Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2020/2021

Ginásio Académico

José Silva (A68243)
Francisco Peixoto (A84668)
Renato Gomes (A84696)
Ricardo Gomes (A93785)

Novembro, 2020

Conteúdo

Índice de Figuras

Introdução

No âmbito da Unidade Curricular de Bases de Dados, foi realizada uma base de dados relacional cujo tema escolhido pelo grupo foi um Ginásio Académico.

Ao longo deste relatório serão apresentados todos os passos para criação do sistema em causa, desde o Modelo Conceptual até ao Modelo Físico.

Numa primeira fase de contextualização, são apresentados os requisitos levantados, assim como a apresentação de um Modelo ER, utilizando a ferramenta brModelo. É também detalhado e apresentado um modelo lógico utilizando o MySQL Workbench.

Finalmente, é apresentada a implementação física da base de dados, representando então a passagem do modelo lógico para o modelo físico, assim como a passagem dos requisitos estabelecidos anteriormente pelo utilizador e pelo grupo para a linguagem SQL.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados no âmbito da utilização e funcionamento de um ginásio universitário.

Palavras-Chave: Bases de Dados Relacionais, Modelo Conceptual, Modelo Lógico, Modelo Físico, MySQL Workbench, SQL, Análise de Requisitos, Atributos, Entidades, Relacionamentos

1. Definição do SGBD[1]

1.1. Definição do Sistema

A fim de simplificar o armazenamento e tratamento de dados de um Ginásio Académico, pretende-se criar um sistema de Base de Dados capaz de suportar diversas operações tais como, inserção e eliminação de novos funcionários, clientes inscritos e mesmo até aulas, controlo de sessões de nutrição bem como o controlo de sessões de PT.

Para esta Base de Dados está prevista a capacidade de conseguir guardar determinadas tabelas associadas às diferentes entidades, e até mesmo informação relativa à interação de diferentes entidades.

Iniciaremos esta implementação pela breve contextualização da situação, descrevendo assim, com mais detalhe, os motivos que levaram à criação deste sistema.

1.2. Contextualização

Uma instituição académica pretende abrir um ginásio aberto tanto a público externo ao núcleo de estudantes como a todos os inscritos no ano letivo e funcionários da mesma, na presente altura de pandemia de Covid-19.

Para tal é necessário um método eficaz de manter o registo de todas as diversas inscrições e permitir assim um fácil e rápido acesso às instalações por parte dos utentes, mas também um modo seguro de organizar toda a informação dos inscritos de forma a que caso seja diagnosticado uma ocorrência de Covid-19, o mesmo poderá ser rapidamente identificado pelo horário no qual frequentou a acomodação e os restantes utilizadores serão posteriormente avisados da ocorrência para prevenção de mais transmissões.

1.3. Análise da Viabilidade do Projeto

Com o projeto relativo à abertura do novo ginásio pretende-se preencher um vazio no mercado atual, pelo que será algo de novo no contexto atual desta universidade. Mas para isto, é necessária uma análise mais aprofundada do tema, de forma a garantir que há uma boa infraestrutura capaz de alcançar os objetivos propostos.

Com o intuito de realizar esta façanha, é importante ter em conta a questão administrativa relativamente à gestão do ginásio. Posto isto, foram elaborados planos de crescimento sustentável que têm por base um aumento credível no que toca às inscrições, dado o número de alunos que frequentam o campus ser avultado.

[1] Sistema de Gestão de Base de Dados

É de salientar ainda que, quanto maior for a quantidade de clientes inscritos, mais funcionários serão necessários contratar, fazendo este projeto aumentar o seu volume de dados e consequentemente a receita.

Para além disso, esta iniciativa permite uma resposta rápida às mais diversas necessidades de quem frequenta o campus, podendo agora usufruir das instalações do ginásio a preços muito abaixo do nível de mercado, que só se conseguem no âmbito académico.

1.4. Motivação e Objetivos

O desenvolvimento da Base de Dados, motivado primeiramente pelos diversos pontos apresentados em cima, procura também simplificar as relações entre utente/funcionário do ginásio, facilitar o acesso à organização, consulta de dados, registos e permitir a rápida entrada e saída dos usuários com os diversos tipos de inscrição dependendo do género de utente (estudante, estudante-atleta, funcionário, externo, etc).

2. Levantamento e Análise de requisitos

Os requisitos relacionados com a base de dados foram obtidos através de entrevistas aos possíveis utilizadores da mesma, tal como PTs, nutricionistas, alunos ou mesmo atletas e membros do staff do ginásio onde foram compreendidos e documentados os seus requisitos.

2.1. Entrevista aos potenciais utilizadores da BD

Como já dito anteriormente, foi realizada uma entrevista aos potenciais utilizadores da base de dados, tendo sido este o método de levantamento de requisitos utilizados. Na entrevista foram feitas perguntas que nos permitissem saber de que forma deveríamos organizar a base de dados de modo a que fosse de fácil acesso para todos. Durante estas entrevistas foram feitas as seguintes perguntas:

1- Como funciona o ginásio?
2- Tipo de membros?
3- Quantos atletas federados?
4- Quantos vão ao nutricionista do ginásio?
5- Quantos planos de adesão e as suas diferenças?
6- Quantos requisitam um Personal Trainer?

Requisitos de identificação:

- Cada membro ao ser acrescentado ao sistema deve fornecer o seu nome, data de nascimento, localidade, altura e peso, sendo associado o ID de membro e plano de inscrição.
- Cada nutricionista contém um ID, nome, e o ID do ginásio
- Cada consulta contém um ID, bem como o ID do nutricionista e o do membro que vai à consulta
- Cada aula contém um nome e um ID, bem como o ID do membro inscrito e o do PT.
- Cada tipo de inscrição tem um ID e nome associado
- Cada PT tem um ID e nome associado bem como o ID do ginásio onde trabalha.
- Cada membro do staff tem um id e um nome, e o ID do ginásio.
- Cada ginásio tem um nome e um ID.

Requisitos de exploração:

Um membro pode:

- Dado o seu ID, aceder às suas marcações de consultas e de aulas, bem como o plano de inscrição.

Um nutricionista pode:

- Dado o seu ID, aceder às suas marcações de consultas.
- Dado um ID de um membro, aceder às informações do mesmo
- Dado um ID de uma consulta, aceder às informações da mesma.

Um PT pode:

- Dado um ID de um membro, aceder à sua informação.
- Dado um ID de uma aula, aceder às suas informações
- Dado o seu ID pode aceder às marcações das aulas.

Um membro do staff do ginásio pode:

- Dado um ID de um membro do ginásio, aceder às informações do mesmo, como aulas e consultas.
- Dado um ID de um nutricionista, aceder às informações do mesmo, incluindo as suas consultas.
- Dado um ID de um PT, aceder às informações do mesmo, incluindo as suas aulas.
- Dado um ID de um tipo de inscrição, saber quais os membros que nela estão inscritos

Requisitos de controlo:

- Um membro do staff do ginásio pode marcar consultas e aulas para os membros
- Um membro do staff do ginásio pode alterar informação sobre qualquer membro
- Um nutricionista pode alterar informação sobre qualquer consulta que tenha sido responsável.
- Um PT pode alterar informação sobre qualquer aula que tenha sido responsável.

3. Modelação Conceptual

3.1. Apresentação da abordagem de modelação realizada

De forma a criar o Modelo Conceptual foram definidos todos os requisitos necessários para a implementação da base de dados. Neste modelo podemos verificar as várias relações que as entidades apresentam umas com as outras, sendo isto bastante importante para o desenvolvimento de qualquer base de dados.

De forma a representar o Modelo, foram seguidos os seguintes passos:

I. Foram identificadas as Entidades;
II. Foram identificadas as Relações entre Entidades;
III. Foi determinado o domínio dos Atributos;
IV. Foram determinadas as chaves primárias, candidatas e estrangeiras;
V. Foi elaborado o diagrama ER;
VI. Validou-se o modelo de dados com o utilizador;

3.2. Identificação e caraterização de Entidades

As principais Entidades são:

- Membro: entidade que representa um membro do ginásio. Esta entidade interage com o sistema para se inscrever em Aulas e em Consultas.
- Ginásio: entidade que representa o ginásio, ao qual estão associados os seus Membros e o seu Staff.
- Aula: entidade que representa as aulas frequentadas pelo membro.
- Consulta: entidade que representa as consultas efetuadas pelo membro.

Tabela 1 - Identificação e caraterização de Identidades

Entidades	Descrição	Aliases	Ocorrências
Membro	Quem se inscreve em aulas e consultas de um ginásio	Cliente	Um membro inscreve-se no ginásio, e posteriormente em aulas e consultas.
Ginásio	Ao qual estão associados os seus membros e os seus funcionários	Estabelecimento	Um ginásio regista os membros inscritos e os elementos do seu staff
Staff	Funcionário que trabalha no ginásio	Funcionário	Funcionário do ginásio que realiza diversas tarefas
PT	Funcionário do ginásio que é PT	Funcionário	Funcionário do ginásio que dá aulas
Nutricionista	Funcionário do ginásio que é nutricionista	Funcionário	Funcionário do ginásio que dá consultas
Consulta	Consulta dada a um membro do ginásio	Consulta	Uma consulta é realizada por um nutricionista a um membro
Aula	Aula realizada no ginásio por um membro	Aula	Uma aula é dada por um PT a um membro
Tipo de Inscrição	Tipo de inscrição de um membro	Tipo de Inscrição	Tipo de inscrição que um membro faz no momento de inscrição no ginásio

3.3. Identificação e caracterização de relacionamentos

No modelo conceptual construído existem os seguintes relacionamentos:

Um membro está associado a um ginásio que frequenta. Um membro só está inscrito num ginásio e um ginásio pode ter vários membros inscritos, logo a relação é de N para 1.

Um membro está associado a um tipo de inscrição. Um membro apenas pode estar associado a um tipo de inscrição, mas um tipo de inscrição pode ter vários membros associados, logo a relação é de N para 1.

Um membro está associado a uma consulta. Um membro pode realizar várias consultas, mas uma consulta apenas pode ter um membro associado, logo a relação é de 1 para N.

Um membro está associado a uma aula. Um membro pode participar em várias aulas, mas uma aula apenas é dada a um membro, logo a relação é de 1 para N.

Um elemento do staff está associado ao ginásio onde trabalha. Um elemento do staff apenas pode trabalhar num ginásio, mas um ginásio pode ter vários elementos do staff, logo a relação é de N para 1.

Um PT está associado a uma aula. Um PT pode dar várias aulas, mas uma aula apenas pode ser dada por um PT, logo a relação é de 1 para N.

Um PT está associado ao ginásio onde trabalha. Um PT apenas pode trabalhar num ginásio, mas um ginásio pode ter vários PTs, logo a relação é de N para 1.

Um nutricionista está associado a uma consulta. Um nutricionista pode realizar várias consultas, mas uma consulta apenas pode ser dada por um nutricionista, logo a relação é de 1 para N.

Um nutricionista está associado ao ginásio onde trabalha. Um nutricionista apenas pode trabalhar num ginásio, mas um ginásio pode ter vários nutricionistas, logo a relação é de N para 1.

Tabela 2 - Identificação e caracterização das Associações dos Atributos com as Entidades

Entidade	**Relação**	**Entidade**	Multiplicidade
Membro	Inscrito	Ginásio	(1, N)
Membro	Possui	Tipo de Inscrição	(N, 1)
Membro	Tem	Consulta	(1, N)
Membro	Participa	Aula	(1, N)
Staff	Trabalha	Ginásio	(N, 1)
PT	Realiza	Aula	(1, N)
PT	Trabalha	Ginásio	(N, 1)
Nutricionista	Realiza	Consulta	(1, N)
Nutricionista	Trabalha	Ginásio	(N, 1)

3.4. Detalhe das Entidades e caracterização de Atributos

Tabela 3 - Detalhes sobre as Entidades

Entidades	Atributos	Descrição	Tipo de Dados	Null	MultiValores	Derivados
Ginásio	id_ginásio	Identificador do ginásio	INT	Não	Não	Não
	nome_ginásio	Nome do ginásio	VARCHAR (45)	Sim	Não	Não
Membro	id_membro	Identificador do membro	INT	Não	Não	Não
	nome_membro	Nome do membro	VARCHAR (45)	Sim	Não	Não
	peso_membro	Peso do membro	DECIMAL	Sim	Não	Não
	altura_membro	Altura do membro	DECIMAL	Sim	Não	Não
	localidade_membro	Localidade do membro	VARCHAR (45)	Sim	Não	Não
	data_nascimento_membro	Data de nascimento do membro	DATE	Sim	Não	Não
Tipo de inscrição	id_tipo	Identificador do tipo de inscrição	INT	Não	Não	Não
	nome_tipo	Nome do tipo de inscrição	VARCHAR (45)	Sim	Não	Não
Aulas	id_aulas	Identificador das aulas	INT	Não	Não	Não
	nome_aulas	Nome das aulas	VARCHAR (45)	Sim	Não	Não
	data_aulas	Data das aulas	DATETIME	Não	Não	Não
PT	id_pt	Identificador do personal trainer	INT	Não	Não	Não
	nome_pt	Nome do personal trainer	VARCHAR (45)	Sim	Não	Não
Staff	id_staff	Identificador do elemento do staff	INT	Não	Não	Não
	nome_staff	Nome do elemento do staff	VARCHAR (45)	Sim	Não	Não
Nutricionista	id_nutricionista	Identificador do nutricionista	INT	Não	Não	Não
	nome_nutricionista	Nome do nutricionista	VARCHAR (45)	Sim	Não	Não
Consulta	id_consulta	Identificador da consultai	INT	Não	Não	Não
	data_consulta	Data da consulta	DATETIME	Não	Não	Não
	tipo_consulta	Tipo da consulta	VARCHAR (45)	Sim	Não	Não

3.5. Apresentação do diagrama ER

Utilizando a ferramenta brModelo construímos o seguinte Diagrama ER, através deste diagrama podemos observar que todos os requisitos vistos anteriormente são comprimdos.

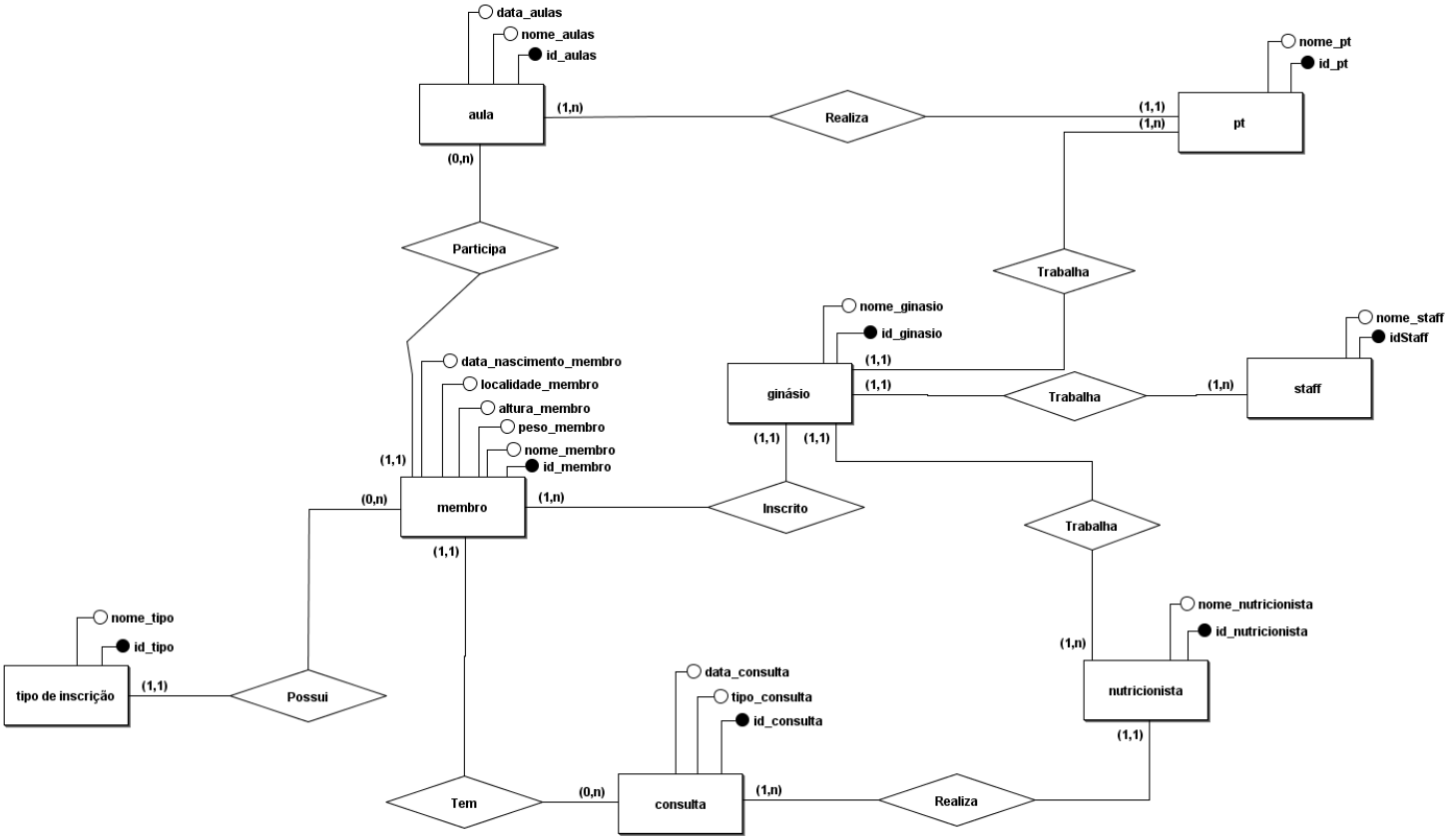

Figura 1 - Diagrama ER

4. Modelação Lógica

4.1. Construção e validação do Modelo de dados Lógico

Nesta fase do projeto, de acordo com o trabalho realizado no Modelo Conceptual, elaboramos o Modelo Lógico. Começou-se por criar uma tabela para cada entidade do Modelo Conceptual.

É importante realçar que quando existem relações entre tabelas com multiplicidade N:M, é necessária também a criação de uma tabela para essa relação, como é o caso da relação Membro-Personal Trainer ou Membro-Nutricionista.

Posto isto, procedeu-se ao preenchimento de cada tabela com os respetivos atributos definidos no Modelo Conceptual.

4.2. Desenho do Modelo Lógico

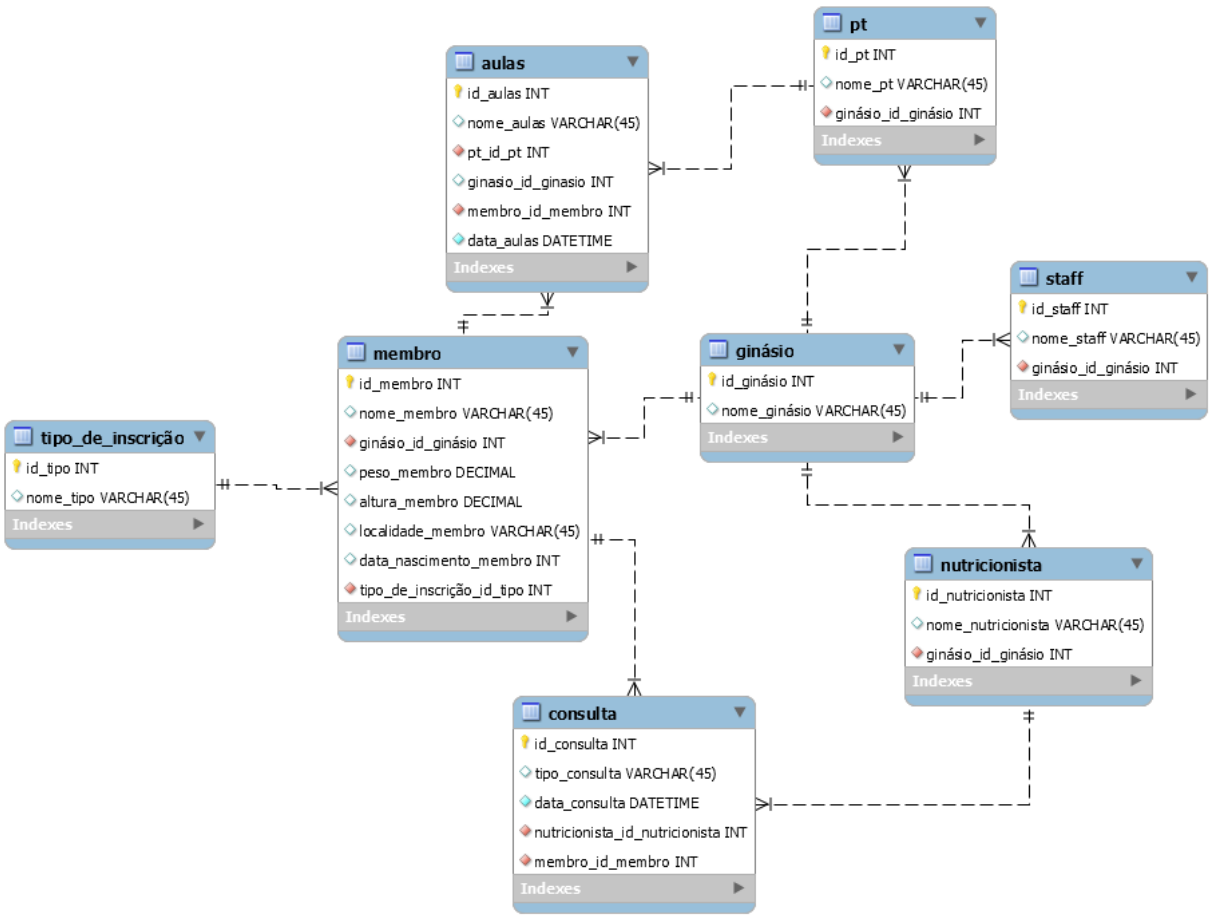

Figura 2 - Modelo Lógico

4.3. Validação do modelo através da normalização

A função principal da normalização é identificar um conjunto adequado de relações que suportem os requisitos da base de dados, para tal é necessário identificar as dependências funcionais existentes entre os atributos.

Primeira Forma Normal (1FN)

Uma relação está na 1ª Forma Normal se:

- Cada atributo contém apenas valores atómicos;

- Não há conjuntos de atributos repetidos a descrever a mesma característica.

Segunda Forma Normal (2FN)

Uma relação está na 2ª Forma Normal se:

- Pertencer à 1FN;

- Todos os atributos que não são chaves, dependem completamente da chave primária.

Terceira Forma Normal (3FN)

Uma relação está na 3ª Forma Normal se:

- Pertencer à 2FN;

- Todos os atributos que não são chaves, dependem completamente da chave primária e não dependem de um outro atributo que, por sua vez, dependesse da chave primária.

Posto isto, sabemos que o nosso Modelo Lógico segue as três primeiras regras de normalização:

1FN – Confirmamos que, em todas as tabelas, cada valor da sua chave primária corresponde a uma única linha da respetiva tabela. Podemos verificar isto nas tabelas Ginásio e Tipo de Inscrição, onde cada um tem o seu ID, que é único. Já, por exemplo, nas tabelas Membro e PT, têm chaves compostas constituídas por chaves estrangeiras. Tendo em conta que sabemos que as chaves estrangeiras destas são únicas, sabemos, então, por transitividade, que as chaves compostas destas tabelas também são únicas.

2FN – Verificou-se que não existem dependências parciais no Modelo Lógico, porque os dados que seriam repetidos nas tabelas têm a sua própria tabela (e.g.: tabela Consulta ou tabela Aulas).

3FN – Verificou-se que o modelo não tem dependências transitivas, isto é, não há atributos redundantes nas tabelas.

4.4. Revisão do modelo lógico com o utilizador

Tal como anteriormente, a nossa aplicação foi posta à prova por utilizadores, a fim de perceber se as suas necessidades estavam a ser tidas em conta e a ser respeitadas.

Partindo do princípio que o Utilizador não tem conhecimentos sobre Bases de Dados ou SGBD, procuramos não envolver demasiados termos e conceitos técnicos, de modo a facilitar a compreensão da aplicação para o mesmo. Deste modo, o Modelo Lógico foi mostrado e explicado detalhadamente, em linguagem corrente, possibilitando também que todas as dúvidas que surgissem fossem devidamente esclarecidas. Em suma, após todas as explicações necessárias, concluímos que o modelo foi aprovado pelos utilizadores.

5. Implementação Física

5.1. Seleção do Sistema de Gestão de Base de Dados

Nesta fase do projeto o grupo realizou a migração do Modelo Lógico para o Modelo Físico, que envolve a escolha de um Sistema de Gestão de Base de dados, neste caso, o sistema usado foi a ferramenta MySQL Workbench.

5.2. Tradução do esquema lógico para o SGBD escolhido em SQL

Após a transcrição do Modelo Lógico para o SGBD e de forma a proceder com a criação de tabelas, usa-se a ferramenta *Forward Engeneering* disponibilizada pelo MySQL Workbench.

```
CREATE TABLE IF NOT EXISTS `mydb`.`membro` (
  `id_membro` INT NOT NULL,
  `nome_membro` VARCHAR(45) NULL,
  `ginásio_id_ginásio` INT NOT NULL,
  `peso_membro` DECIMAL NULL,
  `altura_membro` DECIMAL NULL,
  `localidade_membro` VARCHAR(45) NULL,
  `data_nascimento_membro` INT NULL,
  `tipo_de_inscrição_id_tipo` INT NOT NULL,
  PRIMARY KEY (`id_membro`),
  INDEX `fk_membro_ginásio1_idx` (`ginásio_id_ginásio` ASC) VISIBLE,
  CONSTRAINT `fk_membro_tipo de inscrição`
    FOREIGN KEY (`tipo_de_inscrição_id_tipo`)
    REFERENCES `mydb`.`tipo_de_inscrição` (`id_tipo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_membro_ginásio1`
    FOREIGN KEY (`ginásio_id_ginásio`)
    REFERENCES `mydb`.`ginásio` (`id_ginásio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 3 - Criação da tabela Membro

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`ginásio` (
  `id_ginásio` INT NOT NULL,
  `nome_ginásio` VARCHAR(45) NULL,
  PRIMARY KEY (`id_ginásio`))
ENGINE = InnoDB;
```

Figura 4 - Criação da tabela Ginásio

```
CREATE TABLE IF NOT EXISTS `mydb`.`nutricionista` (
  `id_nutricionista` INT NOT NULL,
  `nome_nutricionista` VARCHAR(45) NULL,
  `ginásio_id_ginásio` INT NOT NULL,
  PRIMARY KEY (`id_nutricionista`),
  INDEX `fk_nutricionista_ginásio1_idx` (`ginásio_id_ginásio` ASC) VISIBLE,
  CONSTRAINT `fk_nutricionista_ginásio1`
    FOREIGN KEY (`ginásio_id_ginásio`)
    REFERENCES `mydb`.`ginásio` (`id_ginásio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 5 - Criação da tabela Nutricionista

```
CREATE TABLE IF NOT EXISTS `mydb`.`pt` (
  `id_pt` INT NOT NULL,
  `nome_pt` VARCHAR(45) NULL,
  `ginásio_id_ginásio` INT NOT NULL,
  PRIMARY KEY (`id_pt`),
  INDEX `fk_pt_ginásio1_idx` (`ginásio_id_ginásio` ASC) VISIBLE,
  CONSTRAINT `fk_pt_ginásio1`
    FOREIGN KEY (`ginásio_id_ginásio`)
    REFERENCES `mydb`.`ginásio` (`id_ginásio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 6 - Criação da tabela PT

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`tipo_de_inscrição` (
  `id_tipo` INT NOT NULL,
  `nome_tipo` VARCHAR(45) NULL,
  PRIMARY KEY (`id_tipo`))
ENGINE = InnoDB;
```

Figura 7 - Criação da tabela Tipo de Inscrição

```
CREATE TABLE IF NOT EXISTS `mydb`.`staff` (
  `id_staff` INT NOT NULL,
  `nome_staff` VARCHAR(45) NULL,
  `ginásio_id_ginásio` INT NOT NULL,
  PRIMARY KEY (`id_staff`),
  INDEX `fk_staff_ginásio1_idx` (`ginásio_id_ginásio` ASC) VISIBLE,
  CONSTRAINT `fk_staff_ginásio1`
    FOREIGN KEY (`ginásio_id_ginásio`)
    REFERENCES `mydb`.`ginásio` (`id_ginásio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 8 - Criação da tabela Staff

```
CREATE TABLE IF NOT EXISTS `mydb`.`consulta` (
  `id_consulta` INT NOT NULL,
  `tipo_consulta` VARCHAR(45) NULL,
  `data_consulta` DATETIME NOT NULL,
  `nutricionista_id_nutricionista` INT NOT NULL,
  `membro_id_membro` INT NOT NULL,
  PRIMARY KEY (`id_consulta`),
  INDEX `fk_consulta_nutricionista1_idx` (`nutricionista_id_nutricionista` ASC) VISIBLE,
  CONSTRAINT `fk_consulta_membro1`
    FOREIGN KEY (`membro_id_membro`)
    REFERENCES `mydb`.`membro` (`id_membro`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_consulta_nutricionista1`
    FOREIGN KEY (`nutricionista_id_nutricionista`)
    REFERENCES `mydb`.`nutricionista` (`id_nutricionista`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 9 - Criação da tabela Consulta

```sql
CREATE TABLE IF NOT EXISTS `mydb`.`aulas` (
  `id_aulas` INT NOT NULL,
  `nome_aulas` VARCHAR(45) NULL,
  `pt_id_pt` INT NOT NULL,
  `ginasio_id_ginasio` INT NULL,
  `membro_id_membro` INT NOT NULL,
  `data_aulas` DATETIME NOT NULL,
  PRIMARY KEY (`id_aulas`),
  INDEX `fk_aulas_membro1_idx` (`membro_id_membro` ASC) VISIBLE,
  INDEX `fk_aulas_pt1_idx` (`pt_id_pt` ASC) VISIBLE,
  CONSTRAINT `fk_aulas_pt1`
    FOREIGN KEY (`pt_id_pt`)
    REFERENCES `mydb`.`pt` (`id_pt`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_aulas_membro1`
    FOREIGN KEY (`membro_id_membro`)
    REFERENCES `mydb`.`membro` (`id_membro`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 10 - Criação da tabela Aulas

5.3. Tradução das interrogações do Utilizador para SQL

```sql
DELIMITER //
drop procedure if exists getAulasPorMembro;
create procedure getAulasPorMembro (IN id_membro INT)

    begin
        select a.nome_aulas as Aula, a.data_aulas as 'Data'
        from aulas a
        inner join membro m on a.membro_id_membro = m.id_membro
        where a.membro_id_membro = id_membro;
    end //

DELIMITER //
```

Figura 11- Dado um ID de um membro ver as aulas que este está inscrito

```
DELIMITER //

drop procedure if exists getInfoAula;
create procedure getInfoAula (IN id_aula INT)

    begin

        select m.nome_membro as Nome, a.nome_aulas as Tipo, p.nome_pt as PT, a.data_aulas as 'Data'
        from aulas a
        inner join membro m on m.id_membro = a.membro_id_membro
        inner join pt p on p.id_pt = a.pt_id_pt
        where a.id_aulas = id_aula;
    end //

DELIMITER //
```

Figura 12- Dado o ID de uma aula saber todas as suas informações

```
DELIMITER //

drop procedure if exists getConsultasPorMembro;
create procedure getConsultasPorMembro (IN id_membro INT)

    begin

        select c.tipo_consulta as Tipo , c.data_consulta as 'Data' , n.nome_nutricionista as Nutricionista
        from consulta c
        left join nutricionista as n on c.nutricionista_id_nutricionista = n.id_nutricionista
        where c.membro_id_membro = id_membro;
    end //

DELIMITER //
```

Figura 13- Dado o ID de um membro saber quais consultas tem marcadas

```
DELIMITER //

drop procedure if exists getInfoConsulta;
create procedure getInfoConsulta (IN id_consulta INT)

    begin

        select m.nome_membro as Nome, c.tipo_consulta as Tipo, n.nome_nutricionista as Nutricionista, c.data_consulta as 'Data'
        from consulta c
        inner join membro m on m.id_membro = c.membro_id_membro
        inner join nutricionista n on n.id_nutricionista = c.nutricionista_id_nutricionista
        where c.id_consulta = id_consulta;
    end //

DELIMITER //
```

Figura 14- Dado o ID de uma consulta saber as informações dessa consulta

```
DELIMITER //

drop procedure if exists getInfoMembro;
create procedure getInfoMembro (IN id_membro INT)

    begin

        select m.nome_membro as Nome, m.peso_membro as Peso, m.altura_membro as Altura, m.data_nascimento_membro as 'Data Nascimento', m.localidade_membro as Localidade, i.nome_tipo as 'Tipo de Inscrição'
        from membro m
        inner join tipo_de_inscrição i on i.id_tipo = m.tipo_de_inscrição_id_tipo
        where m.id_membro = id_membro;
    end //

DELIMITER //
```

Figura 15- Dado um ID de um membro saber todas as suas informações

```
DELIMITER //

drop procedure if exists getInfoNutricionista;
create procedure getInfoNutricionista (IN id_nutricionista INT)

    begin

        select n.nome_nutricionista as Nome, n.ginásio_id_ginásio as Ginásio
        from nutricionista n
        where n.id_nutricionista = id_nutricionista;
    end //

DELIMITER //
```

Figura 16- Dado o ID de um nutricionista saber todas as suas informações

```
DELIMITER //

drop procedure if exists getMarcacaoConsultas;
create procedure getMarcacaoConsultas (IN id_nutricionista INT)

    begin

        select c.tipo_consulta as Tipo, m.nome_membro as Paciente, c.data_consulta as 'Data', c.id_consulta as ID, n.nome_nutricionista as Nutricionista, n.ginásio_id_ginásio as Ginásio
        from consulta c
        inner join membro m on m.id_membro = c.membro_id_membro
        inner join nutricionista n on n.id_nutricionista = c.nutricionista_id_nutricionista
        where c.nutricionista_id_nutricionista = id_nutricionista;
    end //

DELIMITER //
```

Figura 17- Dado o ID de um nutricionista saber as consultas que tem marcadas

```
DELIMITER //

drop procedure if exists getInfoPT;
create procedure getInfoPT (IN id_pt INT)

    begin

        select p.nome_pt as Nome, p.ginásio_id_ginásio as Ginásio
        from pt p
        where p.id_pt = id_pt;
    end //

DELIMITER //
```

Figura 18- Dado o ID de um PT saber as suas informações

```
DELIMITER //

drop procedure if exists getMarcacaoAulas;
create procedure getMarcacaoAulas (IN id_pt INT)

    begin

        select a.nome_aulas as Tipo, m.nome_membro as Membro, a.data_aulas as 'Data', p.nome_pt as PT, a.id_aulas as ID, a.ginasio_id_ginasio as Ginásio
        from aulas a
        inner join membro m on m.id_membro = a.membro_id_membro
        inner join pt p on p.id_pt = a.pt_id_pt
        where a.pt_id_pt = id_pt;
    end //

DELIMITER //
```

Figura 19- Dado o ID de um PT saber quais aulas tem marcadas

```
DELIMITER //

drop procedure if exists getMarcacoesMembro;


create procedure getMarcacoesMembro (IN id_membro INT)

    begin
        select  m.nome_membro as Nome, m.id_membro as 'ID Membro' , a.id_aulas as 'Aula ID', a.nome_aulas as Aula, a.data_aulas as 'Data', c.id_consulta as 'Consulta ID', c.tipo_consulta as 'Tipo de Consulta'
        from aulas a
        inner join membro m on a.membro_id_membro = m.id_membro
        inner join consulta c on a.membro_id_membro = c.membro_id_membro
        where a.membro_id_membro = id_membro;
    end //

DELIMITER //
```

Figura 20- Dado o ID de um membro saber quais aulas e consultas tem marcadas

```
DELIMITER //

drop procedure if exists getInfoTipoInscricao;
create procedure getInfoTipoInscricao (IN id_tipo INT)

    begin

        select i.nome_tipo as Tipo, m.nome_membro as Nome
        from tipo_de_inscrição i
        inner join membro m on m.tipo_de_inscrição_id_tipo = i.id_tipo
        where m.tipo_de_inscrição_id_tipo = id_tipo;
    end //

DELIMITER //
```

Figura 21- Dado o ID de um tipo de inscrição saber quais membros dela fazem parte

```
DELIMITER //

drop procedure if exists getTipoInscricao;
create procedure getTipoInscricao (IN id_membro INT)

    begin

        select i.nome_tipo as Tipo, m.nome_membro as Nome
        from tipo_de_inscrição i
        inner join membro m on m.tipo_de_inscrição_id_tipo = i.id_tipo
        where m.id_membro = id_membro;
    end //

DELIMITER //
```

Figura 22- Dado um ID de um membro saber qual o seu tipo de inscrição

5.4. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Entidades	Atributos	Tipo de Dados	Tamanho (em bytes)		Subtotal (em bytes)
Ginásio	id_ginásio	INT	4	1	50
	nome_ginásio	VARCHAR (45)	46		
Membro	id_membro	INT	4	70	7630
	nome_membro	VARCHAR (45)	46		
	peso_membro	DECIMAL	5		
	altura_membro	DECIMAL	5		
	localidade_membro	VARCHAR (45)	46		
	data_nascimento_membro	DATE	3		
Tipo de inscrição	id_tipo	INT	4	3	150
	nome_tipo	VARCHAR (45)	46		
Aulas	id_aulas	INT	4	10	580
	nome_aulas	VARCHAR (45)	46		
	data_aulas	DATETIME	8		
PT	id_pt	INT	4	10	500
	nome_pt	VARCHAR (45)	46		
Staff	id_staff	INT	4	10	500
	nome_staff	VARCHAR (45)	46		
Nutricionista	id_nutricionista	INT	4	10	500
	nome_nutricionista	VARCHAR (45)	46		
Consulta	id_consulta	INT	4	10	580
	data_consulta	DATETIME	8		
	tipo_consulta	VARCHAR (45)	46		
				Total	10490

Com o modelo atual e assumindo uma taxa de crescimento anual de 15%, chegou-se aos seguintes dados:

- Ano 1: 10490 bytes
- Ano 2: 12064 bytes
- Ano 3: 13874 bytes

5.5. Definição e caracterização das vistas de utilização em SQL

As Vistas são um ponto a realçar na implementação de uma base de dados, já que permitem que o utilizador tenha apenas acesso a uma tabela temporária e não à tabela onde são realmente armazenados os dados, sendo, portanto, um reforço na segurança.

```
create view mediaidade as
select avg(TIMESTAMPDIFF(year, data_nascimento_membro, CURDATE())) from membro;
```

Figura 23 - Vista referente à média de idades dos membros

```
create view top10alturas as
select altura_membro from membro order by altura_membro DESC
limit 10;
```

Figura 24 - Vista referente ao top 10 de alturas dos membros

5.6. Revisão do sistema Implementado

Após a elaboração do sistema, foram realizados os testes das Queries, a inserção de dados nas tabelas e posteriormente a manipulação dos dados.

Concluímos, portanto, que o produto final correspondeu às expectativas iniciais do grupo.

6. Conclusão e Análise critica

A elaboração deste sistema de base de dados, apesar de ter-se demonstrado um processo trabalhoso, foi uma experiência enriquecedora, quer a nível prático da elaboração e tratamento dos dados para as aplicações realizadas, mas também porque permitiu ao grupo testar e aprofundar os conhecimentos obtidos nesta matéria.

De modo geral, as etapas que levaram à construção de todo o projeto foram realizadas com o devido planeamento e cuidado, sempre com o intuito de tornar o sistema o mais prático e eficiente possível de forma a corresponder às expectativas inicialmente propostas.

O grupo está, portanto, satisfeito com o trabalho realizado e com todo o processo desenvolvido ao longo da criação do sistema, é de salientar que consoante o sistema vá crescendo, é necessária a devida manutenção do mesmo.

7. Referências Bibliográficas

- Slides da UC fornecidos pela equipa docente;
- Thomas Connolly, Carolyn Begg. Fourth edition. "Database Systems - A Practical Approach to Design, Implementation, and Management".
- BR Modelo - http://www.sis4.com/brModelo/
- W3Schools - https://www.w3schools.com/sql/