Polo Centro – Araraquara – SP

Kervini Ribeiro da Silva

Curso: Desenvolvimento Full Stack

Disciplina: Vamos Manter as Informações?

Turma: 9001

3º Semestre

1º Procedimento - Criando o Banco de Dados

Objetivo: Identificar os requisitos do sistema e modelar o DER do sistema, com base no DER desenvolver a estrutura do banco usando instruções DDL.

Análise e Conclusão:

a. A cardinalidade 1x1 é identificada a partir de uma chave estrangeira (FK) que faz referência a uma chave primária (PK) em outra tabela e essa FK deve fazer parte da PK da tabela, garantindo a unicidade do registro.

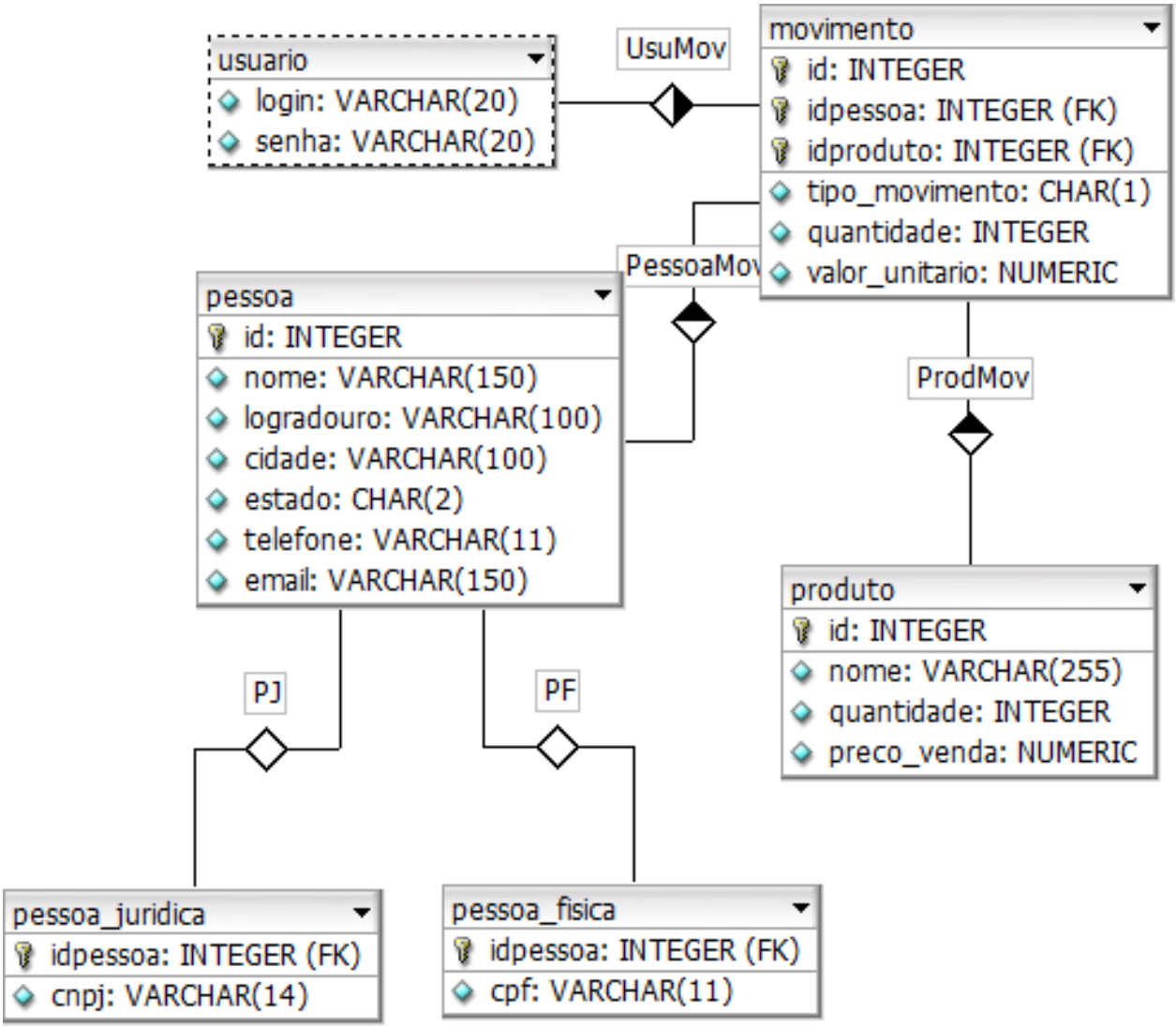
   Já a cardinalidade 1xN, significa que a tabela da esquerda pode ser relacionar com mais de um registro da tabela da direita, então a FK da tabela da direita que faz relação com a da esquerda não deve fazer parte PK sozinha, tem que ou ser parte de uma PK composta ou fora da PK.

   Por fim a cardinalidade NxN significa a possibilidade dos registros, tanto da esquerda quanto da direita, podem se relacionar com mais de um registro do outro lado do relacionamento, portanto apenas a restrição de PK e FK não é suficiente para garantir o relacionamento. Por isso é criada uma terceira tabela com a PK de ambas as tabelas exportadas e servindo como PK da tabela relacional.

b. A herança em um banco relacional é definida por uma tabela exportando sua PK para a tabela mais especifica como uma FK garantindo a relação.

c. O SSMS é uma ferramenta visual, que nos permite visualizar os objetos do banco de dados e manipulá-los de forma simples, com conexão fácil e segura.

GITHUB:https://github.com/Kervini/Mundo3-Nivel2.git


usuario
login: VARCHAR(20)
senha: VARCHAR(20)
UsuMov
movimento
id: INTEGER
idpessoa: INTEGER (FK)
idproduto: INTEGER (FK)
tipo_movimento: CHAR(1)
quantidade: INTEGER
valor_unitario: NUMERIC
PessoaMov
pessoa
id: INTEGER
nome: VARCHAR(150)
logradouro: VARCHAR(100)
cidade: VARCHAR(100)
estado: CHAR(2)
telefone: VARCHAR(11)
email: VARCHAR(150)
ProdMov
produto
id: INTEGER
nome: VARCHAR(255)
quantidade: INTEGER
preco_venda: NUMERIC
PJ
PF
pessoa_juridica
idpessoa: INTEGER (FK)
cnpj: VARCHAR(14)
pessoa_fisica
idpessoa: INTEGER (FK)
cpf: VARCHAR(11)

```sql
CREATE DATABASE loja;
GO
USE loja;
GO
CREATE TABLE produto (
  id INTEGER NOT NULL IDENTITY,
  nome VARCHAR(255) NOT NULL,
  quantidade INTEGER NOT NULL,
  preco_venda NUMERIC(14,2) NOT NULL,
CONSTRAINT pk_produto PRIMARY KEY(id));
GO
CREATE TABLE pessoa (
  id INTEGER NOT NULL, nome VARCHAR(150) NOT NULL, logradouro VARCHAR(100) NOT NULL,
  cidade VARCHAR(100) NOT NULL, estado CHAR(2) NOT NULL, telefone VARCHAR(11) NOT NULL,
  email VARCHAR(150) NOT NULL,
CONSTRAINT pk_pessoa PRIMARY KEY(id));
GO
CREATE TABLE usuario (
  id INTEGER NOT NULL IDENTITY, [login] VARCHAR(20) NOT NULL, senha VARCHAR(20) NOT NULL,
  CONSTRAINT pk_usuario PRIMARY KEY(id));
GO
CREATE TABLE pessoa_juridica (
  idpessoa INTEGER NOT NULL, cnpj VARCHAR(14) NOT NULL,
  CONSTRAINT pk_pessoa_juridica PRIMARY KEY(idpessoa),
  CONSTRAINT fk_pj_pessoa FOREIGN KEY(idpessoa) REFERENCES pessoa(id));
GO
CREATE TABLE pessoa_fisica (
  idpessoa INTEGER NOT NULL, cpf VARCHAR(11) NOT NULL,
  CONSTRAINT pk_pessoa_fisica PRIMARY KEY(idpessoa),
  CONSTRAINT fk_pf_pessoa FOREIGN KEY(idpessoa) REFERENCES pessoa(id));
GO
CREATE TABLE movimento (
  id INTEGER NOT NULL IDENTITY, idusuario INTEGER NOT NULL, idpessoa INTEGER NOT NULL,
  idproduto INTEGER NOT NULL, tipo_movimento CHAR(1) NOT NULL,
  quantidade INTEGER NOT NULL, valor_unitario NUMERIC(14,2) NOT NULL,
  CONSTRAINT pk_movimento PRIMARY KEY(id),
  CONSTRAINT fk_movimento_usuario FOREIGN KEY(idusuario) REFERENCES usuario(id),
  CONSTRAINT fk_movimento_pessoa FOREIGN KEY(idpessoa) REFERENCES pessoa(id),
  CONSTRAINT fk_movimento_produto FOREIGN KEY(idproduto) REFERENCES produto(id),
  CONSTRAINT chk_tipo_movimento CHECK (tipo_movimento = 's' or tipo_movimento = 'e')
 );
GO
CREATE SEQUENCE sqc_pessoa_id
    AS INT START WITH 1 INCREMENT BY 1 ;
GO
```

2 º Procedimento – Alimentando a Base

Objetivo: Alimentar a base definida no procedimento anterior usando instruções DML

Análise e Conclusão:

a. A sequence é uma ferramenta mais poderosa que o identity com mais possibilidades e podendo ser manuseada pelo usuário em vários objetos do banco, já o identity é mais específico e de uso restrito.

b. As FKs permitem que o banco de dados mantenha sua integridade relacional. Os registros são ligados e gerenciados pela FK e isso é um recurso muito importante para garantir a veracidade das informações extraídas do banco.

c. O agrupamento é feito através de junções de tabela usando a instrução JOIN, que faz uso de igualdades entre duas ou mais tabela para criar o plano cartesiano entre as entidades especificadas.

GITHUB:https://github.com/Kervini/Mundo3-Nivel2.git

```sql
INSERT INTO usuario (login, senha)
VALUES ('op1', 'op1'),
       ('op2', 'op2');

INSERT INTO produto (nome, quantidade, preco_venda)
VALUES ('Banana', 100, 5.00),
       ('Laranja', 500, 2.00),
       ('Manga', 800, 4.00);

INSERT INTO pessoa (nome, logradouro, cidade, estado, telefone, email)
VALUES (NEXT VALUE FOR sqc_pessoa_id, 'Joao', 'Rua 12, casa 3, Quitanda', 'Riacho do Sul', 'PA', '1111-1111', 'joao@riacho.com');

INSERT INTO pessoa_fisica (idpessoa, cpf)
VALUES (convert(int, (SELECT current_value FROM sys.sequences WHERE name = 'sqc_pessoa_id')), '11111111111');

INSERT INTO pessoa (nome, logradouro, cidade, estado, telefone, email)
VALUES (NEXT VALUE FOR sqc_pessoa_id, 'JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '1212-1212', 'jjc@riacho.com');

INSERT INTO pessoa_juridica (idpessoa, cnpj)
VALUES (convert(int, (SELECT current_value FROM sys.sequences WHERE name = 'sqc_pessoa_id')), '2222222222222');

INSERT INTO movimento (idusuario, idpessoa, idproduto, tipo_movimento, quantidade, valor_unitario)
VALUES (1, 1, 1, 's', 20, 5.00),
       (1, 1, 2, 's', 15, 2.00),
       (2, 1, 2, 's', 10, 2.00),
       (1, 2, 2, 'e', 15, 5.00),
       (1, 2, 3, 'e', 20, 4.00);

SELECT p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pf.cpf
FROM pessoa p
INNER JOIN pessoa_fisica pf ON p.id = pf.idpessoa;

SELECT p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pj.cnpj
FROM pessoa p
INNER JOIN pessoa_juridica pj ON p.id = pj.idpessoa;
```

```sql
SELECT p.nome, p2.nome AS fornecedor, m.quantidade, m.valor_unitario,
       (m.valor_unitario * m.quantidade) AS valor_total
FROM movimento m
INNER JOIN produto p ON m.idproduto = p.id
INNER JOIN pessoa p2 ON m.idpessoa = p2.id
INNER JOIN pessoa_juridica pj ON p2.id = pj.idpessoa
WHERE tipo_movimento = 'e';

SELECT p.nome, p2.nome AS comprador, m.quantidade,m.valor_unitario,
    (m.valor_unitario * m.quantidade) AS valor_total
FROM movimento m
INNER JOIN produto p ON m.idproduto = p.id
INNER JOIN pessoa p2 ON m.idpessoa = p2.id
INNER JOIN pessoa_fisica pf ON p2.id = pf.idpessoa
WHERE tipo_movimento = 's';

SELECT p.id, p.nome, sum(m.valor_unitario * m.quantidade) AS 'valor total'
FROM movimento m
INNER JOIN produto p ON m.idproduto = p.id
WHERE tipo_movimento = 'e'
GROUP BY p.id, p.nome;

SELECT p.id, p.nome, sum(m.valor_unitario * m.quantidade) AS 'valor total'
FROM movimento m
INNER JOIN produto p ON m.idproduto = p.id
WHERE tipo_movimento = 's'
GROUP BY p.id, p.nome;

SELECT id, [login]
FROM usuario
WHERE id not in
    (SELECT DISTINCT idusuario FROM movimento WHERE tipo_movimento = 'e');

SELECT u.id, u.[login], sum(m.valor_unitario * m.quantidade) AS 'valor total'
FROM movimento m
INNER JOIN usuario u ON m.idusuario = u.id
WHERE tipo_movimento = 'e'
GROUP BY u.id, u.[login];
```

```sql
SELECT u.id, u.[login], sum(m.valor_unitario * m.quantidade) AS 'valor total'
FROM movimento m
INNER JOIN usuario u ON m.idusuario = u.id
WHERE tipo_movimento = 's'
GROUP BY u.id, u.[login];

SELECT avg(valor_unitario) AS 'valor medio'
FROM movimento
WHERE tipo_movimento = 's'
GROUP BY idproduto
```