

**Disciplina:** RPG0015 - Vamos manter as informações!

**Nome:** João Gilberto dos Santos

**Turma:** 2022.4

## 1º Título da Prática: Criando o Banco de Dados

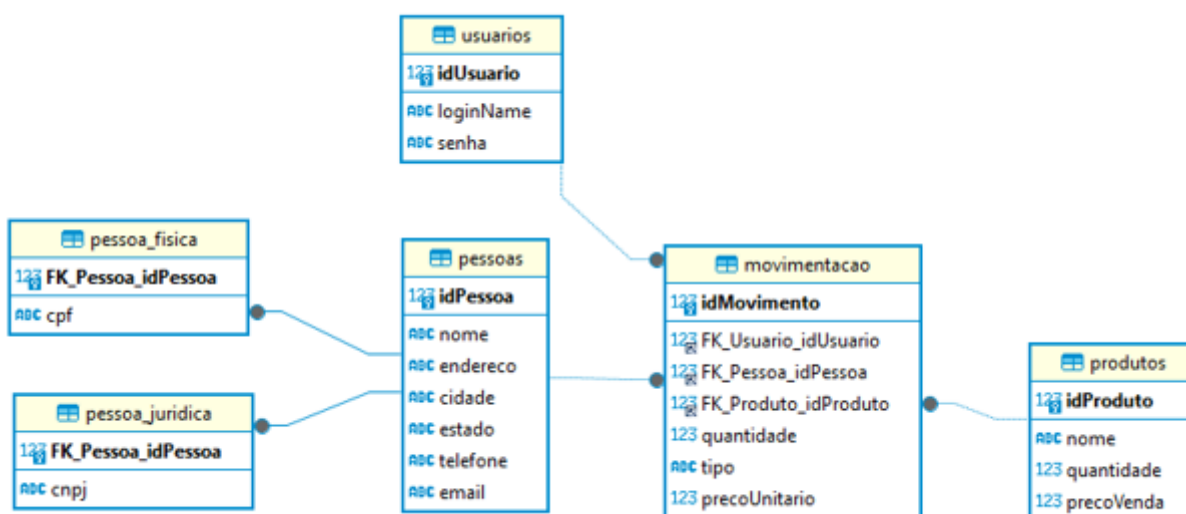
## 2º Objetivo da Prática

O objetivo deste relatório é demonstrar os resultados obtidos após a implementação de uma modelagem de banco de dados (DDL e DML) referente a um sistema de movimentação de produtos, onde o mesmo apresenta as seguintes entidades:

1. pessoa física;
2. pessoa jurídica;
3. usuário;
4. movimentação;
5. produto.

Tal implementação será utilizada para a posteriori para o desenvolvimento de um sistema que será escrito na linguagem JAVA

**Imagem 1:** Entidade-Relacionamento (DER).



### 3º 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),
endereco VARCHAR(255),
cidade VARCHAR(255),
estado CHAR(2),
telefone VARCHAR(15),
email VARCHAR(255),
CONSTRAINT CPK_Pessoa PRIMARY KEY CLUSTERED(idPessoa ASC)
);
GO

CREATE TABLE PessoaFisica(
FK_Pessoa_idPessoa INTEGER NOT NULL,
cpf VARCHAR(11) NOT NULL,
CONSTRAINT CPK_PessoaFisica PRIMARY KEY CLUSTERED(FK_Pessoa_idPessoa ASC),
CONSTRAINT CFK_Pessoa_PessoaFisica FOREIGN KEY(FK_Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)
ON UPDATE CASCADE
ON DELETE CASCADE
);
GO

CREATE TABLE PessoaJuridica(
FK_Pessoa_idPessoa INTEGER NOT NULL,
cnpj VARCHAR(14) NOT NULL,
CONSTRAINT CPK_PessoaJuridica PRIMARY KEY CLUSTERED(FK_Pessoa_idPessoa ASC),
CONSTRAINT CFK_Pessoa_PessoaJuridica FOREIGN KEY(FK_Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)
ON UPDATE CASCADE
ON DELETE CASCADE
);
GO

CREATE TABLE Usuario(
idUserario INTEGER NOT NULL IDENTITY,
loginName VARCHAR(20) NOT NULL,
senha VARCHAR(20) NOT NULL,
CONSTRAINT CPK_Usuario PRIMARY KEY CLUSTERED(idUsuario ASC)
);
GO

CREATE TABLE Produto(
idProduto INTEGER NOT NULL IDENTITY,
nome VARCHAR(255) NOT NULL,
quantidade INTEGER,
precoVenda NUMERIC,
CONSTRAINT CPK_Produto PRIMARY KEY CLUSTERED(idProduto ASC)
);
GO

CREATE TABLE Movimento(
idMovimento INTEGER NOT NULL IDENTITY,
FK_Usuario_idUsuario INTEGER NOT NULL,
FK_Pessoa_idPessoa INTEGER NOT NULL,
FK_Produto_idProduto INTEGER NOT NULL,
quantidade INTEGER,
tipo CHAR(1),
precoUnitario NUMERIC,
CONSTRAINT CPK_Movimento PRIMARY KEY CLUSTERED(idMovimento ASC),
CONSTRAINT CFK_Usuario_Movimento FOREIGN KEY(FK_Usuario_idUsuario) REFERENCES Usuario(idUsuario)
ON UPDATE CASCADE
ON DELETE CASCADE,
CONSTRAINT CFK_Pessoa_Movimento FOREIGN KEY(FK_Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)
ON UPDATE CASCADE
ON DELETE CASCADE,
CONSTRAINT CFK_Produto_Movimento FOREIGN KEY(FK_Produto_idProduto) REFERENCES Produto(idProduto)
ON UPDATE CASCADE
ON DELETE CASCADE
);
GO
```

## 4º Os resultados da execução dos códigos:

SQLQuery9 - 3\*Script...Gilberto (Old) (74) - Microsoft SQL Server Management Studio

Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda

Loja Executar

SQLQuery9 - 3\*Script...Gilberto (Old) (74) SQLQuery9 - 2\*Script...Gilberto (Old) (66) SQLQuery9 - 1\*Script...Gilberto (Old) (54)

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

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

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

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

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

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

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

-- item (h)
SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as compras
FROM Movimento m
INNER JOIN Usuario u ON m.FK_Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.loginName;

-- item (i)
SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as vendas
FROM Movimento m
INNER JOIN Usuario u ON m.FK_Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.loginName;
```

100 %

Resultados Mensagens

idPessoa	nome	endereco	cidade	estado	telefone	email	cpf
1	João Gilberto	Avenida Conde da Boa Vista, 1044	Recife	PE	3304-2431	gigi@hotmail.com	06772345477
2	Bruno Cavalcante	Professor Jose Amarino dos Reis, 289	Recife	PE	3389-3000	calvo@ig.com.br	01114549834
3	Ridaiso	Tamboara, 17	Recife	PE	3400-3400	rsadinha@yahoo.com	00879462300

idPessoa	nome	endereco	cidade	estado	telefone	email	cnpj
4	Panificadora Andrea	Beco do Pavão, 89	Recife	PE	3000-3232	andrea@panificadora.com.br	05311244000109
5	BiBike	Avenida Norte, 3409	Recife	PE	3100-9999	bibike@bibiki.com.br	07455190000111

idMovimento	FK_Usuario_idUsuario	FK_Pessoa_idPessoa	FK_Produto_idProduto	quantidade	tipo	precoUnitario	fornecedor	Produto	quantidade	precoUnitario	total
1	1	1	1	10	E	1	João Gilberto	Parafuso	10	1	10
2	3	1	3	30	E	3	Ridaiso	Cola	30	3	90

idMovimento	FK_Usuario_idUsuario	FK_Pessoa_idPessoa	FK_Produto_idProduto	quantidade	tipo	precoUnitario	comprador	Produto	quantidade	precoUnitario	total
1	2	2	2	20	S	2	Bruno Cavalcante	Prego	20	2	40

nome	compras
Cola	90
Parafuso	10

nome	vendas
Prego	40

idUsuario	loginName	senha
1	2	op2

loginName	compras
op1	100

loginName	vendas
op2	40

nome	media
Prego	2.000000

Consulta executada com êxito.

DESKTOP-J0D1HA4 (16.0 RTM) DESKTOP-J0D1HA4João G... Loja 00:00:00 15 linhas

Pronto Li 66 Col 18 Car 18 INS

## 5º Análise e Conclusão:

**A** - Bancos de dados relacionais, a cardinalidade define a relação numérica entre entidades, especificando quantas instâncias de uma entidade podem se relacionar com quantas instâncias de outra. As três cardinalidades básicas são:

### 1x1 (Um para Um):

- **Descrição:** Uma instância em uma entidade se relaciona com no máximo uma instância na outra entidade.
- **Implementação:**
  - **Chave Primária:** A chave primária de uma entidade pode ser usada como chave estrangeira na outra entidade, ou vice-versa.
  - **Coluna de Flag:** Uma coluna adicional pode ser usada para indicar a presença ou ausência de uma correspondência (por exemplo, "professor\_id" na tabela "aluno" com flag "professor\_atribuido").

**Exemplo:** Colaborador pode ter apenas único endereço principal (1x1).

### 1xN (Um para Muitos):

- **Descrição:** Uma instância em uma entidade se relaciona com zero, uma ou várias instâncias na outra entidade.
- **Implementação:**
  - **Chave Estrangeira:** A chave primária da entidade "um" é armazenada como chave estrangeira na entidade "muitos".
  - **Coluna de Flag:** Uma coluna adicional pode ser usada para indicar o relacionamento principal (por exemplo, "departamento\_id" na tabela "funcionário" com flag "departamento\_principal").

**Exemplo:** Um departamento pode ter vários funcionários (1xN).

### NxN (Muitos para Muitos):

- **Descrição:** Uma instância em uma entidade se relaciona com zero, uma ou várias instâncias na outra entidade, e vice-versa.
- **Implementação:**
  - **Tabela de Junção:** Uma tabela separada é criada para armazenar as relações entre as entidades. Essa tabela contém as chaves primárias de ambas as entidades originais.
  - **Colunas Adicionais:** Atributos adicionais podem ser incluídos na tabela de junção para armazenar informações específicas da relação.

**Exemplo:** Um aluno pode estar matriculado em vários cursos, e um curso pode ter vários alunos matriculados (NxN).

**B** - Bancos de dados relacionais não suportam herança de forma nativa, como em linguagens de programação orientadas a objetos. No entanto, existem diferentes estratégias para modelar hierarquias de entidades e relacionamentos de herança nesses bancos de dados. As três principais abordagens são:

### 1. Tabela por Hierarquia:

- **Descrição:** Uma única tabela armazena os dados de todas as entidades na hierarquia.
- **Vantagens:**
  - Simplicidade de implementação.
  - Boa visualização da hierarquia completa.
- **Desvantagens:**
  - Redundância de dados, especialmente para atributos comuns.
  - Dificuldade em recuperar dados específicos de uma entidade.
  - Ineficiência em consultas que não acessam todos os atributos da hierarquia.

### 2. Tabela por Classe Concreta:

- **Descrição:** Cada entidade concreta (subclasse) possui sua própria tabela, armazenando apenas seus atributos específicos.
- **Vantagens:**
  - Elimina redundância de dados.
  - Otimiza consultas para entidades específicas.
- **Desvantagens:**
  - Complexidade de implementação e manutenção.
  - Dificuldade em visualizar a hierarquia completa.
  - Necessidade de junções entre tabelas para recuperar dados de superclasses.

### 3. Tabela Única com Discriminador:

- **Descrição:** Uma única tabela armazena os dados de todas as entidades, com uma coluna adicional que indica o tipo de entidade ("discriminador").
- **Vantagens:**
  - Equilíbrio entre simplicidade e flexibilidade.
  - Evita redundância de dados para atributos comuns.
- **Desvantagens:**
  - Pode exigir consultas mais complexas em alguns casos.
  - Dificuldade em visualizar a hierarquia completa sem ferramentas específicas.

**C** - SQL Server Management Studio (SSMS), ferramenta de gerenciamento de banco de dados da Microsoft, oferece diversos recursos que aprimoram a produtividade nas tarefas relacionadas ao banco de dados, desde a criação e o desenvolvimento até a administração e a manutenção. Entre os principais benefícios do SSMS, podemos destacar:

- Interface gráfica intuitiva
- Gerenciamento centralizado
- Automação de tarefas
- Visualização de dados
- Depuração e otimização de consultas
- Monitoramento de desempenho
- Segurança aprimorada
- Suporte para diferentes versões
- Integração com outras ferramentas
- Atualizações frequentes

Em resumo, o SQL Server Management Studio é uma ferramenta poderosa e versátil que auxilia os profissionais de banco de dados a aumentar sua produtividade e eficiência em diversas tarefas relacionadas ao gerenciamento de banco de dados. Com sua interface amigável, recursos abrangentes e diversas funcionalidades, o SSMS se torna um aliado essencial para administradores de banco de dados, desenvolvedores e analistas de dados

## 1º Alimentando a Base:

## 2º Objetivo da Prática:

- identificar os requisitos do sistema transformando em um modelo adequado.
- ferramentas de modelagem pra bases de dados relacionais.
- criar uma sintaxe SQL na criação de estruturas do banco (DDL).
- explorar a sintaxe SQL na consulta e manipulação de dados (DML)

## 3º Códigos solicitados:

```
INSERT INTO Pessoa(idPessoa,nome,endereco,cidade,estado,telefone,email)
VALUES (NEXT VALUE FOR orderPessoa, 'João Gilberto', 'Avenida Conde da Boa Vista,
1044', 'Recife', 'PE', '3304-2431', 'gigi@hotmail.com'),
(NEXT VALUE FOR orderPessoa, 'Bruno Cavalcante', 'Professor Jose Amarino dos Reis,
289', 'Recife', 'PE', '3389-3000', 'calvo@ig.com.br'),
(NEXT VALUE FOR orderPessoa, 'Ridalso', 'Tamboara,
17', 'Recife', 'PE', '3400-3400', 'risadinha@yahoo.com'),
(NEXT VALUE FOR orderPessoa, 'Panificadora Andrea', 'Beco do Pavão,
89', 'Recife', 'PE', '3000-3232', 'andrea@panificadora.com.br'),
(NEXT VALUE FOR orderPessoa, 'BiBike', 'Avenida Norte,
3409', 'Recife', 'PE', '3100-9999', 'bibike@bibiki.com.br');
```

```
INSERT INTO PessoaFisica(FK_Pessoa_idPessoa,cpf)
VALUES (1, '06772345477'),
(2, '01114549834'),
(3, '00879462300');
```

```
INSERT INTO PessoaJuridica(FK_Pessoa_idPessoa,cnpj)
VALUES (4, '05311244000109'),
(5, '07455190000111');
```

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

```
INSERT INTO Produto(nome,quantidade,precoVenda)
VALUES ('Parafuso', 300, '0.30'),
('Prego', 600, '0.20'),
('Cola', 900, '1.70');
```

```
INSERT INTO Movimento(FK_Usuario_idUsuario,FK_Pessoa_idPessoa,FK_Produto_idProduto,quantidade,tipo,precoUnitario)
VALUES (1,1,1,10, 'E', 1.00),
(2,2,2,20, 'S', 2.00),
(1,3,3,30, 'E', 3.00);
```

### 3.1º Códigos solicitados:

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

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

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

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

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

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

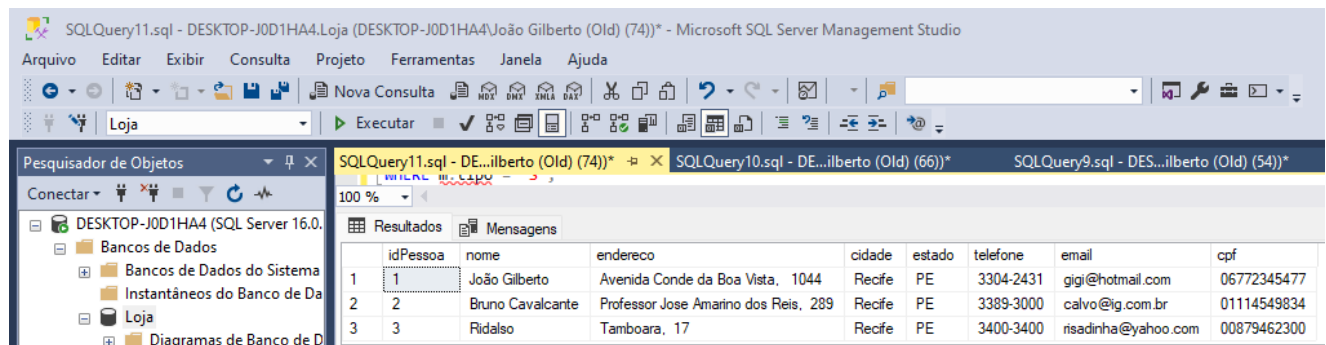
-- item (g)
SELECT u.*
FROM Usuario u
LEFT JOIN Movimento m ON u.idUsuario = m.FK_Usuario_idUsuario AND m.tipo = 'E'
WHERE m.idMovimento IS NULL;

-- item (h)
SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as compras
FROM Movimento m
INNER JOIN Usuario u ON m.FK_Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.loginName;

-- item (i)
SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as vendas
FROM Movimento m
INNER JOIN Usuario u ON m.FK_Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.loginName;

-- item (j)
SELECT pr.nome, SUM(m.precoUnitario * m.quantidade) / SUM(m.quantidade) as media
FROM Movimento m
INNER JOIN Produto pr ON m.FK_Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S'
GROUP BY pr.nome;
```

### Imagem 01 - Dados completos das pessoas físicas.



SQLQuery11.sql - DESKTOP-J0D1HA4.Loja (DESKTOP-J0D1HA4João Gilberto (Old) (74))\* - Microsoft SQL Server Management Studio

Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda

Loja Executar

Pesquisador de Objetos

Conectar

DESKTOP-J0D1HA4 (SQL Server 16.0.)

Bancos de Dados

Bancos de Dados do Sistema

Instantâneos do Banco de Dados

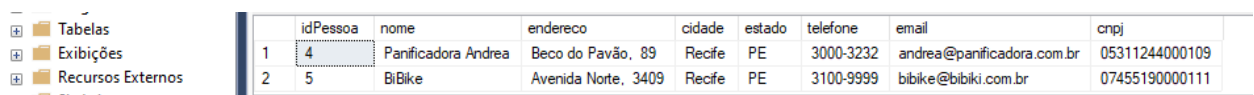
Loja

Diagramas de Banco de Dados

Resultados

	idPessoa	nome	endereco	cidade	estado	telefone	email	cpf
1	1	João Gilberto	Avenida Conde da Boa Vista, 1044	Recife	PE	3304-2431	gigi@hotmail.com	06772345477
2	2	Bruno Cavalcante	Professor Jose Amarino dos Reis, 289	Recife	PE	3389-3000	calvo@ig.com.br	01114549834
3	3	Ridalso	Tamboara, 17	Recife	PE	3400-3400	risadilha@yahoo.com	00879462300

### Imagem 02 - Dados completos de pessoas jurídicas.



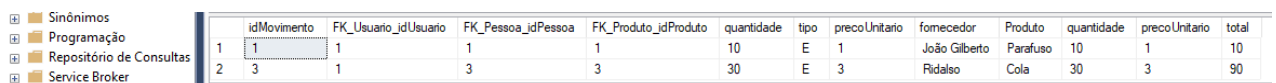
Tabelas

Exibições

Recursos Externos

	idPessoa	nome	endereco	cidade	estado	telefone	email	cnpj
1	4	Panificadora Andrea	Beco do Pavão, 89	Recife	PE	3000-3232	andrea@panificadora.com.br	05311244000109
2	5	BiBike	Avenida Norte, 3409	Recife	PE	3100-9999	bibike@bibiki.com.br	07455190000111

### Imagem 03 - Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.



Sinônimos

Programação

Repositório de Consultas

Service Broker

	idMovimento	FK_Usuario_idUsuario	FK_Pessoa_idPessoa	FK_Produto_idProduto	quantidade	tipo	precoUnitario	fornecedor	Produto	quantidade	precoUnitario	total
1	1	1	1	1	10	E	1	João Gilberto	Parafuso	10	1	10
2	3	1	3	3	30	E	3	Ridalso	Cola	30	3	90

### Imagem 04 - Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total.



Armazenamento

Segurança

Segurança

	idMovimento	FK_Usuario_idUsuario	FK_Pessoa_idPessoa	FK_Produto_idProduto	quantidade	tipo	precoUnitario	comprador	Produto	quantidade	precoUnitario	total
1	2	2	2	2	20	S	2	Bruno Cavalcante	Prego	20	2	40

### Imagem 05 - Valor total das entradas agrupadas por produto.



Programação

Repositório de Consultas

Service Broker

Armazenamento

	nome	compras
1	Cola	90
2	Parafuso	10

### Imagem 06 - Valor total das saídas agrupadas por produto.



Segurança

Segurança

Objetos de Servidor

	nome	vendas
1	Prego	40

### Imagem 07 - Operadores que não efetuaram movimentações de entrada (compra).



Objetos de Servidor

Replicação

Gerenciamento

	idUsuario	loginName	senha
1	2	op2	op2

### Imagem 08 - Valor total de entrada, agrupado por operador.



XEvent Profiler

	loginName	compras
1	op1	100

### Imagem 09 - Valor total de saída, agrupado por operador.



	loginName	vendas
1	op2	40



## 4º Resultados da execução dos códigos:

SQLQuery9 - 3\*Script...Gilberto (Old) (74)) - Microsoft SQL Server Management Studio

Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda

Loja Executar

SQLQuery9 - 3\*Script...Gilberto (Old) (74)) SQLQuery9 - 2\*Script...Gilberto (Old) (66)) SQLQuery9 - 1\*Script...Gilberto (Old) (54))

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

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

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

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

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

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

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

-- item (h)
SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as compras
FROM Movimento m
INNER JOIN Usuario u ON m.FK_Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.loginName;

-- item (i)
SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as vendas
FROM Movimento m
INNER JOIN Usuario u ON m.FK_Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.loginName;
```

100 %

Resultados Mensagens

idPessoa	nome	endereco	cidade	estado	telefone	email	cpf
1	João Gilberto	Avenida Conde da Boa Vista, 1044	Recife	PE	3304-2431	gigi@hotmail.com	06772345477
2	Bruno Cavalcante	Professor Jose Amarino dos Reis, 289	Recife	PE	3389-3000	calvo@ig.com.br	01114549834
3	Ridaleo	Tamboara, 17	Recife	PE	3400-3400	rsadinha@yahoo.com	00873462300

idPessoa	nome	endereco	cidade	estado	telefone	email	cnpj	
1	4	Panificadora Andrea	Beco do Pavão, 89	Recife	PE	3000-3232	andrea@panificadora.com.br	05311244000109
2	5	BiBike	Avenida Norte, 3409	Recife	PE	3100-9999	bibike@bibiki.com.br	07455190000111

idMovimento	FK_Usuario_idUsuario	FK_Pessoa_idPessoa	FK_Produto_idProduto	quantidade	tipo	precoUnitario	fornecedor	Produto	quantidade	precoUnitario	total
1	1	1	1	10	E	1	João Gilberto	Parafuso	10	1	10
2	3	1	3	30	E	3	Ridaleo	Cola	30	3	90

idMovimento	FK_Usuario_idUsuario	FK_Pessoa_idPessoa	FK_Produto_idProduto	quantidade	tipo	precoUnitario	comprador	Produto	quantidade	precoUnitario	total
1	2	2	2	20	S	2	Bruno Cavalcante	Prego	20	2	40

nome	compras
Cola	90
Parafuso	10

nome	vendas
Prego	40

idUsuario	loginName	senha	
1	2	op2	op2

loginName	compras
op1	100

loginName	vendas
op2	40

nome	media
Prego	2.000000

Consulta executada com êxito.

DESKTOP-J0D1HA4 (16.0 RTM) | DESKTOP-J0D1HA4\João G... | Loja | 00:00:00 | 15 linhas

Pronto Li 66 Col 18 Car 18 INS

## 5º Análise e Conclusão:

### A - Quais as diferenças no uso de sequence e identity?

- **Identity:** É uma propriedade de coluna, o que significa que está vinculada a uma coluna específica em uma tabela. Cada tabela só pode ter uma coluna identity.
- **Sequence:** É um objeto de banco de dados independente, não vinculado a nenhuma tabela específica. Uma sequence pode ser usada por várias colunas em diferentes tabelas

### B - Qual a importância das chaves estrangeiras para a consistência do banco?

As chaves estrangeiras é essencial para a construção de bancos de dados relacionais confiáveis e consistentes. Ela também garantem a integridade referencial dos dados, que por sua vez impedem anomalias, facilita a manipulação de informações, contribuem para a segurança e qualidade dos dados armazenados. Implementar chaves estrangeiras de forma adequada, os desenvolvedores e administradores de bancos de dados garantem a confiabilidade e precisão das informações, otimizando o desempenho e a confiabilidade do sistema.

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

- **Projeção (SELECT):** Seleciona colunas específicas de uma relação.
- **Seleção (WHERE):** Filtra tuplas de uma relação com base em uma condição booleana.
- **Produto cartesiano (FROM ... JOIN ...):** Combina duas relações em uma nova relação que contém todas as combinações possíveis de tuplas das relações originais.
- **União (UNION):** Combina duas relações com o mesmo esquema, unindo tuplas distintas de ambas as relações.
- **Intersecção (INTERSECT):** Retorna apenas as tuplas que estão presentes em ambas as relações.
- **Diferença (EXCEPT):** Retorna as tuplas que estão presentes na primeira relação, mas não na segunda.
- **Renomeação de colunas (AS):** Atribui novos nomes às colunas de uma relação.

### D - Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

#### Exemplo de Agrupamento:

Considere uma tabela `Pedidos` com colunas `id_pedido`, `cliente`, `data_pedido` e `valor_total`. Para analisar o valor total dos pedidos por cliente, podemos utilizar a seguinte consulta:

#### SQL

```
SELECT cliente, SUM(valor_total) AS total_por_cliente
FROM Pedidos
GROUP BY cliente;
```

#### Requisito Obrigatório:

O requisito fundamental para realizar o agrupamento em SQL é a presença de pelo menos uma coluna na cláusula `GROUP BY`. Essa coluna define como os dados serão divididos em grupos distintos, e as funções de agregação são aplicadas a cada grupo formado.

