Centro Universitário Estácio do Ceará - Campus Centro

Curso: Desenvolvimento Full Stack

Disciplina: Nível 1: Iniciando o Caminho Pelo Java

Número da Turma: RPG0015

Semestre Letivo: 3

Integrantes: Fernando Rocha Fonteles Filho

Repositorio Git: https://github.com/fernandorff/EstacioFullStack_Mundo3_Nivel2

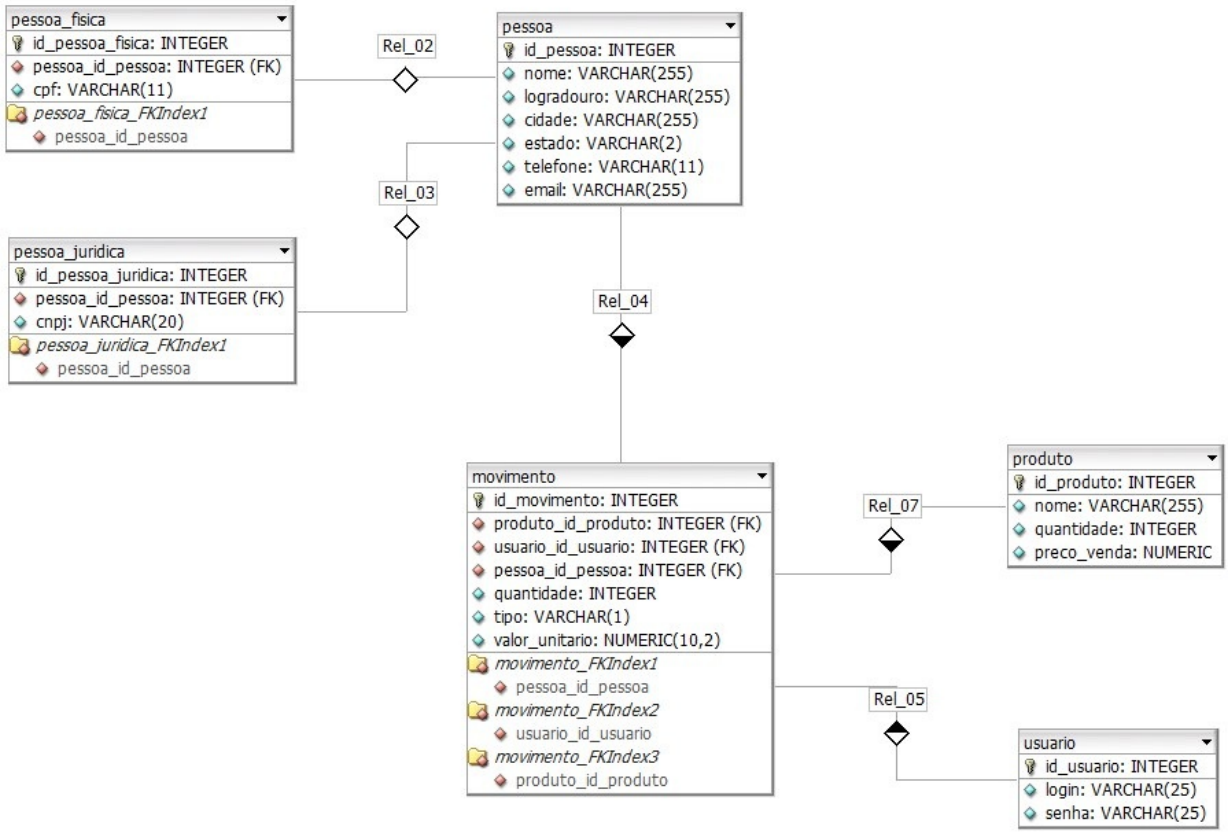
Título da Prática: 1º Procedimento | Criando o Banco de Dados

Objetivos 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.

Códigos solicitados neste roteiro de aula:

- Tabelas relacionadas no DBDesigner



- Esquema de criação das tabelas do banco de dados

```
CREATE TABLE pessoa (
  id_pessoa INTEGER IDENTITY NOT NULL PRIMARY KEY,
  nome VARCHAR(255) NULL,
  logradouro VARCHAR(255) NULL,
  cidade VARCHAR(255) NULL,
  estado CHAR(2) NULL,
  telefone VARCHAR(11) NULL,
  email VARCHAR(255) NULL
);

CREATE TABLE pessoa_fisica (
  id_pessoa_fisica INTEGER IDENTITY NOT NULL PRIMARY KEY,
  pessoa_id_pessoa INTEGER NOT NULL CHECK (pessoa_id_pessoa > 0),
  cpf VARCHAR(11) NULL
);

CREATE TABLE pessoa_juridica (
  id_pessoa_juridica INTEGER IDENTITY NOT NULL PRIMARY KEY,
  pessoa_id_pessoa INTEGER NOT NULL CHECK (pessoa_id_pessoa > 0),
  cnpj VARCHAR(20) NULL
);

CREATE TABLE movimento (
  id_movimento INTEGER IDENTITY NOT NULL PRIMARY KEY,
  pessoa_id_pessoa INTEGER NOT NULL CHECK (pessoa_id_pessoa > 0),
```

```
  produto_id_produto INTEGER NOT NULL CHECK (produto_id_produto > 0),
  usuario_id_usuario INTEGER NOT NULL CHECK (usuario_id_usuario > 0),
  quantidade INTEGER NULL CHECK (quantidade > 0),
  tipo CHAR(1) NULL,
  valor_unitario NUMERIC(10, 2) NULL
);

CREATE TABLE produto (
  id_produto INTEGER IDENTITY NOT NULL PRIMARY KEY,
  nome VARCHAR(255) NULL,
  quantidade INTEGER NULL CHECK (quantidade > 0),
  precoVenda NUMERIC(10, 2) NULL
);

CREATE TABLE usuario (
  id_usuario INTEGER IDENTITY NOT NULL PRIMARY KEY,
  login VARCHAR(25) NULL,
  senha VARCHAR(25) NULL
);

CREATE INDEX pessoa_fisica_FKIndex1 ON pessoa_fisica(pessoa_id_pessoa);

CREATE INDEX pessoa_juridica_FKIndex1 ON pessoa_juridica(pessoa_id_pessoa);

CREATE INDEX movimento_FKIndex1 ON movimento(usuario_id_usuario);
CREATE INDEX movimento_FKIndex2 ON movimento(produto_id_produto);
CREATE INDEX movimento_FKIndex3 ON movimento(pessoa_id_pessoa);
```

Análise e Conclusão

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

As diferentes cardinalidades são implementadas em um banco de dados relacional de acordo com as regras de normalização. As regras de normalização são um conjunto de princípios que ajudam a garantir a integridade e a consistência dos dados em um banco de dados relacional.

- Cardinalidade 1X1

A cardinalidade 1X1 é implementada em um banco de dados relacional fundindo as duas entidades envolvidas no relacionamento em uma única tabela. Isso ocorre porque cada elemento de uma entidade A se relaciona com um e somente um elemento de outra entidade B.

- Cardinalidade 1XN ou N1

A cardinalidade 1XN ou N1 é implementada em um banco de dados relacional adicionando uma chave estrangeira à tabela que possui cardinalidade mínima 1. A chave estrangeira é um atributo que faz referência à chave primária da tabela que possui cardinalidade máxima N.

- Cardinalidade NXN

A cardinalidade NXN é implementada em um banco de dados relacional criando uma nova entidade para armazenar dados das entidades que estão se relacionando inicialmente. Essa nova entidade é chamada de entidade associativa.

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

O tipo de relacionamento que deve ser utilizado para representar o uso de herança em bancos de dados relacionais é o relacionamento de supertipo/subtipo. Esse relacionamento é uma relação de generalização/especialização, onde uma entidade supertipo é um conjunto mais geral de entidades, e uma entidade subtipo é um conjunto mais específico de entidades.

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

O SSMS oferece uma variedade de recursos que podem ajudar os usuários a realizar tarefas de forma mais rápida e eficiente, incluindo:

- Visualização e gerenciamento de dados: O SSMS fornece uma variedade de ferramentas para visualizar e gerenciar dados, incluindo tabelas, views, procedimentos armazenados e outros objetos de banco de dados.
- Gerenciamento de banco de dados: O SSMS fornece uma variedade de ferramentas para gerenciar bancos de dados, incluindo criação, alteração e exclusão de bancos de dados, além de gerenciamento de usuários e permissões.
- Gerenciamento de objetos: O SSMS fornece uma variedade de ferramentas para gerenciar objetos de banco de dados, incluindo criação, alteração e exclusão de objetos, além de gerenciamento de dependências.

Título da Prática: 2º Procedimento | Alimentando a Base

- Esquema de inserção de dados nas tabelas

```sql
-- 1. b. dados completos do usuario

insert into usuario (login, senha)
VALUES ('op1', 'op1'), ('op2', 'op2');


-- 1. c. dados completos do produto

insert into usuario (nome, quantidade, precoVenda)
VALUES ('Banana', 100, 5.00), ('Laranja', 500, 2.00), ('Manga', 800, 4.00);


-- 2. b. Incluir na tabela pessoa os dados comuns

INSERT INTO pessoa (nome, logradouro, cidade, estado, telefone, email)
VALUES
('João da Silva', 'Rua das Flores', 'São Paulo', 'SP', '11999999999',
'joao.silva@email.com'),
('Maria da Silva', 'Rua dos Pinheiros', 'São Paulo', 'SP', '11999999999',
'maria.silva@email.com'),
('Empresa XPTO', 'Rua dos Negócios', 'São Paulo', 'SP', '11999999999',
'empresa.xpto@email.com');


-- 2. c. Incluir em pessoa física o CPF, efetuando o relacionamento com pessoa.

INSERT INTO pessoa_fisica (pessoa_id_pessoa, cpf)
VALUES (1, '12345678900'), (2, '98765432100');


-- 2. d. Incluir em pessoa jurídica o CNPJ, relacionando com pessoa.

INSERT INTO pessoa_juridica (pessoa_id_pessoa, cnpj)
VALUES (3, '12345678000101');
```

- Esquema de buscas na tabelas

```sql
-- 4. a. Dados completos de pessoas físicas

SELECT P.id_pessoa, P.nome, P.logradouro, P.cidade, P.estado, P.telefone,
P.email, PF.cpf
FROM Pessoa AS P
INNER JOIN pessoa_fisica AS PF ON P.id_pessoa = PF.id_pessoa;


-- 4. b. Dados completos de pessoas jurídicas

SELECT P.id_pessoa, P.nome, P.logradouro, P.cidade, P.estado, P.telefone,
P.email, PJ.cnpj
FROM Pessoa AS P
INNER JOIN pessoa_juridica AS PJ ON P.id_pessoa = PJ.id_pessoa;
```

```sql
-- 4. c. Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total

SELECT M.tipo AS tipo_movimentacao, P.nome AS Produto, F.nome AS Fornecedor, M.quantidade AS Quantidade, M.valor_unitario AS preco_unitario, (M.quantidade * M.valor_unitario) AS valor_total
FROM Movimento AS M
INNER JOIN Pessoa AS F ON M.id_pessoa = F.id_pessoa
INNER JOIN Produto AS P ON M.id_produto = P.id_produto
WHERE M.tipo = 'E';


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

SELECT M.tipo AS tipo_movimentacao, P.nome AS Produto, PF.nome AS Comprador, M.quantidade AS Quantidade, M.valor_unitario AS preco_unitario, (M.quantidade * M.valor_unitario) AS valor_total
FROM Movimento AS M
INNER JOIN Pessoa AS PF ON M.id_pessoa = PF.id_pessoa
INNER JOIN Produto AS P ON M.id_produto = P.id_produto
WHERE M.tipo = 'S';


-- 4. e. Valor total das entradas agrupadas por produto

SELECT P.nome AS produto_entrada,
    SUM(M.quantidade) AS quantidade_total,
    SUM(M.quantidade * M.valor_unitario) AS valor_total
FROM Movimento AS M
INNER JOIN Produto AS P ON M.id_produto = P.id_produto
WHERE M.tipo = 'E'
GROUP BY P.nome;


-- 4. f. Valor total das saídas agrupadas por produto

SELECT P.nome AS produto_saida,SUM(M.quantidade) AS quantidade_total,
SUM(M.quantidade * M.valor_unitario) AS valor_total
FROM Movimento AS M
INNER JOIN Produto AS P ON M.id_produto = P.id_produto
WHERE M.tipo = 'S'
GROUP BY P.nome;


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

SELECT U.login AS Operador
FROM Usuario AS U
WHERE NOT EXISTS (SELECT 1 FROM Movimento AS M WHERE M.Usuario_id_usuario = U.id_usuario AND M.tipo = 'E');
```

```sql
-- 4. h. Valor total de entrada, agrupado por operador

SELECT U.login AS operador_entrada, SUM(M.quantidade) AS quantidade_total,
SUM(M.quantidade * M.valor_unitario) AS valor_total
FROM Movimento AS M
INNER JOIN Usuario AS U ON M.id_usuario = U.id_usuario
WHERE M.tipo = 'E'
GROUP BY U.login;


-- 4. i. Valor total de saída, agrupado por operador

SELECT U.login AS operador_saida, SUM(M.quantidade) AS quantidade_total,
SUM(M.quantidade * M.valor_unitario) AS valor_total
FROM Movimento AS M
INNER JOIN Usuario AS U ON M.id_usuario = U.id_usuario
WHERE M.tipo = 'S'
GROUP BY U.login;


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

SELECT P.nome AS produto_saida, SUM(M.quantidade) AS quantidade_total,
SUM(M.quantidade * M.valor_unitario) / SUM(M.quantidade) AS valor_ponderado,
SUM(M.quantidade * M.valor_unitario) AS valor_total
FROM Movimento AS M
INNER JOIN Pessoa AS PF ON M.id_pessoa = PF.id_pessoa
INNER JOIN Produto AS P ON M.id_produto = P.id_produto
WHERE M.tipo = 'S'
GROUP BY P.nome;
```

Análise e Conclusão

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

A principal diferença entre eles é que sequence é um objeto de banco de dados, enquanto identity é uma propriedade de uma coluna.

Sequence

- É um objeto de banco de dados que armazena um valor sequencial.
- Pode ser usado em qualquer coluna, não apenas em colunas de tipo inteiro.
- Pode ser usado para gerar valores sequenciais para várias colunas.
- Pode ser reinicializado ou revertido.

Identity

- É uma propriedade de uma coluna que permite que o banco de dados gere valores sequenciais para essa coluna.
- Só pode ser usado em colunas de tipo inteiro.
- Só pode ser usado para gerar valores sequenciais para uma coluna.

- Não pode ser reinicializado ou revertido.

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

As chaves estrangeiras são importantes para a consistência do banco de dados porque ajudam a garantir que os dados nas tabelas relacionadas sejam consistentes. Isso é feito exigindo que o valor de uma chave estrangeira corresponda ao valor da chave primária da tabela relacionada.

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

Os operadores do SQL que pertencem à álgebra relacional são:

- SELECT
- PROJECT
- JOIN:
- UNION
- INTERSECT
- EXCEPT

Os operadores do SQL que pertencem ao cálculo relacional são:

- COUNT
- MAX
- MIN
- SUM
- AVG

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

O agrupamento em consultas SQL é feito usando a cláusula GROUP BY. A cláusula GROUP BY especifica as colunas que serão usadas para agrupar os resultados da consulta.

O requisito obrigatório para o agrupamento é que a cláusula GROUP BY contenha pelo menos uma coluna. Se a cláusula GROUP BY não conter nenhuma coluna, a consulta retornará uma tabela com uma única linha e uma única coluna, contendo o valor da função de agregação na linha.