Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

André Luis Gonçalves Carvalhal - Matrícula: 20220318540

Campus Barra World - Desenvolvimento Full-Stack
Vamos manter as informações! – 9001 – 2023.3

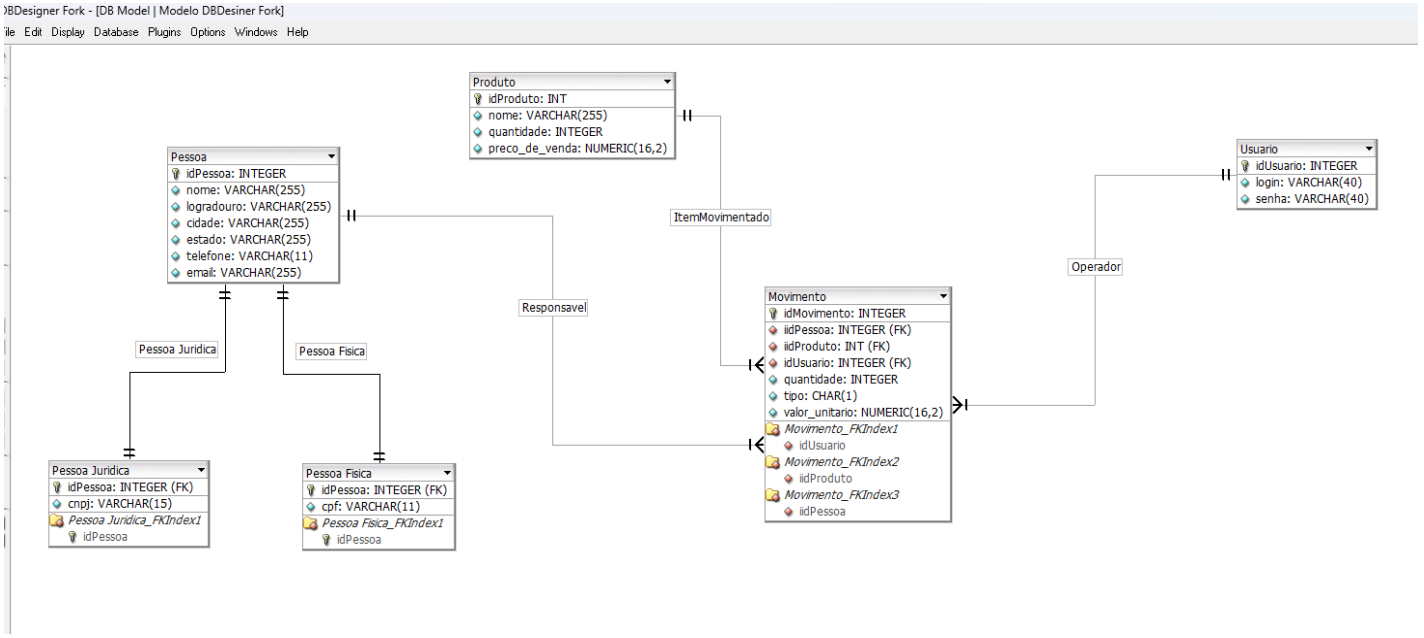
Objetivo da Prática

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.**

Link GitHub

https://github.com/ANDREC1986/RPG0015-202203185403

1º Procedimento | Criando o Banco de Dados



Criar-database-e-tabelas.sql

```
GO
CREATE DATABASE loja;

GO
use loja

CREATE TABLE usuario (
	idUsuario int identity(1,1) primary key not null,
	login varchar(40) not null,
	senha varchar(40) not null
)

CREATE TABLE pessoa (
	idPessoa int primary key not null,
	nome varchar(255) not null,
	logradouro varchar(255) not null,
	cidade varchar(255) not null,
	estado varchar(2) not null,
	telefone varchar(11) not null,
	email varchar(255) not null
)

CREATE TABLE pessoa_juridica (
	idPessoa int foreign key references pessoa(idPessoa) not null primary key,
	cnpj varchar(15) not null
)

CREATE TABLE pessoa_fisica (
	idPessoa int foreign key references pessoa(idPessoa) primary key,
	cpf varchar(11) not null
)

CREATE TABLE produto (
	idProduto int identity(1,1) primary key,
```

```
	nome varchar(255) not null,
	quantidade int not null,
	preco_de_venda numeric(16,2) not null
)

CREATE TABLE movimento(
	idMovimento int identity(1,1) primary key not null,
	idUsuario int foreign key references usuario(idUsuario) not null,
	idPessoa int foreign key references pessoa(idPessoa) not null,
	idProduto int foreign key references produto(idProduto) not null,
	quantidade int not null,
	tipo char(1) not null,
	valor_unitario numeric(16,2) not null
)

CREATE SEQUENCE idPessoa_SEQ
			AS INT
			START WITH 7
			INCREMENT BY 8
```

Inserindo dados.sql

```
GO
use loja
INSERT INTO usuario (login,senha) VALUES ('op1','op1')
INSERT INTO usuario (login,senha) VALUES ('op2','op2')

INSERT INTO produto (nome,preco_de_venda,quantidade) VALUES ('Banana',5.00,100)
INSERT INTO produto (nome,preco_de_venda,quantidade) VALUES ('UVA',5.00,100)
INSERT INTO produto (nome,preco_de_venda,quantidade) VALUES ('Laranja',2.00,500)
INSERT INTO produto (nome,preco_de_venda,quantidade) VALUES ('Manga',4.10,800)
DELETE FROM produto WHERE idProduto=2

DECLARE @ID INT

SET @ID = next value for idPessoa_SEQ
INSERT INTO pessoa (idPessoa,nome,logradouro,cidade,estado,telefone,email) VALUES (@ID,'Joao','Rua 12, casa 3,
Quitanda','Riacho do Sul','PA','1111-1111','joao@riacho.com')
INSERT INTO pessoa_fisica (idPessoa,cpf) VALUES (@ID,'11111111111');

DECLARE @ID2 INT
SET @ID2 = next value for idPessoa_SEQ
INSERT INTO pessoa (idPessoa,nome,logradouro,cidade,estado,telefone,email) VALUES (@ID2,'JJC','Rua 11, Centro','Riacho do
Norte','PA','1212-1212','jjc@riacho.com')
INSERT INTO pessoa_juridica (idPessoa,cnpj) VALUES (@ID2,'222222222222222');

INSERT INTO movimento (idUsuario,idPessoa,idProduto,quantidade,tipo,valor_unitario) VALUES (1,7,1,20,'S',4.00);
INSERT INTO movimento (idUsuario,idPessoa,idProduto,quantidade,tipo,valor_unitario) VALUES (1,7,3,15,'S',2.00);
INSERT INTO movimento (idUsuario,idPessoa,idProduto,quantidade,tipo,valor_unitario) VALUES (2,7,3,10,'S',3.00);
INSERT INTO movimento (idUsuario,idPessoa,idProduto,quantidade,tipo,valor_unitario) VALUES (1,15,3,15,'E',5.00);
INSERT INTO movimento (idUsuario,idPessoa,idProduto,quantidade,tipo,valor_unitario) VALUES (1,15,4,20,'E',4.00);
```

Conclusão:

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

No caso de A^1 para B^1 - Cada entrada de A só pode ter relação com uma entrada de B e vice-versa. Sendo a chave primária de A se torna chave estrangeira em B e só pode ser relacionada a um elemento de B.

No caso A^1 para B^n : Cada elemento de A pode ter relacionamento com N elementos de B. Sendo que a chave primária de A torna-se chave estrangeira de B.

No caso de A^n para B^n é criado um novo elemento de relação, onde as chaves primárias de A e B são relacionadas. E N elementos de A podem se relacionar com N elementos de B.

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

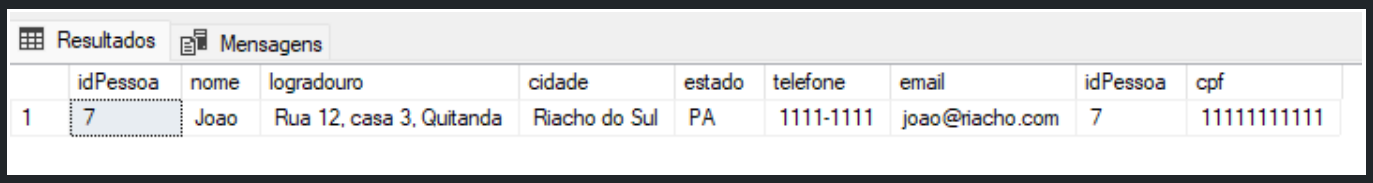
O relacionamento A^1 para B^1, onde o elemento B recebe a chave primária A. Sendo assim B só existe se A existir também, criando assim uma dependência/herança.

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

Permitindo a criação, visualização e edição de bancos de dados de forma visual, sem a utilização de comandos SQL. Mapeamento da estrutura de relação do banco de dados. Permitindo, importar, exportar e executar scripts com arquivos .sql. Fornecendo assistência na criação de comandos SQL através do Intellisense do editor de scripts. Fornecendo assistentes para exportação e importação de dados, integração entre bancos. E permitindo agendamento e automatização de funções de backup e manutenção do banco.

2º Procedimento | Alimentando a Base

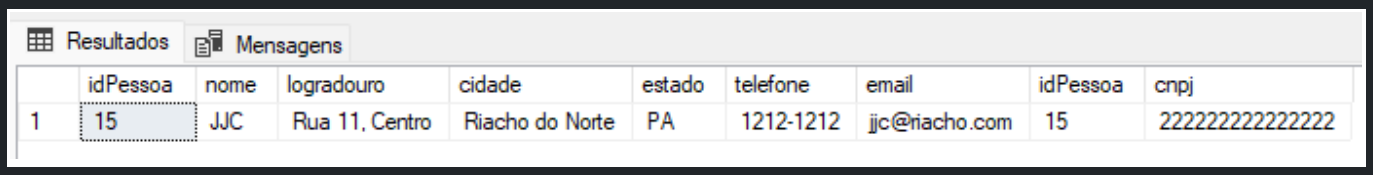
```
SELECT * FROM pessoa, pessoa_fisica WHERE pessoa.idPessoa = pessoa_fisica.idPessoa;
```



Resultados | Mensagens

	idPessoa	nome	logradouro	cidade	estado	telefone	email	idPessoa	cpf
1	7	Joao	Rua 12, casa 3, Quitanda	Riacho do Sul	PA	1111-1111	joao@riacho.com	7	11111111111

```
SELECT * FROM pessoa, pessoa_juridica WHERE pessoa.idPessoa = pessoa_juridica.idPessoa;
```



Resultados | Mensagens

	idPessoa	nome	logradouro	cidade	estado	telefone	email	idPessoa	cnpj
1	15	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com	15	222222222222222

```
SELECT *, quantidade * valor_unitario as valor_total FROM movimento WHERE tipo = 'E'
```

Resultados | Mensagens

	idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valor_unitario	valor_total
1	6	1	15	3	15	E	5.00	75.00
2	7	1	15	4	20	E	4.00	80.00

SELECT *, quantidade * valor_unitario as valor_total FROM movimento WHERE tipo = 'S'

Resultados | Mensagens

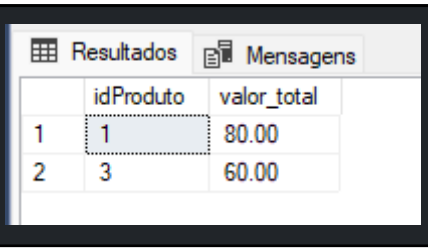
	idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valor_unitario	valor_total
1	1	1	7	1	20	S	4.00	80.00
2	2	1	7	3	15	S	2.00	30.00
3	3	2	7	3	10	S	3.00	30.00

SELECT idProduto, SUM(quantidade * valor_unitario) as valor_total FROM movimento WHERE tipo = 'E' Group by idProduto

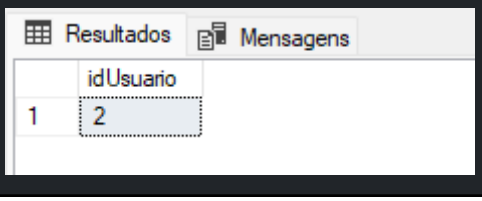
Resultados | Mensagens

	idProduto	valor_total
1	3	75.00
2	4	80.00

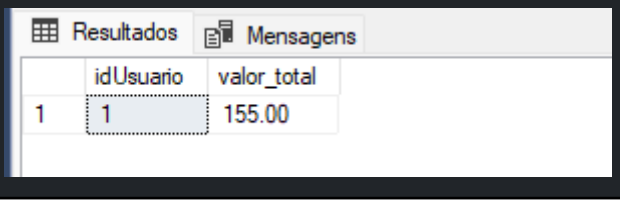
SELECT idProduto, SUM(quantidade * valor_unitario) as valor_total FROM movimento WHERE tipo = 'S' Group by idProduto



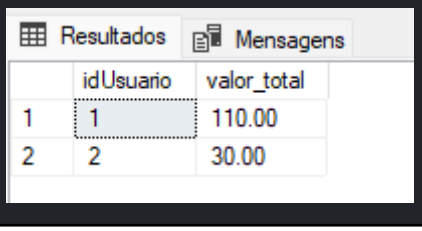
SELECT idUsuario from movimento EXCEPT SELECT idUsuario from movimento WHERE tipo = 'E'



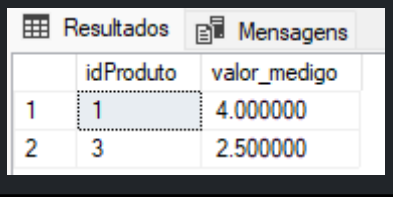
SELECT idUsuario, SUM(quantidade * valor_unitario) as valor_total FROM movimento WHERE tipo = 'E' Group by idUsuario



SELECT idUsuario, SUM(quantidade * valor_unitario) as valor_total FROM movimento WHERE tipo = 'S' Group by idUsuario



SELECT idProduto, AVG(valor_unitario) as valor_medigo FROM movimento WHERE tipo = 'S' Group by idProduto



Resultados | Mensagens

	idProduto	valor_medigo
1	1	4.000000
2	3	2.500000

Conclusão:

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

Embora sejam semelhantes no aspecto em que podem ter seu tipo, valor inicial, incremento, definidos na sua criação. Se diferenciam quando uma identity é um método de auto-incremento uma coluna específica em uma tabela. Já um sequence é um objeto separado, que fornece um valor que pode ser utilizado como chave em diversas tabelas diferentes, além de ter os parâmetros de mínimo, máximo e ciclo determinados.

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

As chaves estrangeiras garantem a consistência do banco de dados, pois garantem que não sejam criadas entradas com valores que não respeitem as regras de negócio.

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

SELECT, UNION, INTERSECT, EXCEPT, CROSS JOIN e JOIN são operadores da álgebra relacional, o cálculo relacional é implementado através do SELECT, PROJECT e pela capacidade de definir alias para atributos e/ou relações com o AS.

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

O agrupamento de consultas o 'GROUP BY', é requisito que todas as colunas listadas sejam utilizadas no 'GROUP BY' e que aquelas que não serão agrupadas possuam um fator agregador, 'SUM', 'COUNT', 'AVG', 'MAX' ou 'MIN'.

Conclusão:

A prática poderia ter incluído a criação de um TRIGGER para que quando fosse executado um INSERT na tabela MOVIMENTO verificasse se o idUsuario era compatível com o TIPO inserido. Preservando ainda mais a regra de negócios proposta no enunciado.