Projeto MarketPlace

db_marketplace.sql

Sistema de Gerenciamento de Banco de Dados de um Comércio Eletrônico.

Apresentação do Banco de Dados: Marketplace

Visão Geral:

O banco de dados foi desenvolvido para um projeto de **Marketplace**, onde compradores e vendedores interagem em uma plataforma para a compra e venda de produtos. A estrutura envolve diversas tabelas que armazenam dados essenciais, como categorias de produtos, informações de clientes, transações e muito mais.

Requisitos Principais do Sistema:

- **Registro de Usuários:** Um formulário simples para que novos usuários possam se cadastrar.
- **Gerenciamento de Produtos:** Opções para que os vendedores possam adicionar, editar e remover produtos.

- **Gestão de Pedidos:** Sistema para automatizar o recebimento e acompanhamento dos pedidos.
- **Sistema de Pagamento Integrado:** Inclusão de métodos de pagamento seguros e variados.

Estrutura Principal:

1. **Esquema do Banco de Dados:**
    - O banco de dados é configurado com o esquema **"mydb"**, utilizando o **charset UTF-8** para garantir a compatibilidade com textos em múltiplos idiomas. Isso é crucial para um ambiente de marketplace global, onde podem haver usuários de diferentes regiões com diferentes alfabetos e símbolos.
2. **Tabelas Principais:**
    - **Categoria:**
        - A tabela armazena as categorias de produtos disponíveis no marketplace.
        - Cada produto pertence a uma categoria, o que facilita a organização do comércio.
    - **Produto:**
        - A tabela inclui detalhes dos itens vendidos no marketplace, como nome, descrição, preço, e categoria.
    - **Cliente:**
        - Comporta dados pessoais e informações de conta dos usuários, como nome, email, endereço, etc.
    - **Vendedor:**
        - Inclui as informações sobre os Vendedores.
    - **Fornecedor:**
        - Inclui as informações sobre os Fonrcedores.

Considerações Importantes:

- **Integridade e Relacionamentos:**

    - O banco de dados tem chaves estrangeiras que conectam tabelas como Produtos, Categoria e Transações, garantindo a integridade referencial. Por exemplo, cada produto pode ter uma chave estrangeira referenciando uma categoria específica.
- **Características Técnicas:**
    - Campos importantes como NOT NULL são usados em várias colunas, garantindo que os dados críticos não sejam deixados em branco.
    - A estrutura também parece focar na escalabilidade, permitindo o aumento de categorias, produtos e usuários à medida que o marketplace cresce.

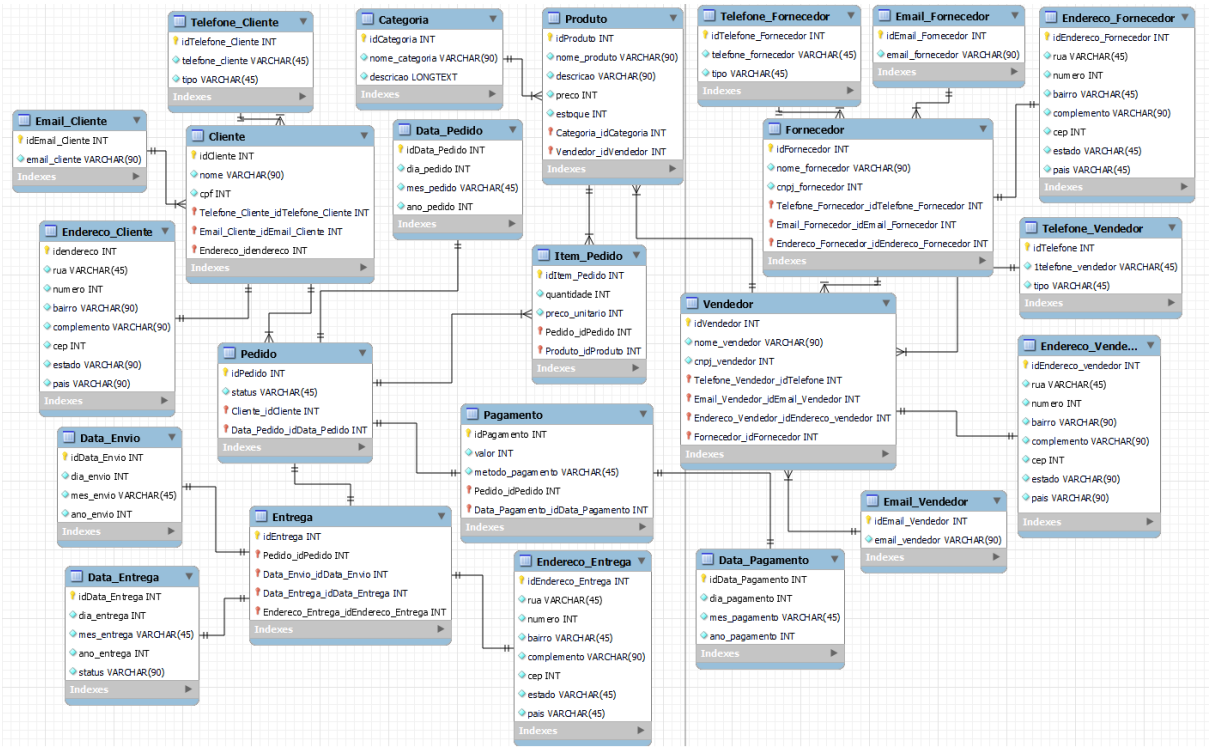
Objetivo:

Este banco de dados foi projetado para suportar as operações de um marketplace online, gerenciando eficientemente categorias de produtos, detalhes dos itens à venda, e informações sobre transações e usuários.

Modelagem Conceitual - Estrutura Relacional

Principais Entidades do Sistema:

- Vendedor: Inclui tanto vendedores.
- Cliente: Inclui os clientes do comércio.
- Fornecedor: Armazena dados sobre os fornecedores.
- Produtos: Artigos cadastrados pelos vendedores, contendo descrição, valor e imagens.
- Pedidos: Representa as transações realizadas pelos compradores.
- Pagamentos: Detalhes sobre os métodos de pagamento e o status dos mesmos.



Relações entre as Entidades

- Usuários podem ser tanto vendedores quanto clientes.
- Produtos são associados a um vendedor e comprados por vários clientes.
- Pedidos são realizados por compradores e podem conter vários produtos.
- Pagamentos são vinculados a pedidos e registram o método utilizado.

Implementação no Banco de Dados

1. Criar Banco de Dados no MySQL para armazenar informações do sistema.
2. Definir as tabelas para entidades como Usuários, Produtos, Pedidos e Pagamentos.
3. Estabelecer chaves estrangeiras para garantir a integridade dos dados.
4. Criar índices para melhorar o desempenho das consultas.

```
-- Criação do banco de dados para o projeto Marketplace Avali
-- MySQL Script generated by MySQL Workbench
-- Fri Sep  6 08:50:11 2024
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering
```

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STI

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
/*-- Criação do esquema 'marketplace' com charset UTF8.
-- O charset UTF8 é utilizado para garantir compatibilidade c
-- possibilitando a inserção de caracteres especiais, como ac
-- Isso é importante em um ambiente de marketplace, que pode
-- Imformações gerais: nos compos onde houver "NOT NULL" dete

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8
USE `mydb` ;

-- Criação da tabela "Categoria".
-- Essa tabela armazena as categorias de produtos disponíveis
-- Cada produto pertence a uma categoria, o que facilita a or

-- -----------------------------------------------------
-- Table `mydb`.`Categoria`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Categoria` (
  `idCategoria` INT NOT NULL auto_increment, -- Este campo é
  `nome_categoria` VARCHAR(90) NOT NULL, -- Este campo armaze
  `descricao` longtext not null, -- Contém a descrição de cad
  PRIMARY KEY (`idCategoria`))
ENGINE = InnoDB;

-- Criação da tabela "Telefone_Vendedor"
-- Tabela 'Telefone_Vendedor' criada para armazenar os telefo
-- A tabela está normalizada para evitar duplicação e garanti
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Telefone_Vendedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Telefone_Vendedor` (
  `idTelefone` INT NOT NULL auto_increment, -- Identificador
  `1telefone_vendedor` VARCHAR(45) NOT NULL, -- Número de tel
  `tipo` VARCHAR(45) NOT NULL, -- Tipo de telefone (ex: celul
  PRIMARY KEY (`idTelefone`)) -- Define uma chave primária co
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Email_Vendedor"
-- Essa tabela armazena os endereços de e-mail dos vendedores
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Email_Vendedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Email_Vendedor` (
  `idEmail_Vendedor` INT NOT NULL auto_increment,  -- Identif
  `email_vendedor` VARCHAR(90) NOT NULL, -- Endereço de e-mai
  PRIMARY KEY (`idEmail_Vendedor`)) -- Define uma chave primá
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Endereço_Vendedor"
-- Essa tabela armazena os endereços dos vendedores.
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Endereco_Vendedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Endereco_Vendedor` (
  `idEndereco_vendedor` INT NOT NULL auto_increment,  -- Iden
  `rua` VARCHAR(45) NOT NULL,  -- Nome da rua do endereço.
  `numero` INT NOT NULL,   -- Número do endereço.
  `bairro` VARCHAR(90) NOT NULL,  -- Nome do bairro.
  `complemento` varchar(90), -- Uma informção opcional caso h
  `cep` INT NOT NULL,  -- Código postal do endereço.
  `estado` VARCHAR(90) NOT NULL, -- Nome do estado do endereç
  `pais` VARCHAR(90) NOT NULL, -- País do endereço
  PRIMARY KEY (`idEndereco_vendedor`)) -- Define uma chave pr
```

```
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Telefone_Fornecedor"
-- Essa tabela armazena os números de telefone dos fornecedor
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Telefone_Fornecedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Telefone_Fornecedor` (
  `idTelefone_Fornecedor` INT NOT NULL auto_increment,  -- Id
  `telefone_fornecedor` VARCHAR(45) NOT NULL,  -- Número do t
  `tipo` VARCHAR(45) NOT NULL, -- Tipo de telefone (ex: celul
  PRIMARY KEY (`idTelefone_Fornecedor`)) -- Define uma chave
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Email_Fornecedor"
-- Essa tabela armazena os emails dos fornecedores.
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Email_Fornecedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Email_Fornecedor` (
  `idEmail_Fornecedor` INT NOT NULL auto_increment, -- Identi
  `email_fornecedor` VARCHAR(90) NOT NULL, -- Endereço de ema
  PRIMARY KEY (`idEmail_Fornecedor`)) -- Define uma chave pri
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Endereço_Fornecedor"
-- Essa tabela armazena o endereço dos fornecedores.
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Endereco_Fornecedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Endereco_Fornecedor` (
  `idEndereco_Fornecedor` INT NOT NULL auto_increment,  -- Id
  `rua` VARCHAR(45) NOT NULL, -- Nome da rua do endereço.
  `numero` INT NOT NULL,  -- Número do endereço.
  `bairro` VARCHAR(45) NOT NULL, -- Bairro onde o fornecedor
```

```
  `complemento` varchar(90), -- Uma informção opcional caso h
  `cep` INT NOT NULL,  -- Código Postal do endereço.
  `estado` VARCHAR(45) NOT NULL, -- Estado do endereço do for
  `pais` VARCHAR(45) NOT NULL, -- País do fornecedor.
  PRIMARY KEY (`idEndereco_Fornecedor`)) -- Define uma chave
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Fornecedor"
-- Essa tabela armazena as informações dos fornecedores.
-- -----------------------------------------------------
-- Table `mydb`.`Fornecedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Fornecedor` (
  `idFornecedor` INT NOT NULL auto_increment,  -- Identificad
  `nome_fornecedor` VARCHAR(90) NOT NULL, -- Nome do forneced
  `cnpj_fornecedor` INT NOT NULL,  -- CNPJ do fornecedor, dev
  `Telefone_Fornecedor_idTelefone_Fornecedor` INT NOT NULL,
  `Email_Fornecedor_idEmail_Fornecedor` INT NOT NULL,  -- Cha
  `Endereco_Fornecedor_idEndereco_Fornecedor` INT NOT NULL,
  PRIMARY KEY (`idFornecedor`, `Telefone_Fornecedor_idTelefon
  garantindo que a combinação dos campos 'idFornecedor', 'Tel
  INDEX `fk_Fornecedor_Telefone_Fornecedor1_idx` (`Telefone_F
  INDEX `fk_Fornecedor_Email_Fornecedor1_idx` (`Email_Fornece
  INDEX `fk_Fornecedor_Endereco_Fornecedor1_idx` (`Endereco_F
  CONSTRAINT `fk_Fornecedor_Telefone_Fornecedor1` -- Define u
    FOREIGN KEY (`Telefone_Fornecedor_idTelefone_Fornecedor`)
    REFERENCES `mydb`.`Telefone_Fornecedor` (`idTelefone_Forn
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fornecedor_Email_Fornecedor1` -- Define uma
    FOREIGN KEY (`Email_Fornecedor_idEmail_Fornecedor`)
    REFERENCES `mydb`.`Email_Fornecedor` (`idEmail_Fornecedor
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fornecedor_Endereco_Fornecedor1` -- Define u
    FOREIGN KEY (`Endereco_Fornecedor_idEndereco_Fornecedor`)
    REFERENCES `mydb`.`Endereco_Fornecedor` (`idEndereco_Forn
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Vendedor"
-- Essa tabela armazena as informações dos Vendedores.
-- -----------------------------------------------------
-- Table `mydb`.`Vendedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Vendedor` (
  `idVendedor` INT NOT NULL auto_increment, -- Identificador
  `nome_vendedor` VARCHAR(90) NOT NULL, -- Nome do vendedor.
  `cnpj_vendedor` INT NOT NULL, -- CNPJ do vendedor, deve ser
  `Telefone_Vendedor_idTelefone` INT NOT NULL, -- Número de t
  `Email_Vendedor_idEmail_Vendedor` INT NOT NULL, -- Endereço
  `Endereco_Vendedor_idEndereco_vendedor` INT NOT NULL, -- En
  `Fornecedor_idFornecedor` INT NOT NULL, -- ID do fornecedor
  PRIMARY KEY (`idVendedor`, `Telefone_Vendedor_idTelefone`,
-- 'Email_Vendedor_idEmail_Vendedor', 'Endereco_Vendedor_idEn
-- Isso garante que cada vendedor seja único em combinação co
  INDEX `fk_Vendedor_Telefone_Vendedor1_idx` (`Telefone_Vende
  INDEX `fk_Vendedor_Email_Vendedor1_idx` (`Email_Vendedor_id
  INDEX `fk_Vendedor_Endereco_Vendedor1_idx` (`Endereco_Vende
  INDEX `fk_Vendedor_Fornecedor1_idx` (`Fornecedor_idFornece
  CONSTRAINT `fk_Vendedor_Telefone_Vendedor1`-- Restrição que
    FOREIGN KEY (`Telefone_Vendedor_idTelefone`)
    REFERENCES `mydb`.`Telefone_Vendedor` (`idTelefone`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Vendedor_Email_Vendedor1` -- Restrição que g
    FOREIGN KEY (`Email_Vendedor_idEmail_Vendedor`)
    REFERENCES `mydb`.`Email_Vendedor` (`idEmail_Vendedor`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Vendedor_Endereco_Vendedor1` -- Restrição qu
-- Caso contrário, não será possível inserir ou atualizar dad
    FOREIGN KEY (`Endereco_Vendedor_idEndereco_vendedor`)
    REFERENCES `mydb`.`Endereco_Vendedor` (`idEndereco_vended
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Vendedor_Fornecedor1` -- Restrição que garan
-- Caso contrário, não será possível inserir ou atualizar dad
    FOREIGN KEY (`Fornecedor_idFornecedor`)
    REFERENCES `mydb`.`Fornecedor` (`idFornecedor`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Produto"
-- Essa tabela armazena as informações dos Produtos.
-- -----------------------------------------------------
-- Table `mydb`.`Produto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Produto` (
  `idProduto` INT NOT NULL auto_increment, -- Identificador ú
  `nome_produto` VARCHAR(90) NOT NULL, -- Nome do produto.
  `descricao` VARCHAR(90) NOT NULL, -- Descrição do produto.
  `preco` INT NOT NULL, -- Preço do produto.
  `estoque` INT NOT NULL, -- Quantidade disponível do produto
  `Categoria_idCategoria` INT NOT NULL, -- ID da categoria do
  `Vendedor_idVendedor` INT NOT NULL, -- ID do vendedor que o
  PRIMARY KEY (`idProduto`, `Categoria_idCategoria`, `Vendedo
  INDEX `fk_Produto_Categoria1_idx` (`Categoria_idCategoria` A
  INDEX `fk_Produto_Vendedor1_idx` (`Vendedor_idVendedor` ASC
  CONSTRAINT `fk_Produto_Categoria1`  -- Restrição de integri
    FOREIGN KEY (`Categoria_idCategoria`)
    REFERENCES `mydb`.`Categoria` (`idCategoria`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Produto_Vendedor1` /*Restringe o campo 'Vend
  Se um vendedor for removido da tabela 'Vendedor', a ação 'N
    FOREIGN KEY (`Vendedor_idVendedor`)
    REFERENCES `mydb`.`Vendedor` (`idVendedor`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta
```

```
-- Criação da tabela "Telefone_Cliente"
-- Essa tabela armazena os números de telefone dos Clientes.
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Telefone_Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Telefone_Cliente` (
  `idTelefone_Cliente` INT NOT NULL auto_increment, -- Identi
  `telefone_cliente` VARCHAR(45) NOT NULL, -- Número de telef
  `tipo` VARCHAR(45) NOT NULL, -- Tipo de telefone (ex: celul
  PRIMARY KEY (`idTelefone_Cliente`)) -- Define uma chave pri
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Email_Cliente"
-- Essa tabela armazena os números de emails dos Clientes.
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Email_Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Email_Cliente` (
  `idEmail_Cliente` INT NOT NULL auto_increment, -- Identific
  `email_cliente` VARCHAR(90) NOT NULL, -- Endereço de e-mail
  PRIMARY KEY (`idEmail_Cliente`)) -- Define uma chave primár
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Endereço_Cliente"
-- Essa tabela armazena o endereço dos Clientes.
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Endereco_Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Endereco_Cliente` (
  `idendereco` INT NOT NULL auto_increment,  -- Identificador
  `rua` VARCHAR(45) NOT NULL, -- Nome da rua do endereço.
  `numero` INT NOT NULL, -- Número do imóvel no endereço.
  `bairro` VARCHAR(90) NOT NULL, -- Bairro onde o endereço es
  `complemento` varchar(90), -- Uma informção opcional caso h
  `cep` INT NOT NULL, -- Código de Endereçamento Postal do en
```

```
  `estado` VARCHAR(90) NOT NULL, -- Estado onde o endereço es
  `pais` VARCHAR(90) NOT NULL,  -- País onde o endereço está
  PRIMARY KEY (`idendereco`)) -- Define 'idendereco' como cha
ENGINE = InnoDB; -- Define o mecanismo de armazenamento da ta

-- Criação da tabela "Cliente"
-- Essa tabela armazena as informações sobre os Clientes.

-- -----------------------------------------------------
-- Table `mydb`.`Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Cliente` (
  `idCliente` INT NOT NULL auto_increment,-- Identificador ún
  `nome` VARCHAR(90) NOT NULL,-- Identificador único para cad
  `cpf` INT NOT NULL,-- Número do CPF do cliente
  `Telefone_Cliente_idTelefone_Cliente` INT NOT NULL, -- Refe
  `Email_Cliente_idEmail_Cliente` INT NOT NULL,-- Referência
  `Endereco_idendereco` INT NOT NULL, -- Referência ao endere
  PRIMARY KEY (`idCliente`, `Telefone_Cliente_idTelefone_Clie
  INDEX `fk_Cliente_Telefone_Cliente1_idx` (`Telefone_Cliente
  INDEX `fk_Cliente_Email_Cliente1_idx` (`Email_Cliente_idEma
  INDEX `fk_Cliente_Endereco1_idx` (`Endereco_idendereco` ASC
  CONSTRAINT `fk_Cliente_Telefone_Cliente1`  -- Define a chav
    FOREIGN KEY (`Telefone_Cliente_idTelefone_Cliente`)
    REFERENCES `mydb`.`Telefone_Cliente` (`idTelefone_Cliente
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Cliente_Email_Cliente1`  -- Define a chave e
    FOREIGN KEY (`Email_Cliente_idEmail_Cliente`)
    REFERENCES `mydb`.`Email_Cliente` (`idEmail_Cliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Cliente_Endereco1` -- Define a chave estrang
    FOREIGN KEY (`Endereco_idendereco`)
    REFERENCES `mydb`.`Endereco_Cliente` (`idendereco`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Criação da tabela `Data_Pedido`
-- Essa tabela armazena as datas dos pedidos.
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Data_Pedido`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Data_Pedido` (
  `idData_Pedido` INT NOT NULL auto_increment,-- Identificado
  `dia_pedido` INT NOT NULL,-- Dia do pedido.
  `mes_pedido` VARCHAR(45) NOT NULL,-- Mês do pedido.
  `ano_pedido` INT NOT NULL, -- Ano do pedido.
  PRIMARY KEY (`idData_Pedido`))
ENGINE = InnoDB;

-- Criação da tabela `Pedido`
-- Tabela para armazenar informações sobre pedidos realizados
-- Cada pedido é identificado por um 'idPedido' e está associ
-- -----------------------------------------------------
-- Table `mydb`.`Pedido`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Pedido` (
  `idPedido` INT NOT NULL auto_increment,-- Identificador úni
  `status` VARCHAR(45) NOT NULL, -- Status atual do pedido (p
  `Cliente_idCliente` INT NOT NULL,-- Referência ao cliente q
  `Data_Pedido_idData_Pedido` INT NOT NULL,-- Referência à da
  PRIMARY KEY (`idPedido`, `Cliente_idCliente`, `Data_Pedido_
  Isso garante que cada pedido para um cliente em uma data es
  INDEX `fk_Pedido_Cliente1_idx` (`Cliente_idCliente` ASC) ,-
  INDEX `fk_Pedido_Data_Pedido1_idx` (`Data_Pedido_idData_Ped
  CONSTRAINT `fk_Pedido_Cliente1`  -- Restrições de chave est
    FOREIGN KEY (`Cliente_idCliente`)
    REFERENCES `mydb`.`Cliente` (`idCliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pedido_Data_Pedido1`  -- Restrições de chave
    FOREIGN KEY (`Data_Pedido_idData_Pedido`)
    REFERENCES `mydb`.`Data_Pedido` (`idData_Pedido`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- Criação da tabela `Item_Pedido`
-- Tabela para armazenar informações sobre pedidos realizados
-- Cada item de pedido inclui uma quantidade e o preço unitár
-- Está associado a um pedido específico e a um produto espec
-- -----------------------------------------------------
-- Table `mydb`.`Item_Pedido`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Item_Pedido` (
  `idItem_Pedido` INT NOT NULL auto_increment,-- Identificado
  `quantidade` INT NOT NULL,-- Quantidade do produto no pedid
  `preco_unitario` INT NOT NULL, -- Preço unitário do produto
  `Pedido_idPedido` INT NOT NULL, -- Referência ao pedido ao
  `Produto_idProduto` INT NOT NULL, -- Referência ao produto
  PRIMARY KEY (`idItem_Pedido`, `Pedido_idPedido`, `Produto_i
  Garante que cada item em um pedido seja único por combinaçã
  INDEX `fk_Item_Pedido_Pedido1_idx` (`Pedido_idPedido` ASC)
  INDEX `fk_Item_Pedido_Produto1_idx` (`Produto_idProduto` AS
  CONSTRAINT `fk_Item_Pedido_Pedido1`  -- Restrições de chave
    FOREIGN KEY (`Pedido_idPedido`)
    REFERENCES `mydb`.`Pedido` (`idPedido`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Item_Pedido_Produto1`  -- Restrições de chav
    FOREIGN KEY (`Produto_idProduto`)
    REFERENCES `mydb`.`Produto` (`idProduto`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- Criação da tabela `Data_Pagamento`
-- Essa tabela armazena as datas dos pagamentos.
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Data_Pagamento`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Data_Pagamento` (
  `idData_Pagamento` INT NOT NULL auto_increment,-- Identific
  `dia_pagamento` INT NOT NULL, -- Dia do pagamento.
  `mes_pagamento` VARCHAR(45) NOT NULL, -- Mês do pagamento (
  `ano_pagamento` INT NOT NULL, -- Ano do pagamento.
  PRIMARY KEY (`idData_Pagamento`))
ENGINE = InnoDB;

-- Criação da tabela `Pagamento`
-- Tabela para armazenar informações sobre os pagamentos real
-- Contém detalhes do pagamento, incluindo o valor, método de
-- -----------------------------------------------------
-- Table `mydb`.`Pagamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Pagamento` (
  `idPagamento` INT NOT NULL auto_increment, -- Identificador
  `valor` INT NOT NULL, -- Valor do pagamento.
  `metodo_pagamento` VARCHAR(45) NOT NULL,-- Método utilizado
  `Pedido_idPedido` INT NOT NULL, -- Referência para o pedido
  `Data_Pagamento_idData_Pagamento` INT NOT NULL, -- Referênc
  PRIMARY KEY (`idPagamento`, `Pedido_idPedido`, `Data_Pagame
  INDEX `fk_Pagamento_Pedido1_idx` (`Pedido_idPedido` ASC) ,
  INDEX `fk_Pagamento_Data_Pagamento1_idx` (`Data_Pagamento_i
  CONSTRAINT `fk_Pagamento_Pedido1` -- Restrições de chave es
    FOREIGN KEY (`Pedido_idPedido`)
    REFERENCES `mydb`.`Pedido` (`idPedido`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pagamento_Data_Pagamento1` -- Restrições de
    FOREIGN KEY (`Data_Pagamento_idData_Pagamento`)
    REFERENCES `mydb`.`Data_Pagamento` (`idData_Pagamento`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- Criação da tabela `Data_Envio`
-- Essa tabela armazena as datas de envio dos pedidos.
```

```
-- A tabela está normalizada para evitar duplicação e garanti

-- -----------------------------------------------------
-- Table `mydb`.`Data_Envio`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Data_Envio` (
  `idData_Envio` INT NOT NULL auto_increment,-- Identificador
  `dia_envio` INT NOT NULL, -- Dia em que o envio ocorreu.
  `mes_envio` VARCHAR(45) NOT NULL,-- Mês em que o envio ocor
  `ano_envio` INT NOT NULL, -- Ano em que o envio ocorreu.
  PRIMARY KEY (`idData_Envio`))
ENGINE = InnoDB;

-- Criação da tabela `Data_Entrega`
-- Essa tabela armazena as datas de entrega dos pedidos.
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Data_Entrega`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Data_Entrega` (
  `idData_Entrega` INT NOT NULL auto_increment, -- Identifica
  `dia_entrega` INT NOT NULL,-- Dia da entrega.
  `mes_entrega` VARCHAR(45) NOT NULL,-- Mês da entrega (em te
  `ano_entrega` INT NOT NULL,-- Ano da entrega.
  `status` VARCHAR(90) NOT NULL,-- Status da entrega (por exe
  PRIMARY KEY (`idData_Entrega`))
ENGINE = InnoDB;

-- Criação da tabela "Endereço_Entrega"
-- Tabela para armazenar endereços de entrega dos pedidos.
-- Contém detalhes sobre o endereço completo de entrega, incl
-- A tabela está normalizada para evitar duplicação e garanti
-- -----------------------------------------------------
-- Table `mydb`.`Endereco_Entrega`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Endereco_Entrega` (
  `idEndereco_Entrega` INT NOT NULL auto_increment,-- Identif
  `rua` VARCHAR(45) NOT NULL,-- Rua do endereço de entrega.
```

```
  `numero` INT NOT NULL, -- Número do endereço de entrega.
  `bairro` VARCHAR(45) NOT NULL, -- Bairro do endereço de ent
  `complemento` varchar(90), -- Uma informção opcional caso h
  `cep` INT NOT NULL,-- Código de Endereçamento Postal (CEP)
  `estado` VARCHAR(45) NOT NULL, -- Estado do endereço de ent
  `pais` VARCHAR(45) NOT NULL,-- País do endereço de entrega.
  PRIMARY KEY (`idEndereco_Entrega`))
ENGINE = InnoDB;

-- Criação da tabela "Entrega"
-- Tabela que registra informações sobre as entregas realizad
-- -----------------------------------------------------
-- Table `mydb`.`Entrega`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Entrega` (
  `idEntrega` INT NOT NULL auto_increment,-- Identificador ún
  `Pedido_idPedido` INT NOT NULL, -- Identificador do pedido
  `Data_Envio_idData_Envio` INT NOT NULL, -- Identificador da
  `Data_Entrega_idData_Entrega` INT NOT NULL, -- Identificado
  `Endereco_Entrega_idEndereco_Entrega` INT NOT NULL,-- Ident
  PRIMARY KEY (`idEntrega`, `Pedido_idPedido`, `Data_Envio_id
  INDEX `fk_Entrega_Pedido1_idx` (`Pedido_idPedido` ASC) ,  -
  INDEX `fk_Entrega_Data_Envio1_idx` (`Data_Envio_idData_Envi
  INDEX `fk_Entrega_Data_Entrega1_idx` (`Data_Entrega_idData_
  INDEX `fk_Entrega_Endereco_Entrega1_idx` (`Endereco_Entrega_
  CONSTRAINT `fk_Entrega_Pedido1` -- Define as chaves estrang
    FOREIGN KEY (`Pedido_idPedido`)
    REFERENCES `mydb`.`Pedido` (`idPedido`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Entrega_Data_Envio1` -- Define as chaves est
    FOREIGN KEY (`Data_Envio_idData_Envio`)
    REFERENCES `mydb`.`Data_Envio` (`idData_Envio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Entrega_Data_Entrega1` -- Define as chaves e
    FOREIGN KEY (`Data_Entrega_idData_Entrega`)
    REFERENCES `mydb`.`Data_Entrega` (`idData_Entrega`)
```

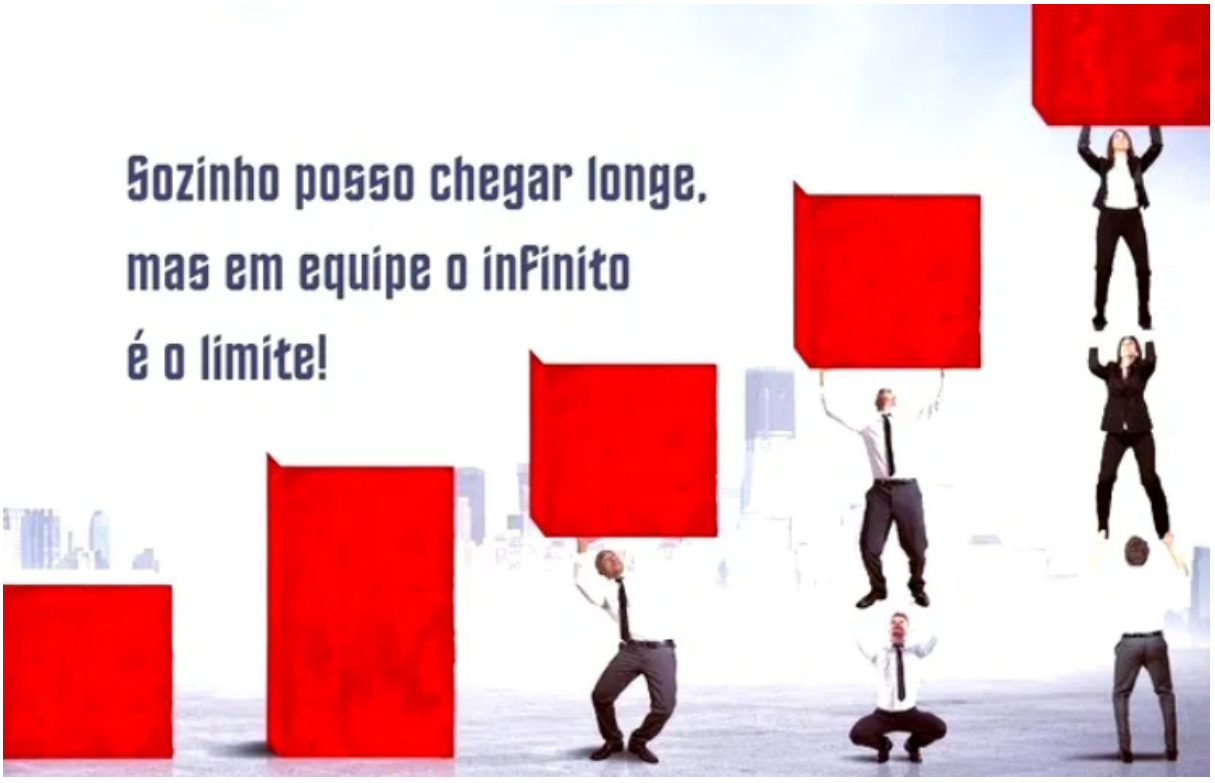
```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Entrega_Endereco_Entrega1` -- Define as chav
    FOREIGN KEY (`Endereco_Entrega_idEndereco_Entrega`)
    REFERENCES `mydb`.`Endereco_Entrega` (`idEndereco_Entrega
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Validação e Testes

1. Inserir dados fictícios para testar a funcionalidade.
2. Executar consultas SQL para verificar o armazenamento e recuperação de dados.
3. Verificar se as chaves estrangeiras estão funcionando corretamente.
4. Realizar ajustes conforme necessário.

Conclusão

O modelo relacional para a plataforma de comércio eletrônico foi implementado com sucesso. Ele garante organização, integridade dos dados e eficiência nas operações.



Este projeto foi desenvolvido por **Maria Fernanda Ribeiro Corrales e Guido Fernandes da Guarda**. Trata-se da PARTE 1 da segunda avaliação do curso de Administrador de Banco de Dados, oferecido pelo SENAI - Taguatinga - DF em parceria com o Programa DF INOVATECH. Mentorado pela Professora Mirka Juliet.