Trabalho II– Implementação de Banco de Dados de Sistema Banco de Dados

Nicollas Castelo Branco Costa Lopes Soares[1]

[1]Tecnologia em Gestão de Dados – Universidade Federal do Piauí (UFPI)
[2]Centro de Educação Aberta e a Distância (CEAD)
[3] Prof. Dr. Arlino Henrique Magalhães de Araújo
Teresina-PI-Brasil

nicollas.soares@ufpi.edu.br

Abstract. *This document presents the guidelines and evaluation criteria for* ***Task II*** *of the Database Technology in course. This file will demonstrate a practical implementation of a database, going through the entire creation process (code, SQL queries, and modeling, all the way to the final product).*

Resumo. *Este documento apresenta as diretrizes e os critérios de avaliação para a* ***Tarefa II*** *do curso Tecnologia em Banco de Dados. Este arquivo demonstrará uma implementação prática de um banco de dados passando por todo o processo de criação (códigos,consultas SQL e modelagem até o produto final).*

1. Introdução

Sabe-se que, no contexto hodierno, as empresas vêm precisando de profissionais que impulsionem e facilitem a organização de estatísticas do local. No entanto, muitas organizações ainda utilizam softwares como o Excel para o armazenamento e o gerenciamento desses dados. Apesar de ser uma ferramenta útil, o processo pode se tornar trabalhoso, principalmente quando há um grande volume de informações a serem tratadas.

Com o avanço das tecnologias, os Grandes Dados (**Big Data**) passaram a desempenhar um papel fundamental no mercado, tornando o armazenamento e a análise de grandes volumes de informações algo mais simples, ágil e eficiente, por meio da utilização dos Bancos de Dados. Esses sistemas permitem não apenas guardar dados, mas também organizá-los, cruzá-los e extrair insights valiosos que contribuem diretamente para a tomada de decisões estratégicas.

Além disso, a adoção de bancos de dados modernos e integrados tem possibilitado maior segurança, escalabilidade e automação dos processos empresariais, reduzindo erros humanos e aumentando a produtividade. Dessa forma, investir em tecnologias de gerenciamento de dados e em profissionais capacitados nessa área tornou-se essencial para empresas que desejam se manter competitivas e inovadoras no cenário atual.

2. MINI-MUNDO

Dando prosseguimento ao Trabalho 1, eu optei por desenvolver o diagrama lógico do meu Banco de Dados, tomando como referência um futuro software voltado para o gerenciamento de uma academia. O objetivo principal desse sistema é facilitar o processo de cadastro de alunos, bem como otimizar a organização, consulta e manutenção das informações dentro da empresa.

Por meio dessa estrutura, será possível armazenar e relacionar diversos tipos de dados essenciais para o funcionamento da academia, garantindo maior controle, eficiência e precisão nas operações diárias. Entre as principais entidades que compõem o banco de dados, destacam-se: ALUNO, INSTRUTOR, PLANO, TREINO, TREINO–EXERCÍCIO e PAGAMENTOS, cada uma delas contendo suas respectivas variáveis e atributos específicos, que serão apresentados de forma detalhada posteriormente.

Esse projeto tem como finalidade demonstrar a importância da modelagem de dados na construção de sistemas eficazes e bem estruturados. A criação do diagrama lógico permitirá compreender de forma mais clara os relacionamentos entre as tabelas e a maneira como as informações circulam dentro do sistema, servindo de base para o desenvolvimento de um software funcional, seguro e de fácil utilização.

Além disso, o banco de dados proposto contribuirá significativamente para a automatização de tarefas que, até então, são realizadas manualmente, reduzindo erros e aumentando a produtividade da equipe administrativa. Dessa forma, o sistema proporcionará uma melhor experiência tanto para os gestores da academia quanto para os alunos, que poderão ter seus dados e planos gerenciados de maneira mais prática e eficiente.

3. DIAGRAMA LÓGICO DO BANCO DE DADOS

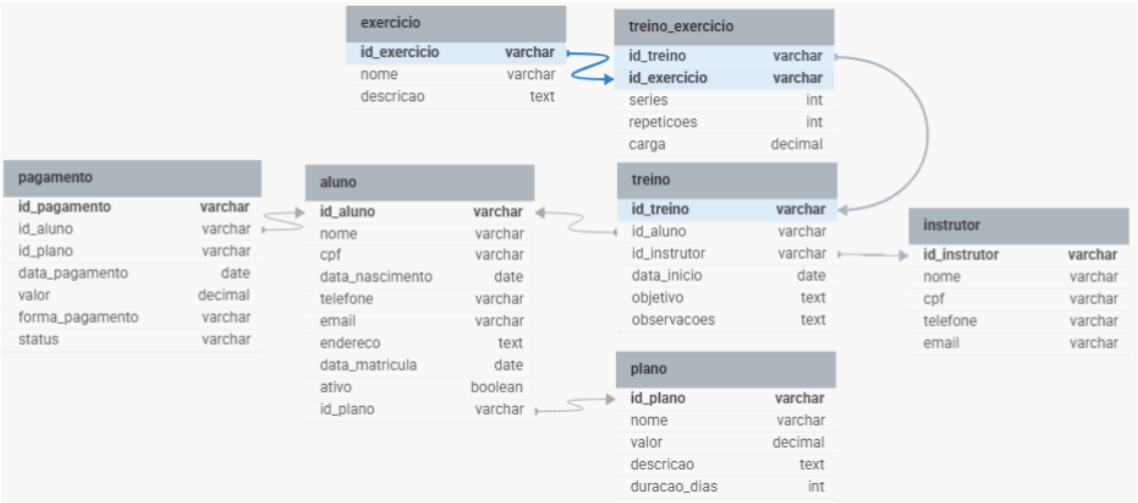

Figure 1. Diagrama lógico da academia

.

4. Criação do banco e tabelas

Nessa seção irei apresentar o comando 'CREATE DATABASE()' que irá gerar O **BANCO DE DADOS**. Já o comando CREATE TABLE () irá gerar as tabelas utilizados no modelo lógico.

```
CREATE DATABASE academia;

CREATE TABLE plano ();
CREATE TABLE aluno ();
CREATE TABLE instrutor ();
CREATE TABLE treino ();
CREATE TABLE exercício ();
CREATE TABLE treino_exercício ();
CREATE TABLE pagamento ();
```

Figure 2. Caption

5. Povoamento e consultas do BD

Para configurar esse Banco de Dados de maneira à facilitar a organização de dados dos clientes e evitar falta estrutura dos dados, separei cada tabela e destrinchei os comandos SQL utilizados para esse povoamento. **OBS: nas fotos coloquei cochetes para facilitar a visualização, na hora de passar para o Banco de Dados os cochetes serão retirados para não dar erros de sintaxe**.

```
CREATE TABLE [plano] (
  [id_plano] nvarchar(50) NOT NULL,
  [nome] nvarchar(50) NOT NULL,
  [valor] decimal(10,2) NOT NULL,
  [descricao] nvarchar(300) NOT NULL,
  [duracao_dias] int NOT NULL,
);
```

Figure 3. Plano dos alunos

Os planos terão as variáveis "ID PLANO,NOME, VALOR, DESCRIÇÃO,DURAÇÃO DIAS" sendo todos NOT NULL(Não podem estar em branco).

```
CREATE TABLE [aluno] (
    [id_aluno] nvarchar(50) NOT NULL,
    [nome] nvarchar(255) NOT NULL,
    [cpf] nvarchar(14) NOT NULL UNIQUE,
    [data_nascimento] date NOT NULL,
    [telefone] nvarchar(15) NOT NULL,
    [email] nvarchar(100) NOT NULL,
    [endereco] nvarchar(100) NOT NULL,
    [data_matricula] date NOT NULL,
    [ativo] bit NOT NULL,
    [id_plano] nvarchar(50) NOT NULL,
);
```

Figure 4. Dados do aluno cadastrado

```
CREATE TABLE [instrutor] (
  [id_instrutor] nvarchar(50) NOT NULL,
  [nome] nvarchar(100) NOT NULL,
  [cpf] nvarchar(14) NOT NULL UNIQUE,
  [telefone] nvarchar(15) NOT NULL,
  [email] nvarchar(100) NOT NULL,
);
```

Figure 5. Cadastro dos instrutores

Nas tabelas ALUNO e INSTRUTOR terão todos os dados deles para que possam entrar na academia, como:
id*aluno: identificador único de cada aluno;
nome: nome completo do aluno;
cpf: número do CPF, utilizado para identificação e controle;
nascimento: data de nascimento do aluno;
telefone: número de contato;
email: endereço eletrônico para comunicação;
endereço: local de residência do aluno;
data*matricula: data em que o aluno foi cadastrado na academia;
ativo: campo que indica se o aluno está com a matrícula ativa ou inativa;
id*plano: chave estrangeira que referencia o plano contratado pelo aluno, relacionando-se com a tabela de planos da academia. e **INSTRUTOR**(id instrutor, nome, cpf, telefone, email).

```
CREATE TABLE [treino_exercicio] (
  [id_treino] nvarchar(50) NOT NULL,
  [id_exercicio] nvarchar(50) NOT NULL,
  [series] int NOT NULL,
  [repeticoes] int NOT NULL,
  [carga] decimal(5,2) NOT NULL,
);
```

Figure 6. treinos*exercícios

```
CREATE TABLE [exercicio] (
  [id_exercicio] nvarchar(50) NOT NULL,
  [nome] nvarchar(100) NOT NULL,
  [descricao] nvarchar(300) NOT NULL,
);
```

Figure 7. exercício

A tabela **EXERCICIO** tem como objetivo armazenar todos os exercícios cadastrados no sistema, servindo como base para a criação dos treinos.

id*exercicio: identificador único de cada exercício (chave primária);
nome: nome do exercício, permitindo rápida identificação;
descricao: campo destinado à explicação detalhada sobre o exercício, podendo incluir informações sobre a forma de execução, músculos trabalhados e observações técnicas.

Já a tabela **TREINO*EXERCICIO** é uma tabela de relacionamento entre os treinos e os exercícios. Ela define quais exercícios pertencem a cada treino, além de registrar informações específicas de execução, como quantidade de séries, repetições e carga utilizada.

id*treino: identificador do treino ao exercício.
id*exercicio: identificador do exercício incluído no treino (chave estrangeira que referencia a tabela EXERCICIO).
series: número de séries a serem realizadas para o exercício.
repeticoes: quantidade de repetições por série.
carga: peso ou resistência aplicada durante a execução do exercício, armazenado.

```
CREATE TABLE [treino] (
  [id_treino] nvarchar(50) NOT NULL,
  [id_aluno] nvarchar(50) NOT NULL,
  [id_instrutor] nvarchar(50) NOT NULL,
  [data_inicio] date NOT NULL,
  [objetivo] nvarchar(300) NOT NULL,
  [observacoes] nvarchar(300) NOT NULL,
);
```

Figure 8. tabela de treinos

A tabela TREINO tem como finalidade armazenar todas as informações referentes aos planos de treino elaborados para os alunos da academia.

id*treino: identificador de cada treino.

id*aluno: chidentifica o aluno responsável por esse treino.

id*instrutor: indica o instrutor responsável pela elaboração e acompanhamento do treino.

data*inicio: data em que o treino foi iniciado.

objetivo: descrição detalhada do propósito do treino.

observacoes: campo destinado a anotações.

```
CREATE TABLE [pagamento] (
  [id_pagamento] nvarchar(50) NOT NULL,
  [id_aluno] nvarchar(50) NOT NULL,
  [id_plano] nvarchar(50) NOT NULL,
  [data_pagamento] date NOT NULL,
  [valor] decimal(10,2) NOT NULL,
  [forma_pagamento] nvarchar(50) NOT NULL,
  [status] nvarchar(20) NOT NULL,
);
```

Figure 9. Meios de pagamento

A tabela PAGAMENTO tem como finalidade registrar todas as transações financeiras realizadas pelos alunos da academia, garantindo o controle e a transparência dos pagamentos efetuados. Cada registro corresponde a um pagamento específico de um aluno, relacionado a um determinado plano contratado, tendo como atributos:
id*pagamento: identificador único de cada pagamento;
id*aluno: chave estrangeira que identifica o aluno;
id*plano: chave estrangeira que referencia o plano contratado pelo aluno;
data*pagamento: data em que o pagamento foi efetuado;
valor: valor total pago pelo aluno, armazenado em formato decimal para representar corretamente valores monetários;
forma*pagamento: meio de pagamento utilizado na transação, podendo ser, por exemplo, dinheiro, cartão de crédito, cartão de débito, transferência bancária ou PIX;
status: campo que indica a situação do pagamento, como “pago”, “pendente” ou “atrasado.

6. Chaves primárias(PK)

As chaves primárias são geralmente ID’s que garantem que aquele número não poderá se repetir como o id de cada um (id*aluno, id*instrutor, id*treino). Essa mecânica facilita não hora de identificar algum dado com facilidade, para isso criei uma imagem com todas as chaves primárias de cada tabela criada nesse projeto.

```
ALTER TABLE aluno
ADD CONSTRAINT pk_aluno PRIMARY KEY (id_aluno)
ALTER TABLE instrutor
ADD CONSTRAINT pk_instrutor PRIMARY KEY (id_instrutor)
ALTER TABLE exercicio
ADD CONSTRAINT pk_exercicio PRIMARY KEY (id_exercicio)
ALTER TABLE plano
ADD CONSTRAINT pk_plano PRIMARY KEY (id_plano)

ALTER TABLE treino
ADD CONSTRAINT pk_treino PRIMARY KEY (id_treino)
ALTER TABLE treino_exercicio
ADD CONSTRAINT pk_treino_exercicio PRIMARY KEY
(id_treino, id_exercicio)
ALTER TABLE pagamento
ADD CONSTRAINT pk_pagamento PRIMARY KEY
(id_pagamento)
```

Figure 10. Caption

7. Chaves estrangeiras(FK)

As **chaves estrangeiras (FK)** têm a função de criar vínculos entre duas tabelas, conectando os dados de forma lógica e garantindo a integridade dentro do banco de dados.

Elas estabelecem relacionamentos entre tabelas, permitindo que as informações fiquem organizadas e associadas corretamente.

References

```
ALTER TABLE aluno
ADD CONSTRAINT fk_aluno_plano
FOREIGN KEY (id_plano) REFERENCES plano(id_plano)

ALTER TABLE treino
ADD CONSTRAINT fk_treino_aluno
FOREIGN KEY (id_aluno) REFERENCES aluno(id_aluno);

ALTER TABLE treino
ADD CONSTRAINT fk_treino_instrutor
FOREIGN KEY (id_instrutor) REFERENCES instrutor(id_instrutor)


ALTER TABLE treino_exercicio
ADD CONSTRAINT fk_treino_exercicio_treino
FOREIGN KEY (id_treino) REFERENCES treino(id_treino);

ALTER TABLE treino_exercicio
ADD CONSTRAINT fk_treino_exercicio_exercicio
FOREIGN KEY (id_exercicio) REFERENCES exercicio(id_exercicio)

ALTER TABLE pagamento
ADD CONSTRAINT fk_pagamento_aluno
FOREIGN KEY (id_aluno) REFERENCES aluno(id_aluno)
```