Campus de Mogi Guaçu – SP

Aluno: João Lucas Menicuci

Matrícula: 2023 0421 4425

Curso: Desenvolvimento Full-Stack

Turma: 2023.1

Disciplina: RPG0015 – Vamos manter as informações!

Missão Prática Nível 2 – Mundo 3

Mogi Guaçu

2024

I. Primeiro Procedimento

1. Título da Prática: “Criando o Banco de Dados”.

2. 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.

3. Códigos de Desenvolvimento: Na seção “III. Anexo” deste relatório. Link para o repositório do GitHub: https://github.com/joaomenicuci/mp-mundo3-nivel2.

4. Resultados:

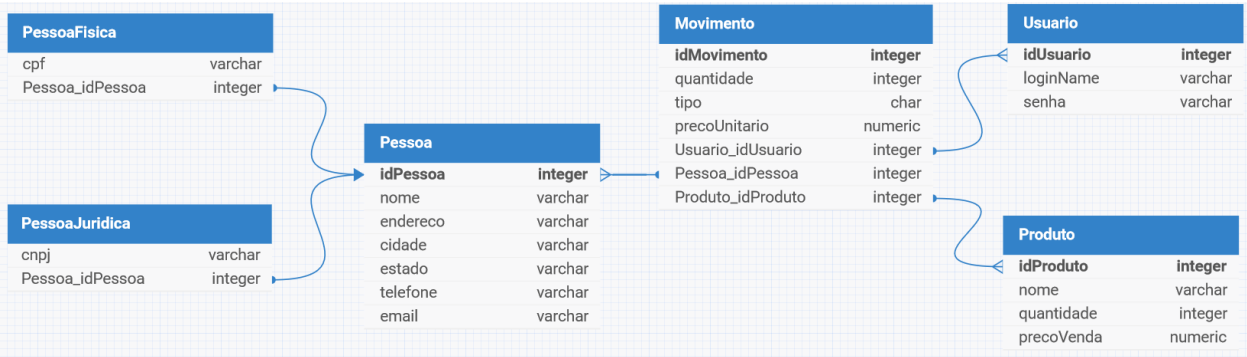

Figura 1: Diagrama Entidade Relacionamento (DER).

5. Análise e Conclusão:

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

   A cardinalidade 1X1 é necessário ter duas tabelas, onde a primeira é precisa ter uma chave primária e a segunda uma chave estrangeira com referencia a chave da tabela 1. Neste caso, os valores das chaves devem ser únicos, e os valores não podem ser repetidos.

   Para a cardinalidade 1XN também é necessário ter duas tabelas, e estabelecer chaves primárias distintas para cada uma delas, e criar uma chave estrangeira

para a tabela 2, com referencia para a chave primária da primeira tabela. A diferença é que os valores da coluna da chave estrangeiras podem ser repetidos.

Já a cardinalidade NXN ocorre quando N linhas de uma tabela estão relacionados com N linhas de outra tabela. Em questão das chaves, as tabelas precisam possuir uma chave primária, e as tabelas intermediárias precisam ter duas chaves estrangeiras, com referencia as chaves primárias das tabelas primárias.

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

O tipo de relacionamento utilizado para representar o uso de herança em bancos de dados relacionais é a generalização/especialização. Esta herança permite organizar as entidades em hierarquia, onde uma superclasse pode ter uma ou mais entidades de subclasse.

A utilização de herança reduz a redundância de dados e melhora a legibilidade do modelo de dados.

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

O SQL Server Management Studio é uma das melhores ferramentas para fazer o gerenciamento de banco de dados, onde melhora a produtividade de forma significativa.

Como destaque desta ferramenta está a automação de tarefas, onde é possível criar scripts para automatizar tarefas repetitivas, e assim economizar tempo. Outro detalhe relevante é a sua integração com produtos da própria Microsoft, como o Visual Studio Code e o Azure Data Studio.

Com o SQL Server Management Studio é possível monitorar o desempenho do banco de dados, aonde pode se detectar possíveis limitações e gargalos do sistema, além de monitorar a saúde do banco de dados, espaço em disto e uso da CPU do computador/servidor.

II. Segundo Procedimento

1. Título da Prática: “Alimentando a Base”.

2. 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.

3. Códigos de Desenvolvimento: Na seção “III. Anexo” deste relatório. Link para o repositório do GitHub: https://github.com/joaomenicuci/mp-mundo3-nivel2.

4. Resultados:

idPeessoa	nome	endereco	cidade	estado	telefone	email
1	Alfredo	Rua A, 100	São Paulo	SP	3891-1000	alfredo@gmail.com
2	Bruno	Rua B, 200	Rio de Janeiro	RJ	3891-2000	bruno@gmail.com
3	Carla	Rua C, 300	Belo Horizonte	MG	3891-3000	carla@gmail.com
4	Domingos Supermercados	Rua D, 400	Curitiba	PR	3891-4000	domingos@gmail.com
5	Escritório 5	Rua E, 500	Vitória	ES	3891-5000	escritorio@gmail.com

Tabela 1: Tabela Pessoa

Pessoa_idPessoa	cpf
1	11111111111
2	22222222222
3	33333333333

Tabela 2: Tabela Pessoa Física

Pessoa_idPessoa	
4	44444444444444
5	55555555555555

Tabela 3: Pessoa Jurídica

idUsuario	loginName	Senha
1	op1	op1
2	op2	op2

Tabela 4: Tabela Usuário

idProduto	nome	quantidade	precoVenda
1	Arroz	10	15
2	Feijão	20	8
3	Farofa	25	5

Tabela 5: Tabela Produto

idMovimento	Usuario_idUsuario	Pessoa_idPessoa	Produto_idProduto	quantidade	tipo	precoUnitario
1	1	1	1	10	E	15
2	2	2	2	20	S	8
3	1	3	3	25	E	5

Tabela 6: Tabela Movimento.

5. Análise e Conclusão:

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

    Tanto quando o uso de *sequence* e *identify* são utilizadas para geração de numeração automática, porém a função *identify* é dependente da coluna da tabela onde é aplicada, já *sequence* é independente da coluna.

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

    As chaves estrangeiras são de extrema importância quando trabalhamos com Sistemas de Gerenciamento de Banco de Dados, pois elas que mantem a integridade das tabelas do sistema. Caso seja feita alguma alteração, exclusão ou inserção em uma tabela, esta alteração será refletida para todas as tabelas que possuem a mesma chave estrangeira vinculada.

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

    Os operadores de álgebra relacional são seleção, união, projeção, diferença, junção e produto cartesiano. Já as de cálculo relacional possuem equivalência com as de álgebra relacional, ou em outras palavras, eles são idênticos entre si.

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

    O agrupamento ocorre com o uso do código GROUP BY, onde o mesmo agrupa linhas baseadas em uma função. As funções de agrupamento podem ser SUM, AVG, MAX, MIN, dentre outras.

III. Anexo

- script1.sql

```
USE Loja;
GO

CREATE SEQUENCE orderPessoa
AS INT
START WITH 1
INCREMENT BY 1;
```

```
CREATE TABLE Pessoa(
  idPessoa INTEGER NOT NULL,
  nome VARCHAR(255),
  endereco VARCHAR(255),
  cidade VARCHAR(255),
  estado CHAR(2),
  telefone VARCHAR(15),
  email VARCHAR(255),
  CONSTRAINT Pessoa PRIMARY KEY CLUSTERED(idPessoa ASC)
);
GO

CREATE TABLE PessoaFisica(
  Pessoa_idPessoa INTEGER NOT NULL,
  cpf VARCHAR(11) NOT NULL,
  CONSTRAINT PessoaFisica PRIMARY KEY CLUSTERED(Pessoa_idPessoa ASC),
  CONSTRAINT Pessoa_PessoaFisica FOREIGN KEY(Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
GO

CREATE TABLE PessoaJuridica(
  Pessoa_idPessoa INTEGER NOT NULL,
  cnpj VARCHAR(14) NOT NULL,
  CONSTRAINT PessoaJuridica PRIMARY KEY CLUSTERED(Pessoa_idPessoa ASC),
  CONSTRAINT Pessoa_PessoaJuridica FOREIGN KEY(Pessoa_idPessoa) REFERENCES
Pessoa(idPessoa)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
GO

CREATE TABLE Usuario(
  idUsuario INTEGER NOT NULL IDENTITY,
  loginName VARCHAR(20) NOT NULL,
  senha VARCHAR(20) NOT NULL,
  CONSTRAINT CPK_Usuario PRIMARY KEY CLUSTERED(idUsuario ASC)
);
GO

CREATE TABLE Produto(
  idProduto INTEGER NOT NULL IDENTITY,
  nome VARCHAR(255) NOT NULL,
  quantidade INTEGER,
  precoVenda NUMERIC,
  CONSTRAINT Produto PRIMARY KEY CLUSTERED(idProduto ASC)
);
GO

CREATE TABLE Movimento(
  idMovimento INTEGER  NOT NULL IDENTITY,
  Usuario_idUsuario INTEGER NOT NULL,
  Pessoa_idPessoa INTEGER NOT NULL,
  Produto_idProduto INTEGER NOT NULL,
  quantidade INTEGER,
  tipo CHAR(1),
  precoUnitario NUMERIC,
```

```
  CONSTRAINT Movimento PRIMARY KEY CLUSTERED(idMovimento ASC),
  CONSTRAINT Usuario_Movimento FOREIGN KEY(Usuario_idUsuario) REFERENCES
Usuario(idUsuario)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  CONSTRAINT Pessoa_Movimento FOREIGN KEY(Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  CONSTRAINT Produto_Movimento FOREIGN KEY(Produto_idProduto) REFERENCES
Produto(idProduto)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
GO
```

- script2.sql

```
INSERT INTO Pessoa(idPessoa,nome,endereco,cidade,estado,telefone,email)
  VALUES (NEXT VALUE FOR orderPessoa, 'Alfredo','Rua A, 100','São Paulo','SP','3891-
1000','alfredo@gmail.com'),
    (NEXT VALUE FOR orderPessoa, 'Bruno','Rua B, 200','Rio de Janeiro','RJ','3891-
2000','bruno@gmail.com'),
    (NEXT VALUE FOR orderPessoa, 'Carla','Rua C, 300','Belo Horizonte','MG','3891-
3000','carla@gmail.com'),
    (NEXT VALUE FOR orderPessoa, 'Domingos Supermercado','Rua D,
400','Curitiba','PR','3891-4000','domingos@gmail.com'),
    (NEXT VALUE FOR orderPessoa, 'Escritório 5','Rua E, 500','Vitoria','ES','3891-
5000','escritorio@gmail.com');

INSERT INTO PessoaFisica(Pessoa_idPessoa,cpf)
  VALUES (1,'11111111111'),
    (2,'22222222222'),
    (3,'33333333333');

INSERT INTO PessoaJuridica(Pessoa_idPessoa,cnpj)
  VALUES (4,'44444444444444'),
    (5,'55555555555555');

INSERT INTO Usuario(loginName,senha)
  VALUES ('op1','op1'),
    ('op2','op2');

INSERT INTO Produto(nome,quantidade,precoVenda)
  VALUES ('Arroz',10,'15.00'),
    ('Feijão',20,'8.00'),
    ('Farofa',25,'5.00');

INSERT INTO
Movimento(Usuario_idUsuario,Pessoa_idPessoa,Produto_idProduto,quantidade,tipo,precoUnitar
io)
  VALUES (1,1,1,10,'E',15.00),
    (2,2,2,20,'S',8.00),
    (1,3,3,30,'E',5.00);
```

- script3.sql

```
SELECT p.*, pf.cpf
FROM Pessoa p
INNER JOIN PessoaFisica pf ON p.idPessoa = pf.Pessoa_idPessoa;

SELECT p.*, pj.cnpj
FROM Pessoa p
INNER JOIN PessoaJuridica pj ON p.idPessoa = pj.Pessoa_idPessoa;

SELECT m.*, p.nome as fornecedor, pr.nome as Produto, m.quantidade, m.precoUnitario,
(m.quantidade * m.precoUnitario) as total
FROM Movimento m
INNER JOIN Pessoa p ON p.idPessoa = m.Pessoa_idPessoa
INNER JOIN Produto pr ON pr.idProduto = m.Produto_idProduto
WHERE m.tipo = 'E';

SELECT m.*, p.nome as comprador, pr.nome as Produto, m.quantidade, m.precoUnitario,
(m.quantidade * m.precoUnitario) as total
FROM Movimento m
INNER JOIN Pessoa p ON m.Pessoa_idPessoa = p.idPessoa
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S';

SELECT pr.nome, SUM(m.quantidade * m.precoUnitario) as compras
FROM Movimento m
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'E'
GROUP BY pr.nome;

SELECT pr.nome, SUM(m.quantidade * m.precoUnitario) as vendas
FROM Movimento m
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S'
GROUP BY pr.nome;

SELECT u.*
FROM Usuario u
LEFT JOIN Movimento m ON u.idUsuario = m.Usuario_idUsuario AND m.tipo = 'E'
WHERE m.idMovimento IS NULL;

SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as compras
FROM Movimento m
INNER JOIN Usuario u ON m.Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.loginName;

SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as vendas
FROM Movimento m
INNER JOIN Usuario u ON m.Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.loginName;

SELECT pr.nome, SUM(m.precoUnitario * m.quantidade) / SUM(m.quantidade) as media
FROM Movimento m
INNER JOIN Produto pr ON m.Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S'
GROUP BY pr.nome;
```