



## Missão Prática | Nível 2 | Mundo 3

### Vamos manter as informações!

Rian Joseph Ramos Felizardo - 202202923931

POLO BARREIRO - Belo Horizonte, MG

Nível 2 - Vamos Manter as Informações? – 2023.1 – 3º Semestre Letivo

Repositório - [desenvolvimento-sistemas-mundo-3/nivel-02/source\\_at\\_main](https://github.com/rianjrp/desenvolvimento-sistemas-mundo-3/nivel-02/source_at_main) · [rianjrp/desenvolvimento-sistemas-mundo-3 \(github.com\)](https://github.com/rianjrp/desenvolvimento-sistemas-mundo-3)

### Objetivo da Prática

Descreva nessa seção qual o objetivo da sua prática. Todos os Relatórios de Práticas deverão ser confeccionados em arquivo no formato PDF, com a **Logo da Universidade, nome do Campus, nome do Curso, nome da Disciplina, número da Turma, semestre letivo**. Além disso, o projeto deve ser armazenado em um repositório no GIT e o respectivo endereço deve constar na documentação e essa documentação deve estar no GIT. O código deve estar versionado no GIT de forma organizada.

**Lembre-se que a organização contará pontos.**

### OBJETIVOS

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)

#### 1º Procedimento | Criando o Banco de Dados

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 1º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

## DBDESIGNER FORK

<b>Usuario</b> idUsuario: INTEGER nome: VARCHAR(100) senha: VARCHAR(50)	
<b>Pessoa</b> idPessoa : INTEGER nome: VARCHAR(100) endereco: VARCHAR(255) telefone: VARCHAR(15)	<b>Movimentos</b> idMovimentos: INTEGER tipo: VARCHAR(10) quantidade: INTEGER precoUnitario: DECIMAL dataMovimento: DATE
<b>PessoaFisica</b> idPessoaFisica: INTEGER cpf: VARCHAR(11) dataNascimento: DATE localizacao: VARCHAR	<b>Produto</b> idProduto: INTEGER nome: VARCHAR(100) quantidade: INTEGER precoVenda: DECIMAL
<b>PessoaJuridica</b> idPessoaJuridica : INTEGER cnpj: VARCHAR(14) razaoSocial: VARCHAR(40) localizacao: VARCHAR	

A Base completamente modelada dentro do DBFORK, utilizando os parametros corretos para serem implementados dentro do SSMS.

# SSMS

```
scriptSQL.sql - MAR...01.Loja (loja (167))
USE Loja;
GO

CREATE TABLE Pessoas (
    Id INT PRIMARY KEY IDENTITY(1,1),
    Nome VARCHAR(100) NOT NULL,
    Endereco VARCHAR(255),
    Telefone VARCHAR(15)
);
GO

CREATE TABLE PessoasFisicas (
    Id INT PRIMARY KEY FOREIGN KEY REFERENCES Pessoas(Id),
    CPF VARCHAR(11) NOT NULL
);
GO

CREATE TABLE PessoasJuridicas (
    Id INT PRIMARY KEY FOREIGN KEY REFERENCES Pessoas(Id),
    CNPJ VARCHAR(14) NOT NULL
);
GO

CREATE TABLE Usuarios (
    Id INT PRIMARY KEY IDENTITY(1,1),
    Nome VARCHAR(100) NOT NULL,
    Senha VARCHAR(50) NOT NULL
);
GO

CREATE TABLE Produtos (
    Id INT PRIMARY KEY IDENTITY(1,1),
    Nome VARCHAR(100) NOT NULL,
    Quantidade INT NOT NULL,
    Preco DECIMAL(10, 2) NOT NULL
);
GO

CREATE TABLE Movimentos (
    Id INT PRIMARY KEY IDENTITY(1,1),
    Tipo VARCHAR(10) CHECK (Tipo IN ('Compra', 'Venda')),
    IdProduto INT FOREIGN KEY REFERENCES Produtos(Id),
    IdPessoa INT FOREIGN KEY REFERENCES Pessoas(Id),
    Quantidade INT NOT NULL,
    PrecoUnitario DECIMAL(10, 2) NOT NULL,
    DataMovimento DATETIME DEFAULT GETDATE()
);
GO

CREATE SEQUENCE SeqPessoa
    START WITH 1
    INCREMENT BY 1
    MINVALUE 1
    NO MAXVALUE
    CACHE 10;
GO
```

Após a modelagem no DBFORK, está a criação (DDL) do código de construção completo da Base de Dados, seguindo todos os procedimentos como descrito no enunciado do trabalho.

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

### **Cardinalidade 1x1**

- **Implementação:** Tabela A e Tabela B têm um registro associado a um único registro da outra.
- **Exemplo:** Usuarios e Perfis, onde Perfis tem uma chave estrangeira apontada UsuarioId.

### **2. Cardinalidade 1XN**

- **Implementação:** Tabela A pode ter vários registros em Tabela B, mas B tem apenas um registro em A.
- **Exemplo:** Clientes e Pedidos, onde Pedidos tem uma coluna ClienteId.

### **3. Cardinalidade NxN**

- **Implementação:** Registros em A podem se associar a vários registros em B e vice-versa.
- **Exemplo:** Estudantes e Cursos, com uma tabela de junção Inscricoes contendo EstudanteId e CursoId.

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

A escolha depende das necessidades do sistema, onde a forma de tabelas separadas é a mais comum.

- Tabela Única:

Onde é uma única tabela para todas as classes. Com a vantagem de simplicidade.

- Tabelas Separadas:

Onde é usado uma tabela para a classe pai e tabelas separadas para as subclasses. Mantendo a vantagem de uma estrutura mais organizada.

- Tabelas de Junção:

Onde é usado tabelas separadas para cada subclasse, sem tabela PAI. Mantendo a vantagem de uma otimização para subclasses.

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

O SSMS aumenta a produtividade no gerenciamento de BD's por meio de uma interface mais intuitiva facilitadora na navegação e visualização das estruturas, visualização dos dados e o monitoramento do banco de dados.

Contando com o editor de consultas que auxilia / facilita a escrita em SQL, e com suas ferramentas de backup, restauração e gerenciamento de usuários.

Observe que os tópicos acima seguem exatamente o que está na Atividade Prática exigida.

2º Procedimento | Alimentando a Base

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 2º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

MOVIMENTAÇÕES

Resultados

Mensagens

	Id	Nome	Endereco	Telefone
1	1	Algusto Nascimento	Rua Adalberto, 123	123456789
2	2	Sempre Vida	Avenida Olinda, 456	987654321
3	3	Joao	Rua 12, casa 3, Quitanda	1111-1111
4	5	JJC	Rua 11, Centro, Riacho Norte	1212-1212

	Id	Nome	Senha
1	2	op1	op1
2	3	op2	op2

	Id	Nome	Quantidade	Preco
1	2	Banana	100	5.00
2	3	Laranja	500	2.00
3	4	Manga	800	4.00
4	5	Maca	300	7.50
5	6	Couve	240	1.50

	Id	Tipo	IdProduto	IdPessoa	Quantidade	PrecoUnitario	DataMovimento
1	11	Venda	2	2	10	7.50	2024-10-09 19:48:40.500
2	13	Venda	3	2	103	7.50	2024-10-09 19:49:44.277
3	14	Compra	4	1	103	7.50	2024-10-09 19:51:10.930

	Id	CPF
1	1	12345678901
2	2	11111111111

	Id	CNPJ
1	2	12345678000195
2	5	2222222222214

Movimentações em DML abaixo.

```

SQLQuery1.sql - MAR...01.Loja (loja (60))*
-- Consulta para calcular o valor total das entradas agrupadas por produto
SELECT
    p.Nome AS Produto,
    SUM(m.Quantidade * m.PrecoUnitario) AS ValorTotalEntradas
FROM
    Movimentos m
JOIN
    Produtos p ON m.IdProduto = p.Id
WHERE
    m.Tipo = 'Compra' -- Filtra apenas as compras
GROUP BY
    p.Nome; -- Agrupa os resultados pelo nome do produto

-- Consulta para calcular o valor total das saídas agrupadas por produto
SELECT
    p.Nome AS Produto,
    SUM(m.Quantidade * m.PrecoUnitario) AS ValorTotalSaidas
FROM
    Movimentos m
JOIN
    Produtos p ON m.IdProduto = p.Id
WHERE
    m.Tipo = 'Venda' -- Filtra apenas as vendas
GROUP BY
    p.Nome; -- Agrupa os resultados pelo nome do produto

-- Consulta para listar operadores que nao efetuaram movimentacoes de entrada (compra)
SELECT
    u.Nome AS Operador
FROM
    Usuarios u
LEFT JOIN
    Movimentos m ON u.Id = m.IdPessoa AND m.Tipo = 'Compra' -- Faz um join com a tabela de movimentos
WHERE
    m.Id IS NULL; -- Filtra para mostrar apenas operadores sem compras

-- Consulta para calcular o valor total de entradas, agrupado por operador
SELECT
    u.Nome AS Operador,
    SUM(m.Quantidade * m.PrecoUnitario) AS ValorTotalEntradas
FROM
    Movimentos m
JOIN
    Usuarios u ON m.IdPessoa = u.Id
WHERE
    m.Tipo = 'Compra' -- Filtra apenas as compras
GROUP BY
    u.Nome; -- Agrupa os resultados pelo nome do operador

```

```

SQLQuery1.sql - MAR...01.Loja (loja (60))*
-- Consulta para calcular o valor total de entradas, agrupado por operador
SELECT
    u.Nome AS Operador,
    SUM(m.Quantidade * m.PrecoUnitario) AS ValorTotalEntradas
FROM
    Movimentos m
JOIN
    Usuarios u ON m.IdPessoa = u.Id
WHERE
    m.Tipo = 'Compra' -- Filtra apenas as compras
GROUP BY
    u.Nome; -- Agrupa os resultados pelo nome do operador

-- Consulta para calcular o valor total de saídas, agrupado por operador
SELECT
    u.Nome AS Operador,
    SUM(m.Quantidade * m.PrecoUnitario) AS ValorTotalSaidas
FROM
    Movimentos m
JOIN
    Usuarios u ON m.IdPessoa = u.Id
WHERE
    m.Tipo = 'Venda' -- Filtra apenas as vendas
GROUP BY
    u.Nome; -- Agrupa os resultados pelo nome do operador

-- Consulta para calcular o valor medio de venda por produto, utilizando media ponderada
SELECT
    p.Nome AS Produto,
    SUM(m.Quantidade * m.PrecoUnitario) / SUM(m.Quantidade) AS MediaPonderada
FROM
    Movimentos m
JOIN
    Produtos p ON m.IdProduto = p.Id
WHERE
    m.Tipo = 'Venda' -- Filtra apenas as vendas
GROUP BY
    p.Nome; -- Agrupa os resultados pelo nome do produto

```

SQLQuery1.sql - MAR...01.Loja (loja (60))\*

```
-- Consulta para calcular o valor total das entradas agrupadas por produto
SELECT
    p.Nome AS Produto,
    SUM(m.Quantidade * m.PrecoUnitario) AS ValorTotalEntradas
FROM
    Movimentos m
JOIN
    Produtos p ON m.IdProduto = p.Id
WHERE
    m.Tipo = 'Compra' -- Filtra apenas as compras
GROUP BY
    p.Nome; -- Agrupa os resultados pelo nome do produto

-- Consulta para calcular o valor total das saídas agrupadas por produto
SELECT
    p.Nome AS Produto,
    SUM(m.Quantidade * m.PrecoUnitario) AS ValorTotalSaidas
FROM
    Movimentos m
JOIN
    Produtos p ON m.IdProduto = p.Id
WHERE
    m.Tipo = 'Venda' -- Filtra apenas as vendas
GROUP BY
    p.Nome; -- Agrupa os resultados pelo nome do produto

-- Consulta para listar operadores que nao efetuaram movimentacoes de entrada (compra)
```

100 %

Resultados Mensagens

	Produto	ValorTotalEntradas
1	Laranja	100.00
2	Manga	3172.50

  

	Produto	ValorTotalSaidas
1	Banana	75.00
2	Laranja	872.50
3	Manga	800.00

  

	Operador
1	op2

  

	Operador	ValorTotalEntradas
1	op1	2400.00

  

	Operador	ValorTotalSaidas
1	op1	1747.50

  

	Produto	MediaPonderada
1	Banana	7.500000
2	Laranja	7.721238
3	Manga	4.000000

Consulta executada com êxito. | MARGARETH\MSSQLSERVER01 (16... | loja (60) | Loja | 00:00:00 | 11 linhas

Após realização de algumas movimentações e finalizado o enunciado do 2º procedimento, obtive resultado positivo em todas as consultas, sendo elas consulta geral, registro de movimentações, produtos, usuários, tipos de pessoa que herdaram da tabela Pessoa.



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

O *identity* é uma propriedade de coluna que gera automaticamente números únicos em uma tabela.

Já o *sequence* é um objeto separado que gera números únicos e pode ser utilizado em várias tabelas, onde oferece mais flexibilidade.

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

As chaves estrangeiras garantem a integridade referencial, mantendo a segurança de validação dos relacionamentos entre as tabelas, ajudando a manter a consistência do banco de dados.

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

Os operadores da Álgebra relacional são alguns como SELECT, PROJECT, JOIN, UNION e INTERSECT.

Já os de Cálculo Relacional se baseiam em expressões lógicas e incluem os operadores como EXISTS e IN, e os quantificadores como ALL e SOME.

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

O agrupamento é feito com o operador GROUP BY. O requisito obrigatório é que todas as colunas no SELECT que não estão agregadas devem ser incluídas na cláusula GROUP BY.

Observe que os tópicos acima seguem exatamente o que está na Atividade Prática exigida.

## Conclusão

Elabore uma análise crítica da sua Missão Prática.

## **Análise crítica do trabalho**

O projeto se baseia em uma modelagem de dados simples com DBFORK e Implementação com SSMS, utilizando DDL e DML para construir um sistema de compra e venda. A DDL é responsável por definir a estrutura do banco de dados, criando tabelas como Usuarios, Produtos e Movimentos, o que organiza bem os dados de maneira lógica.

A DML, por sua vez, é utilizada para manipular esses dados com operações de inserção, atualização, deleção e consultas (CRUD). Isso facilita o cadastro de usuários e o registro de movimentações de compra e venda, tornando o sistema funcional e simples.

A segurança é um aspecto crucial para este projeto, especialmente no que diz respeito ao armazenamento de senhas e à proteção de informações. Implementar boas práticas de segurança seria essencial para garantir mais confiabilidade no sistema.

Em resumo, a combinação de DDL e DML cria uma ótima base para o sistema, que pode ser aprimorada para atender a diferentes necessidades no futuro.