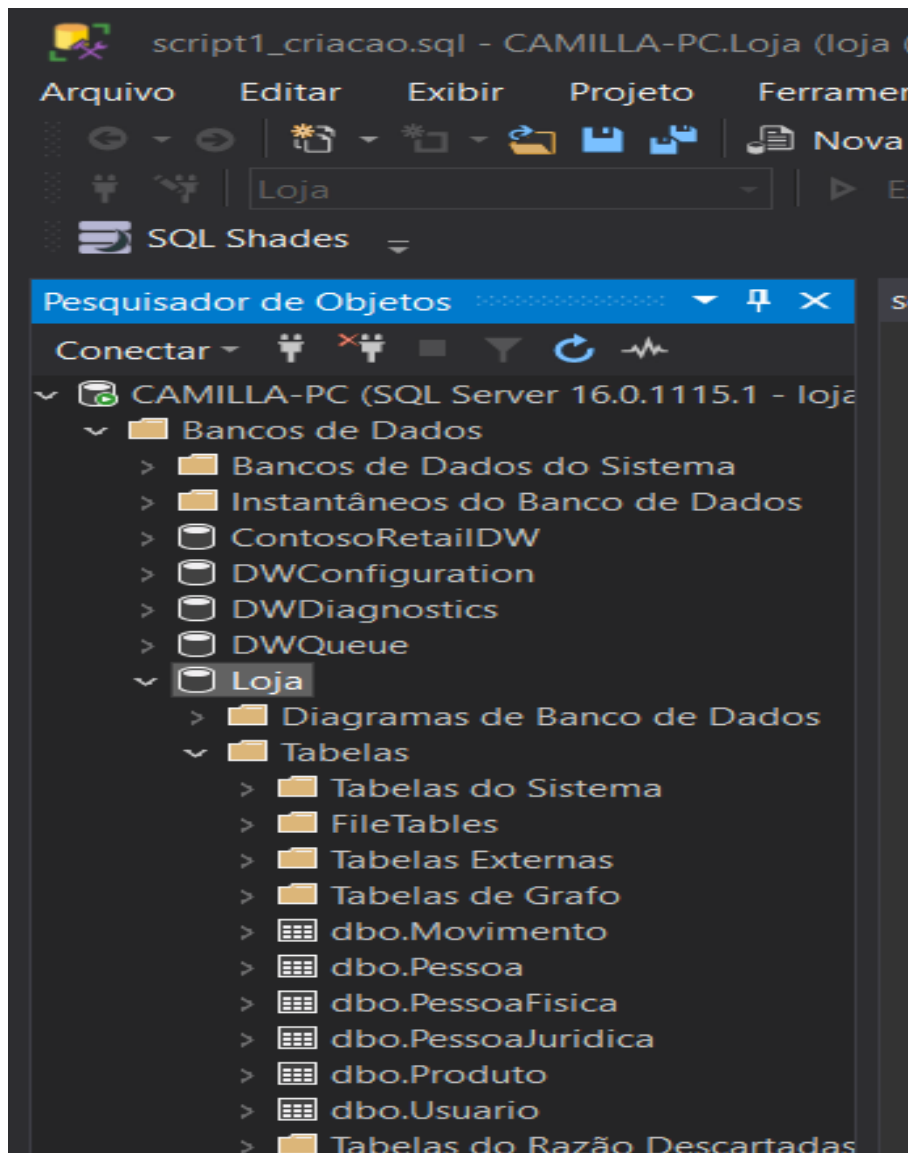
```
CREATE TABLE Produto (  
    idProduto INTEGER NOT NULL CONSTRAINT PK_Produto PRIMARY KEY,  
    nome VARCHAR(255) NOT NULL,  
    quantidade INTEGER,  
    precoVenda NUMERIC(5, 2)  
);  
GO
```

```
CREATE TABLE Movimento (  
    idMovimento INTEGER NOT NULL CONSTRAINT PK_Movimento PRIMARY KEY,  
    idUsuario INTEGER NOT NULL,  
    idPessoa INTEGER NOT NULL,  
    idProduto INTEGER,  
    quantidade INTEGER,  
    tipo CHAR(1),  
    valorUnitario NUMERIC (5, 2),  
    CONSTRAINT FK_Movimento_Usuario FOREIGN KEY (idUsuario) REFERENCES Usuario  
(idUsuario),  
    CONSTRAINT FK_Movimento_Pessoa FOREIGN KEY (idPessoa) REFERENCES Pessoa (idPessoa),  
    CONSTRAINT FK_Movimento_Produto FOREIGN KEY (idProduto) REFERENCES Produto (idProduto)  
);  
GO
```

Resultados:

Modelagem do banco de dados:





*Criação do banco no SSMS.*

## **Análise e Conclusão:**

### **I. Implementação das diferentes cardinalidades em um banco de dados relacional:**

1 para 1: Cada registro em uma tabela está associado a um único registro em outra tabela, e vice-versa, usando chaves primárias e estrangeiras.

1 para N: Um registro em uma tabela pode estar associado a vários registros em outra tabela, mas cada registro nesta segunda tabela está associado a apenas um registro na primeira tabela, usando uma chave estrangeira na tabela secundária.

N para N: Vários registros em uma tabela podem estar associados a vários registros em outra tabela, geralmente implementado por meio de uma tabela de junção que mapeia as associações entre as duas tabelas.

## II. Herança em bancos de dados relacionais:

Melhor tipo de relacionamento:  
Generalização/Especialização (IS-A)

Motivo: A tabela genérica (superclasse) é herdada pela tabela específica (subclasse).

## III. Melhoria da produtividade com o SQL Server Management Studio (SSMS):

Interface Gráfica Amigável: O SSMS fornece uma interface intuitiva e fácil de usar para interagir com o banco de dados, permitindo a execução de consultas, criação de tabelas e gerenciamento de permissões de forma visual.

## 2º Procedimento / Alimentando a Base

### Códigos Solicitados:

#### Inserindo os dados:

```
USE Loja;
```

```
INSERT INTO Usuario (login, senha)
VALUES ('op1', 'op1'),
('op2', 'op2'),
('op3', 'op3'),
('op4', 'op4');
```

```
INSERT INTO Produto (idProduto, nome, quantidade, precoVenda)
VALUES ('1', 'Banana', '100', '5.00'),
('3', 'Laranja', '500', '2.00'),
('4', 'Manga', '800', '4.00');
```

```

INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email)
VALUES
(NEXT VALUE FOR ordemPessoaId, 'Alana', 'Rua X, 10', 'Manaus', 'AM', '1111-1111',
'alana@gmail.com'),
(NEXT VALUE FOR ordemPessoaId, 'Breno', 'Rua Y, 20', 'Rio de Janeiro', 'RJ', '2222-
2222', 'breno@gmail.com'),
(NEXT VALUE FOR ordemPessoaId, 'Caio', 'Rua Z, 30', 'Porto Alegre', 'RS', '3333-3333',
'caio@gmail.com'),
(NEXT VALUE FOR ordemPessoaId, 'Distribuidora Diamante', 'Avenida A, 40', 'Curitiba',
'PR', '4444-4444', 'diamante@gmail.com'),
(NEXT VALUE FOR ordemPessoaId, 'Empresa Estrela', 'Avenida B, 50', 'Recife', 'PE',
'5555-5555', 'estrela@gmail.com');

INSERT INTO PessoaFisica (idPessoa, cpf)
VALUES (1, '11111111111'),
(2, '22222222222'),
(3, '33333333333');

INSERT INTO PessoaJuridica (idPessoa, cnpj)
VALUES (4, '44444444444444'),
(5, '55555555555555');

INSERT INTO Movimento (idMovimento, idUsuario, idPessoa, idProduto, quantidade, tipo,
valorUnitario)
VALUES (1, 1, 5, 1, 40, 'E', 5.00),
(3, 2, 3, 3, 20, 'S', 2.00),
(5, 1, 4, 4, 60, 'E', 4.00),
(6, 2, 1, 1, 15, 'S', 5.00),
(7, 4, 2, 4, 25, 'S', 4.00),
(9, 3, 5, 3, 50, 'E', 2.00);

```

## **Consultas solicitadas:**

```

/* Dados completos de pessoas físicas. */
SELECT *
FROM PessoaFisica
INNER JOIN Pessoa ON PessoaFisica.idPessoa = Pessoa.idPessoa

/* Dados completos de pessoas jurídicas. */
SELECT *
FROM PessoaJuridica
INNER JOIN Pessoa ON PessoaJuridica.idPessoa = Pessoa.idPessoa

/* Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e
valor total. */
SELECT
Produto.nome AS 'produto', Pessoa.nome AS 'fornecedor',
Movimento.quantidade, Movimento.valorUnitario,
Movimento.quantidade * Movimento.valorUnitario AS 'valorTotal'
FROM Movimento
INNER JOIN Produto ON Movimento.idProduto = Produto.idProduto
INNER JOIN Pessoa ON Movimento.idPessoa = Pessoa.idPessoa
WHERE Movimento.tipo = 'E';

```

```

/* Movimentações de saída, com produto, comprador, quantidade,
preço unitário e valor total. */
SELECT
Produto.nome AS 'produto', Pessoa.nome AS 'comprador',
Movimento.quantidade, Movimento.valorUnitario,
Movimento.quantidade * Movimento.valorUnitario AS 'valorTotal'
FROM Movimento
INNER JOIN Produto ON Movimento.idProduto = Produto.idProduto
INNER JOIN Pessoa ON Movimento.idPessoa = Pessoa.idPessoa
WHERE Movimento.tipo = 'S';

/* Valor total das entradas agrupadas por produto. */
SELECT
Produto.nome AS 'produto',
SUM(Movimento.quantidade * Movimento.valorUnitario) AS 'valorTotalEntrada'
FROM Movimento
JOIN Produto ON Movimento.idProduto = Produto.idProduto
WHERE Movimento.tipo = 'E'
GROUP BY Produto.nome;

/* Valor total das saídas agrupadas por produto. */
SELECT
Produto.nome AS 'produto',
SUM(Movimento.quantidade * Movimento.valorUnitario) AS 'valorTotalSaida'
FROM Movimento
JOIN Produto ON Movimento.idProduto = Produto.idProduto
WHERE Movimento.tipo = 'S'
GROUP BY Produto.nome;

/* Operadores que não efetuaram movimentações de entrada
(compra). */
SELECT DISTINCT
Usuario.idUsuario, Usuario.login, Movimento.idMovimento
FROM Usuario
LEFT JOIN Movimento ON Usuario.idUsuario = Movimento.idUsuario AND Movimento.tipo = 'E'
WHERE idMovimento IS NULL;

/* Valor total de entrada, agrupado por operador. */
SELECT
Usuario.login AS 'operador',
SUM(Movimento.quantidade * Movimento.valorUnitario) AS 'valorTotalEntrada'
FROM Movimento
JOIN Usuario ON Movimento.idUsuario = Usuario.idUsuario
WHERE Movimento.tipo = 'E'
GROUP BY Usuario.login;

/* Valor total de saída, agrupado por operador. */
SELECT
Usuario.login AS 'operador',
SUM(Movimento.quantidade * Movimento.valorUnitario) AS 'valorTotalSaida'
FROM Movimento
JOIN Usuario ON Movimento.idUsuario = Usuario.idUsuario
WHERE Movimento.tipo = 'S'
GROUP BY Usuario.login;

/* Valor médio de venda por produto, utilizando média ponderada. */
SELECT
SUM(Movimento.valorUnitario * Movimento.quantidade) / SUM(Movimento.quantidade) AS
'médiaPonderada'
FROM Movimento
WHERE Movimento.tipo = 'S';

```

Resultados:

Tabelas após inserir os dados:

100 %

ResultadosMensagens

	idUsuario	login	senha
1	1	op1	op1
2	2	op2	op2
3	3	op3	op3
4	4	op4	op4

	idProduto	nome	quantidade	precoVenda
1	1	Banana	100	5.00
2	3	Laranja	500	2.00
3	4	Manga	800	4.00

	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	1	Alana	Rua X, 10	Manaus	AM	1111-1111	alana@gmail.com
2	2	Breno	Rua Y, 20	Rio de Janeiro	RJ	2222-2222	breno@gmail.com
3	3	Caio	Rua Z, 30	Porto Alegre	RS	3333-3333	caio@gmail.com
4	4	Distribuidora Diamante	Avenida A, 40	Curitiba	PR	4444-4444	diamante@gmail.com
5	5	Empresa Estrela	Avenida B, 50	Recife	PE	5555-5555	estrela@gmail.com

	idPessoa	cpf
1	1	11111111111
2	2	22222222222
3	3	33333333333

	idPessoa	cnpj
1	4	4444444444444
2	5	5555555555555

	idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valorUnitario
1	1	1	5	1	40	E	5.00
2	3	2	3	3	20	S	2.00
3	5	1	4	4	60	E	4.00
4	6	2	1	1	15	S	5.00
5	7	4	2	4	25	S	4.00
6	9	3	5	3	50	E	2.00



## Consultas:

script3\_consultas.sq...-PC.Loja (loja (51))

100 %

Resultados

Mensagens

	idPessoa	cpf	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	1	11111111111	1	Alana	Rua X, 10	Manaus	AM	1111-1111	alana@gmail.com
2	2	22222222222	2	Breno	Rua Y, 20	Rio de Janeiro	RJ	2222-2222	breno@gmail.com
3	3	33333333333	3	Caio	Rua Z, 30	Porto Alegre	RS	3333-3333	caio@gmail.com

	idPessoa	cnpj	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	4	4444444444444	4	Distribuidora Diamante	Avenida A, 40	Curitiba	PR	4444-4444	diamante@gmail.com
2	5	5555555555555	5	Empresa Estrela	Avenida B, 50	Recife	PE	5555-5555	estrela@gmail.com

	produto	fornecedor	quantidade	valorUnitario	valorTotal
1	Banana	Empresa Estrela	40	5.00	200.00
2	Manga	Distribuidora Diamante	60	4.00	240.00
3	Laranja	Empresa Estrela	50	2.00	100.00

	produto	comprador	quantidade	valorUnitario	valorTotal
1	Laranja	Caio	20	2.00	40.00
2	Banana	Alana	15	5.00	75.00
3	Manga	Breno	25	4.00	100.00

	produto	valorTotalEntrada
1	Banana	200.00
2	Laranja	100.00
3	Manga	240.00

	produto	valorTotalSaida
1	Banana	75.00
2	Laranja	40.00
3	Manga	100.00

	idUserario	login	idMovimento
1	2	op2	NULL
2	4	op4	NULL

	operador	valorTotalEntrada
1	op1	440.00
2	op3	100.00

	operador	valorTotalSaida
1	op2	115.00
2	op4	100.00

	médiaPonderada
1	3.583333

## **Análise e Conclusão:**

### **I. Quais as diferenças no uso de sequence e identity?**

SEQUENCE: Gera valores sequenciais únicos, mesmo que a inserção seja feita por várias conexões concorrentemente. Os valores são armazenados em um objeto separado.

IDENTITY: Gera valores sequenciais únicos dentro de uma única conexão. Os valores são armazenados na própria tabela.

### **II. Qual a importância das chaves estrangeiras para a consistência do banco?**

Inserções inválidas: Impedem a inserção de dados em uma tabela filha que não existam na tabela pai.

Atualizações inválidas: Impedem a atualização ou exclusão de dados em uma tabela pai que tenham referências em tabelas filhas.

Exclusões em cascata: Permitem que dados em tabelas filhas sejam excluídos automaticamente quando os dados correspondentes na tabela pai forem excluídos.

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

Álgebra Relacional:

- União (UNION)
- Interseção (INTERSECT)
- Diferença (EXCEPT)
- Produto Cartesiano (CROSS JOIN)
- Junção (JOIN)
- Projeção (SELECT)

Cálculo Relacional:

- Restrição (WHERE)
- Agrupamento (GROUP BY)
- Ordenação (ORDER BY)

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

O agrupamento é feito usando o GROUP BY. Ele é usado para combinar linhas de dados em grupos com base em valores semelhantes em uma ou mais colunas.

Requisito obrigatório:

- A cláusula GROUP BY deve especificar todas as colunas não agregadas na cláusula SELECT.

**Repositório GIT:**

- O projeto foi armazenado em um repositório no GIT. O endereço do repositório é: <https://github.com/Camilla-Alves/mundo3-nivel2>