3 parte do trabalho (PARTE FINAL)
José,Miguel,Heron,Pablo,João Pedro Mendes de Oliveira

RELATORIO (PARTE FINAL DO TRABALHO)

O presente relatório apresenta e descreve detalhadamente a construção do banco de dados meuBanco, cujo objetivo é oferecer uma solução completa para o gerenciamento de estádios, contemplando eventos, ingressos, clientes, funcionários, lojas, produtos e operações internas. A criação do banco foi iniciada com a exclusão de qualquer versão anterior e a geração de um novo ambiente, garantindo organização e consistência desde o início. A escolha do mecanismo InnoDB foi fundamental por permitir o uso adequado de chaves estrangeiras e assegurar integridade referencial em toda a modelagem.

A estrutura começa com a tabela setor, responsável por registrar cada área interna de um estádio, como arquibancadas, camarotes e setores especiais. Essa tabela contém informações importantes, como nome, tipo e capacidade, e se relaciona diretamente com a tabela de ingressos. A tabela ingressos registra dados como preço, assento, status e o setor ao qual o ingresso pertence, garantindo que cada ticket vendido esteja vinculado a um setor válido e devidamente cadastrado.

A tabela estadio centraliza as informações principais dos estádios, incluindo nome, endereço e capacidade total. A partir dessa tabela, estabelecem-se relações fundamentais com eventos, lojas e estacionamento. A tabela eventos é a base para o registro de todas as atividades realizadas no estádio, armazenando data, horário, classificação indicativa e o estádio responsável. A modelagem inclui ainda duas especializações: evento_esportivo e evento_cultural, que utilizam a mesma chave primária da tabela de eventos, evitando redundância e organizando os diferentes tipos de eventos.

O sistema também utiliza a tabela pessoa, que funciona como base para duas especializações: clientes e funcionários. Essa escolha evita duplicação de informações pessoais e garante integridade no cadastro. Funcionários recebem atributos específicos, como salário e função, enquanto clientes possuem registro de compras e data de nascimento, sempre vinculados ao CPF já existente na tabela pessoa.

A modelagem inclui ainda a tabela lojas, responsáveis pelos estabelecimentos internos do estádio. Cada loja possui nome, localização e vínculo com um estádio. Os relacionamentos mais complexos, como funcionários que trabalham em lojas, são resolvidos com a tabela trabalha, que representa um relacionamento muitos-para-

muitos entre funcionários e lojas. De forma semelhante, a venda de produtos pelas lojas é representada pela tabela loja_vende, enquanto o consumo desses produtos por clientes é registrado na tabela consome.

A tabela produto registra os itens disponibilizados nas lojas, com informações como tipo, descrição e quantidade em estoque. Já os fornecedores são gerenciados pela tabela fornecedores, e seu relacionamento com as lojas ocorre por meio da tabela loja_fornecedores, permitindo que múltiplas lojas sejam supridas por múltiplos fornecedores.

O sistema contempla ainda o controle de estacionamento por meio da tabela estacionamento, que registra o ticket, o preço e a placa do veículo, sempre vinculado ao estádio correspondente. Para completar, a venda de ingressos pelos estádios é administrada pela tabela vende_ingresso, representando um relacionamento entre estádios e os ingressos comercializados.

Em conclusão, o banco de dados meuBanco apresenta uma modelagem robusta, clara e totalmente estruturada com base nos princípios da integridade referencial e da normalização. A organização das tabelas, os relacionamentos planejados e as especializações garantem um sistema eficiente e flexível.

CODIGO SQL:

```
DROP DATABASE meuBanco;

CREATE DATABASE meuBanco;

USE meuBanco;


-- TABELA SETOR


CREATE TABLE setor (

   idSetor INT AUTO_INCREMENT PRIMARY KEY,
```

```
    nome VARCHAR(100),
    tipo VARCHAR(50),
    capacidade INT
) ENGINE=InnoDB;

-- TABELA INGRESSOS
CREATE TABLE ingressos (
    id_ingresso INT AUTO_INCREMENT PRIMARY KEY,
    preco DECIMAL(10,2),
    assento VARCHAR(10),
    status VARCHAR(20),
    idSetor INT,
    FOREIGN KEY (idSetor) REFERENCES setor(idSetor)
) ENGINE=InnoDB;

-- TABELA ESTADIO
CREATE TABLE estadio (
    idEstadio INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    endereco VARCHAR(200),
    capacidade INT
) ENGINE=InnoDB;

-- TABELA EVENTOS
```

```
CREATE TABLE eventos (
   id_Evento INT AUTO_INCREMENT PRIMARY KEY,
   data DATE,
   horario TIME,
   classificacao_indicativa VARCHAR(20),
   idEstadio INT,
   FOREIGN KEY (idEstadio) REFERENCES estadio(idEstadio)
) ENGINE=InnoDB;

-- TABELA EVENTO ESPORTIVO
CREATE TABLE evento_esportivo (
   id_Evento INT PRIMARY KEY,
   modalidade VARCHAR(50),
   competicao VARCHAR(50),
   FOREIGN KEY (id_Evento) REFERENCES eventos(id_Evento)
) ENGINE=InnoDB;

-- TABELA EVENTO CULTURAL
CREATE TABLE evento_cultural (
   id_Evento INT PRIMARY KEY,
   tipo VARCHAR(50),
   FOREIGN KEY (id_Evento) REFERENCES eventos(id_Evento)
) ENGINE=InnoDB;
```

```
-- TABELA PESSOA
CREATE TABLE pessoa (
   cpf VARCHAR(14) PRIMARY KEY,
   nome VARCHAR(100),
   telefone VARCHAR(20),
   email VARCHAR(100)
) ENGINE=InnoDB;

-- TABELA FUNCIONARIOS
CREATE TABLE funcionarios (
   cpf VARCHAR(14) PRIMARY KEY,
   salario DECIMAL(10,2),
   funcao VARCHAR(50),
   FOREIGN KEY (cpf) REFERENCES pessoa(cpf)
) ENGINE=InnoDB;

-- TABELA CLIENTES
CREATE TABLE clientes (
   cpf VARCHAR(14) PRIMARY KEY,
   compras INT,
   data_de_nascimento DATE,
   FOREIGN KEY (cpf) REFERENCES pessoa(cpf)
) ENGINE=InnoDB;
```

```
-- TABELA LOJAS
CREATE TABLE lojas (
   idLoja INT AUTO_INCREMENT PRIMARY KEY,
   nome VARCHAR(100),
   localizacao VARCHAR(100),
   idEstadio INT,
   FOREIGN KEY (idEstadio) REFERENCES estadio(idEstadio)
) ENGINE=InnoDB;

-- TABELA TRABALHA (FUNCIONARIOS x LOJAS)
CREATE TABLE trabalha (
   cpf VARCHAR(14),
   idLoja INT,
   PRIMARY KEY (cpf, idLoja),
   FOREIGN KEY (cpf) REFERENCES funcionarios(cpf),
   FOREIGN KEY (idLoja) REFERENCES lojas(idLoja)
) ENGINE=InnoDB;

-- TABELA PRODUTOS
CREATE TABLE produto (
   codigo_produto INT AUTO_INCREMENT PRIMARY KEY,
   tipo VARCHAR(100),
   descricao VARCHAR(200),
   estoque INT
```

```
) ENGINE=InnoDB;


-- LOJA VENDE PRODUTOS (N:N)
CREATE TABLE loja_vende (
   idLoja INT,
   codigo_produto INT,
   PRIMARY KEY (idLoja, codigo_produto),
   FOREIGN KEY (idLoja) REFERENCES lojas(idLoja),
   FOREIGN KEY (codigo_produto) REFERENCES produto(codigo_produto)
) ENGINE=InnoDB;


-- CLIENTE CONSOME PRODUTOS (N:N)
CREATE TABLE consome (
   cpf VARCHAR(14),
   codigo_produto INT,
   quantidade INT,
   PRIMARY KEY (cpf, codigo_produto),
   FOREIGN KEY (cpf) REFERENCES clientes(cpf),
   FOREIGN KEY (codigo_produto) REFERENCES produto(codigo_produto)
) ENGINE=InnoDB;


-- TABELA FORNECEDORES
CREATE TABLE fornecedores (
   idFornecedores INT AUTO_INCREMENT PRIMARY KEY,
```

```
    endereco VARCHAR(200),
    telefone VARCHAR(20)
) ENGINE=InnoDB;

-- LOJA x FORNECEDORES (N:N)
CREATE TABLE loja_fornecedores (
    idLoja INT,
    idFornecedores INT,
    PRIMARY KEY (idLoja, idFornecedores),
    FOREIGN KEY (idLoja) REFERENCES lojas(idLoja),
    FOREIGN KEY (idFornecedores) REFERENCES fornecedores(idFornecedores)
) ENGINE=InnoDB;

-- TABELA ESTACIONAMENTO
CREATE TABLE estacionamento (
    ticket INT AUTO_INCREMENT PRIMARY KEY,
    preco DECIMAL(10,2),
    placa_veiculo VARCHAR(10),
    idEstadio INT,
    FOREIGN KEY (idEstadio) REFERENCES estadio(idEstadio)
) ENGINE=InnoDB;

-- ESTADIO VENDE INGRESSOS (1:N)
CREATE TABLE vende_ingresso (
```

```
  idEstadio INT,
  id_ingresso INT,
  PRIMARY KEY (idEstadio, id_ingresso),
  FOREIGN KEY (idEstadio) REFERENCES estadio(idEstadio),
  FOREIGN KEY (id_ingresso) REFERENCES ingressos(id_ingresso)
) ENGINE=InnoDB;
```

CODIGO DE APLICAÇÃO:

No terminal digite: pip install mysql-connector-python

import mysql.connector from mysql.connector import Error

class DatabaseManager: def *init*(self): self.connection = None

```
def connect(self):
    try:
        # Configure com suas credenciais
        self.connection = mysql.connector.connect(
            host='localhost',
            user='root',
            password='1234567809',
            database='testdb'
        )
        print('Conectado ao MySQL com sucesso!')
        return True
    except Error as e:
        print(f'Erro ao conectar: {e}')
        return False

def create_table(self):
    try:
        cursor = self.connection.cursor()
        create_table_sql = """
            CREATE TABLE IF NOT EXISTS users (
                id INT AUTO_INCREMENT PRIMARY KEY,
```

```
                name VARCHAR(100) NOT NULL,
                email VARCHAR(100) NOT NULL UNIQUE,
                created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            )
        """
        cursor.execute(create_table_sql)
        print('Tabela users criada ou verificada com sucesso!')
    except Error as e:
        print(f'Erro ao criar tabela: {e}')

def insert_user(self, name, email):
    try:
        cursor = self.connection.cursor()
        sql = "INSERT INTO users (name, email) VALUES (%s, %s)"
        cursor.execute(sql, (name, email))
        self.connection.commit()
        print(f'Usuario inserido com ID: {cursor.lastrowid}')
    except Error as e:
        print(f'Erro ao inserir usuario: {e}')

def select_users(self):
    try:
        cursor = self.connection.cursor()
        cursor.execute("SELECT * FROM users")
        rows = cursor.fetchall()

        print('\nLista de Usuarios:')
        print('----------------------------------------')
        for row in rows:
            print(f"ID: {row[0]} | Nome: {row[1]} | Email: {row[2]}
| Criado: {row[3]}")
        print('----------------------------------------')
        print(f"Total: {len(rows)} usuarios\n")
    except Error as e:
        print(f'Erro ao buscar usuarios: {e}')

def show_menu(self):
    print('\nMENU DO BANCO DE DADOS')
    print('1. Inserir usuario')
    print('2. Listar usuarios')
    print('3. Sair')
```

```
def start(self):
    if not self.connect():
        return

    self.create_table()

    while True:
        self.show_menu()
        option = input('Escolha uma opcao: ')

        if option == '1':
            name = input('Nome: ')
            email = input('Email: ')
            self.insert_user(name, email)

        elif option == '2':
            self.select_users()

        elif option == '3':
            print('Saindo...')
            break

        else:
            print('Opcao invalida!')

    if self.connection:
        self.connection.close()


#Executar o programa

if __name__ == "__main__":
 db_manager = DatabaseManager()
 db_manager.start()
```