DESENVOLVIMENTO FULL STACK
POLO SETOR O – CEILÂNDIA - DF
PERÍODO 2024.2 FLEX
DISCIPLINA: VAMOS MANTER AS INFORMAÇÕES
MAYCON MOURA

Título da Prática

Alimentando a Base SQL e realização de consultas

Objetivo da Prática

O objetivo desta prática é demonstrar as habilidades básicas para a modelagem de um banco de dados relacional utilizando o SQL Server, além de utilizar a sintaxe SQL para a criação das estruturas necessárias e a realização de consultas. O projeto inclui a criação de tabelas para usuários, pessoas físicas e jurídicas, produtos, e movimentações de compra e venda, bem como a inserção e consulta de dados.

Códigos

```
-- Dados completos de pessoas físicas
        SELECT
           p.id_pessoa,
           p.nome,
           p.tipo,
           p.endereco,
           p.telefone,
           p.email,
           pf.cpf
        FROM
           Pessoa p
        JOIN
           PessoaFisica pf
        ON
           p.id_pessoa = pf.id_pessoa;
```

```sql
-- Dados completos de pessoas jurídicas
	SELECT
	   p.id_pessoa,
	   p.nome,
	   p.tipo,
	   p.endereco,
	   p.telefone,
	   p.email,
	   pj.cnpj
	FROM
	   Pessoa p
	JOIN
	   PessoaJuridica pj
	ON
	   p.id_pessoa = pj.id_pessoa;

-- Movimentações de entrada
	SELECT
	   m.id_compra,
	   p.nome AS fornecedor,
	   pr.nome AS produto,
	   m.quantidade,
	   m.preco_unitario,
	   m.quantidade * m.preco_unitario AS valor_total,
	   m.data_movimento
	FROM
	   Movimento m
	JOIN
	   Pessoa p ON m.id_pessoa = p.id_pessoa
	JOIN
	   Produto pr ON m.id_produto = pr.id_produto
	WHERE
	   m.tipo = 'E';

-- Movimentações de saída
	SELECT
	   m.id_compra,
	   p.nome AS comprador,
	   pr.nome AS produto,
	   m.quantidade,
	   m.preco_unitario,
	   m.quantidade * m.preco_unitario AS valor_total,
	   m.data_movimento
	FROM
	   Movimento m
	JOIN
	   Pessoa p ON m.id_pessoa = p.id_pessoa
```

```
    JOIN
        Produto pr ON m.id_produto = pr.id_produto
    WHERE
        m.tipo = 'S';

-- Valor total das entradas agrupadas por produto
    SELECT
        pr.nome AS produto,
        SUM(m.quantidade * m.preco_unitario) AS valor_total_entradas
    FROM
        Movimento m
    JOIN
        Produto pr ON m.id_produto = pr.id_produto
    WHERE
        m.tipo = 'E'
    GROUP BY
        pr.nome;

-- Valor total das saídas agrupadas por produto
    SELECT
        pr.nome AS produto,
        SUM(m.quantidade * m.preco_unitario) AS valor_total_saidas
    FROM
        Movimento m
    JOIN
        Produto pr ON m.id_produto = pr.id_produto
    WHERE
        m.tipo = 'S'
    GROUP BY
        pr.nome;

-- Operadores que não efetuaram movimentações de entrada
    SELECT
        u.id_usuario,
        u.nome,
        u.email
    FROM
        Usuario u
    WHERE
        u.id_usuario NOT IN (
            SELECT DISTINCT
                m.id_usuario
            FROM
                Movimento m
            WHERE
                m.tipo = 'E'
        );
```

```
-- Valor total de entrada, agrupado por operador
	SELECT
	   u.nome AS operador,
	   SUM(m.quantidade * m.preco_unitario) AS valor_total_entradas
	FROM
	   Movimento m
	JOIN
	   Usuario u ON m.id_usuario = u.id_usuario
	WHERE
	   m.tipo = 'E'
	GROUP BY
	   u.nome;

-- Valor total de saída, agrupado por operador
	SELECT
	   u.nome AS operador,
	   SUM(m.quantidade * m.preco_unitario) AS valor_total_saidas
	FROM
	   Movimento m
	JOIN
	   Usuario u ON m.id_usuario = u.id_usuario
	WHERE
	   m.tipo = 'S'
	GROUP BY
	   u.nome;

-- Valor médio de venda por produto, utilizando média ponderada
	SELECT
	   pr.nome AS produto,
	   SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS
	media_ponderada_vendas
	FROM
	   Movimento m
	JOIN
	   Produto pr ON m.id_produto = pr.id_produto
	WHERE
	   m.tipo = 'S'
	GROUP BY
	   pr.nome;
```

Resultados

Dados completos de pessoas físicas.

Resultados | Mensagens

	id_pessoa	nome	tipo	endereco	telefone	email	cpf
1	1	João	F	Rua A, 123	1234-5678	joao@mail.com	12345678901

Dados completos de pessoas jurídicas.

Resultados | Mensagens

	id_pessoa	nome	tipo	endereco	telefone	email	cnpj
1	3	JJC	J	Avenida B, 456	9876-5432	contato@empresa.com	12345678000199

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

Resultados | Mensagens

	id_compra	fornecedor	produto	quantidade	preco_unitario	valor_total	data_movimento
1	7	JJC	Manga	15	5.00	75.00	2024-07-23 00:00:00.000
2	9	JJC	Pera	20	4.00	80.00	2024-07-23 00:00:00.000

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

Resultados | Mensagens

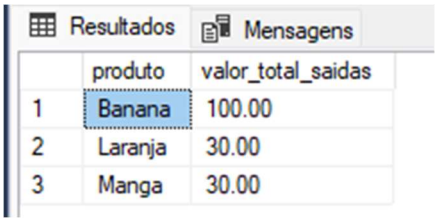
	id_compra	comprador	produto	quantidade	preco_unitario	valor_total	data_movimento
1	4	João	Banana	20	5.00	100.00	2024-07-23 00:00:00.000
2	5	João	Laranja	15	2.00	30.00	2024-07-23 00:00:00.000
3	6	João	Manga	10	3.00	30.00	2024-07-23 00:00:00.000

Valor total das entradas agrupadas por produto.

Resultados | Mensagens

	produto	valor_total_entradas
1	Manga	75.00
2	Pera	80.00

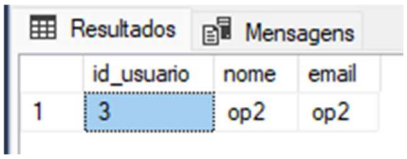
Valor total das saídas agrupadas por produto.



Resultados | Mensagens

	produto	valor_total_saidas
1	Banana	100.00
2	Laranja	30.00
3	Manga	30.00

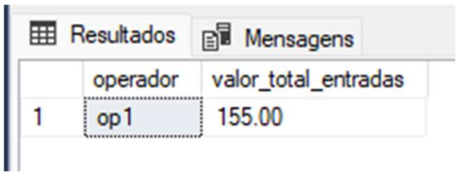
Operadores que não efetuaram movimentações de entrada (compra).



Resultados | Mensagens

	id_usuario	nome	email
1	3	op2	op2

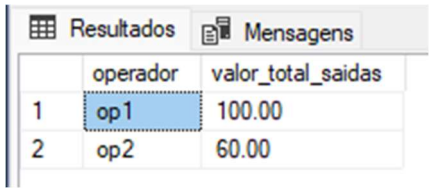
Valor total de entrada, agrupado por operador.



Resultados | Mensagens

	operador	valor_total_entradas
1	op1	155.00

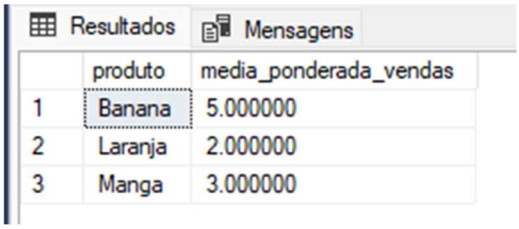
Valor total de saída, agrupado por operador.



Resultados | Mensagens

	operador	valor_total_saidas
1	op1	100.00
2	op2	60.00

Valor médio de venda por produto, utilizando média ponderada.



Resultados | Mensagens

	produto	media_ponderada_vendas
1	Banana	5.000000
2	Laranja	2.000000
3	Manga	3.000000

Análise e Conclusão

Quais as diferenças no uso de sequence e identity?

- **Identity**:
  - **Definição**: O IDENTITY é uma propriedade usada para colunas em tabelas do SQL Server que auto-incrementam seus valores automaticamente sempre que uma nova linha é inserida.
  - **Uso**: É especificada diretamente na definição da coluna da tabela.
  - **Exemplo**: id_usuario INT IDENTITY(1,1) PRIMARY KEY.

    - **Limitação**: Está limitada à tabela onde é definida e não pode ser compartilhada entre diferentes tabelas.
- **Sequence**:
    - **Definição**: Uma SEQUENCE é um objeto de banco de dados independente que gera números em uma sequência definida. Pode ser usado para gerar valores únicos em várias tabelas.
    - **Uso**: É criada separadamente e referenciada nas colunas das tabelas.
    - **Exemplo**: CREATE SEQUENCE s_pessoa_id START WITH 1 INCREMENT BY 1;.
    - **Flexibilidade**: Pode ser compartilhada entre múltiplas tabelas e pode ser reutilizada em diferentes contextos.

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

- **Consistência Referencial**: As chaves estrangeiras garantem que os dados em uma tabela (tabela filha) correspondam a dados válidos em outra tabela (tabela pai). Isso impede a inserção de valores inválidos ou a deleção de registros que ainda são referenciados.
- **Integridade dos Dados**: Mantém a integridade dos dados ao assegurar que relações entre tabelas sejam válidas.
- **Operações de Cascata**: Permitem a definição de ações em cascata (por exemplo, ON DELETE CASCADE) que automaticamente atualizam ou deletam registros dependentes, simplificando a manutenção do banco de dados.

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

- **Álgebra Relacional**:
    - **Select**: SELECT operador básico para recuperar dados de uma tabela.
    - **Project**: SELECT column1, column2 seleciona colunas específicas.
    - **Join**: INNER JOIN, LEFT JOIN, RIGHT JOIN combinam registros de duas tabelas baseadas em uma condição de correspondência.
    - **Union**: UNION combina o resultado de duas consultas.
    - **Difference**: EXCEPT retorna registros de uma consulta que não aparecem na outra.
    - **Rename**: AS renomeia colunas ou tabelas.
- **Cálculo Relacional**:
    - **Existential Quantifier**: EXISTS verifica se a subconsulta retorna algum registro.
    - **Universal Quantifier**: NOT EXISTS verifica se a subconsulta não retorna registros.
    - **Logical Connectives**: AND, OR, NOT usados em condições de filtro.

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

- **Agrupamento**:
    - **Uso do GROUP BY**: O agrupamento em consultas SQL é feito utilizando a cláusula GROUP BY, que agrupa linhas que possuem valores iguais em colunas especificadas.

    - **Exemplo**: SELECT coluna1, SUM(coluna2) FROM tabela GROUP BY coluna1;.
- **Requisito Obrigatório**:
    - **Colunas Selecionadas**: Todas as colunas listadas no SELECT que não são usadas em funções de agregação (como SUM, COUNT, AVG) devem ser incluídas na cláusula GROUP BY.
    - **Exemplo**: Se coluna1 está no SELECT e não é uma função de agregação, deve estar no GROUP BY.

Análise e Conclusão

- **Diferenças no uso de sequence e identity**: IDENTITY é usado diretamente na tabela e é limitado a uma tabela, enquanto SEQUENCE é um objeto separado que pode ser usado em múltiplas tabelas, proporcionando maior flexibilidade.
- **Importância das chaves estrangeiras**: As chaves estrangeiras garantem a integridade referencial e a consistência dos dados, evitando referências inválidas e permitindo ações em cascata para manter a consistência.
- **Operadores de álgebra relacional e cálculo relacional**: A álgebra relacional inclui operadores como SELECT, JOIN, UNION, etc., enquanto o cálculo relacional usa quantificadores como EXISTS e NOT EXISTS, além de conectivos lógicos.
- **Agrupamento em consultas**: O agrupamento é feito usando GROUP BY e todas as colunas no SELECT que não são agregadas devem estar incluídas no GROUP BY.