	<p>Universidade Estácio</p> <p>Campus Polo Centro - Barão de Cocais – MG</p> <p>Curso de Desenvolvimento Full Stack</p> <p>Relatório da Missão Prática 2 – Mundo 3</p>
Disciplina:	RPG0015 - Vamos Manter as Informações
Nome:	Wesley Borges do Carmo de Oliveira – 202305150171
Turma:	9001 – 3º Semestre

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

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

1º Procedimento | Criando o Banco de Dados

Resultados

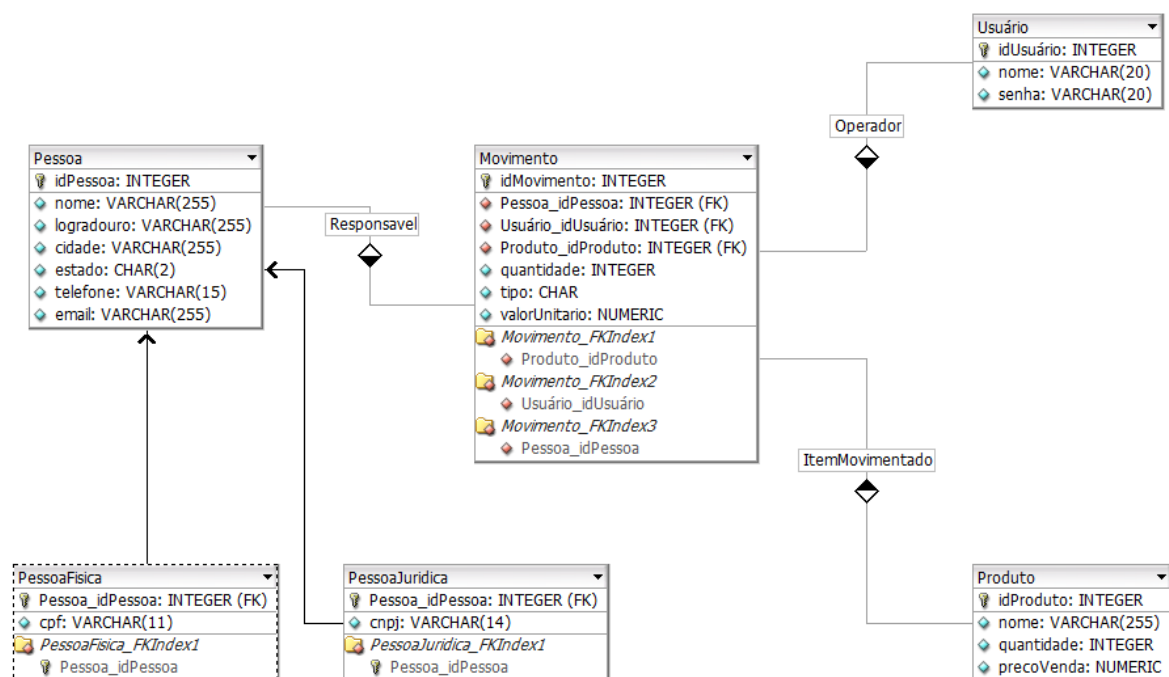


Figura 01: Diagrama Entidade-Relacionamento (DER).

Códigos solicitados

```
USE Loja;  
GO
```

```
CREATE SEQUENCE orderPessoa  
as INT  
START WITH 1  
INCREMENT BY 1;
```

```
CREATE TABLE Pessoa (  
idPessoa INTEGER NOT NULL,  
nome VARCHAR(255),  
logradouro VARCHAR(255),  
cidade VARCHAR(255),  
estado CHAR(2),  
telefone VARCHAR(15),  
email VARCHAR(255),  
PRIMARY KEY(idPessoa));  
GO
```

```
CREATE TABLE PessoaFisica (  
Pessoa_idPessoa INTEGER NOT NULL,  
cpf VARCHAR(11) NOT NULL,  
PRIMARY KEY(Pessoa_idPessoa),  
FOREIGN KEY(Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)  
ON UPDATE CASCADE  
ON DELETE CASCADE  
);  
GO
```

```
CREATE TABLE PessoaJuridica (  
Pessoa_idPessoa INTEGER NOT NULL,  
cnpj VARCHAR(14) NOT NULL,  
PRIMARY KEY(Pessoa_idPessoa),  
FOREIGN KEY(Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)  
ON UPDATE CASCADE  
ON DELETE CASCADE  
);  
GO
```

```
CREATE TABLE Usuário (  
idUserário INTEGER NOT NULL,  
nome VARCHAR(20) NOT NULL,  
senha VARCHAR(20) NOT NULL,  
PRIMARY KEY(idUsuário));  
GO
```

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

```

GO

CREATE TABLE Movimento (
idMovimento INTEGER NOT NULL,
Pessoa_idPessoa INTEGER NOT NULL,
Usuário_idUsuário INTEGER NOT NULL,
Produto_idProduto INTEGER NOT NULL,
quantidade INTEGER,
tipo CHAR(1),
valorUnitario NUMERIC,
PRIMARY KEY(idMovimento) ,
FOREIGN KEY(Produto_idProduto)
REFERENCES Produto(idProduto)
ON UPDATE CASCADE
ON DELETE CASCADE,
FOREIGN KEY(Usuário_idUsuário)
REFERENCES Usuário(idUsuário)
ON UPDATE CASCADE
ON DELETE CASCADE,
FOREIGN KEY(Pessoa_idPessoa)
REFERENCES Pessoa(idPessoa)
ON UPDATE CASCADE
ON DELETE CASCADE);
GO

```

Figura 02: Script de criação das tabelas do banco de dados.

Análise e Conclusão

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

Assim como o próprio nome supõe, o relacionamento 1X1 define que um item de uma entidade só poderá se relacionar com um item de outra entidade. Por exemplo, supondo que temos as entidades Cliente e Endereço que se relacionam de forma 1X1, um cliente só poderá possuir um endereço, que também só pode estar relacionado a um cliente.

Relacionamento de Um-para-Muitos (1 X N) ou Muitos-para-Um (N X 1): Um elemento de uma entidade A pode se relacionar com mais de um elemento de outra entidade B.

Quando há uma cardinalidade 1XN ou NX1 ocorre a inclusão da chave primária da tabela que possui cardinalidade mínima 1 na tabela que possui cardinalidade máxima N. Assim fazendo referência a sua tabela de origem.

O relacionamento NXN define que um item de uma tabela pode se relacionar com vários itens de uma outra tabela e vice-versa. Por exemplo, podemos determinar que um pedido pode possuir diversos produtos relacionados a ele, assim como um mesmo produto pode estar relacionado a diversos pedidos diferentes.

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

Em bancos de dados relacionais, o tipo de relacionamento utilizado para representar herança é a generalização/especialização. Essa técnica permite organizar as entidades em uma hierarquia, onde uma entidade superclasse (geralmente mais abrangente) pode ter uma ou mais entidades subclasse (mais específicas).

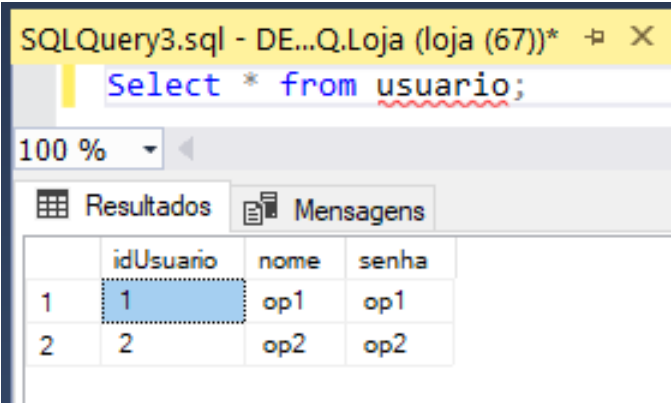
- c) Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?

O SQL Server Management Studio é uma ferramenta essencial para administradores de banco de dados e desenvolvedores que trabalham com o SQL Server. Com seus recursos avançados e interface intuitiva, o SSMS facilita o gerenciamento e a manutenção de bancos de dados, permitindo um trabalho mais eficiente e produtivo.

- O SSMS possui uma interface gráfica intuitiva, com uma navegação fácil e organizada. Isso permite que os usuários encontrem rapidamente as opções e recursos necessários para gerenciar seus bancos de dados.
- O SSMS possui um editor de consultas avançado, que oferece recursos como realce de sintaxe, sugestões de código e execução de consultas em tempo real. Isso facilita o desenvolvimento e a execução de consultas complexas.
- Com o SSMS, os usuários podem criar e editar tabelas visualmente por meio do designer de tabelas. Isso simplifica o processo de criação e modificação de estruturas de tabela.
- O SSMS permite que os usuários monitorem o desempenho do servidor de banco de dados, identificando gargalos e otimizando consultas. Ele fornece informações detalhadas sobre o uso de recursos, bloqueios, consultas em execução e muito mais.
- O SSMS oferece recursos abrangentes de segurança, permitindo que os administradores de banco de dados gerenciem permissões de usuário, criem logins e configurem políticas de segurança. Isso garante a proteção dos dados armazenados no banco de dados.
- O SSMS é compatível com várias versões do SQL Server, permitindo que os usuários gerenciem bancos de dados em diferentes ambientes. Isso é especialmente útil para organizações que possuem bancos de dados em diferentes versões do SQL Server.
- O SSMS se integra perfeitamente a outras ferramentas e serviços da Microsoft, como o Visual Studio e o Azure. Isso permite que os usuários acessem recursos adicionais e estendam as funcionalidades do SSMS.

2º Procedimento | Alimentando a Base

Resultados



SQLQuery3.sql - DE...Q.Loja (loja (67))*

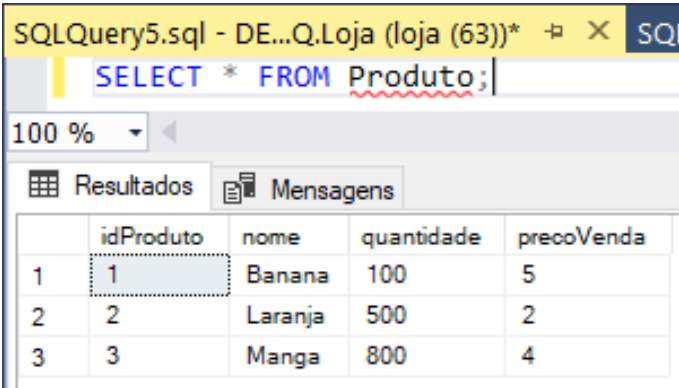
```
Select * from usuario;
```

100 %

Resultados Mensagens

	idUsuario	nome	senha
1	1	op1	op1
2	2	op2	op2

Figura 03: Tabela Usuario.



SQLQuery5.sql - DE...Q.Loja (loja (63))*

```
SELECT * FROM Produto;
```

100 %

Resultados Mensagens

	idProduto	nome	quantidade	precoVenda
1	1	Banana	100	5
2	2	Laranja	500	2
3	3	Manga	800	4

Figura 04: Tabela Produto.



create_script01.sql...NQ.Loja (loja (62)) insert_script02.sql...2NQ.Loja (loja (53))* SQLQuery4.sql - DE...

```
SELECT * FROM Pessoa;
```

100 %

Resultados Mensagens

	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	1	João Silva	Rua das Flores, 123	São Paulo	SP	11987654321	joao.silva@example.com
2	2	Maria Oliveira	Avenida Paulista, 456	São Paulo	SP	11987654322	maria.oliveira@example.com
3	3	Pedro Santos	Praça da Sé, 789	Rio de Janeiro	RJ	21987654323	pedro.santos@example.com
4	4	Ana Souza	Rua XV de Novembro, 101	Curitiba	PR	41987654324	ana.souza@example.com
5	5	Carlos Pereira	Avenida Brasil, 202	Belo Horizonte	MG	31987654325	carlos.pereira@example.com

Figura 05: Tabela Pessoa.

create_script01.sql...NQ.Loja (loja (62))

```
SELECT * FROM PessoaFisica;
```

100 %

Resultados Mensagens

	Pessoa_idPessoa	cpf
1	1	12345678900
2	2	23456789011
3	3	34567890122

Figura 06: Tabela PessoaFisica.

create_script01.sql...NQ.Loja (loja (62))

```
SELECT * FROM PessoaJuridica;
```

100 %

Resultados Mensagens

	Pessoa_idPessoa	cnpj
1	4	98765432000188
2	5	12345678000199

Figura 07: Tabela PessoaJuridica.

insert_script02.sql...2NQ.Loja (loja (53))* select_script03.sql...2NQ.Loja (loja (60)) SQLQ

```
SELECT * FROM Movimento;
```

100 %

Resultados Mensagens

	idMovimento	Pessoa_idPessoa	Usuario_idUsuario	Produto_idProduto	quantidade	tipo	valorUnitario
1	1	1	1	1	5	S	4
2	2	3	2	3	8	S	3
3	3	2	1	1	35	E	4
4	4	5	2	2	25	E	6

Figura 08: Tabela Movimento.

create_script01.sql...NQ.Loja (loja (62)) insert_script02.sql...2NQ.Loja (loja (53))* SQLQuery4.sql - DE...Q.Loja (loja (62))

```
SELECT p.*, pf.cpf FROM
Pessoa p
INNER JOIN PessoaFisica pf ON p.idPessoa = pf.Pessoa_idPessoa;
```

100 %

Resultados Mensagens

	idPessoa	nome	logradouro	cidade	estado	telefone	email	cpf
1	1	João Silva	Rua das Flores, 123	São Paulo	SP	11987654321	joao.silva@example.com	12345678900
2	2	Maria Oliveira	Avenida Paulista, 456	São Paulo	SP	11987654322	maria.oliveira@example.com	23456789011
3	3	Pedro Santos	Praça da Sé, 789	Rio de Janeiro	RJ	21987654323	pedro.santos@example.com	34567890122

Figura 09: Consulta de dados completos das pessoa físicas.

create_script01.sql...NQ.Loja (loja (62)) insert_script02.sql...2NQ.Loja (loja (53)) SQLQuery4.sql - DE...Q.Loja (loja (60))*

```

SELECT p.*, pj.cnpj
FROM Pessoa p
INNER JOIN PessoaJuridica pj ON p.idPessoa = pj.Pessoa_idPessoa;

```

100 %

Resultados Mensagens

	idPessoa	nome	logradouro	cidade	estado	telefone	email	cnpj
1	4	Ana Souza	Rua XV de Novembro, 101	Curitiba	PR	41987654324	ana.souza@example.com	98765432000188
2	5	Carlos Pereira	Avenida Brasil, 202	Belo Horizonte	MG	31987654325	carlos.pereira@example.com	12345678000199

Figura 10: Consulta de dados completos das pessoa jurídicas.

insert_script02.sql...2NQ.Loja (loja (53))* select_script03.sql...2NQ.Loja (loja (60))* SQLQuery5.sql - DE...Q.Loja (loja (63))* create_scri

```

SELECT m.*, p.nome as fornecedor, pr.nome as Produto, m.quantidade, m.valorUnitario, (m.quantidade * m.valorU
FROM Movimento m
INNER JOIN Pessoa p ON p.idPessoa = m.Pessoa_idPessoa
INNER JOIN Produto pr ON pr.idProduto = m.Produto_idProduto
WHERE m.tipo = 'E';

```

100 %

Resultados Mensagens

	idMovimento	Pessoa_idPessoa	Usuario_idUsuario	Produto_idProduto	quantidade	tipo	valorUnitario	fornecedor	Produto	quantidade	valorUnitario	total
1	3	2	1	1	35	E	4	Maria Oliveira	Banana	35	4	140
2	4	5	2	2	25	E	6	Carlos Pereira	Laranja	25	6	150

Figura 11: Consulta de dados de movimento entrada.

insert_script02.sql...2NQ.Loja (loja (53))* select_script03.sql...2NQ.Loja (loja (60)) SQLQuery5.sql - DE...Q.Loja (loja (63))* create_scri

```

SELECT m.*, p.nome as comprador, pr.nome as Produto, m.quantidade, m.valorUnitario, (m.quantidade * m.valorU
FROM Movimento m
INNER JOIN Pessoa p ON m.Pessoa_idPessoa = p.idPessoa
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S';

```

100 %

Resultados Mensagens

	idMovimento	Pessoa_idPessoa	Usuario_idUsuario	Produto_idProduto	quantidade	tipo	valorUnitario	comprador	Produto	quantidade	valorUnitario	total
1	1	1	1	1	5	S	4	João Silva	Banana	5	4	20
2	2	3	2	3	8	S	3	Pedro Santos	Manga	8	3	24

Figura 12: Consulta de dados de movimento saída.

insert_script02.sql...2NQ.Loja (loja (53))* select_script03.sql...2NQ.Loja (loja (60)) SC

```

SELECT pr.nome, SUM(m.quantidade * m.valorUnitario) as valor_total
FROM Movimento m
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'E'
GROUP BY pr.nome;

```

100 %

Resultados Mensagens

	nome	valor_total
1	Banana	140
2	Laranja	150

Figura 13: Consulta de dados do valor total de entrega.

insert_script02.sql...2NQ.Loja (loja (53))* select_script03.sql...2NQ.Loja (loja (60))

```

SELECT pr.nome, SUM(m.quantidade * m.valorUnitario) as valor_total
FROM Movimento m
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S'
GROUP BY pr.nome;

```

100 %

Resultados Mensagens

	nome	valor_total
1	Banana	20
2	Manga	24

Figura 14: Consulta de dados do valor total de saída.

insert_script02.sql...2NQ.Loja (loja (53))* select_script03.sql...2NQ.Loja (loja (60))

```

SELECT u.nome, SUM(m.quantidade * m.valorUnitario) as valor_total
FROM Movimento m
INNER JOIN Usuario u ON m.Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.nome

```

100 %

Resultados Mensagens

	nome	valor_total
1	op1	140
2	op2	150

Figura 15: Consulta de dados de operadores que não fizeram movimento de entrega.

insert_script02.sql...2NQ.Loja (loja (53))* select_script03.sql...2NQ.Loja (loja (60)) SQLQuery5.sql - DE...Q.Loja (loja (63))*

```

SELECT u nome, SUM(m.quantidade * m.valorUnitario) as valor_total
FROM Movimento m
INNER JOIN Usuario u ON m.Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.nome

```

100 %

Resultados Mensagens

	nome	valor_total
1	op1	20
2	op2	24

Figura 16: Consulta de dados do valor total de entrega, agrupado por operador.

insert_script02.sql...2NQ.Loja (loja (53))*		select_script03.sql...2NQ.Loja (loja (60))	SQLQuery5.sql - DE
<pre> SELECT pr.nome, SUM(m.quantidade * m.valorUnitario) / SUM(m.quantidade) as media FROM Movimento m INNER JOIN Produto pr ON m.Produto_idProduto = pr.IdProduto WHERE m.tipo = 'S' GROUP BY pr.nome; </pre>			
100 %			
Resultados		Mensagens	
	nome	media	
1	Banana	4.000000	
2	Manga	3.000000	

Figura 17: Consulta de dados do valor médio ponderado de venda por produto

Códigos solicitados

```

INSERT INTO Usuario(nome,senha)
VALUES('op1','op1'),
('op2','op2');

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

INSERT INTO Pessoa(idPessoa,nome,logradouro,cidade,estado,telefone,email)
VALUES(NEXT VALUE FOR orderPessoa, 'João Silva', 'Rua das Flores, 123',
'São Paulo', 'SP', '11987654321', 'joao.silva@example.com'),
(NEXT VALUE FOR orderPessoa, 'Maria Oliveira', 'Avenida Paulista, 456', 'São
Paulo', 'SP', '11987654322', 'maria.oliveira@example.com'),
(NEXT VALUE FOR orderPessoa, 'Pedro Santos', 'Praça da Sé, 789', 'Rio de
Janeiro', 'RJ', '21987654323', 'pedro.santos@example.com'),
(NEXT VALUE FOR orderPessoa, 'Ana Souza', 'Rua XV de Novembro, 101',
'Curitiba', 'PR', '41987654324', 'ana.souza@example.com'),
(NEXT VALUE FOR orderPessoa, 'Carlos Pereira', 'Avenida Brasil, 202', 'Belo
Horizonte', 'MG', '31987654325', 'carlos.pereira@example.com');

INSERT INTO PessoaFisica(Pessoa_idPessoa, cpf)
VALUES(1, '12345678900'),
(2, '23456789011'),
(3, '34567890122');

INSERT INTO PessoaJuridica(Pessoa_idPessoa,cnpj)
VALUES(4, '98765432000188'),
(5, '12345678000199');

INSERT INTO Movimento(Pessoa_idPessoa, Usuario_idUsuario, Produto_idProduto,
quantidade, tipo, valorUnitario)
VALUES(1,1,1,5,'S',4.00),
(3,2,3,8,'S',3.00),
(2,1,1,35,'E',4.00),
(5,2,2,25,'E',6.00);

```

Figura 18: Script de inserção de dados ao banco de dados.

```

--(a)
SELECT p.*, pf.cpf
FROM Pessoa p
INNER JOIN PessoaFisica pf ON p.idPessoa = pf.Pessoa_idPessoa;

--(b)
SELECT p.*, pj.cnpj
FROM Pessoa p
INNER JOIN PessoaJuridica pj ON p.idPessoa = pj.Pessoa_idPessoa;

--(c)
SELECT m.*, p.nome as fornecedor, pr.nome as Produto, m.quantidade,
m.valorUnitario, (m.quantidade * m.valorUnitario) as total
FROM Movimento m
INNER JOIN Pessoa p ON p.idPessoa = m.Pessoa_idPessoa
INNER JOIN Produto pr ON pr.idProduto = m.Produto_idProduto
WHERE m.tipo = 'E';

--(d)
SELECT m.*, p.nome as comprador, pr.nome as Produto, m.quantidade,
m.valorUnitario, (m.quantidade * m.valorUnitario) as total
FROM Movimento m
INNER JOIN Pessoa p ON m.Pessoa_idPessoa = p.idPessoa
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S';

--(e)
SELECT pr.nome, SUM(m.quantidade * m.valorUnitario) as valor_total
FROM Movimento m
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'E'
GROUP BY pr.nome;

--(f)
SELECT pr.nome, SUM(m.quantidade * m.valorUnitario) as valor_total
FROM Movimento m
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S'
GROUP BY pr.nome;

--(g)
SELECT u.*
FROM Usuario u
LEFT JOIN Movimento m ON u.idUsuario = m.Usuario_idUsuario AND m.tipo = 'E'
WHERE m.idMovimento IS NULL;

--(h)
SELECT u.nome, SUM(m.quantidade * m.valorUnitario) as valor_total
FROM Movimento m
INNER JOIN Usuario u ON m.Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.nome;

```

```

--(i)
SELECT u.nome, SUM(m.quantidade * m.valorUnitario) as valor_total
FROM Movimento m
INNER JOIN Usuario u ON m.Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.nome;

--(j)
SELECT pr.nome, SUM(m.quantidade * m.valorUnitario) / SUM(m.quantidade) as
media
FROM Movimento m
INNER JOIN Produto pr ON m.Produto_idProduto = pr.IdProduto
WHERE m.tipo = 'S'
GROUP BY pr.nome;

```

Figura 19: Script de consulta de dados do banco de dados.

Análise e Conclusão

a) Quais as diferenças no uso de sequence e identity?

Uma das diferenças entre SEQUENCE e IDENTITY está no fato de que as SEQUENCES são acionadas sempre quando forem necessárias, sem dependência de tabelas e campos no banco, onde pode ser chamada diretamente por aplicativos, nas SEQUENCES, também podemos obter o novo valor antes de usá-lo em um comando, diferente do IDENTITY, onde não podemos obter um novo valor. Além disso, com o IDENTITY não podemos gerar novos valores em uma instrução UPDATE, enquanto que com SEQUENCE, já podemos.

Uma das grandes utilidades em IDENTITY, está no fato de podermos trabalhar com o mesmo na utilização de TRANSAÇÕES de INSERT, pois, só iremos gerar um próximo valor a partir do momento que o comando for executado, ou seja, que a transação for aceita, ao contrário de uma SEQUENCE, que uma vez chamado seu próximo valor, mesmo que ocorra um erro de transação, o valor é alterado.

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

Além de ajudar a descrever o relacionamento nos modelos, as chaves estrangeiras são usados principalmente pra manter a integridade dos dados, ou seja imagine que você tem duas tabelas ligadas por uma chave estrangeira e tem dados na tabela B ligados a uma especifica linha na tabela A, então você, se você tentar deletar esta linha especifica o banco vai lhe impedir e vai enviar um erro.

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

A álgebra relacional é uma forma de cálculo sobre conjuntos ou relações, ela recebia pouca atenção até a publicação do modelo relacional de dados de E.F Codd, em 1970. Codd propôs tal álgebra como uma base para linguagens de consulta em banco de dados.

Há seis operações fundamentais na álgebra relacional, Seleção, Projeção, Produto cartesiano, União, Diferença entre conjuntos, Renomear.

O cálculo relacional é um modelo formal que se baseia na lógica de predicados e que permite manipular relações no modelo relacional, ele possui o mesmo poder expressivo da álgebra relacional. Uma expressão do cálculo relacional é igualmente uma relação que representa o resultado de uma consulta à base de dados.

As expressões do cálculo podem ser especificadas em termos de variáveis sobre os tuplos, cálculo relacional por tuplos (ou CRT), ou em termos de variáveis sobre o domínio dos atributos, cálculo relacional por domínios (ou CRD).

O cálculo relacional é uma linguagem não-procedimental. Nas expressões do cálculo não se especifica o modo de obter o resultado mas sim o tipo de informação que se pretende obter. Isto difere da álgebra relacional onde é necessário especificar a sequência de operações a aplicar para obter o resultado. A linguagem SQL baseia-se em parte no cálculo relacional por tuplos.

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

O agrupamento em consultas é realizado através da cláusula "GROUP BY", utilizada para agrupar linhas baseada em uma função aplicada sobre uma coluna. O requisito obrigatório, além do uso da cláusula GROUP BY, é incluir alguma função de agrupamento tal como: SUM, COUNT, AVG, MAX, MIN, entre outras.