Estácio - Mundo 3 - Missão Nível 2

Faculdade Estácio - Polo Centro – Belo Horizonte – MG

Curso: Desenvolvimento Full Stack.

Disciplina: Nível 2: Vamos Manter as Informações !

RPG0015.

Semestre Letivo: 3

Integrante: André Luiz Ferreira da Silva

Repositório: https://github.com/Andre-Luiz22/m3-n2

IDE: Sql Server Management Studio.

Título da Prática

Vamos manter as informações!

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 realcionais.
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 de sintaxe SQL, na plataforma SQL Server.

Arquivos do Projeto

```
use loja
INSERT INTO Usuario (login, senha)
VALUES
  ('op1', 'op1'),
  ('op2', 'op2'),
  ('op3', 'op3'),
  ('op4', 'op4');

SELECT * FROM Usuario;

INSERT INTO Produto (nome, quantidade, precoVenda)
VALUES
  ('banana', 100, 5.0),
  ('laranja', 500, 2.0),
  ('Manga', 800, 4.0),
  ('Pera', 300, 6.0),
  ('Abacaxi', 300, 3.0);

SELECT * FROM Produto;

INSERT INTO Pessoa (nome, rua, cidade, estado, telefone, email)
VALUES
  ('Joao', 'Rua 12, casa3, Quitanda', 'Riacho do Sul', 'PA', '11111111', 'joao@riacho.com'),
```

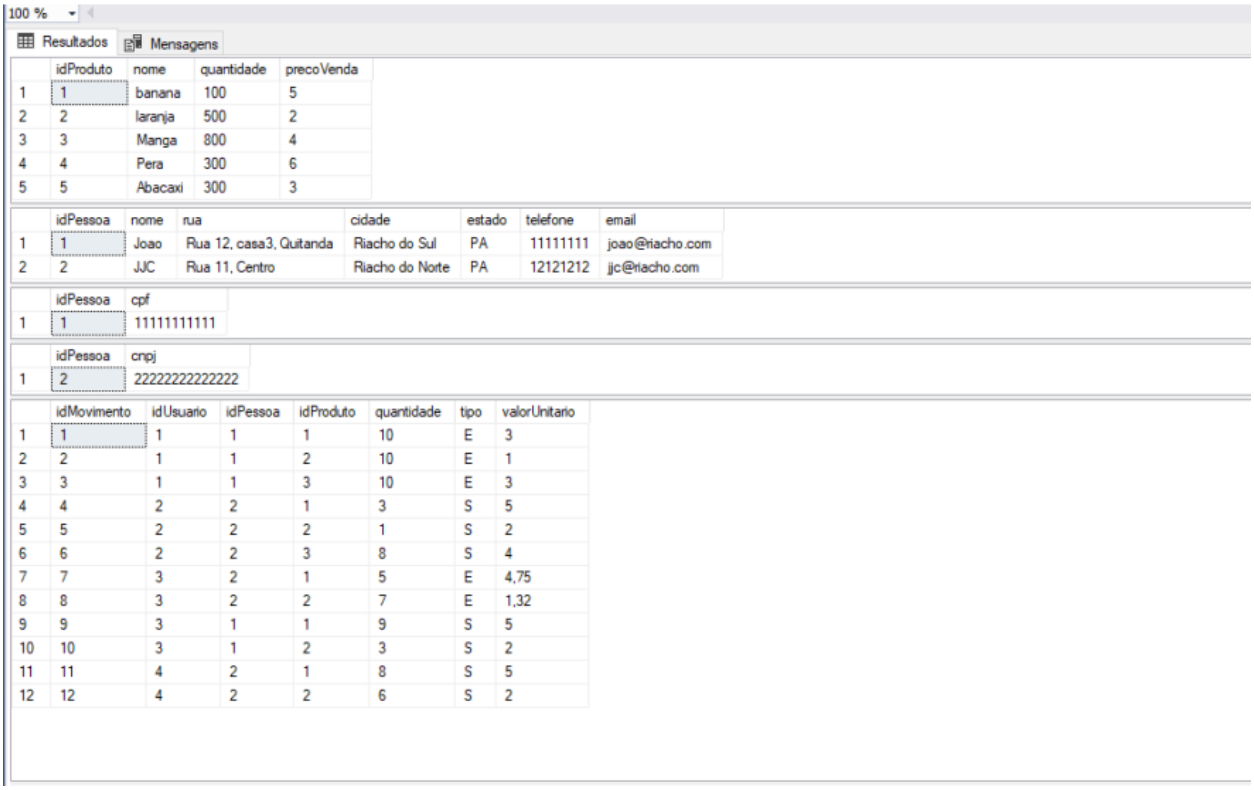
```
  ('JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '12121212', 'jjc@riacho.com');


SELECT * FROM Pessoa;


INSERT INTO PessoaFisica (idPessoa, cpf)
VALUES
  (1, '11111111111');


SELECT * FROM PessoaFisica;


INSERT INTO PessoaJuridica (idPessoa, cnpj)
VALUES
  (2, '22222222222222');


SELECT * FROM PessoaJuridica;


INSERT INTO Movimento (idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario)
VALUES
  (1, 1, 1, 10, 'E', 3.0),
  (1, 1, 2, 10, 'E', 1.0),
  (1, 1, 3, 10, 'E', 3.0),
  (2, 2, 1, 3, 'S', 5.0),
  (2, 2, 2, 1, 'S', 2.0),
  (2, 2, 3, 8, 'S', 4.0),
  (3, 2, 1, 5, 'E', 4.75),
  (3, 2, 2, 7, 'E', 1.32),
  (3, 1, 1, 9, 'S', 5.0),
  (3, 1, 2, 3, 'S', 2.0),
```

(4, 2, 1, 8, 'S', 5.0),

(4, 2, 2, 6, 'S', 2.0);

SELECT * FROM Movimento;

Resultado



100 %

Resultados | Mensagens

	idProduto	nome	quantidade	precoVenda
1	1	banana	100	5
2	2	laranja	500	2
3	3	Manga	800	4
4	4	Pera	300	6
5	5	Abacaxi	300	3

	idPessoa	nome	rua	cidade	estado	telefone	email
1	1	Joao	Rua 12, casa3, Quitanda	Riacho do Sul	PA	11111111	joao@riacho.com
2	2	JJC	Rua 11, Centro	Riacho do Norte	PA	12121212	jjc@riacho.com

	idPessoa	cpf
1	1	11111111111

	idPessoa	cnpj
1	2	22222222222222

	idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valorUnitario
1	1	1	1	1	10	E	3
2	2	1	1	2	10	E	1
3	3	1	1	3	10	E	3
4	4	2	2	1	3	S	5
5	5	2	2	2	1	S	2
6	6	2	2	3	8	S	4
7	7	3	2	1	5	E	4,75
8	8	3	2	2	7	E	1,32
9	9	3	1	1	9	S	5
10	10	3	1	2	3	S	2
11	11	4	2	1	8	S	5
12	12	4	2	2	6	S	2

Análise e Conclusão:

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

As sequências (sequences) e as colunas de identidade (identity columns) são duas maneiras de gerar valores automaticamente para uma coluna em um banco de dados relacional. Ambos são frequentemente usados para criar chaves primárias únicas ou outras colunas que requerem valores exclusivos. Aqui estão as principais diferenças entre eles:

1. Sequências (Sequences):
    - Uma sequência é um objeto de banco de dados que gera uma série de valores numéricos em ordem crescente ou decrescente.
    - As sequências são independentes das tabelas e podem ser usadas em várias tabelas dentro do mesmo banco de dados.
    - Elas podem ser compartilhadas entre diferentes usuários e sessões de conexão.
    - As sequências podem ser usadas para gerar valores para qualquer tipo de coluna, não apenas inteiros, e podem ser configuradas com incrementos e limites específicos.

2. Colunas de Identidade (Identity Columns):
    - Uma coluna de identidade é uma coluna em uma tabela que é automaticamente preenchida com valores sequenciais à medida que novas linhas são inseridas na tabela.
    - As colunas de identidade são específicas para uma tabela e estão vinculadas a essa tabela. Cada tabela pode ter no máximo uma coluna de identidade.
    - A geração de valores para colunas de identidade é normalmente feita pelo próprio mecanismo de banco de dados, e os valores gerados são específicos para essa tabela.

- Em sistemas como SQL Server, MySQL e SQL Anywhere, as colunas de identidade são comumente usadas para gerar valores exclusivos para chaves primárias.

Enquanto as sequências são objetos de banco de dados independentes usados para gerar valores numéricos em ordem, as colunas de identidade são específicas para uma tabela e são usadas para gerar valores automáticos sequenciais para uma coluna específica dessa tabela. A escolha entre usar uma sequência ou uma coluna de identidade depende das necessidades específicas do sistema e das funcionalidades oferecidas pelo mecanismo de banco de dados em uso.

2) Qual a importância das chaves estrangerias para a consistência do banco?

As chaves estrangeiras desempenham um papel fundamental na garantia da consistência dos dados em um banco de dados relacional. Aqui estão algumas das razões pelas quais as chaves estrangeiras são importantes para a consistência do banco:

1. **Integridade Referencial**: As chaves estrangeiras são usadas para estabelecer relacionamentos entre tabelas em um banco de dados relacional. Elas garantem que os dados em uma tabela que faz referência a outra tabela estejam sempre consistentes, evitando referências a registros que não existem ou foram excluídos.
2. **Manutenção da Relação Pai-Filho:** Ao criar um relacionamento de chave estrangeira entre duas tabelas, você está definindo uma relação pai-filho entre elas. Isso significa que os registros na tabela "filha" estão vinculados aos registros correspondentes na tabela "pai". Essa relação ajuda a manter a consistência dos dados e evita inconsistências, como registros órfãos (registros filhos sem um registro pai correspondente).
3. **Evita a Inserção de Dados Inválidos**: As chaves estrangeiras impedem a inserção de dados inválidos que violariam as restrições de integridade referencial. Por exemplo, se uma tabela de pedidos tem uma chave estrangeira que faz referência à tabela de clientes, você não poderá inserir um pedido para um cliente que não existe na tabela de clientes.
4. **Atualizações em Cascata (CASCADE)**: As chaves estrangeiras podem ser configuradas para realizar ações em cascata quando ocorrem operações de atualização ou exclusão na tabela pai. Por exemplo, você pode configurar uma atualização em cascata para que quando o ID de um cliente na tabela de clientes

seja atualizado, todos os pedidos associados a esse cliente na tabela de pedidos também sejam atualizados automaticamente.

5. **Consulta e Manutenção Mais Eficientes**: Ao estabelecer relacionamentos através de chaves estrangeiras, você pode facilmente consultar dados relacionados de maneira eficiente usando junções (joins) entre tabelas. Além disso, as chaves estrangeiras ajudam a garantir que consultas e operações de manutenção mantenham a integridade dos dados em todo o banco de dados.

As chaves estrangeiras desempenham um papel crucial na garantia da consistência dos dados em um banco de dados relacional, ajudando a manter a integridade referencial, prevenir a inserção de dados inválidos e facilitar operações de consulta e manutenção.

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

Na linguagem SQL, há uma sobreposição significativa entre os conceitos da álgebra relacional e do cálculo relacional. No entanto, é possível identificar alguns operadores que são mais diretamente relacionados à álgebra relacional e outros que estão mais associados ao cálculo relacional.

- **Operadores da Álgebra Relacional:**
  - **Seleção (Selection):** Retorna as tuplas que satisfazem uma condição específica.
  - **Projeção (Projection):** Retorna apenas as colunas especificadas das tuplas.
  - **União (Union):** Retorna as tuplas presentes em pelo menos uma das relações especificadas.
  - **Interseção (Intersection):** Retorna as tuplas que são comuns a duas relações.
  - **Diferença (Difference):** Retorna as tuplas que estão em uma relação, mas não na outra.
  - **Produto Cartesiano (Cartesian Product):** Retorna o conjunto de todas as combinações de tuplas de duas relações.
  - **Junção (Join):** Combina tuplas de duas relações com base em uma condição de igualdade entre colunas.

- **Operadores definidos no Cálculo Relacional:**
    - **Operador de Seleção (σ):** Similar à seleção na álgebra relacional, mas expresso usando predicados lógicos.
    - **Operador de Projeção (π):** Similar à projeção na álgebra relacional, mas expresso como uma lista de variáveis ou expressões.
    - **Operador de Ligação (ρ):** Renomeia as variáveis de uma expressão.
    - **Operador de União (∪):** Representa a união de dois conjuntos.
    - **Operador de Interseção (∩):** Representa a interseção de dois conjuntos.
    - **Operador de Diferença (−):** Representa a diferença entre dois conjuntos.
    - **Operador de Produto Cartesiano (×):** Representa o produto cartesiano de dois conjuntos.

É importante observar que, embora a álgebra relacional e o cálculo relacional possuam operadores distintos, muitas implementações de SQL incluem funcionalidades que se sobrepõem, permitindo expressar consultas de maneiras que são conceitualmente semelhantes à álgebra relacional ou ao cálculo relacional.

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

O agrupamento em consultas SQL é feito usando a cláusula GROUP BY. Essa cláusula permite agrupar linhas com base em valores comuns em uma ou mais colunas e, em seguida, aplicar funções de agregação, como SUM, AVG, COUNT, MAX e MIN, às colunas agrupadas.