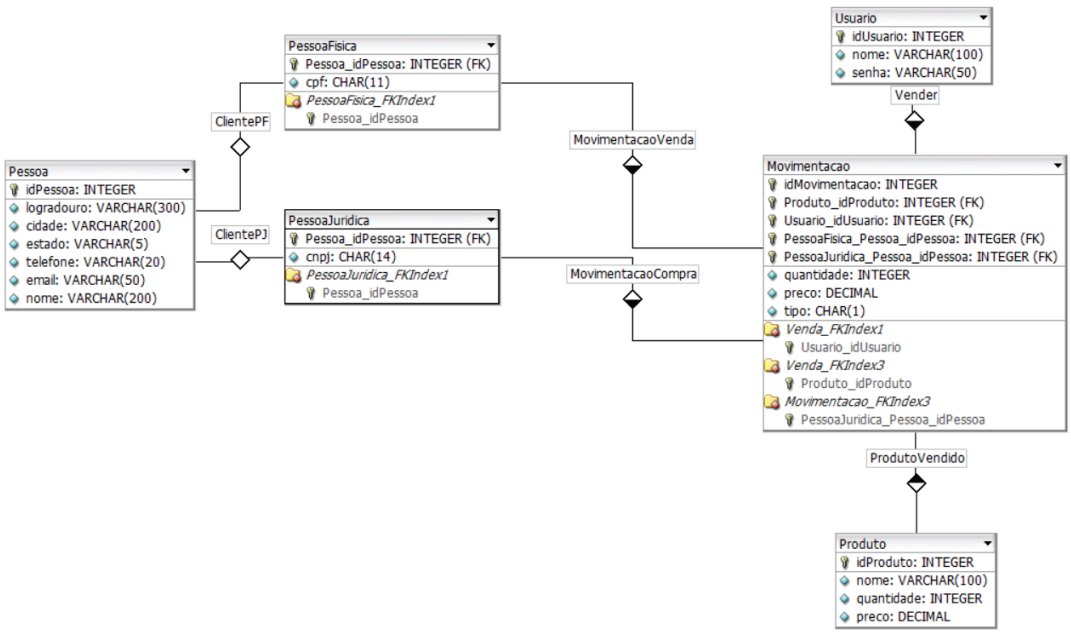
POLO MADUREIRA - RIO DE JANEIRO - RJ
NIVEL 2 : VAMOS MANTER AS INFORMAÇÕES!
TURMA 2024.1 - MUNDO 3
ALUNO: MAYLSON SIMOES

Relatório discente de acompanhamento

Criando o Banco de Dados

Objetivo da Prática

- Identificar os requisitos de um sistema e transformá-los no modelo adequado.
- Utilizar ferramentas de modelagem para bases de dados relacionais.
- Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
- Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
- 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.



```sql
CREATE TABLE usuario (
    idUsuario INTEGER IDENTITY(1,1) PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    senha VARCHAR(50) NOT NULL
);


CREATE TABLE pessoa (
    idPessoa INTEGER PRIMARY KEY,
    nome VARCHAR(200) NOT NULL,
    logradouro VARCHAR(300) NOT NULL,
    cidade VARCHAR(200) NOT NULL,
    estado VARCHAR(5) NOT NULL,
    telefone VARCHAR(20) NOT NULL,
    email VARCHAR(50) NOT NULL
);


CREATE TABLE pessoaFisica (
    idPessoa INTEGER PRIMARY KEY REFERENCES pessoa(idPessoa),
    cpf VARCHAR(11) NOT NULL UNIQUE
);

CREATE TABLE pessoaJuridica (
    idPessoa INTEGER PRIMARY KEY REFERENCES pessoa(idPessoa),
    cnpj VARCHAR(14) NOT NULL UNIQUE
);
```

```sql
CREATE TABLE produto (
    idProduto INTEGER IDENTITY(1,1) PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    quantidade INTEGER NOT NULL,
    preco DECIMAL(10, 2) NOT NULL
);

CREATE TABLE movimentacao (
    idMovimentacao INTEGER IDENTITY(1,1) PRIMARY KEY,
    idUsuario INTEGER NOT NULL,
    idProduto INTEGER NOT NULL,
    idPessoa INTEGER NOT NULL ,
    quantidade INTEGER NOT NULL,
    preco DECIMAL(10, 2) NOT NULL,
    tipo CHAR(1) NOT NULL,
    CONSTRAINT fk_usuario FOREIGN KEY (idUsuario) REFERENCES usuario(idUsuario),
    CONSTRAINT fk_produto FOREIGN KEY (idProduto) REFERENCES produto(idProduto),
    CONSTRAINT fk_pessoa FOREIGN KEY (idPessoa) REFERENCES pessoa(idPessoa)
);

CREATE SEQUENCE seq_idPessoa
    AS INTEGER
    START WITH 1
    INCREMENT BY 1;
```

Análise e Conclusão:

1. Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?

- 1X1

Este tipo de relacionamento ocorre quando cada registro em uma tabela está relacionado a exatamente um registro em outra tabela. Pode ser implementado usando chaves estrangeiras nas duas tabelas, mas com a restrição de que as chaves estrangeiras devem ser únicas (chave única ou **UNIQUE**)

- 1xN

Neste relacionamento, um registro em uma tabela pode estar relacionado a vários registros em outra tabela. Pode ser implementado usando uma chave estrangeira na tabela do lado "N" que referência a chave primária da tabela do lado "1".

- NxN (Muitos para muitos)

Neste relacionamento, vários registros em uma tabela podem estar relacionados a vários registros em outra tabela. Pode ser implementado usando uma tabela intermediária (ou tabela de junção) que contém chaves estrangeiras para as duas tabelas principais.

2. Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?

- **Single Table Inheritance (STI)** (Herança de Tabela Única):

Neste modelo, você coloca todos os dados em uma única tabela, incluindo campos específicos para cada tipo. Você pode adicionar uma coluna de tipo para identificar o tipo de registro.

- **Class Table Inheritance (CTI)** (Herança de Tabela de Classe):

Neste modelo, você cria tabelas separadas para cada tipo de classe e uma tabela base comum. Cada tabela derivada contém uma chave estrangeira que aponta para a tabela base.

3. Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?

- **Interface gráfica intuitiva**: Permite gerenciar bancos de dados com uma interface amigável, facilitando o trabalho com tabelas, procedimentos armazenados, funções, índices, etc.
- **Query Editor**: Possui um editor de consultas que permite escrever, executar e otimizar consultas SQL com recursos como destaque de sintaxe e sugestões automáticas.
- **Gestão visual de objetos**: Você pode visualizar, criar, alterar e excluir objetos do banco de dados usando painéis de gerenciamento.
- **Backup e restauração fáceis**: Permite realizar operações de backup e restauração com poucos cliques.
- **Ferramentas de diagnóstico e análise**: Oferece ferramentas para monitorar o desempenho do banco de dados, como o Activity Monitor e os planos de execução de consultas.
- **Integração com outras ferramentas**: SSMS pode ser integrado com outras ferramentas, como o SQL Profiler, para monitorar eventos e desempenho.
- **Automação de tarefas**: Permite a automação de tarefas repetitivas com scripts ou a criação de jobs.

Alimentando a Base

Inserções

```sql
insert into usuario (nome, senha) values('op1', 'op1');
insert into usuario (nome, senha) values('op2', 'op2');
insert into usuario (nome, senha) values('op3', 'op3');

insert into produto (nome, quantidade, preco) values ('Banana', 100, 5.00);
insert into produto (nome, quantidade, preco) values ('Laranja', 500, 2.00);
insert into produto (nome, quantidade, preco) values ('Manga', 800, 4.00);

insert into pessoa (idPessoa, nome, logradouro, cidade, estado, email, telefone) values (NEXT VALUE FOR seq_idPessoa, 'Joao', 'Rua 12, Ba
insert into pessoa (idPessoa, nome, logradouro, cidade, estado, email, telefone) values (NEXT VALUE FOR seq_idPessoa, 'Pedro', 'Rua 7, Ba
insert into pessoa (idPessoa, nome, logradouro, cidade, estado, email, telefone) values (NEXT VALUE FOR seq_idPessoa, 'Maria', 'Rua 2, Ba
insert into pessoa (idPessoa, nome, logradouro, cidade, estado, email, telefone) values (NEXT VALUE FOR seq_idPessoa, 'Julia', 'Rua 1, Ba

insert into pessoaFisica (idPessoa, cpf) values (1, '11122233344');
insert into pessoaFisica (idPessoa, cpf) values (3, '11122233355');
insert into pessoaJuridica(idPessoa, cnpj) values (2, '11122233344455');
insert into pessoaJuridica(idPessoa, cnpj) values (4, '11122233344466');

insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (1, 2, 4, 200, 4.00, 'E');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (1, 2, 4, 50, 4.50, 'E');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (1, 1, 4, 100, 5.00, 'S')
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (1, 1, 4, 80, 5.00, 'S');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (2, 4, 5, 300, 1.50, 'E');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (2, 4, 5, 100, 1.80, 'E');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (2, 3, 5, 200, 2.00, 'S');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (2, 3, 5, 500, 2.00, 'S');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (2, 4, 6, 400, 3.00, 'E');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (2, 4, 6, 300, 3.20, 'E');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (2, 3, 6, 200, 4.00, 'S');
insert into movimentacao (idUsuario, idPessoa, idProduto, quantidade, preco, tipo) values (1, 3, 6, 200, 4.00, 'S');
```

Consultas

- Dados completos de pessoas físicas.

```sql
/*Dados completos de pessoas fisicas. */
select * from pessoa inner join pessoaFisica on pessoa.idPessoa = pessoaFisica.idPessoa;
```

100 %

Results | Messages

	idPessoa	nome	logradouro	cidade	estado	telefone	email	idPessoa	cpf
1	1	Joao	Rua 12, Bairro 1	Riacho do Norte	PA	11 1234-5678	Joao@email.com	1	11122233344
2	3	Pedro	Rua 7, Bairro 8	Riacho do Sul	PA	11 1234-5678	pedro@email.com	3	11122233355

- Dados completos de pessoas jurídicas.

```sql
select * from pessoa inner join pessoaJuridica on pessoa.idPessoa = pessoaJuridica.idPessoa;
```

100 %

Results | Messages

	idPessoa	nome	logradouro	cidade	estado	telefone	email	idPessoa	cnpj
1	2	Maria	Rua 2, Bairro 7	Riacho do Norte	PA	11 7894-5612	Maria@email.com	2	11122233344455
2	4	Julia	Rua 1, Bairro 10	Riacho do Sul	PA	11 7894-5612	Julia@email.com	4	11122233344466

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

```sql
select pe.nome produto, po.nome fornecedor, m.quantidade, m.preco, (m.preco * m.quantidade) as valorTotal
    from movimentacao m
        inner join pessoa pe on pe.idPessoa = m.idPessoa
        inner join produto po on po.idProduto = m.idProduto
    where tipo = 'E';
```

100 % | Results | Messages

	produto	fornecedor	quantidade	preco	valorTotal
1	Maria	Banana	200	4.00	800.00
2	Julia	Laranja	300	1.50	450.00
3	Maria	Banana	50	4.50	225.00
4	Julia	Laranja	100	1.80	180.00
5	Julia	Manga	400	3.00	1200.00
6	Julia	Manga	300	3.20	960.00

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

```sql
select pe.nome produto, po.nome fornecedor, m.quantidade, m.preco, (m.preco * m.quantidade) as valorTotal
    from movimentacao m
        inner join pessoa pe on pe.idPessoa = m.idPessoa
        inner join produto po on po.idProduto = m.idProduto
    where tipo = 'S';
```

Results | Messages

	produto	fornecedor	quantidade	preco	valorTotal
	Joao	Banana	100	5.00	500.00
	Pedro	Laranja	200	2.00	400.00
	Joao	Banana	80	5.00	400.00
	Pedro	Laranja	500	2.00	1000.00
	Pedro	Manga	200	4.00	800.00
	Pedro	Manga	200	4.00	800.00

- Valor total das entradas agrupadas por produto.

```sql
select p.nome, sum(m.quantidade * m.preco) as totalEntradas
    from movimentacao m
        inner join produto p on p.idProduto = m.idProduto
    where tipo = 'E'
    group by p.nome;
```

Results | Messages

	nome	totalEntradas
	Banana	1025.00
	Laranja	630.00
	Manga	2160.00

- Valor total das saídas agrupadas por produto.

```sql
select p.nome, sum(m.quantidade * m.preco) as TotalSaidas
    from movimentacao m
        inner join produto p on p.idProduto = m.idProduto
    where tipo = 'S'
    group by p.nome;
```

0 %

Results | Messages

	nome	TotalSaidas
	Banana	900.00
	Laranja	1400.00
	Manga	1600.00

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

```sql
select nome from usuario where idUsuario not in (select idUsuario from movimentacao where tipo = 'E');
```

100 %

Results | Messages

	nome
1	op3

- Valor total de entrada, agrupado por operador.

```sql
select o.nome, sum(m.quantidade * m.preco) as TotalEntrada
    from movimentacao m
        inner join usuario o on o.idUsuario = m.idUsuario
    where tipo = 'E'
    group by o.nome;
```

00 %

Results | Messages

	nome	TotalEntrada
1	op1	1025.00
2	op2	2790.00

- Valor total de saída, agrupado por operador.

```sql
select o.nome, sum(m.quantidade * m.preco) as TotalSaidas
    from movimentacao m
        inner join usuario o on o.idUsuario = m.idUsuario
    where tipo = 'S'
    group by o.nome;
```

Results | Messages

nome	TotalSaidas
op1	1700.00
op2	2200.00

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

```sql
select p.nome, sum(m.quantidade * m.preco) / SUM(m.quantidade) as mediaPonderada
    from movimentacao m
        inner join produto p on p.idProduto = m.idProduto
    where tipo = 'S'
    group by p.nome;
```

Results | Messages

nome	mediaPonderada
Banana	5.000000
Laranja	2.000000
Manga	4.000000

Análise e Conclusão:

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

Uma **SEQUENCE** é um objeto de banco de dados separado que gera uma sequência de números em um intervalo definido pelo usuário. Ela pode ser usada em várias tabelas e campos, não estando associada diretamente a uma única tabela ou coluna.
Uma coluna **IDENTITY** é uma propriedade definida em uma coluna específica de uma tabela. Ela gera valores automaticamente para essa coluna sempre que um novo registro é inserido. A propriedade **IDENTITY** está associada a uma única tabela e coluna específica.

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

As chaves estrangeiras são importantes para garantir a consistência e integridade referencial dos dados em um banco de dados, garantem que um registro em uma

tabela referenciada exista antes de ser usado em outra tabela e elas asseguram que os relacionamentos entre tabelas permaneçam consistentes.

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

Alguns operadores comuns de SQL que pertencem à álgebra relacional incluem: SELECT, UNION, INTERSECT, EXCEPT E JOIN.

Os operadores de SQL relacionados ao cálculo relacional incluem:
ALL, ANY, WHERE, HAVING, AND, OR E NOT.

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

O agrupamento em consultas é realizado usando a cláusula **GROUP BY** em SQL. Essa cláusula permite agrupar os resultados de uma consulta com base em uma ou mais colunas.
O requisito obrigatório para usar o agrupamento (**GROUP BY**) é que todas as colunas selecionadas na consulta (**SELECT**) que não sejam funções de agregação (como **SUM()**, **AVG()**, **COUNT()**, etc.) devem ser incluídas na cláusula **GROUP BY.** Isso garante que os dados sejam agrupados corretamente.

Repositório git: https://github.com/VascoMay/movimentacaoDB