



Relatório de Práticas

Identificação

- Nome do Campus: [POLO SÃO JOÃO]
- Nome do Curso: [DESENVOLVIMENTO FULL STACK]
- Nome da Disciplina: [VAMOS MANTER AS INFORMAÇÕES?]
- Número da Turma: [9001]
- Semestre Letivo: [2024.3]
- Integrantes da Prática: [MATHEUS MACÊDO SOUSA]

Título da Prática

Desenvolvimento de um Sistema de Gestão de Vendas

Objetivo da Prática

O objetivo desta prática é criar um banco de dados para a gestão de uma loja, incluindo tabelas para usuários, produtos, pessoas e movimentações de compra e venda, além de realizar consultas que permitem extrair informações relevantes sobre o sistema.

-- 1. Criando o Banco de Dados

```
IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = 'loja')
```

```
BEGIN
```

```
    CREATE DATABASE loja;
```

```
END
```

GO

USE loja;

GO

-- 2. Criando Tabelas

-- Tabela de usuários

IF NOT EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME
= 'usuarios')

BEGIN

CREATE TABLE usuarios (

id INT PRIMARY KEY IDENTITY(1,1),

nome VARCHAR(255) NOT NULL,

senha VARCHAR(255) NOT NULL,

tipo VARCHAR(20) CHECK (tipo = 'operador')

);

END

-- Tabela de pessoas

IF NOT EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME
= 'pessoas')

BEGIN

CREATE TABLE pessoas (

id INT PRIMARY KEY IDENTITY(1,1),

```

        tipo VARCHAR(20) CHECK (tipo IN ('fisica', 'juridica')),
        nome VARCHAR(255) NOT NULL,
        cpf VARCHAR(14) UNIQUE,
        cnpj VARCHAR(18) UNIQUE,
        endereco VARCHAR(255),
        telefone VARCHAR(15),
        email VARCHAR(255)
    );

END

-- Tabela de produtos

IF NOT EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME
= 'produtos')

BEGIN

    CREATE TABLE produtos (

        id INT PRIMARY KEY IDENTITY(1,1),
        nome VARCHAR(255) NOT NULL,
        quantidade INT NOT NULL CHECK (quantidade >= 0),
        preco_venda DECIMAL(10, 2) NOT NULL CHECK (preco_venda >= 0)

    );

END

-- Tabela de movimentos

IF NOT EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME
= 'movimentos')

```

BEGIN

```
CREATE TABLE movimentos (  
    id INT PRIMARY KEY IDENTITY(1,1),  
    tipo VARCHAR(20) CHECK (tipo IN ('compra', 'venda')),  
    id_usuario INT NOT NULL,  
    id_produto INT NOT NULL,  
    id_pessoa INT NOT NULL,  
    quantidade INT NOT NULL CHECK (quantidade > 0),  
    preco_unitario DECIMAL(10, 2) NOT NULL CHECK (preco_unitario >= 0),  
    data_movimento DATETIME NOT NULL DEFAULT GETDATE(),  
    FOREIGN KEY (id_usuario) REFERENCES usuarios(id),  
    FOREIGN KEY (id_produto) REFERENCES produtos(id),  
    FOREIGN KEY (id_pessoa) REFERENCES pessoas(id)  
);
```

END

-- 3. Inserindo Dados na Tabela de Usuários se não existirem

```
IF NOT EXISTS (SELECT * FROM usuarios WHERE nome = 'Operador 1')
```

BEGIN

```
INSERT INTO usuarios (nome, senha, tipo) VALUES  
('Operador 1', 'senha123', 'operador'),  
('Operador 2', 'senha456', 'operador');
```

END

-- 4. Inserindo Produtos se não existirem

IF NOT EXISTS (SELECT * FROM produtos WHERE nome = 'Produto A')

BEGIN

INSERT INTO produtos (nome, quantidade, preco_venda) VALUES

('Produto A', 100, 10.00),

('Produto B', 200, 20.00),

('Produto C', 150, 15.50);

END

-- 5. Criando Pessoas Físicas e Jurídicas se não existirem

IF NOT EXISTS (SELECT * FROM pessoas WHERE nome = 'João Silva')

BEGIN

INSERT INTO pessoas (tipo, nome, cpf, cnpj, endereco, telefone, email) VALUES

('fisica', 'João Silva', '123.456.789-00', NULL, 'Rua A, 123', '1234-5678',
'joao@example.com'),

('juridica', 'Empresa XYZ', NULL, '12.345.678/0001-95', 'Av. B, 456', '9876-5432',
'contato@xyz.com');

END

-- 6. Criando Movimentações se não existirem

IF NOT EXISTS (SELECT * FROM movimentos)

BEGIN

-- Movimentações de entrada (compras)

INSERT INTO movimentos (tipo, id_usuario, id_produto, id_pessoa, quantidade,
preco_unitario) VALUES

('compra', 1, 1, 2, 50, 10.00),

```
('compra', 1, 2, 2, 30, 20.00),
```

```
('compra', 2, 3, 1, 20, 15.50);
```

```
-- Movimentações de saída (vendas)
```

```
INSERT INTO movimentos (tipo, id_usuario, id_produto, id_pessoa, quantidade,  
preco_unitario) VALUES
```

```
('venda', 1, 1, 1, 20, 10.00),
```

```
('venda', 2, 2, 1, 10, 20.00),
```

```
('venda', 1, 3, 2, 5, 15.50);
```

```
END
```

```
-- 7. Consultas
```

```
-- a) Dados completos de pessoas físicas
```

```
SELECT
```

```
    p.id AS [ID],
```

```
    p.nome AS [Nome],
```

```
    p.cpf AS [CPF],
```

```
    p.endereco AS [Endereço],
```

```
    p.telefone AS [Telefone],
```

```
    p.email AS [Email]
```

```
FROM pessoas p
```

```
WHERE p.tipo = 'fisica';
```

```
-- b) Dados completos de pessoas jurídicas
```

SELECT

p.id AS [ID],

p.nome AS [Nome],

p.cnpj AS [CNPJ],

p.endereco AS [Endereço],

p.telefone AS [Telefone],

p.email AS [Email]

FROM pessoas p

WHERE p.tipo = 'juridica';

-- c) Movimentações de entrada

SELECT

m.data_movimento AS [Data],

m.tipo AS [Tipo],

p.nome AS [Fornecedor],

pr.nome AS [Produto],

m.quantidade AS [Quantidade],

m.preco_unitario AS [Preço Unitário],

(m.quantidade * m.preco_unitario) AS [Valor Total]

FROM movimentos m

JOIN produtos pr ON m.id_produto = pr.id

JOIN pessoas p ON m.id_pessoa = p.id

WHERE m.tipo = 'compra';

-- d) Movimentações de saída

```
SELECT
    m.data_movimento AS [Data],
    m.tipo AS [Tipo],
    p.nome AS [Comprador],
    pr.nome AS [Produto],
    m.quantidade AS [Quantidade],
    m.preco_unitario AS [Preço Unitário],
    (m.quantidade * m.preco_unitario) AS [Valor Total]
FROM movimentos m
JOIN produtos pr ON m.id_produto = pr.id
JOIN pessoas p ON m.id_pessoa = p.id
WHERE m.tipo = 'venda';
```

-- e) Valor total das entradas agrupadas por produto

```
SELECT
    pr.nome AS [Produto],
    SUM(m.quantidade * m.preco_unitario) AS [Total Entrada]
FROM movimentos m
JOIN produtos pr ON m.id_produto = pr.id
WHERE m.tipo = 'compra'
GROUP BY pr.nome;
```

-- f) Valor total das saídas agrupadas por produto


```
SELECT

    pr.nome AS [Produto],

    SUM(m.quantidade * m.preco_unitario) AS [Total Saída]

FROM movimentos m

JOIN produtos pr ON m.id_produto = pr.id

WHERE m.tipo = 'venda'

GROUP BY pr.nome;
```

-- g) Operadores que não efetuaram movimentações de entrada (compra)

```
SELECT

    u.nome AS [Operador]

FROM usuarios u

LEFT JOIN movimentos m ON u.id = m.id_usuario AND m.tipo = 'compra'

WHERE m.id IS NULL;
```

-- h) Valor total de entrada, agrupado por operador

```
SELECT

    u.nome AS [Operador],

    SUM(m.quantidade * m.preco_unitario) AS [Total Entrada]

FROM movimentos m

JOIN usuarios u ON m.id_usuario = u.id

WHERE m.tipo = 'compra'

GROUP BY u.nome;
```

-- i) Valor total de saída, agrupado por operador

```
SELECT
    u.nome AS [Operador],
    SUM(m.quantidade * m.preco_unitario) AS [Total Saída]
FROM movimentos m
JOIN usuarios u ON m.id_usuario = u.id
WHERE m.tipo = 'venda'
GROUP BY u.nome;
```

-- j) Valor médio de venda por produto (média ponderada)

```
SELECT
    pr.nome AS [Produto],
    SUM(m.quantidade * m.preco_unitario) / NULLIF(SUM(m.quantidade), 0) AS [Média Venda]
FROM movimentos m
JOIN produtos pr ON m.id_produto = pr.id
WHERE m.tipo = 'venda'
GROUP BY pr.nome;
```

SQLQuery17.sql -...KP5M9T\layma (52))

GROUP BY u.nome;

100 %

Resultados Mensagens

ID	Nome	CPF	Endereço	Telefone	Email
1	João Silva	123.456.789-00	Rua A, 123	1234-5678	joao@example.com

ID	Nome	CNPJ	Endereço	Telefone	Email
1	Empresa XYZ	12.345.678/0001-95	Av. B, 456	9876-5432	contato@xyz.com

Data	Tipo	Fornecedor	Produto	Quantidade	Preço Unitário	Valor Total
2024-10-09 19:28:18.487	compra	Empresa XYZ	Produto A	50	10.00	500.00
2024-10-09 19:28:18.487	compra	Empresa XYZ	Produto B	30	20.00	600.00

Data	Tipo	Comprador	Produto	Quantidade	Preço Unitário	Valor Total
2024-10-09 19:28:18.490	venda	João Silva	Produto A	20	10.00	200.00
2024-10-09 19:28:18.490	venda	João Silva	Produto B	10	20.00	200.00

Produto	Total Entrada
Produto A	500.00
Produto B	600.00

Produto	Total Saída
Produto A	200.00

Operador
Operador 2

Operador	Total Entrada
Operador 1	1100.00

Operador	Total Saída
Operador 1	200.00

Resultados da Execução dos Códigos

Após a execução dos códigos, as seguintes ações foram realizadas com sucesso:

- O banco de dados "loja" foi criado.
- As tabelas "usuarios", "pessoas", "produtos" e "movimentos" foram criadas, permitindo a gestão das informações da loja.
- Dados iniciais foram inseridos nas tabelas, incluindo operadores, produtos e pessoas.
- As movimentações de compras e vendas foram registradas corretamente.
- Consultas foram realizadas com êxito, retornando informações sobre pessoas físicas e jurídicas, movimentações e totais agrupados.

Análise e Conclusão

Diferenças no uso de sequence e identity

- **Sequence:** Um objeto de banco de dados que gera números sequenciais independentes, podendo ser utilizado em diversas tabelas.
- **Identity:** Um atributo de coluna que gera automaticamente um número único para cada nova linha inserida na tabela, facilitando o gerenciamento de chaves primárias.

Importância das Chaves Estrangeiras

As chaves estrangeiras são essenciais para garantir a integridade referencial no banco de dados. Elas asseguram que um valor em uma tabela exista em outra, evitando inconsistências e assegurando que os dados se relacionem corretamente.

Operadores do SQL na Álgebra Relacional e Cálculo Relacional

- **Álgebra Relacional:** Inclui operadores como SELECT, JOIN, UNION, e INTERSECT.
- **Cálculo Relacional:** Baseia-se em expressões lógicas, utilizando operadores como EXISTS e NOT EXISTS.

Agrupamento em Consultas

O agrupamento em consultas é realizado com a cláusula **GROUP BY**. É obrigatório utilizar uma função de agregação (como COUNT, SUM, AVG) para compilar os dados agrupados, permitindo análises mais detalhadas e sumarizadas.