	UNIVERSIDADE ESTÁCIO DE SÁ POLO CENTRO – SANTA ROSA – RS **DESENVOLVIMENTO FULL STACK** Relatório da Missão Prática Nível 2 – Mundo 3
Disciplina:	Vamos Manter as Informações?
Aluno/Matrícula:	Anderson Rech - 202304442215
Repositório:	https://github.com/4nderech/Mundo3-Missao-Pratica-N2.git

RPG0015 - Vamos manter as informações!

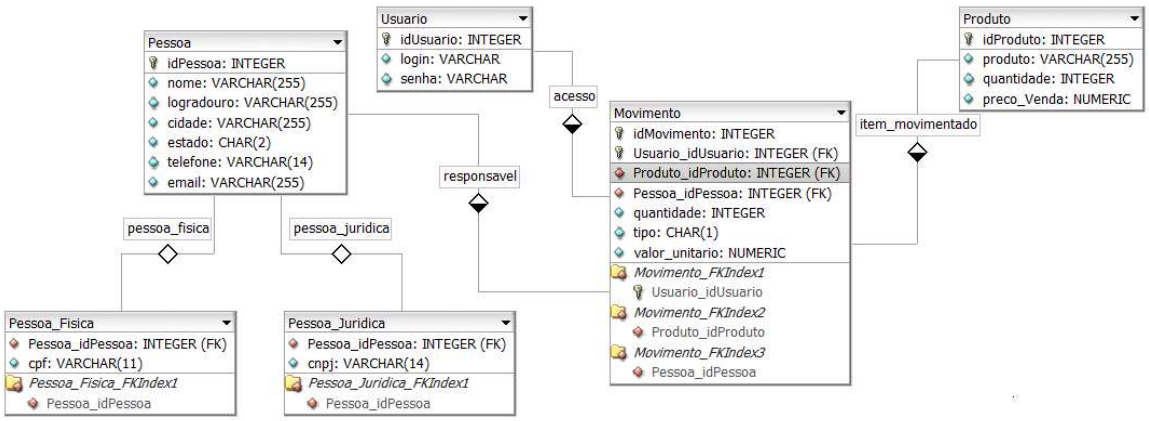
1. Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

2. Objetivos 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.

3. 1º Procedimento | Criando o Banco de Dados

Modelagem do Banco de Dados no DBDesigner Fork:



Script completo para criação do Banco de Dados:

SQLQuery1.sql - AND...PC.Loja (loja (53))*

```
use Loja;
GO

CREATE TABLE [Pessoa] (
    idPessoa Integer NOT NULL IDENTITY,
    nome varchar(255) NOT NULL,
    logradouro varchar(255) NOT NULL,
    cidade varchar(255) NOT NULL,
    estado char(2) NOT NULL,
    telefone varchar(14) NOT NULL,
    email varchar(255) NOT NULL,
    CONSTRAINT [PK_PESSOA] PRIMARY KEY CLUSTERED ([idPessoa] ASC)
)
GO


CREATE TABLE [Pessoa_Fisica] (
    idPessoa Integer NOT NULL,
    cpf varchar(11) NOT NULL,
    CONSTRAINT [PK_PESSOA_FISICA] PRIMARY KEY CLUSTERED ([idPessoa] ASC),
    CONSTRAINT [FK_PESSOA_FISICA_PESSOA] FOREIGN KEY ([idPessoa]) REFERENCES [Pessoa] ([idPessoa])
    ON UPDATE CASCADE
    ON DELETE CASCADE
)
GO

CREATE TABLE [Pessoa_Juridica] (
    idPessoa Integer NOT NULL,
    cnpj varchar(14) NOT NULL,
    CONSTRAINT [PK_PESSOA_JURIDICA] PRIMARY KEY CLUSTERED ([idPessoa] ASC),
    CONSTRAINT [FK_PESSOA_JURIDICA_PESSOA] FOREIGN KEY ([idPessoa]) REFERENCES [Pessoa] ([idPessoa])
    ON UPDATE CASCADE
    ON DELETE CASCADE
)
GO

CREATE TABLE [Usuario] (
    idUsuario Integer NOT NULL IDENTITY,
    login varchar(20) NOT NULL,
    senha varchar(20) NOT NULL,
    CONSTRAINT [PK_USUARIO] PRIMARY KEY CLUSTERED ([idUsuario] ASC)
)
GO

CREATE TABLE [Produto] (
    idProduto Integer NOT NULL IDENTITY,
    nome varchar(255) NOT NULL,
    quantidade Integer NOT NULL,
    preco_Venda numeric NOT NULL,
    CONSTRAINT [PK_PRODUTO] PRIMARY KEY CLUSTERED ([idProduto] ASC)
)
GO

CREATE TABLE [Movimento] (
    idMovimento Integer NOT NULL IDENTITY,
    idPessoa Integer NOT NULL,
    idProduto Integer NOT NULL,
    idUsuario Integer NOT NULL,
    quantidade Integer NOT NULL,
    tipo char(1) NOT NULL,
    valor_unitario numeric NOT NULL,
    CONSTRAINT [PK_MOVIMENTO] PRIMARY KEY CLUSTERED ([idMovimento] ASC)
)
GO

ALTER TABLE [Movimento] WITH CHECK ADD CONSTRAINT [Movimento_fk0] FOREIGN KEY ([idPessoa])
REFERENCES [Pessoa] ([idPessoa])
ON UPDATE CASCADE
ON DELETE CASCADE
GO
ALTER TABLE [Movimento] CHECK CONSTRAINT [Movimento_fk0]
GO

ALTER TABLE [Movimento] WITH CHECK ADD CONSTRAINT [Movimento_fk1] FOREIGN KEY ([idProduto])
REFERENCES [Produto] ([idProduto])
ON UPDATE CASCADE
GO
ALTER TABLE [Movimento] CHECK CONSTRAINT [Movimento_fk1]
GO

ALTER TABLE [Movimento] WITH CHECK ADD CONSTRAINT [Movimento_fk2] FOREIGN KEY ([idUsuario])
REFERENCES [Usuario] ([idUsuario])
ON UPDATE CASCADE
GO
ALTER TABLE [Movimento] CHECK CONSTRAINT [Movimento_fk2]
GO
```

4. Análise e Conclusão:

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

Relacionamento 1x1 (Um para Um) é implementado utilizando uma chave primária em ambas as tabelas, onde a chave primária de uma tabela se torna a chave estrangeira (com uma restrição UNIQUE) na outra tabela, garantindo que cada linha de uma tabela corresponda a no máximo uma linha na outra.

Relacionamento 1xN (Um para Muitos) é implementado utilizando uma chave primária na tabela do lado "um" e uma chave estrangeira na tabela do lado "muitos". A chave estrangeira na tabela do lado "muitos" referencia a chave primária da tabela do lado "um", permitindo que uma linha na tabela do lado "um" se relacione com várias linhas na tabela do lado "muitos".

Relacionamento NxN (Muitos para Muitos) é implementado utilizando uma tabela intermediária que contém chaves estrangeiras de ambas as tabelas associadas. A tabela intermediária tem uma chave primária composta pelas chaves estrangeiras, permitindo que várias linhas de uma tabela se relacionem com várias linhas da outra tabela.

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

Para representar herança em bancos de dados relacionais, utiliza-se um relacionamento 1:1 entre a tabela base (pai) e cada tabela derivada (filha), onde a chave primária da tabela base também atua como chave primária e estrangeira nas tabelas derivadas.

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 (SSMS) melhora a produtividade no gerenciamento de bancos de dados ao oferecer uma interface gráfica intuitiva, ferramentas avançadas para desenvolvimento e depuração de SQL, funcionalidades de gerenciamento e monitoramento de desempenho, além de recursos automatizados para backups, restaurações e manutenção de banco de dados.

5. 2º Procedimento | Alimentando a base

Adicionando dados a base:

Segundo Procedimen...C.Loja (loja (53))

```
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
('Joao', 'Rua 12, casa 3, Quitanda', 'Riacho Sul', 'PA', '1111-1111', 'joao@riacho.com'),
('JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '1212-1212', 'jjc@riacho.com');

INSERT INTO Pessoa_Fisica (idPessoa, cpf)
VALUES
(7, '11111111111');

INSERT INTO Pessoa_Juridica (idPessoa, cnpj)
VALUES
(15, '22222222222222');

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

6. Análise e Conclusão:

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

Sequence é o objeto independente que pode ser usado por múltiplas tabelas, oferecendo maior flexibilidade e controle sobre a geração de números. Identity é vinculado a uma coluna específica de uma tabela, gerando valores automaticamente ao inserir novas linhas.

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

As chaves estrangeiras são cruciais para a consistência do banco de dados porque elas garantem a integridade referencial, assegurando que os valores em uma coluna correspondam a valores válidos em outra tabela, evitando dados órfãos e inconsistentes.

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

Na SQL, operadores da álgebra relacional incluem SELECT para filtrar linhas, PROJECT para selecionar colunas, JOIN para combinar tabelas, UNION para unir resultados removendo duplicatas, INTERSECT para retornar linhas comuns, DIFFERENCE para obter linhas exclusivas de uma consulta e CARTESIAN PRODUCT para combinar todas as linhas de duas tabelas.

Operadores definidos no cálculo relacional incluem EXISTS para verificar a existência de linhas retornadas por uma subconsulta, IN para verificar se um valor está em um conjunto de valores, ANY/SOME para checar se alguma linha satisfaz uma condição, e FOR ALL (geralmente emulado com NOT EXISTS) para verificar se uma condição é verdadeira para todas as linhas.

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

O agrupamento em consultas SQL é feito usando a cláusula GROUP BY. Ela agrupa as linhas que têm valores iguais em colunas especificadas em conjuntos de resumo, como totalizações ou médias. O requisito obrigatório é que todas as colunas mencionadas na cláusula SELECT que não são usadas em funções de agregação (como SUM, COUNT, AVG, etc.) devem estar presentes na cláusula GROUP BY.