



**Curso**  
Desenvolvedor full Stack

**Campus**  
Polo Ingl Rio Verm - Florianópolis - SC

**Nome**  
Yuri Frederick de Sousa Cunha Bernardo Disciplina

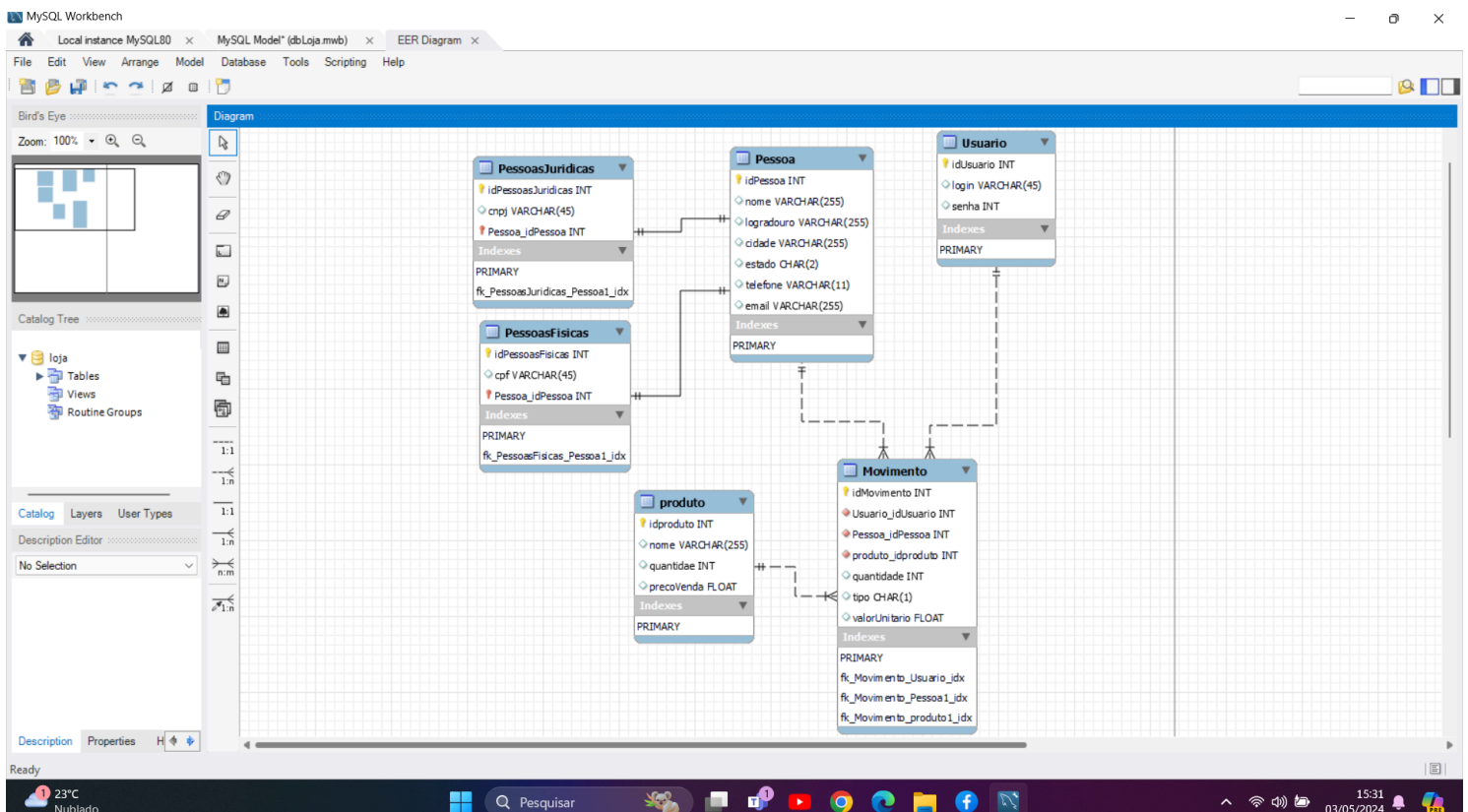
**Disciplina**  
Nível 2: Vamos manter as informações

**3º Semestre**

## Vamos manter as informações!

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

### Modelo lógico :



Utilizando o editor de SQL para criar as estruturas do modelo:

```
1 • CREATE DATABASE loja;
2
3 • USE loja;
4
5 • CREATE TABLE
6 Pessoa (
7     idPessoa INTEGER NOT NULL PRIMARY KEY,
8     nome VARCHAR(255) NOT NULL,
9     rua VARCHAR(255) NOT NULL,
10    cidade VARCHAR(255) NOT NULL,
11    estado CHAR(2) NOT NULL,
12    telefone VARCHAR(11) NOT NULL,
13    email VARCHAR(255) NOT NULL
14 );
15
16 • CREATE TABLE
17 PessoaJuridica (
18     idPessoa INTEGER NOT NULL PRIMARY KEY,
19     cnpj VARCHAR(14) NOT NULL,
20     CONSTRAINT fk_PessoaJuridica_Pessoa FOREIGN KEY (idPessoa) REFERENCES Pessoa (idPessoa)
21 );
```

```
22
23 • CREATE TABLE
24 PessoaFisica (
25     idPessoa INTEGER NOT NULL PRIMARY KEY,
26     cpf VARCHAR(11) NOT NULL,
27     constraint fk_PessoaFisica_Pessoa foreign key (idPessoa) references Pessoa (idPessoa)
28 );
29
30 • CREATE TABLE
31 Produto (
32     idProduto INTEGER NOT NULL PRIMARY KEY,
33     nome VARCHAR(255) NOT NULL,
34     quantidade INTEGER NOT NULL,
35     precoVenda double NOT NULL
36 );
37
38 • CREATE TABLE
39 Usuario (
40     idUsuario INTEGER NOT NULL PRIMARY KEY,
41     login VARCHAR(50) NOT NULL,
42     senha VARCHAR(50) NOT NULL
43 );
```

```

44
45 • CREATE TABLE
46   Movimento (
47     idMovimento INTEGER NOT NULL PRIMARY KEY,
48     idUsuario INTEGER NOT NULL,
49     idPessoa INTEGER NOT NULL,
50     idProduto INTEGER NOT NULL,
51     quantidade INTEGER NOT NULL,
52     tipo CHAR(1) NOT NULL,
53     valorUnitario FLOAT NOT NULL,
54     constraint fk_Movimento_Produto foreign key (idProduto) references Produto (idProduto),
55     constraint fk_Movimento_Usuario foreign key (idUsuario) references Usuario (idUsuario),
56     constraint fk_Movimento_Pessoa foreign key (idPessoa) references Pessoa (idPessoa)

```

## Alimentando a Base:

```

1 • use loja;
2
3 • insert into `loja`.`usuario`(`idUsuario`,`login`,`senha`) values (1,'op1','op1');
4 • insert into `loja`.`usuario`(`idUsuario`,`login`,`senha`) values (2,'op2','op2');
5
6 • select * from usuario;
7
8 • insert into `loja`.`produto`(`idProduto`,`nome`,`quantidade`,`precoVenda`) values (1,'Banana',100,5.00);
9 • insert into `loja`.`produto`(`idProduto`,`nome`,`quantidade`,`precoVenda`) values (3,'Laranja',500,2.00);
10 • insert into `loja`.`produto`(`idProduto`,`nome`,`quantidade`,`precoVenda`) values (4,'Manga',800,4.00);
11
12 • select * from produto;
13
14 • insert into `loja`.`pessoa`(`idPessoa`,`nome`,`rua`,`cidade`,`estado`,`telefone`,`email`) values (7,'Joao','Rua 12.casa 3.Quitanda','Riacho do Sul','PA','1111-1111','joao@riach
15 • insert into `loja`.`pessoafisica`(`idPessoa`,`cpf`) values (7,'11111111111');
16
17
18 • insert into `loja`.`pessoa`(`idPessoa`,`nome`,`rua`,`cidade`,`estado`,`telefone`,`email`) values (15,'JJC','Rua 11.Centro','Riacho do Norte','PA','1212-1212','jcc@riacho.com');
19 • insert into `loja`.`pessoajuridica`(`idPessoa`,`cnpj`) values (15,'22222222222222');
20
21 • insert into `loja`.`movimento`(`idMovimento`,`idUsuario`,`idPessoa`,`idProduto`,`quantidade`,`tipo`,`valorUnitario`) values (1,1,7,1,20,'S',4.00);
22 • insert into `loja`.`movimento`(`idMovimento`,`idUsuario`,`idPessoa`,`idProduto`,`quantidade`,`tipo`,`valorUnitario`) values (4,1,7,3,15,'S',2.00);
23 • insert into `loja`.`movimento`(`idMovimento`,`idUsuario`,`idPessoa`,`idProduto`,`quantidade`,`tipo`,`valorUnitario`) values (5,2,7,3,10,'S',3.00);
24 • insert into `loja`.`movimento`(`idMovimento`,`idUsuario`,`idPessoa`,`idProduto`,`quantidade`,`tipo`,`valorUnitario`) values (7,1,15,3,15,'E',5.00);
25 • insert into `loja`.`movimento`(`idMovimento`,`idUsuario`,`idPessoa`,`idProduto`,`quantidade`,`tipo`,`valorUnitario`) values (8,1,15,4,20,'E',4.00);
26

```

## Consultando os dados inseridos:

```

27   -- Dados completos de pessoas físicas
28 • select
29   pessoa.*,
30   pessoafisica.*
31   from pessoa
32   inner join pessoafisica on
33   pessoafisica.idPessoa = pessoa.idPessoa;
34
35   -- Dados completos de pessoas jurídicas
36 • select
37   pessoa.*,
38   pessoajuridica.*
39   from pessoa
40   inner join pessoajuridica on
41   pessoajuridica.idPessoa = pessoa.idPessoa;
42

```

```
43      -- Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.
44 •   select
45     P.nome,
46     fornecedor.nome as fornecedor,
47     M.quantidade,
48     M.valorUnitario,
49     M.quantidade * M.valorUnitario as valorTotal
50   from produto P
51   inner join movimento M on P.idProduto = M.idProduto,
52   (select
53     pessoa.nome,
54     pessoajuridica.*
55   from pessoa
56   inner join pessoajuridica on
57     pessoajuridica.idPessoa = pessoa.idPessoa
58   )fornecedor
59   where tipo = 'E';

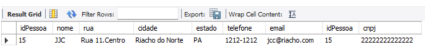
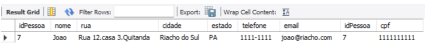
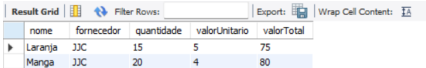
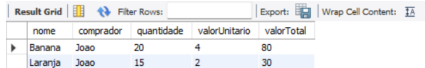
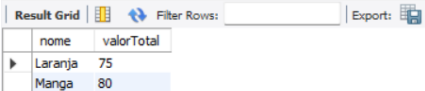
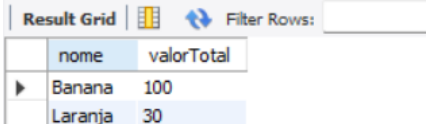
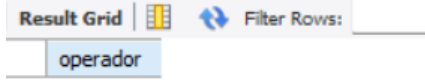
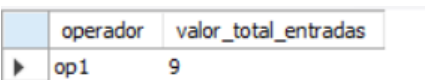
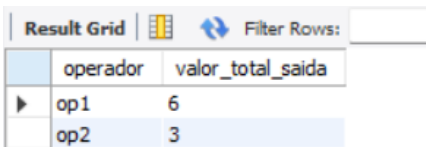
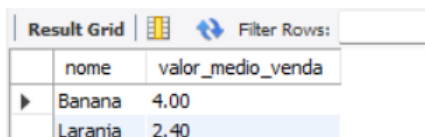
60
61      -- Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total.
62 •   select
63     P.nome,
64     comprador.nome as comprador,
65     M.quantidade,
66     M.valorUnitario,
67     M.quantidade * M.valorUnitario as valorTotal
68   from produto P
69   inner join movimento M on P.idProduto = M.idProduto,
70   (select
71     P.nome,
72     pessoaFisica.*
73   from pessoa P
74   inner join pessoaFisica on
75     pessoaFisica.idPessoa = P.idPessoa
76   )comprador
77   where tipo = 'S';
```

```

78
79 -- Valor total das entradas agrupadas por produto
80 • select
81 P.nome,
82 M.quantidade * M.valorUnitario as valorTotal
83 from produto P
84 inner join movimento M on P.idProduto = M.idProduto
85 where tipo = 'E';
86
87 -- Valor total das saídas agrupadas por produto
88 • select
89 P.nome,
90 M.quantidade * P.precoVenda as valorTotal
91 from produto P
92 inner join movimento M on P.idProduto = M.idProduto
93 where tipo = 'S';

```

## Resultados das execuções das Consultas:

<p>-- Dados completos de pessoas jurídicas</p>  <p>-- Dados completos de pessoas físicas</p> 	<p>-- Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.</p> 	<p>-- Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total.</p> 
<p>-- Valor total das entradas agrupadas por produto</p> 	<p>-- Valor total das saídas agrupadas por produto</p> 	<p>-- Operadores que não efetuaram movimentações de entrada(compra)</p> 
<p>-- Valor total de entrada por operador</p> 	<p>-- Valor total de saída por operador</p> 	<p>-- Valor médio de venda por produto, utilizando média ponderada</p> 

## Análise e Conclusão:

## **Cardinalidades em Bancos de Dados Relacionais:**

As cardinalidades descrevem o relacionamento quantitativo entre duas tabelas (entidades) em um banco de dados relacional. Existem quatro tipos principais de cardinalidade:

1:1 (Um para Um):

Neste tipo de relação, uma instância de uma tabela está associada a no máximo uma instância de outra tabela, e vice-versa. Por exemplo, a relação entre uma pessoa e seu passaporte.

1:N (Um para Muitos):

Aqui, uma instância de uma tabela está associada a zero, uma ou várias instâncias de outra tabela, mas uma instância da segunda entidade está associada a no máximo uma instância da primeira. Exemplo: a relação entre uma mãe e seus filhos.

N:1 (Muitos para Um):

É a inversão da relação um para muitos. Por exemplo, muitos livros podem ser publicados por uma única editora.

N:N (Muitos para Muitos):

Neste caso, várias instâncias de uma entidade podem estar associadas a várias instâncias de outra entidade. Exemplo: a relação entre livros e autores.

## **Herança em Bancos de Dados Relacionais:**

A herança em bancos de dados relacionais é um conceito proveniente da orientação a objetos. Ao modelarmos herança em um banco de dados, representamos a relação entre uma classe "super" (ou pai) e suas subclasses (ou filhas).

Existem diversas estratégias para mapear herança em um banco de dados relacional:

### **Tabela por Hierarquia:**

Mapear toda a hierarquia de classes para uma única tabela, armazenando todos os atributos das classes da hierarquia. Esta abordagem pode ser mais simples, mas pode resultar em colunas nulas para atributos específicos de subclasses.

### **Tabela por Subclasse:**

Criar uma tabela separada para cada classe (subclasse), incluindo os atributos específicos de cada uma. Esta abordagem evita colunas nulas, porém pode gerar mais tabelas e complexidade.

### **Tabela por Classe Abstrata:**

Criar uma tabela para a classe abstrata (superclasse) e tabelas adicionais para as subclasses. As tabelas de subclasses possuem uma chave estrangeira que referencia a tabela da superclasse. Esta abordagem mantém a integridade referencial e evita colunas nulas.

## **SQL Server Management Studio (SSMS):**

O SSMS é uma ferramenta de gerenciamento de banco de dados desenvolvida pela Microsoft. Oferece recursos para configurar, monitorar e administrar instâncias do SQL Server e bancos de dados.

Principais recursos do SSMS incluem:

Interface intuitiva com navegação organizada.

Editor de consultas avançado com realce de sintaxe e sugestões de código.

Designer de tabelas para criação e edição visual de estruturas de tabela.

Monitoramento de desempenho, segurança e gerenciamento de permissões.

Suporte a várias versões do SQL Server e integração com outras ferramentas da Microsoft.

O SSMS permite otimizar tarefas relacionadas ao gerenciamento de bancos de dados, aprimorando a produtividade dos administradores e desenvolvedores.

## **Diferenças entre SEQUENCE e IDENTITY:**

As sequências são invocadas conforme necessário, independentemente de tabelas ou campos específicos no banco de dados. Elas podem ser acessadas diretamente por aplicativos. Além disso, por meio de sequências, é viável obter o novo valor antes de utilizá-lo em um comando. Também é possível gerar novos valores em uma instrução UPDATE.

O identity está vinculado à tabela e associado a uma coluna específica. Não é possível recuperar um novo valor antes de utilizá-lo, e não é viável gerar novos valores em uma instrução UPDATE.

Em síntese, as sequências proporcionam mais flexibilidade e controle em comparação com o identity.

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

As chaves estrangeiras são fundamentais para garantir a integridade referencial dos dados em um banco de dados. Elas desempenham as seguintes funções:

### **Normalização:**

As chaves estrangeiras auxiliam na normalização do banco de dados, reduzindo a redundância de dados e mantendo a consistência das informações.

### **Consistência dos Dados:**

Uma chave estrangeira impede a exclusão de registros de uma tabela pai se houver registros relacionados na tabela filha.

Operadores da Álgebra Relacional e do Cálculo Relacional:

## **Álgebra Relacional:**

**Seleção:** Selecciona tuplas (linhas) que atendem a um predicado ou condição específicos.

**Projeção:** Retorna apenas as colunas especificadas.

**Produto Cartesiano:** Combina todas as tuplas de duas tabelas.

**União:** Combina todas as tuplas de duas tabelas, eliminando duplicatas.

**Diferença entre Conjuntos:** Retorna as tuplas presentes em uma tabela, mas ausentes na outra.

**Renomeação:** Modifica os nomes das colunas.

## **Cálculo Relacional:**

**Seleção:** Selecciona tuplas que satisfaçam uma condição específica.

**Projeção:** Retorna apenas as colunas especificadas.

**Junção:** Combina tuplas de duas tabelas com base em uma condição.

**Divisão:** Retorna as tuplas que correspondem a todas as combinações de valores de uma subconsulta.

**Renomeação:** Altera os nomes das colunas.

## **Agrupamento em Consultas:**

O GROUP BY é uma cláusula poderosa do SQL que permite agrupar registros com base em dados de uma coluna (ou colunas) específica.

Por meio do GROUP BY, é possível calcular métricas como médias, valores máximos e mínimos para cada grupo de registros.

A cláusula HAVING permite filtrar grupos com base em condições após a realização do agrupamento.

Funções agregadas (tais como COUNT, SUM e AVG) são frequentemente empregadas com o GROUP BY para calcular estatísticas para cada grupo.

Link GitHub: [Yuri6406/3Semestre-ProjetoNivel2 \(github.com\)](https://github.com/Yuri6406/3Semestre-ProjetoNivel2)