



Missão Prática | Nível 2 | Mundo 3

Lorena Rosa Borges Sanches - Matrícula: 202204376067

Estácio Campus Uberlândia/MG

Vamos manter a informações! - 2023.2 – 3º Semestre Letivo

Desenvolvimento Full Stack

Link do Git: <https://github.com/LorenaBorgesSanches/Mundo3-Pratica2>

Objetivo da Prática

O propósito do trabalho a seguir é acompanhar a atividade prática elaborada como requisito da disciplina “RPG0015”. Nos tópicos a seguir serão apresentados temas referente a criação de banco de dados. O estudo é dividido em duas partes, o primeiro procedimento é referente a criação do banco de dados, e o segundo procedimento, se trata de alimentar a base, ou seja, as tabelas.

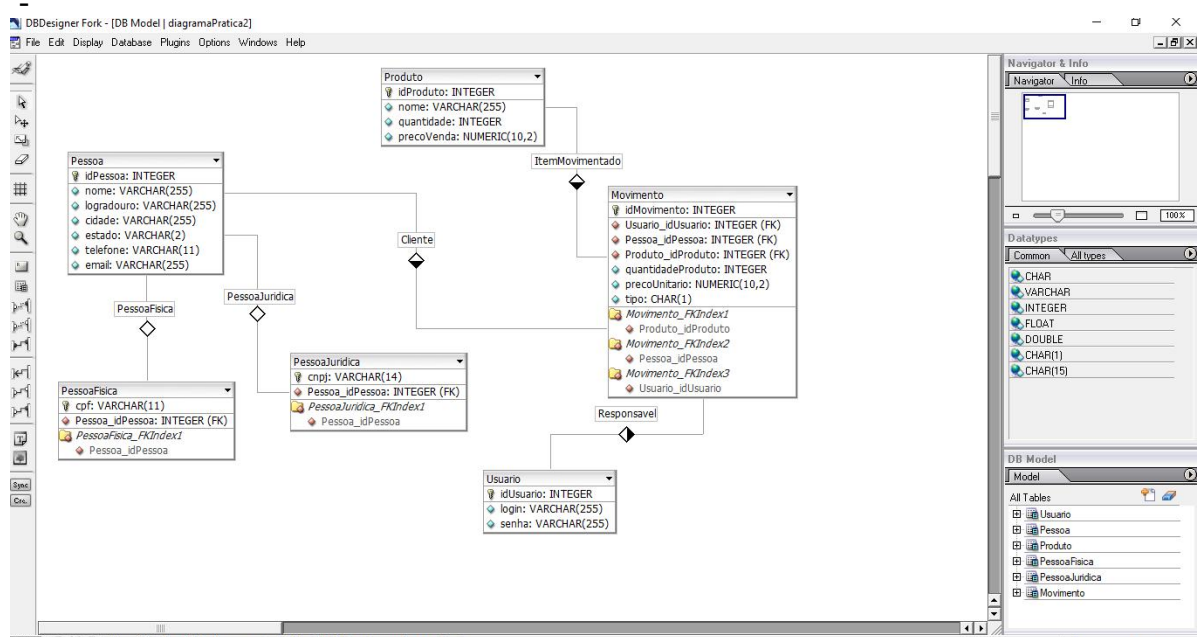
Em cada um dos procedimentos, terão prints sobre a prática, em que foi usado o software DBDesigner Fork e o SQL Server Management Studio, para modelação e criação do banco de dados.

E para finalizar, serão respondidas questões elaboradas pelo tutor da disciplina, afim de justificar os conceitos utilizados.

1º Procedimento – Criando o Banco de Dados

Tópico 1 - É referente baixar/installar a ferramenta de modelagem, DBDesigner Fork.

Tópico 2: Modelação dos dados



Tópico 3 -

A.

The screenshot shows the 'Conectar ao Mecanismo de Banco de Dados' (Connect to Database Mechanism) dialog box for SQL Server. The fields are filled as follows:

- Tipo de servidor: Mecanismo de Banco de Dados
- Nome do servidor: DESKTOP-8PQGG84
- Autenticação: Autenticação do SQL Server
- Logon: loja
- Senha: (masked with asterisks)
- ☐ Lembrar senha

Buttons at the bottom: Conectar, Cancelar, Ajuda, Opções >>.

B.

C.

```
CREATE TABLE Produto (  
    idProduto INTEGER NOT NULL IDENTITY ,  
    nome VARCHAR(255) NOT NULL ,  
    quantidade INTEGER NOT NULL ,  
    precoVenda NUMERIC(10,2) NOT NULL ,  
PRIMARY KEY(idProduto));  
GO
```

```
CREATE TABLE Pessoa (  
    idPessoa INTEGER NOT NULL ,  
    nome VARCHAR(255) NOT NULL ,  
    logradouro VARCHAR(255) ,  
    cidade VARCHAR(255) ,  
    estado VARCHAR(2) ,  
    telefone VARCHAR(11) ,  
    email VARCHAR(255) NOT NULL ,  
PRIMARY KEY(idPessoa));  
GO
```

```
CREATE TABLE Usuario (  
    idUsuario INTEGER NOT NULL IDENTITY ,  
    login VARCHAR(255) NOT NULL ,  
    senha VARCHAR(255) NOT NULL ,  
PRIMARY KEY(idUsuario));  
GO
```

```
CREATE TABLE PessoaJuridica (  
    cnpj VARCHAR(14) NOT NULL ,  
    Pessoa_idPessoa INTEGER NOT NULL ,  
PRIMARY KEY(cnpj) ,  
FOREIGN KEY(Pessoa_idPessoa)  
REFERENCES Pessoa(idPessoa));  
GO
```

```
CREATE INDEX PessoaJuridica_FKIndex1 ON PessoaJuridica (Pessoa_idPessoa);  
GO
```

```
CREATE INDEX IFK_PessoaJuridica ON PessoaJuridica (Pessoa_idPessoa);  
GO
```

```

CREATE TABLE PessoaFisica (
    cpf VARCHAR(11) NOT NULL ,
    Pessoa_idPessoa INTEGER NOT NULL ,
    PRIMARY KEY(cpf) ,
    FOREIGN KEY(Pessoa_idPessoa)
        REFERENCES Pessoa(idPessoa));
GO

CREATE INDEX PessoaFisica_FKIndex1 ON PessoaFisica (Pessoa_idPessoa);
GO

CREATE INDEX IFK_PessoaFisica ON PessoaFisica (Pessoa_idPessoa);
GO

CREATE TABLE Movimento (
    idMovimento INTEGER NOT NULL IDENTITY ,
    Usuario_idUsuario INTEGER NOT NULL ,
    Pessoa_idPessoa INTEGER NOT NULL ,
    Produto_idProduto INTEGER NOT NULL ,
    quantidadeProduto INTEGER NOT NULL ,
    precoUnitario NUMERIC(10,2) NOT NULL ,
    tipo CHAR(1) NOT NULL ,
    PRIMARY KEY(idMovimento) ,
    FOREIGN KEY(Produto_idProduto)
        REFERENCES Produto(idProduto),
    FOREIGN KEY(Pessoa_idPessoa)
        REFERENCES Pessoa(idPessoa),
    FOREIGN KEY(Usuario_idUsuario)
        REFERENCES Usuario(idUsuario));
GO

CREATE INDEX Movimento_FKIndex1 ON Movimento (Produto_idProduto);
GO
CREATE INDEX Movimento_FKIndex2 ON Movimento (Pessoa_idPessoa);
GO
CREATE INDEX Movimento_FKIndex3 ON Movimento (Usuario_idUsuario);
GO

CREATE INDEX IFK_ItemMovimentado ON Movimento (Produto_idProduto);
GO
CREATE INDEX IFK_Cliente ON Movimento (Pessoa_idPessoa);
GO
CREATE INDEX IFK_Responsavel ON Movimento (Usuario_idUsuario);
GO

```


D.

```

CREATE SEQUENCE idPessoaSequence
START WITH 1
INCREMENT BY 1;
GO

```

E.

 criacaoBancoLoja.sql

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

Relacionamento 1X1: é implementado gerando uma chave estrangeira da tabela origem para destino e vice e versa. Por exemplo, dada as tabelas Funcionário e

Notebook, sendo que um Notebook pertencerá apenas a um Funcionário e um Funcionário terá apenas um Notebook. Cria-se uma chave estrangeira de Funcionário em Notebook, e uma chave estrangeira de Notebook em Funcionário.

Relacionamento 1XN: é implementado com apenas uma chave estrangeira, por exemplo, dada as tabelas Empresa e Funcionário, em que, uma empresa pode conter N Funcionários. Neste caso, cria-se uma chave estrangeira de empresa na tabela de Funcionário.

Relacionamento NXN: é implementado através da criação de uma tabela intermediária, contendo chaves estrangeiras das duas tabelas relacionadas. Por exemplo, dada as tabelas Curso e Disciplina, em que um Curso pode conter N disciplinas e uma Disciplina pode estar em N Cursos, cria-se uma tabela Curso_Disciplina contendo as chaves estrangeiras de ambas as tabelas.

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

É mais comum utilizar o relacionamento 1X1 para representar herança em bancos de dados relacionais. Embora, outras abordagens, como a de tabela única também seja possível.

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

O SQLSMS fornece uma interface gráfica amigável, permitindo configurações, consultas, e várias tarefas relacionadas ao contexto do banco de dados sem necessidade de escrever linhas de comando. E também possui um assistente na construção de consultas, sugerindo nomes de tabelas, campos, etc, para que o usuário não precise decorar toda a estrutura de seus bancos de dados. Dessa forma otimiza a produtividade.

Além disso, permite criação e configuração de usuários e papéis, delegação de acesso e várias funcionalidades referentes a parte de acesso e segurança da informação do banco.

2º Procedimento – Alimentando a Base

Tópico 1 -

A. Definição do software a ser utilizado, sendo o SQL Server Management Studio o definido pelo tutor.

B

```
insert into Usuario (login, senha) values ('op1', 'op1');  
insert into Usuario (login, senha) values ('op2', 'op2');
```

C.

```
insert into Produto (nome, quantidade, precoVenda) values  
( 'banana', '100', '5.00'),  
( 'laranja', '500', '2.00'),  
( 'manga', '800', '4.00'),  
( 'maça', '100', '5.70');
```

Tópico 2 -

A.

```
DECLARE @NextValue INT  
SELECT @NextValue = NEXT VALUE FOR idPessoaSequence
```

B.

```
insert into Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email) values  
(@NextValue, 'João', 'Rua 12, casa 3, Quitanda', 'Riacho do Sul', 'PA', '1111-1111', 'joao@riacho.com');
```

C.

```
insert into PessoaFisica(Pessoa_idPessoa, cpf) values  
(@NextValue, '12345678900');
```

D.

```
insert into PessoaJuridica(Pessoa_idPessoa, cnpj) values
(@NextValue, '22222222222');
```

Tópico 3 -

```
insert into Movimento (Usuario_idUsuario, Pessoa_idPessoa, Produto_idProduto, quantidadeProduto, tipo, precoUnitario) values
(1, 2, 1, 20, 'S', 4.00),
(2, 2, 2, 15, 'S', 2.00),
(2, 2, 1, 10, 'S', 3.00),
(1, 3, 2, 15, 'E', 5.00),
(1, 3, 4, 20, 'E', 4.00);
```

Tópico 4 -

A.

```
--Exercicio 4
--letra a:
select * from Pessoa
inner join PessoaFisica on (Pessoa.idPessoa = PessoaFisica.Pessoa_idPessoa);
```

B.

```
--letra b:
select * from Pessoa
inner join PessoaJuridica on (Pessoa.idPessoa = PessoaJuridica.Pessoa_idPessoa);
```

C.

```
--letra c:
select
    p.nome,
    pe.nome,
    m.quantidadeProduto,
    m.precoUnitario,
    m.quantidadeProduto * m.precoUnitario as ValorTotal
from Movimento as m
inner join Produto as p on (m.Produto_idProduto = p.idProduto)
inner join Pessoa as pe on (m.Pessoa_idPessoa = pe.idPessoa)
where tipo = 'E'
```

D.

```
--letra d:
select
    p.nome,
    pe.nome,
    m.quantidadeProduto,
    m.precoUnitario,
    m.quantidadeProduto * m.precoUnitario as ValorTotal
from Movimento as m
inner join Produto as p on (m.Produto_idProduto = p.idProduto)
inner join Pessoa as pe on (m.Pessoa_idPessoa = pe.idPessoa)
where tipo = 'S'
```

E.

```
--letra e:
select p.nome, sum(quantidadeProduto * precoUnitario) as ValorTotal
from Movimento as m
inner join Produto as p on (m.Produto_idProduto = p.idProduto)
where m.tipo = 'E'
group by p.nome
```

F.


```

--letra f:
select p.nome, sum(quantidadeProduto * precoUnitario) as ValorTotal
from Movimento as m
inner join Produto as p on (m.Produto_idProduto = p.idProduto)
where m.tipo = 'S'
group by p.nome

```

G.

```

--letra g:
select u.idUsuario, u.login from
(select idUsuario from Usuario
EXCEPT
select Usuario_idUsuario from Movimento as m
where m.tipo = 'E') as UserSemMovimento
inner join Usuario as u on (u.idUsuario = UserSemMovimento.idUsuario)

```

H.

```

--letra h:
select u.login, sum(quantidadeProduto * precoUnitario) as ValorTotal
from Movimento as m
inner join Usuario as u on (m.Usuario_idUsuario = u.idUsuario)
where m.tipo = 'E'
group by u.login

```

I.

```

--letra i:
select u.login, sum(quantidadeProduto * precoUnitario) as ValorTotal
from Movimento as m
inner join Usuario as u on (m.Usuario_idUsuario = u.idUsuario)
where m.tipo = 'S'
group by u.login

```

J.

```

--letra j:
select p.nome, sum(quantidadeProduto * precoUnitario)/sum(quantidadeProduto) as ValorTotal
from Movimento as m
inner join Produto as p on (m.Produto_idProduto = p.idProduto)
where m.tipo = 'S'
group by p.nome

```

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

O identity é gerenciado pelo próprio banco de dados, gerando um novo elemento a cada inserção na tabela em que está relacionado. Pertence a uma coluna de tabela e não pode ser compartilhado com outras, desta forma, cada tabela que possui identity terá controle individual dos elementos gerados.

O identity é criado para uma coluna específica de uma tabela, e gera novos valores sequenciais e únicos a cada inserção na tabela, enquanto a sequence, é criada para o banco de dados, e por isso, não se prende a uma coluna específica. Ao contrário do identity, um novo elemento de uma sequence tem que ser solicitado explicitamente antes da inserção de um novo elemento na tabela. Mas em contrapartida, um elemento de sequence pode ser utilizado em mais de uma inserção em tabelas distintas.

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

As chaves estrangeiras possuem um papel fundamental na consistência do banco de dados. Elas garantem que os relacionamentos entre tabelas sejam íntegros, pois, para uma chave estrangeira existir, ela necessariamente deve ser uma chave primária de uma outra tabela e seus dados devem ser únicos.

Possuem também, restrições no quesito de inserção e exclusão de dados, portanto, quando for criar ou alterar a chave estrangeira, o sistema vai conferir se ela existe como chave primária em outra tabela, e quando for excluir, chave primária que possui chave estrangeira que a referencia, deverá impedir a exclusão, uma vez geraria inconsistência haver chave estrangeira sem “pai”. E o sistema alertará sobre problema na integridade referencial.

Outra importante característica da chave estrangeira, está no fato de que ela evita redundância de dados. Se a chave estrangeira existe, então pode se afirmar que os valores dela são consistentes em todas as tabelas relacionadas, e ao efetuar alterações, também irá garantir que todas as tabelas sejam atualizadas de maneira coerente.

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

Os operadores Projection, Select, Union, Intersection, Difference, Divide, Product e Join, pertencem a álgebra relacional.

Os operadores descritos acima pertencem a álgebra relacional e ao cálculo relacional, que são essencialmente os mesmos.

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

O agrupamento em consultas é realizado através de uma função de agregação (Count, AVG, Sum, Max, etc) e da inclusão de uma cláusula de Group By, sendo obrigatório que todas as colunas a serem exibidas e agrupadas, devem aparecer na cláusulas de Group By.

Conclusão

Os conceitos utilizados na Missão Prática foram importantes para o aprendizado sobre banco de dados. Desde a criação do diagrama até a criação do banco de dados. Obviamente, há muito o que ser aprofundado nesse tema tão complexo. No entanto, para o primeiro contato com a disciplina, se faz importante aprender sobre o básico.