



- **Nome do Campus:** Polo Mondubim
- **Nome do Curso:** Desenvolvimento Full Stack
- **Nome da Disciplina:** Vamos manter as informações?
- **Número da Turma:** 9001
- **Semestre Letivo:** 2023.3
- **Integrante da Prática:** Samuel Mota Araujo
- **Repositório GIT:** <https://github.com/samuelmotapf>

## **# Relatório de Desenvolvimento - Missão Prática RPG0015 – Exercício2**

### **1. Título da Prática:**

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

### **## Introdução**

Este relatório descreve o desenvolvimento da missão prática intitulada "Vamos manter as informações!", cujo objetivo é a modelagem e implementação de um banco de dados simples, utilizando o SQL Server como plataforma. A prática envolve a identificação de requisitos de um sistema, a transformação desses requisitos em um modelo adequado e a utilização de ferramentas de modelagem para bases de dados relacionais, bem como a exploração da sintaxe SQL para criação de estruturas (DDL) e manipulação de dados (DML).

### **## Objetivos da Prática**

Os objetivos desta missão prática são:

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).

Ao final do exercício, espera-se que o aluno tenha vivenciado a experiência de modelar a base de dados para um sistema simples e implementá-la através da sintaxe SQL na plataforma do SQL Server.

## **## Materiais Necessários**

**Para a realização da prática, são necessários os seguintes materiais:**

1. Ferramenta de modelagem DBDesigner.
2. Banco de dados SQL Server com o gerenciador SQL Server Management Studio.
3. Computador com acesso à Internet.

## **## Desenvolvimento da Prática**

### **### Procedimento 1: Criando o Banco de Dados**

1. **\*\*Download e Execução da Ferramenta de Modelagem:\*\***
  - Acessar o endereço [DBDesigner Fork](<https://sourceforge.net/projects/dbdesigner-fork/>).

- Efetuar o download do DBDesigner Fork no formato Zip.

## 2. **\*\*Modelagem de Dados:\*\***

- Definir o modelo de dados para um sistema com as seguintes características:
  - Cadastro de usuários para acesso ao sistema.
  - Cadastro de pessoas físicas e jurídicas com dados de identificação, localização e contato.
  - Cadastro de produtos com identificador, nome, quantidade e preço de venda.
  - Movimentos de compra e venda efetuados por operadores (usuários) para produtos específicos e entidades específicas.

## 3. **\*\*Utilização do SQL Server Management Studio:\*\***

- Utilizar o SQL Server Management Studio para criar a base de dados modelada no passo anterior.
  - Definir uma sequence para a geração dos identificadores de pessoa, considerando o relacionamento 1x1 com pessoa física ou jurídica.
  - Salvar o script completo para a criação do banco de dados em um arquivo com extensão .sql.

## **## Resultados Esperados**

Os resultados esperados são:

1. Código organizado na modelagem e implementação do banco de dados.
2. Exploração das funcionalidades do DB Designer Fork e do SQL Server Management Studio para modelagem e criação do banco de dados.

3. Demonstração das habilidades básicas para a modelagem da base de dados em um sistema, além do uso da sintaxe SQL para criação das estruturas necessárias.

---

Este relatório documenta o desenvolvimento da missão prática, fornecendo uma visão geral das etapas realizadas e destacando os resultados alcançados. O código desenvolvido atende aos requisitos estabelecidos, demonstrando o entendimento e aplicação dos conceitos abordados na prática.

### 3. Códigos Solicitados:

-- Tabela Usuário

```
CREATE TABLE Usuario (  
  UserID INT PRIMARY KEY,  
  Username VARCHAR(50) UNIQUE,  
  PasswordHash VARCHAR(100),  
  TipoOperador VARCHAR(20),  
  PessoaFisicalD INT,  
  PessoaJuridicalD INT,  
  FOREIGN KEY (PessoaFisicalD) REFERENCES  
  PessoaFisica(PessoaFisicalD),  
  FOREIGN KEY (PessoaJuridicalD) REFERENCES  
  PessoaJuridica(PessoaJuridicalD)  
);
```

-- Tabela PessoaFisica

```
CREATE TABLE PessoaFisica (  
  PessoaFisicalD INT PRIMARY KEY,  
  Nome VARCHAR(100),  
  CPF VARCHAR(14) UNIQUE,
```

```
EnderecoVARCHAR(200),  
    Contato VARCHAR(100)  
);
```

-- Tabela PessoaJuridica

```
CREATE TABLE PessoaJuridica (  
    PessoaJuridicaID INT PRIMARY KEY,  
    NomeEmpresaVARCHAR(100),  
    CNPJ VARCHAR(18) UNIQUE,  
    EnderecoVARCHAR(200),  
    Contato VARCHAR(100)  
);
```

-- Tabela Produto

```
CREATE TABLE Produto (  
    ProdutoID INT PRIMARY KEY,  
    Nome VARCHAR(100),  
    QuantidadeEmEstoque INT,  
    PrecoVendaAtualDECIMAL(10, 2)  
);
```

-- Tabela MovimentoCompra

```
CREATE TABLE MovimentoCompra (  
    MovimentoCompraID INT PRIMARY KEY,  
    ProdutoID INT,  
    PessoaJuridicaID INT,  
    Quantidade INT,  
    PrecoUnitarioDECIMAL(10, 2),
```

```
Data DATE,  
    FOREIGN KEY (ProdutoID) REFERENCES Produto(ProdutoID),  
    FOREIGN KEY (PessoaJuridicaID) REFERENCES  
    PessoaJuridica(PessoaJuridicaID)  
);
```

-- Tabela MovimentoVenda

```
CREATE TABLE MovimentoVenda (  
    MovimentoVendaID INT PRIMARY KEY,  
    ProdutoID INT,  
    PessoaFisicaID INT,  
    Quantidade INT,  
    Data DATE,  
    FOREIGN KEY (ProdutoID) REFERENCES Produto(ProdutoID),  
    FOREIGN KEY (PessoaFisicaID) REFERENCES  
    PessoaFisica(PessoaFisicaID)  
);
```

4. Resultados da Execução dos Códigos:

SQLQuery6.sql - (lo...Jr\muangala Jr (56))\* SQLQuery5.sql - (lo...Jr\muangala Jr (57))\*

```
SELECT *  
FROM PessoaFisica;
```

100 %

Results Messages

	PessoaFisicaID	Nome	CPF	Endereco	Contato
1	1	João da Silva	123.456.789-00	Rua A, 123	joao@email.com
2	2	Maria Oliveira	987.654.321-00	Avenida B, 456	maria@email.com

Imagem de Consulta de pessoa física

SQLQuery6.sql - (lo...Jr\muangala Jr (56))*						
SQLQuery5.sql - (lo...Jr\muangala Jr (57))*						
SQLQuery4.s						
<pre> SELECT MC.ProdutoID, P.Nome AS Produto, PJ.NomeEmpresa AS Fornecedor, MC.Quantida FROM MovimentoCompra MC INNER JOIN Produto P ON MC.ProdutoID = P.ProdutoID INNER JOIN PessoaJuridica PJ ON MC.PessoaJuridicaID = PJ.PessoaJuridicaID; </pre>						
100 %						
Results Messages						
	ProdutoID	Produto	Fornecedor	Quantidade	PrecoUnitario	ValorTotal
1	1	Produto A	Empresa ABC LTDA	20	40.00	800.00

Consulta da tabela produtos

## Importância das Chaves Estrangeiras para a Consistência do Banco:

As chaves estrangeiras desempenham um papel fundamental na garantia da consistência e integridade dos dados em um banco de dados. Elas estabelecem relações entre as tabelas, permitindo a manutenção de integridade referencial, o que é essencial para garantir a consistência dos dados.

## Operadores do SQL e sua relação com a Álgebra e Cálculo Relacional:

Os operadores do SQL têm suas bases conceituais na teoria matemática, especificamente na álgebra relacional e no cálculo relacional. Ambos fornecem fundamentos teóricos para as operações realizadas em bancos de dados relacionais.

## Agrupamento em Consultas e Requisitos Necessários:

A álgebra relacional foi proposta por Edgar F. Codd, o pai do modelo relacional de dados. Ela fornece um conjunto de operadores para



manipular relações (tabelas) em bancos de dados. Alguns dos operadores da álgebra relacional e suas correspondências no SQL são:

1. **Projeção ( $\pi$ ):** Seleciona colunas específicas de uma tabela.
  - **SQL equivalente:** **SELECT**.
2. **Seleção ( $\sigma$ ):** Filtra linhas com base em uma condição específica.
  - **SQL equivalente:** **WHERE**.
3. **União ( $\cup$ ):** Combina duas tabelas, retornando as linhas presentes em ambas.
  - **SQL equivalente:** **UNION**.
4. **Interseção ( $\cap$ ):** Retorna apenas as linhas que aparecem em ambas as tabelas.
  - **SQL equivalente:** Não possui um operador específico, mas pode ser simulado usando **INNER JOIN** ou subconsultas.
5. **Diferença ( $-$ ):** Retorna as linhas presentes em uma tabela, mas não na outra.
  - **SQL equivalente:** **EXCEPT**.