POLO SANTA CRUZ - RIO DE JANEIRO - RJ/UNIVERSIDADE ESTÁCIO DE SÁ

Missão Prática | Nível 2 | Mundo 3

Curso: Desenvolvimento Full Stack
Disciplina Nível 2: RPG0015 – Vamos manter as informações!
Número da Turma: 2024.2
Semestre Letivo: Mundo-3
Aluno: Lukas Cauã Oliveira Xavier
Matrícula: 202305450556
URL GIT: https://github.com/Dev-Lukas2004/TrabalhoBancoN2

1º Título da Prática: Criando o Banco de Dados

2º Objetivo da Prática:

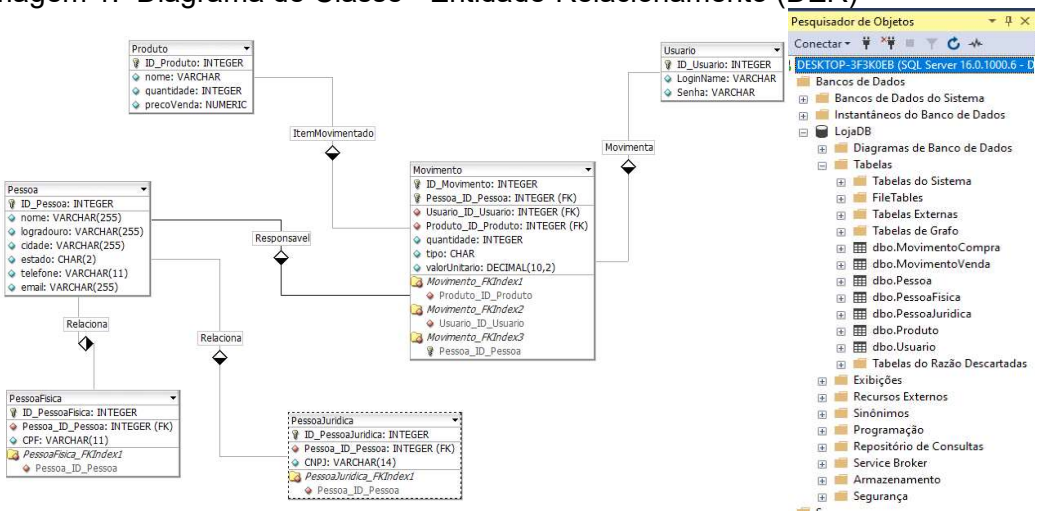
Identificar os requisitos de um sistema e transformá-los no modelo adequado.
Utilizar ferramentas de modelagem para bases de dados relacionais.
Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
Explorar a sintaxe SQL na consulta e manipulação de dados (DML) No final do exercício, o aluno 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.

Tabelas a serem Criadas:

- Usuário
- Pessoa
- Pessoa física;
- Pessoa jurídica;
- Produto;
- Movimento.

1º Procedimento | Criando o Banco de Dados.

Imagem 1: Diagrama de Classe - Entidade-Relacionamento (DER)



Definir o modelo de dados para um sistema com as características apresentadas nos tópicos seguintes:

- Deve haver um cadastro de **usuários** para acesso ao sistema, os quais irão atuar como operadores para a compra e venda de produtos.

```
CREATE TABLE Usuario (
ID_Usuario INT NOT NULL IDENTITY,
LoginName VARCHAR(255) NOT NULL,
Senha VARCHAR(255) NOT NULL,
CONSTRAINT CPK_Usuario PRIMARY KEY CLUSTERED (ID_Usuario ASC)
);
GO
```

- Deve haver um cadastro de pessoas físicas e pessoas jurídicas, com os dados básicos de identificação, localização e contato, diferenciando-se apenas pelo uso de CPF ou CNPJ.

```
CREATE TABLE PessoaFisica (
ID_PessoaFisica INT NOT NULL,
Pessoa_ID_Pessoa INT NOT NULL,
CPF VARCHAR(11) NOT NULL,
CONSTRAINT CPK_PessoaFisica PRIMARY KEY CLUSTERED (ID_PessoaFisica ASC),
CONSTRAINT CFK_Pessoa_PessoaFisica FOREIGN KEY (Pessoa_ID_Pessoa) REFERENCES
Pessoa (ID_Pessoa)
ON UPDATE CASCADE
ON DELETE CASCADE
);
GO

CREATE TABLE PessoaJuridica (
ID_PessoaJuridica INT NOT NULL,
Pessoa_ID_Pessoa INT NOT NULL,
CNPJ VARCHAR(14) NOT NULL,
```

```
CONSTRAINT CPK_PessoaJuridica PRIMARY KEY CLUSTERED (ID_PessoaJuridica ASC),
CONSTRAINT CFK_Pessoa_PessoaJuridica FOREIGN KEY (Pessoa_ID_Pessoa) REFERENCES
Pessoa (ID_Pessoa)
ON UPDATE CASCADE
ON DELETE CASCADE
);
GO
```

- Deve haver um cadastro de produtos, contendo identificador, nome, quantidade e preço de venda.

```
CREATE TABLE Produto (
ID_Produto INT NOT NULL IDENTITY,
nome VARCHAR(255) NOT NULL,
quantidade INT,
precoVenda NUMERIC(10, 2),
CONSTRAINT CPK_Produto PRIMARY KEY CLUSTERED (ID_Produto ASC)
);
GO
```

- Definir uma sequence para geração dos identificadores de pessoa, dado o relacionamento 1x1 com pessoa física ou jurídica.

```
USE Loja;
GO

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

- Os operadores (usuários) poderão efetuar **movimentos** de compra para um determinado produto, sempre de uma pessoa jurídica, indicando a quantidade de produtos e preço unitário.

```
CREATE TABLE Movimento (
ID_Movimento INT NOT NULL IDENTITY,
Pessoa_ID_Pessoa INT NOT NULL,
Usuario_ID_Usuario INT NOT NULL,
Produto_ID_Produto INT NOT NULL,
quantidade INT,
tipo CHAR(1),
valorUnitario DECIMAL(10, 2),
CONSTRAINT CPK_Movimento PRIMARY KEY CLUSTERED (ID_Movimento ASC),
CONSTRAINT CFK_Produto_Movimento FOREIGN KEY (Produto_ID_Produto) REFERENCES
Produto (ID_Produto)
ON UPDATE CASCADE
ON DELETE CASCADE,
CONSTRAINT CFK_Usuario_Movimento FOREIGN KEY (Usuario_ID_Usuario) REFERENCES
Usuario (ID_Usuario)
ON UPDATE CASCADE
ON DELETE CASCADE,
CONSTRAINT CFK_Pessoa_Movimento FOREIGN KEY (Pessoa_ID_Pessoa) REFERENCES Pessoa
(ID_Pessoa)
ON UPDATE CASCADE
ON DELETE CASCADE
);
GO

CREATE TABLE Pessoa (
ID_Pessoa INT NOT NULL DEFAULT NEXT VALUE FOR orderPessoa,
```

```
nome VARCHAR(255),
logradouro VARCHAR(255),
cidade VARCHAR(255),
estado CHAR(2),
telefone VARCHAR(11),
email VARCHAR(255),
CONSTRAINT CPK_Pessoa PRIMARY KEY CLUSTERED (ID_Pessoa ASC)
);
GO
```

Análise e Conclusão do 1º Procedimento:

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

No mundo dos bancos de dados relacionais, as cardinalidades 1:1, 1:N e N:N definem as relações entre as entidades, ditando quantas instâncias de uma entidade podem se associar a quantas instâncias de outra. Cada tipo de cardinalidade possui mecanismos específicos para sua implementação:

1:1 (Um para Um):

- **Chave Primária e Chave Estrangeira:** Nessa relação, cada entidade possui sua própria chave primária. Uma das entidades referencia a chave primária da outra através de uma chave estrangeira. Essa chave estrangeira torna-se obrigatória na tabela referenciada, garantindo que cada registro na tabela referenciada esteja associado a um único registro na tabela principal.
- **Exemplo:** Tabela Cliente (ID_Cliente, Nome, CPF) e Tabela Endereço (ID_Endereço, Rua, Cidade, ID_Cliente). Aqui, cada cliente possui apenas um endereço e cada endereço pertence a um único cliente. A tabela Endereço possui a chave estrangeira ID_Cliente que referencia a chave primária da tabela Cliente.

1:N (Um para Muitos):

- **Chave Primária e Chave Estrangeira:** Semelhante ao 1:1, a entidade principal possui sua chave primária, enquanto a entidade dependente possui chave primária própria e uma chave estrangeira que referencia a chave primária da principal. A chave estrangeira na tabela dependente também é obrigatória.
- **Exemplo:** Tabela Professor (ID_Professor, Nome, Departamento) e Tabela Turma (ID_Turma, Disciplina, ID_Professor). Um professor pode lecionar várias turmas, mas cada turma tem apenas um professor. A tabela Turma possui a chave estrangeira ID_Professor que referencia a chave primária da tabela Professor.

N:N (Muitos para Muitos):

- **Tabela de Junção:** Diferentemente das cardinalidades 1:1 e 1:N, aqui

não há chave estrangeira em nenhuma das tabelas principais. Uma nova tabela é criada, chamada de tabela de junção, contendo chaves primárias de ambas as entidades e colunas adicionais se necessário.

- **Exemplo:** Tabela Aluno (ID_Aluno, Nome, Curso) e Tabela Disciplina (ID_Disciplina, Nome, Professor). Um aluno pode cursar várias disciplinas e uma disciplina pode ter vários alunos. A tabela de junção Aluno_Disciplina (ID_Aluno, ID_Disciplina) seria criada para relacionar os alunos com as disciplinas cursadas.

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

- Existem duas abordagens principais para implementar herança em bancos de dados relacionais:
- **Tabela por classe** (ou tabela única): Nesta abordagem, todas as classes (ou tipos) na hierarquia de herança são representadas em uma única tabela. Esta tabela contém todos os atributos da hierarquia, sendo que para uma instância de uma classe específica, os atributos correspondentes às classes mais gerais são preenchidos com valores nulos.

- **Tabela por subclasse** (ou tabela separada): Aqui, cada classe (ou tipo) na hierarquia de herança é representada por uma tabela separada. A tabela correspondente à classe mais geral contém os atributos comuns a todas as classes na hierarquia. Cada tabela subsequente na hierarquia contém seus próprios atributos exclusivos além dos atributos herdados.

  O relacionamento de generalização/especialização permite modelar essas hierarquias de forma que consultas e operações possam ser feitas eficientemente, levando em consideração a estrutura hierárquica e os relacionamentos entre as entidades (ou tabelas).

- **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) é uma ferramenta robusta da Microsoft projetada para facilitar o gerenciamento de bancos de dados SQL Server. Ela melhora a produtividade em diversas tarefas relacionadas ao gerenciamento de bancos de dados de várias maneiras conforme alguns tópicos abaixo:

- **Interface Gráfica Amigável:** O SSMS fornece uma interface gráfica intuitiva que permite aos administradores de banco de dados e

desenvolvedores realizar tarefas complexas de forma mais eficiente do que usando apenas comandos SQL.

- **Editor SQL Avançado**: Inclui um editor SQL poderoso com recursos como realce de sintaxe, sugestões automáticas, formatação de código e verificação de erros, facilitando a escrita e edição de consultas SQL complexas.

- Gerenciamento de Servidores: Permite conectar e gerenciar facilmente vários servidores SQL Server a partir de uma única interface, possibilitando o monitoramento do status do servidor, configuração de propriedades e gerenciamento de segurança.

- **Ferramentas de Desenvolvimento**: Oferece ferramentas integradas para desenvolvimento de bancos de dados, como o Designer de Tabelas, o Designer de Consultas e o Designer de Procedimentos Armazenados, que simplificam o desenvolvimento de esquemas de banco de dados e a criação de consultas SQL complexas.

- **Administração de Segurança**: Facilita a administração de permissões e segurança no banco de dados, permitindo aos administradores gerenciar logons, usuários, papéis e permissões de forma centralizada.

- **Automação de Tarefas**: Oferece suporte à automação de tarefas administrativas por meio de scripts T-SQL, procedimentos armazenados, jobs do SQL Server Agent e integração com PowerShell, permitindo agendar e executar rotinas de manutenção, backups e outras tarefas programadas.

- **Monitoramento de Desempenho**: Inclui ferramentas para monitoramento e otimização de desempenho, como o Monitor de Atividade e Relatórios de Desempenho, que ajudam a identificar gargalos de desempenho e ajustar configurações para melhorar a eficiência do banco de dados.

- **Integração com Ferramentas de BI**: Possibilita a integração com ferramentas de Business Intelligence (BI), como o SQL Server Reporting Services (SSRS) e o SQL Server Analysis Services (SSAS), facilitando a criação e gerenciamento de relatórios e análises.

Esses recursos combinados ajudam os administradores de banco de dados e desenvolvedores a serem mais produtivos, permitindo-lhes gerenciar, desenvolver e manter bancos de dados SQL Server de maneira eficiente e eficaz através de uma interface unificada e poderosa.