

# Projeto de Banco de Dados - Modelagem Conceitual

### Edson Luiz Pacheco Junior - 202308892185

Polo Jardim Goiás RPG0015 - Vamos manter as informações! - 9001 - 2024.3

# Objetivo da Prática

- 1. Identificar os requisitos de um sistema e transformá-los no modelo adequado.
- 2. Utilizar ferramentas de modelagem para bases de dados relacionais.
- 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 da sintaxe SQL, na plataforma do SQL Server.

# 1º Procedimento | Criando o Banco de Dados

Modelo: Está no final do documento a imagem do modelo.

Códigos de criação:

### SEQUENCE

CREATE SEQUENCE seq\_pessoa\_id

START WITH 1

INCREMENT BY 1

NO CYCLE;

```
    PESSOA
```

```
CREATE TABLE pessoa (
        idPessoa INTEGER PRIMARY KEY DEFAULT NEXT VALUE FOR seq_pessoa_id,
        nome VARCHAR(255) NOT NULL,
        logradouro VARCHAR(255),
 cidade VARCHAR(255),
 estado CHAR(2),
 telefone VARCHAR(11),
 email VARCHAR(255)
  );
PESSOA FÍSICA
 CREATE TABLE pessoa fisica (
      idPessoaFisica INTEGER PRIMARY KEY,
      idPessoa INTEGER UNIQUE,
      cpf CHAR(14),
CONSTRAINT FK_PessoaFisica_Pessoa FOREIGN KEY (idPessoa) REFERENCES
 Pessoa(idPessoa)
  );
PESSOA JURÍDICA
 CREATE TABLE pessoa juridica (
        idPessoaJuridica INTEGER PRIMARY KEY,
        idPessoa INTEGER UNIQUE,
        cnpj CHAR(18),
     CONSTRAINT FK_PessoaJuridica_Pessoa FOREIGN KEY (idPessoa) REFERENCES
 Pessoa(idPessoa)
  );
```

### • PRODUTO

```
CREATE TABLE produto (
       idProduto INTEGER PRIMARY KEY,
       nome VARCHAR(255) NOT NULL,
       quantidade INTEGER NOT NULL,
       precoVenda NUMERIC(10, 2)
 );
USUÁRIO
CREATE TABLE usuario (
       idUsuario INTEGER PRIMARY KEY,
       login VARCHAR(255) NOT NULL,
       senha VARCHAR(255) NOT NULL
 );
MOVIMENTO
 CREATE TABLE movimento (
       idMovimento INTEGER PRIMARY KEY,
       idUsuario INTEGER,
       idPessoa INTEGER,
       idProduto INTEGER,
       quantidade INTEGER NOT NULL,
       tipo CHAR(1),
       valorUnitario NUMERIC(10, 2),
    CONSTRAINT FK_Movimento_Usuario FOREIGN KEY (idUsuario) REFERENCES
Usuario(idUsuario),
  CONSTRAINT FK Movimento Pessoa FOREIGN KEY (idPessoa) REFERENCES
Pessoa(idPessoa),
CONSTRAINT FK Movimento Produto FOREIGN KEY (idProduto) REFERENCES
Produto(idProduto)
```

#### Conclusão:

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

Na cardinalidade 1x1, é criada uma chave estrangeira que referencia a chave primária de outra tabela, neste caso, uma linha da tabela irá corresponder a exatamente uma linha na outra tabela. Na cardinalidade 1xN, a chave primária de uma tabela é referenciada através de uma chave estrangeira em várias linhas de outra tabela.

Na cardinalidade NxN, para esse relacinamento é necessário criar uma tabela de junção que conterá duas chaves estrangeiras, cada uma referenciando a chave primária de outra tabela. Assim permitindo um relacionamento de muitos para muitos, pois permitirá que múltiplas linhas da tabela estarão relacionadas a múltiplas linhas de outra tabela.

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

É um relacionamento 1x1, que é feito pelas tabelas por subclasse. Onde uma tabela (pai) terá todos os atributos que serão compartilhados entre os filhos, e então as tabelas de subclasse (filho) terão atributos mais específicos.

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

A sua interface gráfica facilita muito o uso e gerenciamento, fazendo com que não seja necessário ter que escrever SQL manualmente. Também o intellisense dele facilita muito a escrita de SQL, pois corrige e também ajuda com as sugestões. A possibilidade de fazer backup, restauração e manutenção do banco, o que traz maior segurança e controle.

# 2º Procedimento | Alimentando a Base

Códigos de inserção:

# USUÁRIO

```
INSERT INTO usuario (idUsuario, login, senha)
VALUES (1, 'op1', 'op1');
INSERT INTO usuario (idUsuario, login, senha)
```

### PRODUTO

```
INSERT INTO produto (idProduto, nome, quantidade, precoVenda)
```

VALUES (1, 'Banana', 100, 5.00);

VALUES (2, 'op2', 'op2');

INSERT INTO produto (idProduto, nome, quantidade, precoVenda)

VALUES (3, 'Laranja', 500, 2.00);

INSERT INTO produto (idProduto, nome, quantidade, precoVenda)

VALUES (4, 'Manga', 800, 4.00);

# • PESSOA E PESSOA FÍSICA

DECLARE @idPessoaFisica INTEGER;

SET @idPessoaFisica = NEXT VALUE FOR seq pessoa id;

INSERT INTO pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email)

VALUES (@idPessoaFisica, 'Joao', 'Rua 12, casa 3, Quitanda', 'Riacho do Sul', 'PA', '1111-1111', 'joao@riacho.com');

INSERT INTO pessoa\_fisica (idPessoaFisica, idPessoa, cpf)

VALUES (@idPessoaFisica, @idPessoaFisica, '123.456.789-10');

### PESSOA E PESSOA JURÍDICA

DECLARE @idPessoaJuridica INTEGER;

SET @idPessoaJuridica = NEXT VALUE FOR seq pessoa id;

INSERT INTO pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email)

VALUES (@idPessoaJuridica, 'JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '1212-1212', 'jjc@riacho.com');

INSERT INTO pessoa juridica (idPessoaJuridica, idPessoa, cnpj)

VALUES (@idPessoaJuridica, @idPessoaJuridica, '12.345.678/0001-09');

### MOVIMENTO

INSERT INTO movimento (idMovimento, idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario)

```
VALUES (1, 1, 1, 1, 20, 'S', 4.00);
```

INSERT INTO movimento (idMovimento, idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario)

```
VALUES (4, 1, 1, 3, 15, 'S', 2.00);
```

INSERT INTO movimento (idMovimento, idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario)

```
VALUES (5, 2, 1, 3, 10, 'S', 3.00);
```

INSERT INTO movimento (idMovimento, idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario)

```
VALUES (7, 1, 2, 4, 15, 'E', 5.00);
```

INSERT INTO movimento (idMovimento, idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario)

```
VALUES (8, 1, 2, 4, 20, 'E', 4.00)
```

# Códigos de consulta:

# a. Dados completos de pessoas físicas.

```
SELECT
p.idPessoa,
p.nome,
p.logradouro,
p.cidade,
p.estado,
p.telefone,
p.email,
pf.cpf
FROM pessoa p
JOIN pessoa fisica pf ON p.idPessoa = pf.idPessoa;
```

Ⅲ F	Resultados	Mensagens Mensagens							
			logradouro	cidade	estado	telefone	email	cpf	
1	1	Joao	Rua 12, casa 3, Quitanda	Riacho do Sul	PA	1111-1111	joao@riacho.com	123.456.789-10	

# b. Dados completos de pessoas jurídicas.

```
select
p.idPessoa,
p.nome,
p.logradouro,
p.cidade,
p.estado,
p.telefone,
p.email,
pj.cnpj
```

FROM pessoa p JOIN pessoa\_juridica pj ON p.idPessoa = pj.idPessoa;

Resultados Mensagens

idPessoa nome logradouro cidade estado telefone email cnpj

1 2 JJC Rua 11, Centro Riacho do Norte PA 1212-1212 jjc@riacho.com 12.345.678/0001-09

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

```
SELECT

m.idMovimento,

p.nome AS fornecedor,

pr.nome AS produto,

m.quantidade,

m.valorUnitario,

(m.quantidade * m.valorUnitario) AS valorTotal

FROM movimento m

JOIN pessoa p ON m.idPessoa = p.idPessoa

JOIN pessoa_juridica pj ON p.idPessoa = pj.idPessoa

JOIN produto pr ON m.idProduto = pr.idProduto

WHERE m.tipo = 'E';
```

	Resultados 🖺	Mensagens				
	idMovimento	fomecedor	produto	quantidade	valorUnitario	valorTotal
1	7	JJC	Manga	15	5.00	75.00
2	8	JJC	Manga	20	4.00	80.00

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

### **SELECT**

m.idMovimento,

p.nome AS comprador,

pr.nome AS produto,

m.quantidade,

m.valorUnitario,

(m.quantidade \* m.valorUnitario) AS valorTotal

FROM movimento m

JOIN pessoa p ON m.idPessoa = p.idPessoa

JOIN pessoa\_fisica pf ON p.idPessoa = pf.idPessoa

JOIN produto pr ON m.idProduto = pr.idProduto

WHERE m.tipo = 'S';

Resultados Mensagens						
	idMovimento	comprador	produto	quantidade	valorUnitario	valorTotal
1	1	Joao	Banana	20	4.00	80.00
2	4	Joao	Laranja	15	2.00	30.00
3	5	Joao	Laranja	10	3.00	30.00

# e. Valor total das entradas agrupadas por produto.

### **SELECT**

pr.nome AS produto,

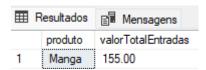
SUM(m.quantidade \* m.valorUnitario) AS valorTotalEntradas

FROM movimento m

JOIN produto pr ON m.idProduto = pr.idProduto

WHERE m.tipo = 'E'

GROUP BY pr.nome;



# f. Valor total das saídas agrupadas por produto.

# SELECT

pr.nome AS produto,

SUM(m.quantidade \* m.valorUnitario) AS valorTotalSaidas

FROM movimento m

JOIN produto pr ON m.idProduto = pr.idProduto

WHERE m.tipo = 'S'

GROUP BY pr.nome;

⊞ F	Resultados	Mensagens			
	produto	valorTotalSaidas			
1	Banana	80.00			
2	Laranja	60.00			

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

SELECT u.idUsuario, u.login

FROM usuario u

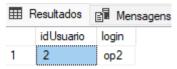
WHERE u.idUsuario NOT IN (

SELECT m.idUsuario

FROM movimento m

WHERE m.tipo = 'E'

);



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

**SELECT** 

u.login AS operador,

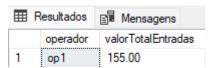
SUM(m.quantidade \* m.valorUnitario) AS valorTotalEntradas

FROM movimento m

JOIN usuario u ON m.idUsuario = u.idUsuario

WHERE m.tipo = 'E'

GROUP BY u.login;



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

**SELECT** 

u.login AS operador,

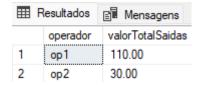
SUM(m.quantidade \* m.valorUnitario) AS valorTotalSaidas

FROM movimento m

JOIN usuario u ON m.idUsuario = u.idUsuario

WHERE m.tipo = 'S'

GROUP BY u.login;



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

SELECT

pr.nome AS produto,

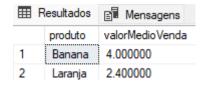
SUM(m.quantidade \* m.valorUnitario) / SUM(m.quantidade) AS valorMedioVenda

FROM movimento m

JOIN produto pr ON m.idProduto = pr.idProduto

WHERE m.tipo = 'S'

GROUP BY pr.nome;



### Conclusão:

a. Quais as diferenças no uso de *sequence* e *identity*?

O identify é uma propriedade de uma coluna em uma tabela, e é atrelado a uma tabela específica, impossibilitando a sua reutilização. O seu controle de valor é automático, o que faz com que tenha menos flexibilidade.

A sequence é um objeto separado no banco de dados que gera valores numéricos em sequência, o que faz com que ela não esteja atrelada a nenhuma tabela, assim podendo ser reutilizada em várias tabelas. Possibilita uso de funções, é possível manipular a sequence podendo reiniciá-la ou ajustá-la conforme necessário, oferecendo melhor flexibilidade e controle.

b. Qual a importância das chaves estrangerias para a consistência do banco?
 Consistência de dados, pois evita a inserção de valores em uma tabela que não têm correspondência em outra tabela. Também ajuda na integridade

referencial e prevenção de dados órfãos, pois se uma linha em uma tabela for deletada, todas as referências devem ser tratadas corretamente, impedindo a inconsistência de dados.

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

Álgebra Relacional: select, select distinct, join, union, except, intersect e cross join.

Cálculo Relacional: Igual (=), Maior que (>), Menor que (<), AND, OR, NOT, EXISTS, FOR ALL e outros.

d. Como é feito o agrupamento em consultas, e qual requisito é obrigatório?
É feito utilizando a cláusula GROUP BY, o requisito obrigatório é que todas as colunas selecionadas que não estejam em uma função agregada devem estar listadas no GROUP BY.

### Conclusão

Elabore uma análise crítica da sua Missão Prática.

Foi uma boa missão prática, em alguns momentos tive dificuldade em entender o que foi pedido, mas creio que consegui fazer corretamente. Não senti tanta dificuldade em fazer pois já tenho um certo conhecimento sobre, mas sempre é bom melhorar e praticar, visto que trabalho na área, mas pouco uso faço de SQL apesar de conhecer. Então me ajudou a praticar e compreender melhor.

### Modelo:

