



Estácio

UNIVERSIDADE ESTÁCIO DE SÁ

CAMPUS POLO CENTRO - MARICÁ - RJ

ALUNO: LUIZ CARLOS MARINHO JUNIOR

MATRÍCULA: 2023.11.17557-1

CURSO: DESENVOLVIMENTO FULL STACK

SEMESTRE LETIVO: 2024.4

DISCIPLINA: NÍVEL 2: VAMOS MANTER AS INFORMAÇÕES

TÍTULO: 2º PROCEDIMENTO | ALIMENTANDO A BASE

RIO DE JANEIRO

2024

LUIZ CARLOS MARINHO JUNIOR

2º PROCEDIMENTO | ALIMENTANDO A BASE

Trabalho prático para aprovação na disciplina de Nível 2: Vamos manter as informações.

Tutora: Prof. Maria B.

RIO DE JANEIRO

2024

SUMÁRIO

| | |
|---|----|
| 1. OBJETIVOS DA PRÁTICA..... | 4 |
| 2. CÓDIGOS SOLICITADOS | 4 |
| 2.1. APRESENTAÇÃO | 4 |
| 2.2. PRINCIPAIS INSTRUÇÕES DO 2ª PROCEDIMENTO | 4 |
| 3. ANÁLISE E CONCLUSÃO..... | 11 |
| 3.1. QUAIS AS DIFERENÇAS NO USO DE SEQUENCE E IDENTITY? | 11 |
| 3.2. QUAL A IMPORTÂNCIA DAS CHAVES ESTRANGERIAS PARA A CONSISTÊNCIA DO BANCO?..... | 11 |
| 3.3. QUAIS OPERADORES DO SQL PERTENCEM À ÁLGEBRA RELACIONAL E QUAIS SÃO DEFINIDOS NO CÁLCULO RELACIONAL?..... | 12 |
| 3.4. COMO É FEITO O AGRUPAMENTO EM CONSULTAS, E QUAL REQUISITO É OBRIGATÓRIO? | 12 |

1. OBJETIVOS DA PRÁTICA

1. Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
2. 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.

2. CÓDIGOS SOLICITADOS

2.1. APRESENTAÇÃO

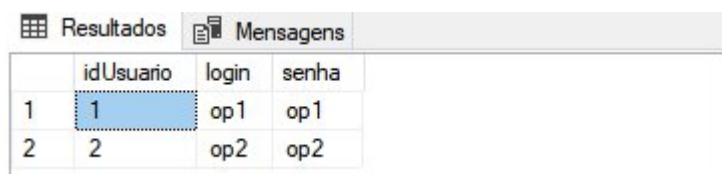
Todos os códigos que serão aqui apresentados, estão disponíveis de forma completa no repositório do Github.

O link para o repositório do Github se encontra logo abaixo:

<https://github.com/luizmarinhojr/loja-database>

2.2. PRINCIPAIS INSTRUÇÕES DO 2ª PROCEDIMENTO

1. Utilizar o SQL Server Management Studio para alimentar as tabelas com dados básicos do sistema:
 - a. Logar como usuário loja, senha loja.
 - b. Utilizar o editor de SQL para incluir dados na tabela de usuários, de forma a obter um conjunto como o apresentado a seguir:



The screenshot shows the 'Resultados' (Results) pane in SQL Server Management Studio. It displays a table with four columns: 'idUsuario', 'login', and 'senha'. The first row has values 1, op1, and op1. The second row has values 2, op2, and op2. The first row is highlighted with a blue background.

| | idUsuario | login | senha |
|---|-----------|-------|-------|
| 1 | 1 | op1 | op1 |
| 2 | 2 | op2 | op2 |

Observação! Usuário op1, com senha op1, e usuário op2, com senha op2, sem criptografia. Para sistemas reais seria necessário armazenar o hash da senha, codificado para Base64.

Código para inclusão dos dados na tabela de usuários:

```

-- INSERT INTO Usuarios (login, senha)
VALUES ('op1', 'op1');

-- INSERT INTO Usuarios (login, senha)
VALUES ('op2', 'op2');

```

- c. Inserir alguns produtos na base de dados, obtendo um conjunto como o que é apresentado a seguir:

| | idProduto | nome | quantidade | precoVenda |
|---|-----------|---------|------------|------------|
| 1 | 1 | Banana | 100 | 5.00 |
| 2 | 3 | Laranja | 500 | 2.00 |
| 3 | 4 | Manga | 800 | 4.00 |

Código para inclusão dos dados na tabela de Produtos:

```

-- INSERT INTO Produtos (nome, quantidade, precoVenda)
VALUES ('Banana', 100, 5.00);

-- INSERT INTO Produtos (nome, quantidade, precoVenda)
VALUES ('Laranja', 400, 2.00);

DELETE FROM Produtos WHERE nome = 'Laranja';

-- INSERT INTO Produtos (nome, quantidade, precoVenda)
VALUES ('Laranja', 500, 2.00);

-- INSERT INTO Produtos (nome, quantidade, precoVenda)
VALUES ('Manga', 800, 4.00);

```

Comando SQL para visualização dos dados da tabela de Produtos:

```

-- SELECT produtoID, nome, quantidade, FORMAT(precoVenda, 'N2') AS precoVenda
FROM Produtos;

```

Resultado da tabela após a inclusão dos dados:

| | 123 produtoID | A-Z nome | 123 quantidade | A-Z precoVenda |
|---|---------------|----------|----------------|----------------|
| 1 | 1 | Banana | 100 | 5.00 |
| 2 | 3 | Laranja | 500 | 2.00 |
| 3 | 4 | Manga | 800 | 4.00 |

2. Criar pessoas físicas e jurídicas na base de dados:

- Obter o próximo id de pessoa a partir da sequence;
- Incluir na tabela pessoa os dados comuns;
- Incluir em pessoa física o CPF, efetuando o relacionamento com pessoa.

Código para inclusão dos dados na tabela Pessoas e PessoasFísicas:

```
● INSERT INTO Pessoas (nome, logradouro, cidade, estado, telefone, email)
  VALUES ('João', 'Rua 12, casa 3, Quitanda', 'Riacho do Norte', 'PA', '1111-1111', 'joao@riacho.com');

● INSERT INTO PessoasFísicas (pessoaID, cpf)
  VALUES (7, '11111111111');
```

Comando SQL para visualização dos dados da tabela de Pessoas e PessoasFísicas:

```
● SELECT * FROM Pessoas p
  JOIN PessoasFísicas pf ON pf.pessoaID = p.pessoaID;
```

Resultado após a inclusão dos dados e visualização com o comando SQL acima:

| | 123 pessoaID | A-Z nome | A-Z logradouro | A-Z cidade | A-Z estado | A-Z telefone | A-Z email | 123 pessoaID | A-Z cpf |
|---|--------------|----------|--------------------------|-----------------|------------|--------------|-----------------|--------------|-------------|
| 1 | 7 | João | Rua 12, casa 3, Quitanda | Riacho do Norte | PA | 1111-1111 | joao@riacho.com | 7 | 11111111111 |

- Incluir em pessoa jurídica o CNPJ, relacionando com pessoa.

Código para inclusão dos dados na tabela Pessoas e PessoasFísicas:

```
● INSERT INTO Pessoas (nome, logradouro, cidade, estado, telefone, email)
  VALUES ('JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '1212-1212', 'jjc@riacho.com');

● INSERT INTO PessoasJuridicas (pessoaID, cnpj)
  VALUES (15, '222222222222');
```

Comando SQL para visualização dos dados da tabela de Pessoas e PessoasJuridicas:

```
● SELECT * FROM Pessoas p
  JOIN PessoasJuridicas pj ON pj.pessoaID = p.pessoaID;
```

Resultado após a inclusão dos dados e visualização com o comando SQL acima:

| | 123 pessoaID | A-Z nome | A-Z logradouro | A-Z cidade | A-Z estado | A-Z telefone | A-Z email | 123 pessoaID | A-Z cnpj |
|---|--------------|----------|----------------|-----------------|------------|--------------|----------------|--------------|--------------|
| 1 | 15 | JJC | Rua 11, Centro | Riacho do Norte | PA | 1212-1212 | jjc@riacho.com | 15 | 222222222222 |

3. Criar algumas movimentações na base de dados, obtendo um conjunto como o que é apresentado a seguir, onde E representa Entrada e S representa Saída.

Código para inclusão dos dados na tabela de Movimentos:

```
● INSERT INTO Movimentos (usuarioID, pessoaID, produtoID, quantidade, tipo, valorUnitario)
  VALUES (1, 7, 1, 20, 'S', 4.00);

● INSERT INTO Movimentos (usuarioID, pessoaID, produtoID, quantidade, tipo, valorUnitario)
  VALUES (1, 7, 3, 15, 'S', 2.00);

● INSERT INTO Movimentos (usuarioID, pessoaID, produtoID, quantidade, tipo, valorUnitario)
  VALUES (2, 7, 3, 10, 'S', 3.00);

● INSERT INTO Movimentos (usuarioID, pessoaID, produtoID, quantidade, tipo, valorUnitario)
  VALUES (1, 15, 3, 15, 'E', 5.00);

● INSERT INTO Movimentos (usuarioID, pessoaID, produtoID, quantidade, tipo, valorUnitario)
  VALUES (1, 15, 4, 20, 'E', 4.00);
```

Comando SQL para visualização dos dados da tabela de Movimentos:

```
● SELECT
  usuarioID,
  pessoaID,
  produtoID,
  quantidade,
  tipo,
  FORMAT(valorUnitario, 'N2')
FROM Movimentos;
```

Resultado após a inclusão dos dados e visualização com o comando SQL acima:

| | 123 usuarioID | 123 pessoaID | 123 produtoID | 123 quantidade | A-Z tipo | A-Z |
|---|---------------|--------------|---------------|----------------|----------|------|
| 1 | 1 | 7 | 1 | 20 | S | 4.00 |
| 2 | 1 | 7 | 3 | 15 | S | 2.00 |
| 3 | 2 | 7 | 3 | 10 | S | 3.00 |
| 4 | 1 | 15 | 3 | 15 | E | 5.00 |
| 5 | 1 | 15 | 4 | 20 | E | 4.00 |

1. Efetuar as seguintes consultas sobre os dados inseridos:
 - a. Dados completos de pessoas físicas.

```
● -- a. Dados completos de pessoas físicas:
  SELECT * FROM Pessoas p
  JOIN PessoasFisicas pf ON p.pessoaID = pf.pessoaID;
```

- b. Dados completos de pessoas jurídicas.

```
-- b. Dados completos de pessoas juridicas:
SELECT * FROM Pessoas p
JOIN PessoasJuridicas pj ON p.pessoaID = pj.pessoaID;
```

- c. Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.

```
-- c. Movimentacoes de entrada, com produto, fornecedor, quantidade, preco unitario e valor total:
SELECT
    m.movimentoID,
    pr.nome AS produto,
    p.nome AS fornecedor,
    m.quantidade,
    FORMAT(m.valorUnitario, 'N2') AS precoUnitario,
    FORMAT((m.quantidade * m.valorUnitario), 'N2') AS valorTotal
FROM
    Movimentos m
JOIN
    Produtos pr ON pr.produtoID = m.produtoID
JOIN
    Pessoas p ON p.pessoaID = m.pessoaID
WHERE
    m.tipo = 'E';
```

- d. Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total.

```
-- d. Movimentacoes de saida, com produto, comprador, quantidade, preco unitario e valor total:
SELECT
    m.movimentoID,
    pr.nome AS produto,
    p.nome AS comprador,
    m.quantidade,
    FORMAT(m.valorUnitario, 'N2') AS precoUnitario,
    FORMAT((m.quantidade * m.valorUnitario), 'N2') AS valorTotal
FROM
    Movimentos m
JOIN
    Produtos pr ON pr.produtoID = m.produtoID
JOIN
    Pessoas p ON p.pessoaID = m.pessoaID
WHERE
    m.tipo = 'S';
```


- e. Valor total das entradas agrupadas por produto.

```
-- e. Valor total das entradas agrupadas por produto:
SELECT
    pr.nome AS produto,
    FORMAT(SUM(m.quantidade * m.valorUnitario), 'N2') AS valorTotalEntrada
FROM
    Movimentos m
JOIN
    Produtos pr ON pr.produtoID = m.produtoID
WHERE
    m.tipo = 'E'
GROUP BY
    pr.nome;
```

- f. Valor total das saídas agrupadas por produto.

```
-- f. Valor total das saidas agrupadas por produto:
SELECT
    pr.nome AS produto,
    FORMAT(SUM(m.quantidade * m.valorUnitario), 'N2') AS valorTotalSaida
FROM
    Movimentos m
JOIN
    Produtos pr ON pr.produtoID = m.produtoID
WHERE
    m.tipo = 'S'
GROUP BY
    pr.nome;
```

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

```
-- g. Operadores que nao efetuaram movimentacoes de entrada (compra):
SELECT
    u.usuarioID,
    u.login AS operador
FROM
    Usuarios u
LEFT JOIN
    Movimentos m ON m.usuarioID = u.usuarioID AND m.tipo = 'E'
WHERE
    m.movimentoID IS NULL;
```

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

```
-- h. Valor total de entrada, agrupado por operador:
SELECT
    u.login AS operador,
    FORMAT(SUM(m.quantidade * m.valorUnitario), 'N2') AS valorTotalEntrada
FROM
    Usuarios u
JOIN
    Movimentos m ON m.usuarioID = u.usuarioID
WHERE
    m.tipo = 'E'
GROUP BY
    u.login;
```

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

```
-- i. Valor total de saída, agrupado por operador:
SELECT
    u.login AS operador,
    FORMAT(SUM(m.quantidade * m.valorUnitario), 'N2') AS valorTotalSaida
FROM
    Usuarios u
JOIN
    Movimentos m ON m.usuarioID = u.usuarioID
WHERE
    m.tipo = 'S'
GROUP BY
    u.login;
```

- j. Valor médio de venda por produto, utilizando média ponderada.

```
-- j. Valor médio de venda por produto, utilizando media ponderada:
SELECT
    pr.nome AS produto,
    SUM(m.quantidade * m.valorUnitario) / SUM(m.quantidade) AS valorMedioVenda
FROM
    Movimentos AS m
JOIN
    Produtos AS pr ON m.produtoID = pr.produtoID
WHERE
    m.tipo = 'S' -- 'S' para saídas
GROUP BY
    pr.nome;
```

3. ANÁLISE E CONCLUSÃO

3.1. QUAIS AS DIFERENÇAS NO USO DE SEQUENCE E IDENTITY?

Um sequence é um objeto independente no banco de dados que gera números sequenciais, dessa forma, ele não está atrelado a especificamente uma tabela ou coluna. O mesmo pode ser utilizado em várias partes do banco de dados, de forma semelhante a uma função em uma linguagem de programação. Já o identity fica obrigatoriamente atrelado a uma coluna e gera números automaticamente quando novas linhas são inseridas. Além dessas diferenças básicas, o sequence é mais flexível, configurável e controlável. O uso de identity é ótimo para simplicidade e valores automáticos associados diretamente a tabelas. Já o uso de sequence é ótimo para maior flexibilidade, reusabilidade e controle em sistemas complexos.

3.2. QUAL A IMPORTÂNCIA DAS CHAVES ESTRANGERIAS PARA A CONSISTÊNCIA DO BANCO?

As chaves estrangeiras ou foreign keys, desempenham um papel crucial para garantir a consistência e a integridade referencial em um banco de dados relacional. Uma chave estrangeira estabelece um vínculo entre uma coluna em uma tabela (a tabela "filha") e a chave primária de outra tabela (a tabela "pai"). Isso assegura que os valores na tabela filha correspondam a valores válidos na tabela pai. Dessa forma, o banco de dados consegue implementar seus relacionamentos e garantir a consistência das informações armazenadas no banco.

3.3. QUAIS OPERADORES DO SQL PERTENCEM À ÁLGEBRA RELACIONAL E QUAIS SÃO DEFINIDOS NO CÁLCULO RELACIONAL?

Operadores Fundamentais da Álgebra Relacional no SQL:

Seleção; Projeção; União; Interseção; Diferença; Produto Cartesiano; Junção; Renomeação; Divisão; e outros componente adicionais, como Atribuição, e mais...

O Cálculo Relacional descreve o que se deseja consultar sem especificar como obtê-lo. Ele é dividido em dois tipos principais: Cálculo Relacional de Tuplas (TRC) e Cálculo Relacional de Domínio (DRC). SQL adota características de ambos. A álgebra relacional tem o foco de ditar como os dados devem ser manipulados e o cálculo relacional tem o foco de ditar o que se deseja consultar. A álgebra relacional possui operadores como JOIN, UNION, SELECT. E o cálculo relacional possui operadores como EXISTS, IN, etc.

3.4. COMO É FEITO O AGRUPAMENTO EM CONSULTAS, E QUAL REQUISITO É OBRIGATÓRIO?

O agrupamento em consultas SQL é realizado utilizando a cláusula GROUP BY, que organiza os resultados em grupos com base em uma ou mais colunas. Após o agrupamento, é possível aplicar funções de agregação, como SUM, COUNT, AVG, MIN, e MAX, para realizar cálculos em cada grupo. Ao usar o GROUP BY, é obrigatório que todas as colunas presentes no SELECT que não são argumentos de funções de agregação sejam incluídas na cláusula GROUP BY. Isso é necessário para que o agrupamento seja consistente e os dados sejam agregados corretamente.

Exemplo de sintaxe SQL:

```
SELECT coluna1, coluna2, FUNCAO_AGREGACAO(coluna3) FROM tabela  
GROUP BY coluna1, coluna2;
```

REFERÊNCIAS

MICROSOFT, SQL Server technical documentation. Disponível em:
<<https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16>>. Acesso em 03 de janeiro de 2025.

W3 Schools, SQL GROUP BY Statement. Disponível em:
<https://www.w3schools.com/sql/sql_groupby.asp>. Acesso em 04 de janeiro de 2025.

ALURA, Conhecendo a Álgebra Relacional para consulta de dados relacionais. Disponível em: <<https://www.alura.com.br/artigos/algebra-relacional#:~:text=A%20%C3%81lgebra%20Relacional%20%C3%A9%20a,extra%C3%A7%C3%B5es%20valiosas%20dos%20dados.>>. Acesso em 04 de janeiro de 2025.