



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

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

Trabalho Prático Base de Dados

Grupo 16

Adriana Gonçalves, a75119

André Sousa, a74813

Bernardo Viseu, a74618

Renato Cruzinha, a75310

Janeiro, 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Trabalho Prático

Adriana Gonçalves, a75119

André Sousa, a74813

Bernardo Viseu, a74618

Renato Cruzinha, a75310

Janeiro, 2020

Resumo

Este projeto foi realizado no âmbito da Unidade Curricular de Bases de Dados, tendo como principal objetivo a planificação de agendamentos de testes clínicos a atletas de várias modalidades e categorias. Este relatório divide-se em duas partes: parte relacional e não relacional. Inicialmente, iremos focar-nos na parte relacional e iremos explicar um pouco mais do problema através de contextualização do tema do relatório, motivações e objetivos, análise de requisitos e a estrutura do trabalho. Posteriormente, apresentaremos o modelo conceptual e todas as entidades e atributos envolvidos no nosso sistema, para que, depois de tudo devidamente identificado, possamos prosseguir para o modelo lógico onde será implementada toda a estrutura desta base de dados. Para finalizar a parte relacional do problema, vamos analisar e validar os dados através do método de normalização das relações existentes.

Na segunda parte do relatório, iremos abordar a parte não relacional do trabalho, onde tivemos de nos basear na base de dados construída na primeira fase, em SQL, e convertê-la para um modelo NoSQL, nomeadamente em Neo4j, que é uma base de dados orientada a grafos. Ao longo deste relatório efetuamos uma análise mais detalhada a todo este processo de conversão, desde a explicação da migração de um modelo de base de dados para o outro até à identificação das vantagens e desvantagens dos modelos de base de dados SQL e NoSQL.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados.

Palavras-Chave: Bases de Dados, Bases de Dados Relacionais, Modelo Conceptual, Modelo Lógico, Análise de Requisitos, Entidades, Atributos, Relacionamentos, MySQL Workbench, SQL, NoSQL, Neo4j, Grafos, Nodos.

Índice

1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	2
1.3. Motivação e Objetivos	2
1.4. Estrutura do Relatório	2
2. Análise e levantamento de Requisitos	4
2.1. Requisitos de dados e transações	4
3. Modelo Conceptual	5
3.1. Identificação das entidades do problema	5
3.2. Identificação dos relacionamentos	7
3.3. Associação dos atributos às entidades e/ou relacionamentos e respetivos domínios dos atributos	12
3.4. Determinar chaves candidatas e primárias	13
3.5. Validação do modelo	17
4. Modelo Lógico	19
4.1. Derivação das relações para modelo lógico	19
4.1.1. Entidades Fortes	20
4.1.2. Entidades Fracas	23
4.1.3. Relacionamentos binários um para muitos (1-N)	23
4.1.4. Relacionamentos binários um para um (1-1)	23
4.1.5. Relacionamentos recursivos um para um (1-1)	24
4.1.6. Relacionamentos Superclasse/Subclasse	24
4.1.7. Relacionamentos binários muitos para muitos (N-M)	24
4.1.8. Relacionamentos complexos	24
4.1.9. Atributos multivalue	24
4.2. Validação das relações através da normalização	25
4.2.1. 1FN – 1ºForma Normal	25
4.2.2. 2FN – 2ºForma Normal	25
4.2.3. 3FN – 3ºForma Normal	25
4.3. Validação das relações	25
4.4. Análise das restrições de integridade	27
5. Modelo Físico	30

5.1. Transcrição do modelo lógico para SGBD.....	30
5.1.1. Desenho das relações base.....	30
5.1.2. Desenho das representações dos dados derivados	34
5.1.3. Desenho das restrições gerais	35
5.2. Organização dos ficheiros e índices	39
5.3. Validação das relações	40
6. Bases de dados NoSQL	41
6.1. Principais Características	42
6.2. SQL vs NoSQL	42
6.3. Vantagens	43
6.4. Limitações	43
6.5. Modelo de Dados NoSQL baseado em grafos	43
6.6. Novo sistema de base de dados	43
6.7. Migração de dados	44
6.7.1. Importação de ficheiros csv para Neo4j	44
6.7.2. Criação de nodos	45
6.7.3. Criação de Indexes	45
6.7.4. Criação de relacionamentos	47
6.8. Queries	48
6.9. Versão final da base de dados em Neo4j	49
7. Conclusões e Trabalho Futuro	50
8. Referências	51
9. Lista de Siglas e Acrónimos	52

Índice de Figuras

Figura 1. Esquema conceptual.....	5
Figura 2. Relacionamento entre Atleta e Modalidade	7
Figura 3. Relacionamento entre Atleta e Clube.....	8
Figura 4. Relacionamento entre Atleta e Agendamento.....	8
Figura 5. Relacionamento entre Atleta e Categoria.....	9
Figura 6. Relacionamento entre Agendamento e Medico.....	9
Figura 7. Relacionamento entre Agendamento e TesteClinico.....	10
Figura 8. Relacionamento entre Medico e Especialidade.....	11
Figura 9. Relacionamento entre TesteClinico e Utensílio.....	11
Figura 10. Entidade Atleta no modelo conceptual.....	13
Figura 11. Entidade Modalidade no modelo conceptual.....	13
Figura 12. Entidade Clube no modelo conceptual.....	14
Figura 13. Entidade Categoria no modelo conceptual.....	14
Figura 14. Entidade Medico no modelo conceptual.....	15
Figura 15. Entidade Especialidade no modelo conceptual.....	15
Figura 16. Entidade Agendamento no modelo conceptual.....	16
Figura 17. Entidade Agendamento no modelo conceptual.....	16
Figura 18. Entidade Utensílio no modelo conceptual.....	17
Figura 19. Modelo Lógico.....	19
Figura 20. Exemplo de integridade relacional.....	29
Figura 21. Criação da tabela Atleta.....	35
Figura 22. Criação da tabela Modalidade.....	36
Figura 23. Criação da tabela Residência.....	36
Figura 24. Criação da tabela Clube.....	36
Figura 25. Criação da tabela Categoria.....	37
Figura 26. Criação da tabela Especialidade.....	37
Figura 27. Criação da tabela Medico.....	37
Figura 28. Criação da tabela Agendamento.....	38
Figura 29. Criação da tabela TesteClinico.....	38
Figura 30. Criação da tabela Utensílio.....	39
Figura 31. Criação da tabela TesteClinicoHasUtensilio.....	39

Figura 32. Resolução da Query 1 em SQL.....	40
Figura 33. Resolução da Query 2 em SQL.....	40
Figura 34. Resolução da Query 3 em SQL.....	41
Figura 35. Resolução da Query 4 em SQL.....	41
Figura 36. Esquema tipo do novo sistema de base de dados.....	44
Figura 37. Script de criação de nodos.....	45
Figura 38. Script de criação de indexes.....	47
Figura 39. Script de criação de relacionamentos.....	47
Figura 40. Modelo final da nova base de dados	

Índice de Tabelas

Tabela 1. Dicionário de relacionamentos do modelo.....	7
Tabela 2. Dicionário de dados dos atributos das entidades do sistema.....	12
Tabela 3. Representação da tabela da entidade Atleta.....	20
Tabela 4. Representação da tabela da entidade Modalidade.....	20
Tabela 5. Representação da tabela da entidade Clube.....	21
Tabela 6. Representação da tabela da entidade Categoria.....	21
Tabela 7. Representação da tabela da entidade Residencia.....	21
Tabela 8. Representação da tabela da entidade Agendamento.....	21
Tabela 9. Representação da tabela da entidade Medico.....	22
Tabela 10. Representação da tabela da entidade TesteClinico.....	22
Tabela 11. Representação da tabela da entidade Especialidade.....	22
Tabela 12. Representação da tabela da entidade Utensílio.....	22
Tabela 13. Tabela de integridade das entidades.....	27
Tabela 14. Tabela das principais diferenças entre SQL e NoSQL.....	43

1. Introdução

1.1. Contextualização

Este trabalho foi-nos proposto no âmbito da unidade curricular de Base de Dados e é pedido que façamos uma análise, um planeamento, uma estrutura e uma implementação de um sistema de base de dados relacional que sirva de suporte ao agendamento de testes clínicos por parte de atletas de várias modalidades/categoria. Sendo assim, neste primeiro capítulo vamos introduzir, brevemente, o nosso projeto. Desta maneira, é necessário definir o contexto no qual se desenvolve o caso de estudo, seguido da sua descrição. De seguida vai ser efetuada a transformação do modelo conceptual para lógico (através da análise das relações existentes no modelo anterior) e por fim, apresentamos o modelo físico e as respetivas resoluções das queries. Relativamente à segunda parte deste relatório, vai focar-se essencialmente na importação da base de dados relacional para uma não relacional. O modelo de base de dados não relacional a ser implementado vai ser o Neo4j, sendo que vamos abordar as suas características, vantagens/desvantagens etc, e ainda migração dos dados.

Apresentação do Caso de Estudo

Pretende desenvolver um Sistema de Gestão de Bases de Dados que permita o agendamento, por parte de atletas, de forma a realizarem testes clínicos dos mais variados tipos. Para isso, numa fase inicial, iremos recolher toda a informação necessária para que a implementação da base de dados seja o mais simples possível, minimizando qualquer problema de organização e de recolha de dados que possa ocorrer. Na segunda parte do trabalho prático o principal objetivo será a implementação de um Sistema de Base de Dados (SBD) não relacional tendo como foco o tema apresentado para a primeira parte, mas desta vez utilizando o Neo4j, ferramenta que iremos utilizar e que se distingue por funcionar à base de grafos. Cada nodo e aresta pode ter um número ilimitado de atributos, sendo que estes podem também ser rotulados de forma a restringir pesquisas.

1.2. Motivação e Objetivos

Desde o início que a intenção foi criar um sistema que regista todos os testes clínicos realizados por atletas de diversas modalidade/categorias. Existem aspetos bastante importantes a ter em conta tais como ter a informação completa e atualizada e sempre disponível para uma gestão de recursos mais eficiente. Na estruturação do problema e elaboração da lista de requisitos deparamo-nos com a possibilidade de existir um grande número de entidades assim como de relacionamentos. Tendo em conta que o objetivo é ter um sistema simples, tiveram de ser tomadas decisões tendo sempre em conta que o sistema teria que possuir o máximo de informação útil que permitisse uma melhor organização. O grande objetivo desta base de dados é modelar um protótipo de um sistema que permita a organização correta de toda a informação relativa aos testes clínicos dos atletas. A viabilidade presente na elaboração desta base de dados vai depender da possibilidade da sua implementação física a nível de terminais de acesso e também da qualidade da informação na atualização de dados. A segunda parte do trabalho tem como objetivo fazer a importação da base de dados relacional para uma não relacional. O modelo de base de dados não relacional a ser implementado vai ser o Neo4j que é uma base de dados que utiliza grafos para guardar todas as informações em si inseridas a nível aplicacional. Para isso realizamos uma migração dos dados que possuímos na base de dados relacional para a nova base de dados a ser criada usando um modelo não relacional. Ao longo da realização desta segunda parte do trabalho fomos adquirindo conhecimentos para fazermos uma diferenciação dos dois modelos usados e saber identificar os pontos fortes e fracos de cada modelo.

1.3. Estrutura do Relatório

Neste relatório iremos apresentar uma descrição detalhada do caso de estudo e de todo o processo de elaboração e implementação da base de dados.

- No primeiro capítulo estará presente a contextualização do problema proposto, tal como os objetivos e motivações que tivemos ao longo do desenvolvimento do projeto.
- No segundo capítulo é onde fazemos a análise e levantamento dos requisitos do sistema, que consiste em descrever sucintamente as entidades presentes na base de dados.
- No capítulo 3 abordamos o conceito de modelo conceptual onde apresentamos as entidades do problema, identificamos as relações entre cada uma das entidades, apresentamos a associação dos atributos às entidades e/ou relacionamentos e os respetivos domínios dos atributos, definimos as chaves primárias e candidatas do sistema, e por último é feita a validação do modelo através de perguntas efetuadas ao programa.

- Depois de bem definido o modelo conceptual, no capítulo 4, abordamos o modelo lógico do problema onde começamos por explicar a transição de um modelo para o outro. Nos tópicos seguintes são analisadas as suas entidades e os relacionamentos contidos no modelo e no final é feita uma validação das suas relações.
- No quinto capítulo é apresentada a tradução do modelo lógico para o modelo físico. São descritas as relações base, os dados derivados, as restrições gerais e é identificada a organização dos ficheiros e índices. No final, realizamos a validação do nosso modelo através do código.
- Depois de concluída a parte relacional do trabalho, no capítulo 6 abordamos a parte não relacional do caso em estudo onde começamos por explicar o que é uma base de dados NoSQL. Sendo que este relatório está dividido em duas partes (relacional e não relacional), falamos das principais características e diferenças comparando com a base de dados anteriormente construída (SQL) mostrando as suas vantagens perante esta, tal como as suas limitações. De seguida, descrevemos o modelo de base de dados NoSQL baseado em grafos, modelo este que foi implementado na nossa base de dados usando o Neo4j. Logo depois é apresentado detalhadamente todo o processo da migração de dados. Antes de apresentarmos a conclusão do trabalho e consequente relatório, apresentamos as queries que representam a resposta às perguntas anteriormente apresentadas na análise do caso em estudo.
- Para concluir o relatório, são apresentadas as conclusões seguidas da bibliografia e a lista de sinónimos e acrónimos

2. Análise e levantamento de requisitos

Neste capítulo iremos abordar todos os requisitos necessários para o funcionamento do nosso sistema, neste caso, agendamento e realização de testes clínicos a atletas. Assim iremos focar-nos principalmente no atleta, a modalidade por ele praticada bem como no teste clínico e o médico que a este está associado.

2.1 Requisitos de dados e transações

De seguida apresentamos os requisitos do funcionamento do sistema do caso em estudo:

Atleta

- Possui um número de identificação único associado ao atleta, um nome, um nº de telemóvel e uma data de nascimento.

Residência

- Possui um número de identificação único associado à residência, uma morada, uma localidade e um código postal.

Modalidade

- Possui um número de identificação único associado à modalidade e a sua designação.

Clube

- Possui um número de identificação, o nome do clube e a respetiva sigla.

Categoria

- Possui um número de identificação único associado à categoria e a designação (100m, dardos, martelo etc).

Médico

- Possui o número de identificação único associado ao médico, o seu nome, a data de quando iniciou serviço.

Especialidade

- Possui o número de identificação único associado a especialidade e a sua designação.

Agendamento

- Possui o número de identificação único associado a um agendamento, bem como a data e hora a que ocorreu.

Teste Clínico

- Possui o número de identificação único associado a um teste clínico, a descrição desse teste, a data e hora a que ocorreu.

Utensílio

- Possui o número de identificação associado a cada utensílio bem como o respetivo nome.

MODELO RELACIONAL

3. MODELO CONCEPTUAL

A elaboração do modelo conceptual constitui a primeira fase da criação de um sistema de gestão de base de dados (SGBD) e tem início após uma pesquisa detalhada dos requisitos. A documentação deste modelo vai ser descrita neste capítulo em diversas fases:

1. Identificação das entidades do problema
2. Identificação dos relacionamentos do projeto
3. Associação dos atributos às entidades e/ou relacionamentos e respectivos domínios dos atributos
4. Determinar chaves candidatas e primárias

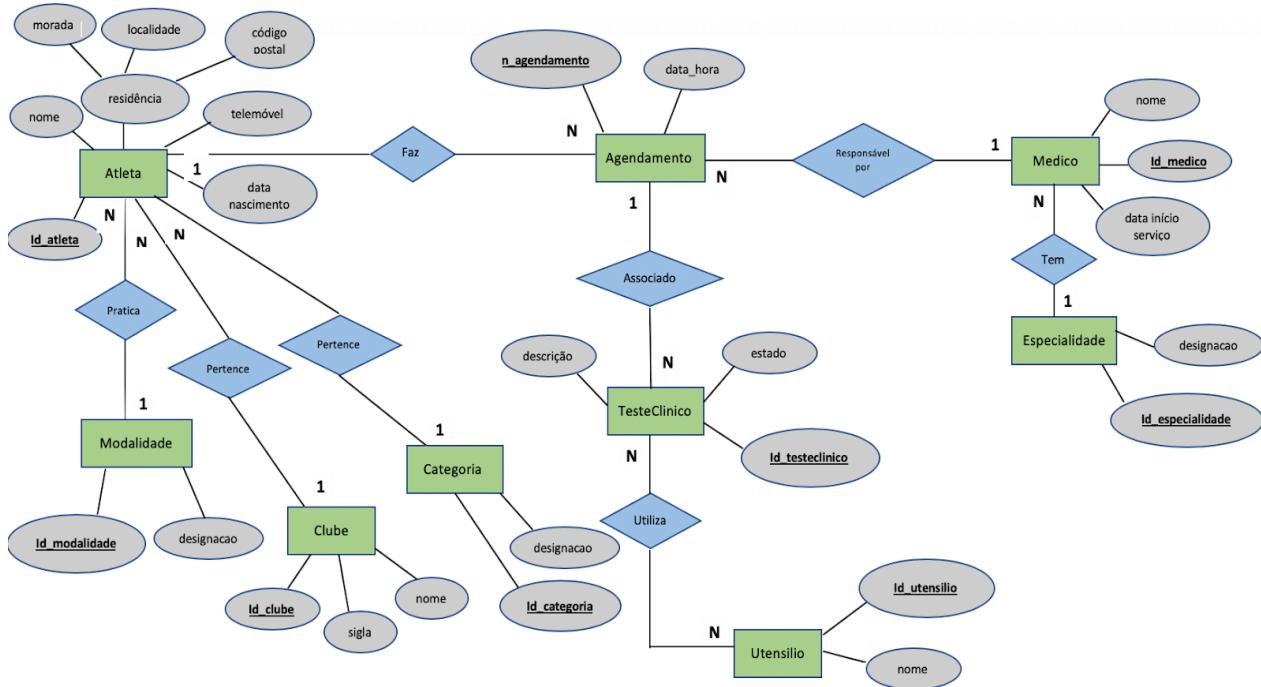


Figura 1. Modelo Conceptual

3.1. Identificação das entidades do problema

A primeira tarefa necessária para a construção do modelo conceptual passa por identificar as entidades do problema. Neste capítulo iremos apresentar as entidades do modelo bem como o seu significado.

Atleta

Uma personagem importante no nosso sistema é a entidade *Atleta*. Este vai ser responsável por fazer o agendamento para a realização dos testes clínicos.

Clube

A entidade *Clube* diz a que clube pertence um dado atleta.

Modalidade

A entidade *Modalidade* terá os dados relativos à modalidade praticada pelos atletas.

Categoria

A entidade *Categoria* dá-nos informações sobre a categoria a que pertence o atleta.

Teste Clínico

A entidade *Teste Clínico* vai conter toda a informação relativa ao estado clínico do atleta. Através do nº do teste clínico teremos acesso à informação acerca do estado de saúde do atleta relativamente a um exame em particular.

Médico

A entidade *Médico* terá a responsabilidade de supervisionar um teste clínico. Este teste pode apenas ter associado um médico responsável, mas o médico pode supervisionar vários testes.

Especialidade

Dá-nos informações relativas à especialidade do médico.

Agendamento

Outro dos elementos importantes no nosso modelo conceptual é a entidade *Agendamento*. Esta entidade é acedida pelo atleta de forma a poder realizar os testes clínicos numa determinada data.

Utensílio

Por último, esta entidade *Utensílio* possibilita a realização dos testes clínicos aos atletas, na medida em que para se realizar um exame é necessário existir utensílios disponíveis.

3.2. Identificação dos relacionamentos do problema

Depois de termos identificado todas as entidades do modelo, procedemos à identificação dos relacionamentos que existem entre elas.

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
<i>Atleta</i>	N	Pratica	1	<i>Modalidade</i>
<i>Atleta</i>	N	Pertence	1	<i>Clube</i>
<i>Atleta</i>	N	Pertence	1	<i>Categoria</i>
<i>TesteClinico</i>	N	Utiliza	N	<i>Utensílio</i>
<i>Atleta</i>	1	Faz	N	<i>Agendamento</i>
<i>Agendamento</i>	1	Associado a	N	<i>TesteClinico</i>
<i>Agendamento</i>	N	À responsabilidade	1	<i>Medico</i>
<i>Medico</i>	N	Tem	1	<i>Especialidade</i>

Tabela 1. Dicionário de relacionamentos do modelo

Atleta e Modalidade:

O relacionamento “Pratica” entre a entidade Atleta e Modalidade caracteriza-se por ter uma cardinalidade de 1 para N, na medida em que um atleta pratica uma modalidade e uma modalidade é praticada por vários atletas.

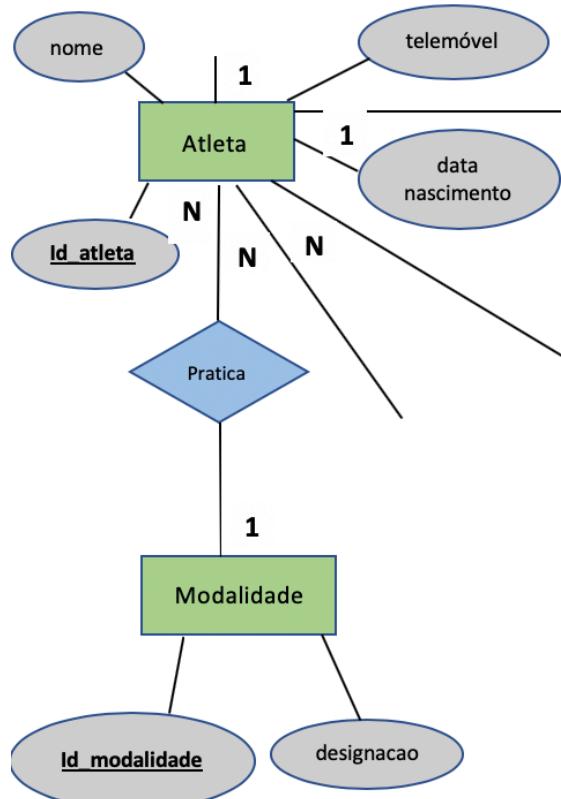


Figura 2. Relacionamento entre Atleta e Modalidade

Atleta e Clube:

O relacionamento “Pertence” entre a entidade Atleta e Clube caracteriza-se por ter uma cardinalidade de 1 para N, na medida em que um atleta pertence exclusivamente a um clube e um clube tem N atletas.

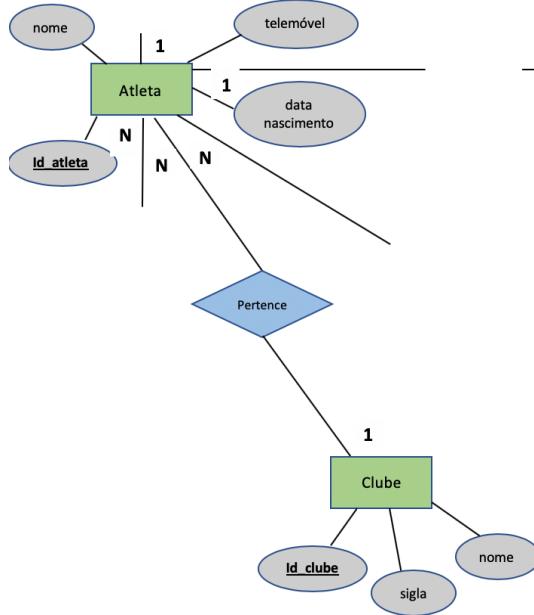


Figura 3. Relacionamento entre Atleta e Clube

Atleta e Agendamento:

O relacionamento “Faz” entre a entidade Atleta e Agendamento caracteriza-se por ter uma cardinalidade de 1 para N, na medida em que um atleta pode fazer vários agendamentos para realizar testes, e um agendamento é feito por um atleta em específico.

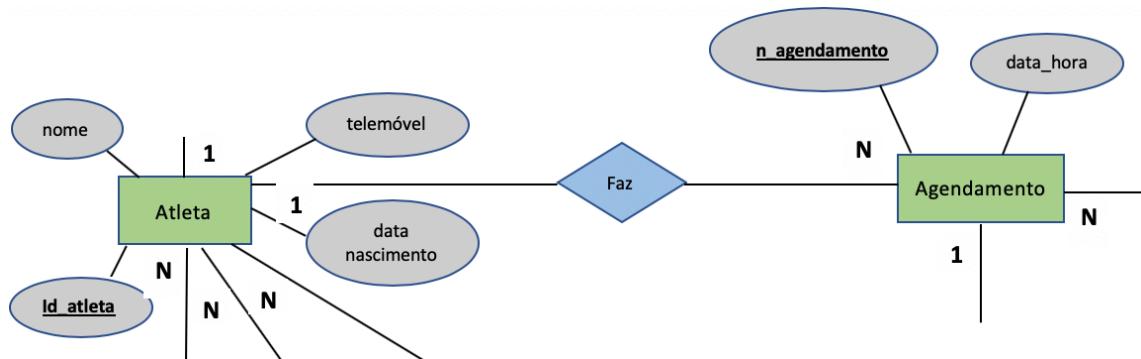


Figura 4. Relacionamento entre Atleta e Agendamento

Atleta e Categoria:

O relacionamento “Pertence” entre a entidade Atleta e Categoria caracteriza-se por ter uma cardinalidade de 1 para N, visto que um atleta pertence a uma categoria e uma categoria nem N atletas.

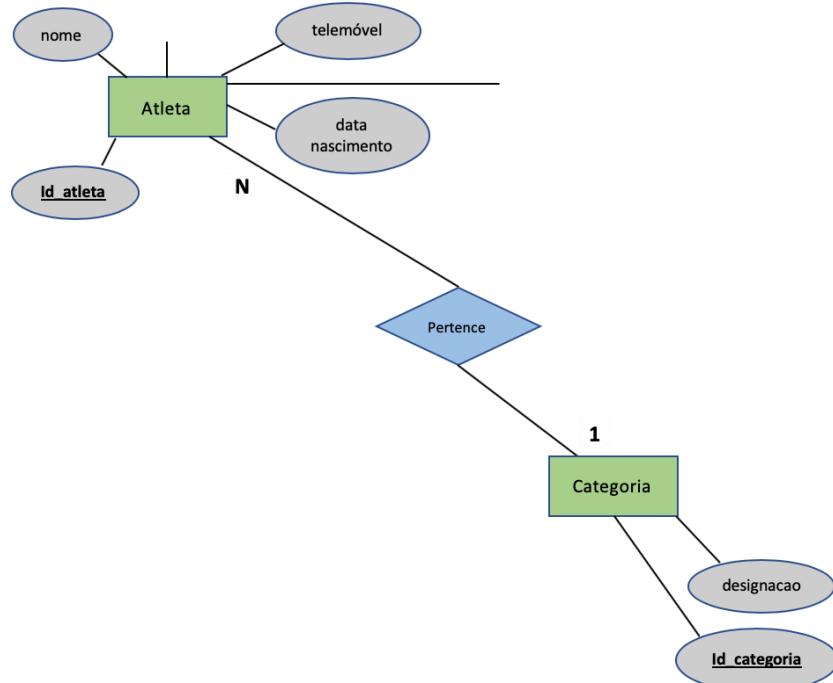


Figura 5. Relacionamento entre Atleta e Categoria

Agendamento e Médico:

O relacionamento “Responsável por” entre a entidade Agendamento e Médico caracteriza-se por ter uma cardinalidade de N para 1, na medida em um agendamento tem apenas um médico responsável pela sua realização e um médico pode ser responsável por vários agendamentos.

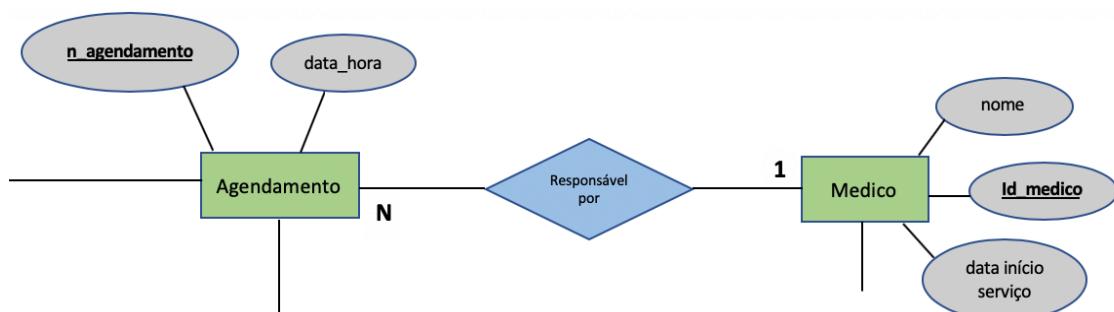


Figura 6. Relacionamento entre Agendamento e Medico

Agendamento e Teste Clínico:

O relacionamento “Associado a” entre a entidade Agendamento e Teste Clínico caracteriza-se por ter uma cardinalidade de 1 para N, na medida em que um agendamento está associado a um teste clínico e um teste clínico está associado a um agendamento.

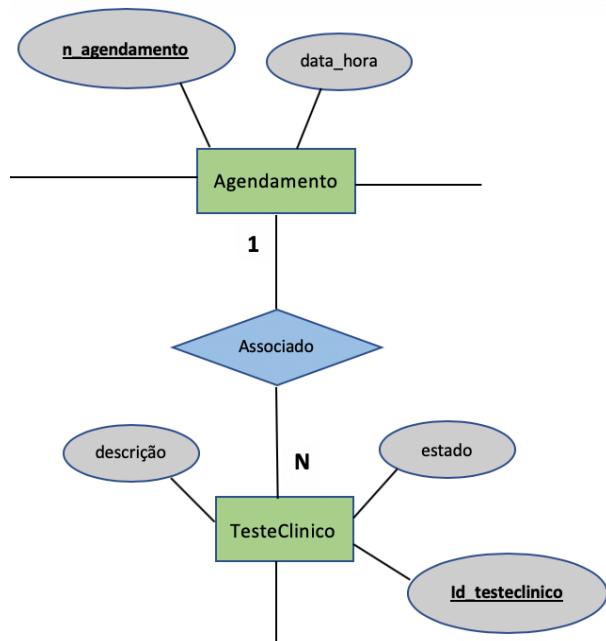


Figura 7. Relacionamento entre Agendamento e TesteClinico

Médico e Especialidade:

O relacionamento “Tem” entre a entidade Médico e Especialidade caracteriza-se por ter uma cardinalidade de N para 1, na medida em um médico tem uma especialidade e uma especialidade tem vários médicos.

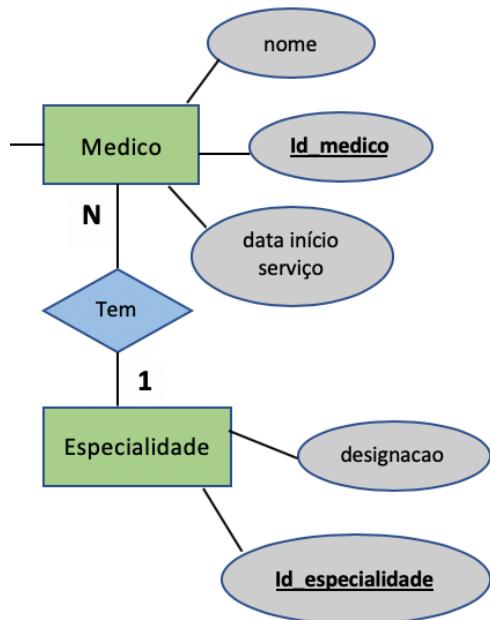


Figura 8. Relacionamento entre Medico e Especialidade

Teste Clínico e Utensílio:

O relacionamento “Utiliza” entre a entidade Teste Clínico e Utensílio caracteriza-se por ter uma cardinalidade de 1 para N, isto é, um teste clínico utiliza vários utensílios e um utensílio é utilizado em apenas um teste.

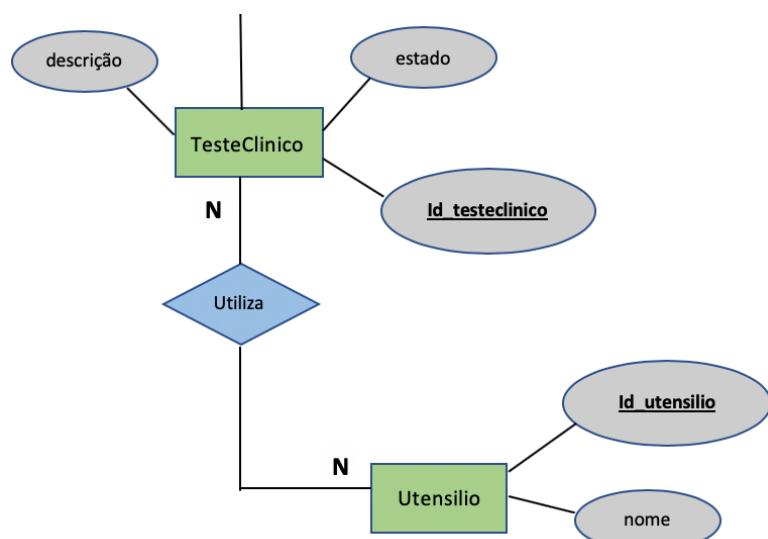


Figura 9. Relacionamento entre TesteClinico e Utensilio

3.3. Associação dos atributos às entidades e/ou relacionamentos e respetivos domínios dos atributos

Entidade	Atributo	Descrição	Tipo de dados e tamanho	Null	Tipo de atributo	Domínio
Atleta	<u>idAtleta</u>	Nº de identificação único do atleta	INT	Não	Chave Primária	Inteiro
	nome	Nome do atleta	VARCHAR (200)	Não	Simples	200 char. variáveis
	dataNascimento	Data de nascimento	DATETIME	Não	Simples	Data
	telemovel	Nº de telemóvel do atleta	INT	Não	Simples	Inteiro
	residência	Informações sobre a residência de um atleta	VARCHAR (200)	Não	Composto → localidade morada codigoPostal	VARCHAR (200) VARCHAR (200) VARCHAR(10)
Clube	<u>idClube</u>	Nº de identificação do clube do atleta	INT	Não	Chave Primária	Inteiro
	nome	Nome do clube	VARCHAR (200)	Não	Simples	200 char. variáveis
	sigla	Sigla do clube	VARCHAR (45)	Não	Simples	45 char. variáveis
Categoria	<u>idCategoria</u>	Nº de identificação da categoria	INT	Não	Chave Primária	Inteiro
	designação	Nome da categoria	VARCHAR (200)	Não	Simples	200 char. variáveis
Modalidade	<u>idModalidade</u>	Nº de identificação da modalidade	INT	Não	Chave Primária	Inteiro
	designacao	Nome da modalidade	VARCHAR (200)	Não	Simples	200 char. variáveis
Agendamento	<u>nºagendamento</u>	Nº que identifica um agendamento	INT	Não	Chave Primária	Inteiro
	data_hora	Horário do agendamento	DATETIME	Não	Simples	Data
TesteClinico	<u>idTesteClinico</u>	Nº que identifica um teste	INT	Não	Chave Primária	Inteiro
	descricao	Nome do teste	VARCHAR (200)	Não	Simples	200 char. variáveis
	estado	Estado de saúde de um atleta (se aprovado muda estado para 'A', caso reprovado para 'R')	VARCHAR (200)	Não	Simples	200 char. variáveis
Medico	<u>idMedico</u>	Nº de identificação único do médico	INT	Não	Chave Primária	Inteiro
	nome	Nome do médico	VARCHAR (200)	Não	Simples	200 char. variáveis
	dataInicioServico	Data de início de trabalho	DATE	Não	Simples	Data
Especialidade	<u>idEspecialidade</u>	Nº de identificação único de uma especialidade	INT	Não	Chave Primária	Inteiro
	designacao	Nome da especialidade	VARCHAR (45)	Não	Simples	45 char. variáveis
Utensilio	<u>idUtensilio</u>	Nº que identifica um utensílio	INT	Não	Chave Primária	Inteiro
	nome	Nome do aparelho usado no teste clínico	VARCHAR (200)	Não	Simples	200 char. variáveis

Tabela 2. Dicionário de dados dos atributos das entidades do sistema

3.4. Determinar chaves candidatas e primárias

Após termos identificado e relacionado todas as entidades e atributos que estão presentes no sistema, teremos agora de decidir quais desses atributos poderão constituir chaves primárias. Inicialmente, convém percebermos que uma chave primária corresponde ao atributo de uma dada entidade capaz de a identificar inequivocamente num registo que a represente. O conjunto de todos os atributos de uma entidade designa-se por chaves candidatas, sendo que os atributos que não foram escolhidos para serem chave primária designam-se por chaves alternadas.

Atleta:

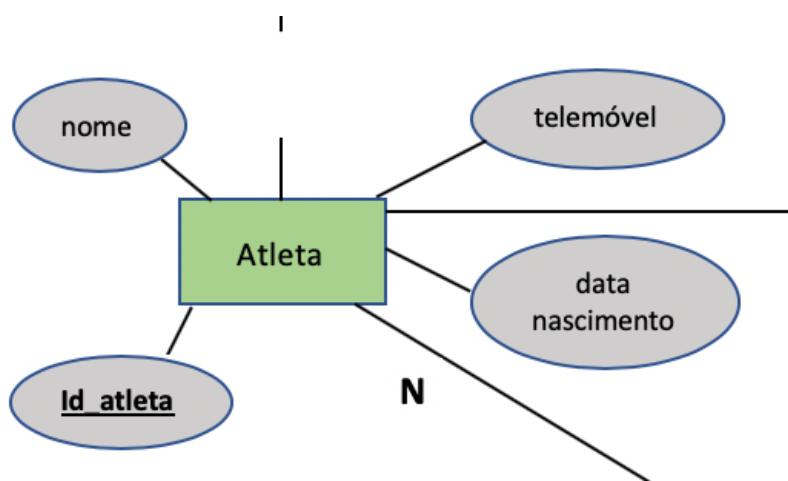


Figura 10. Entidade Atleta no modelo conceptual

Chaves Candidatas: {id_atleta, nome}

Chave Primária: elegemos o id_atleta como chave primária pois é a chave candidata mais simples, otimizando o desempenho de consultas futuras.

Chaves Alternadas: nome

Modalidade:

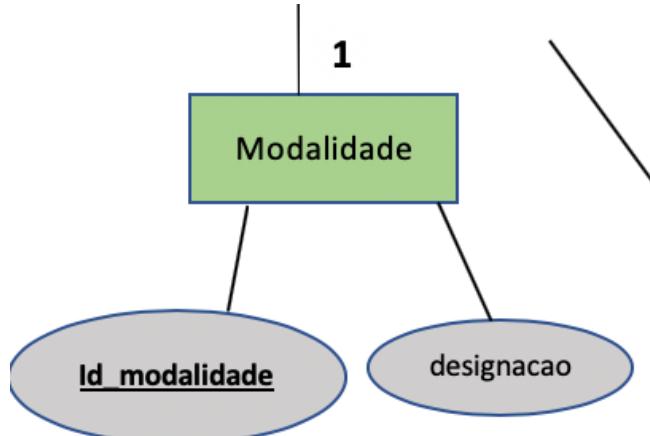


Figura 11. Entidade Modalidade no modelo conceptual

Chaves Candidatas: {id_modalidade, designacao}

Chave Primária: elegemos o id_modalidade como chave primária pois requer menos carga computacional (pois é um inteiro) do que uma designação (que é uma string).

Chaves Alternadas: designação

Clube:

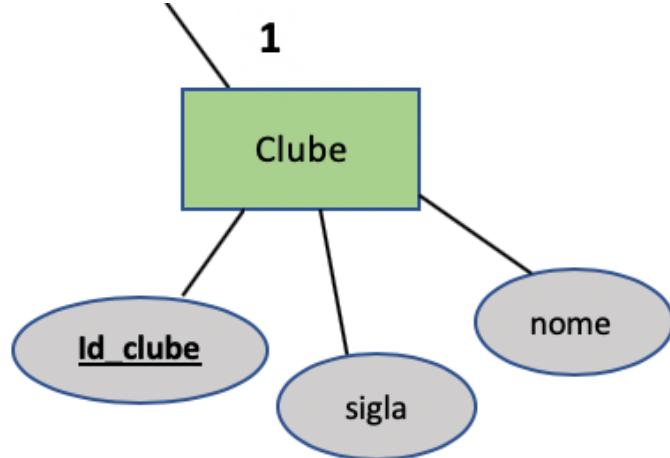


Figura 12. Entidade Clube no modelo conceptual

Chaves Candidatas: {id_clube, nome}

Chave Primária: a chave primária escolhida é id_clube, pois é o único atributo capaz de identificar de forma única uma viagem (por exemplo, o atributo "sigla" não poderia ser pois diferentes clubes podem ter a mesma sigla).

Chaves Alternadas: nome

Categoria:

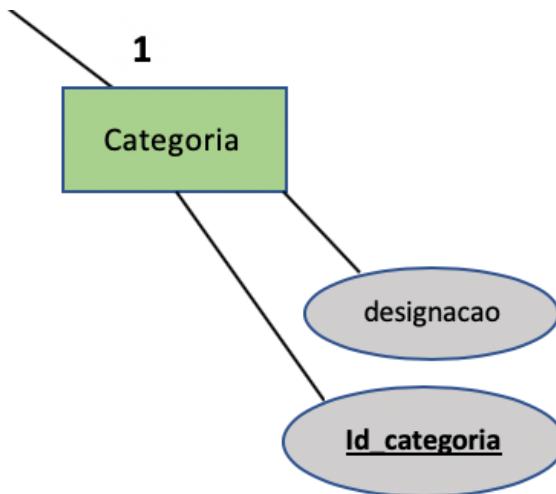


Figura 13. Entidade Categoria no modelo conceptual

Chaves Candidatas: {id_categoria, designacao}

Chave Primária: ambas as chaves são únicas e invariáveis para cada categoria, mas elegemos o id_categoria como chave primária.

Chaves Alternadas: designacao

Médico:

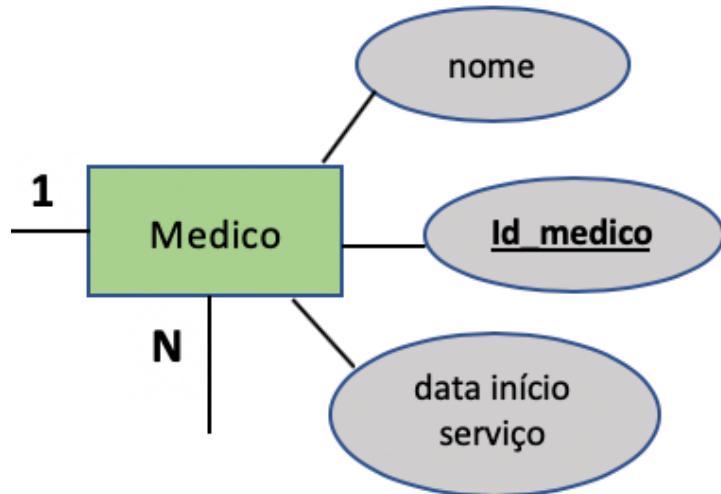


Figura 14. Entidade Medico no modelo conceptual

Chaves Candidatas: {id_medico, nome}

Chave Primária: a chave primária escolhida é o id_medico visto ser mais fácil procurar por um nº de identificador único ao invés de procurar pelo nome do médico (sendo que pode haver médicos com o mesmo nome).

Chaves Alternadas: nome

Especialidade:

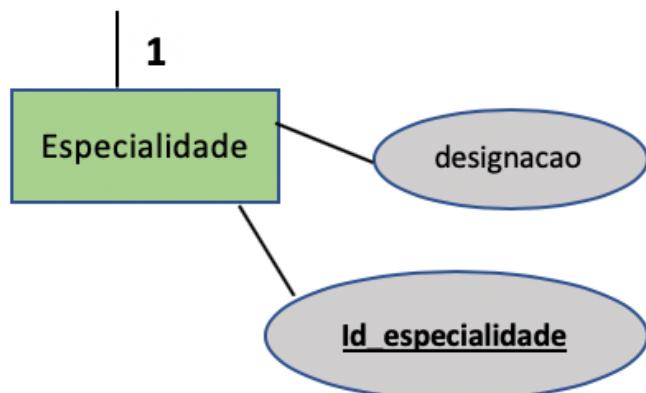


Figura 15. Entidade Especialidade no modelo conceptual

Chaves Candidatas: {id_especialidade}

Chave Primária: a chave primária escolhida é a única chave candidata, o id_especialidade, pois é o único atributo capaz de identificar de forma única uma especialidade.

Chaves Alternadas: nenhuma

Agendamento:

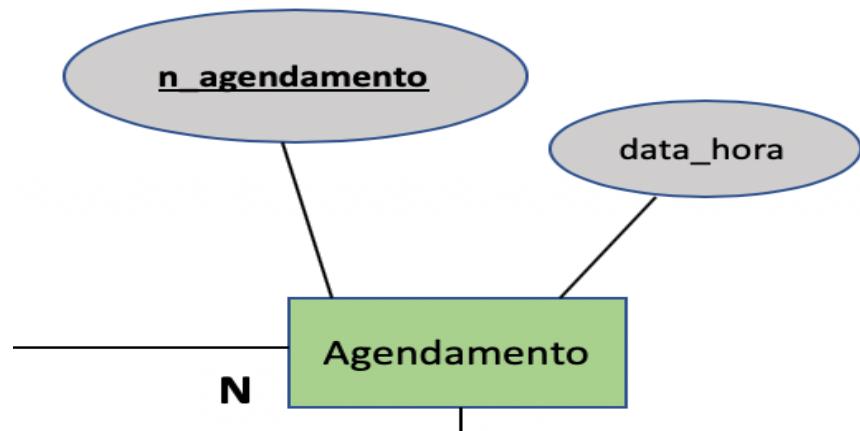


Figura 16. Entidade Agendamento no modelo conceptual

Chaves Candidatas: {n_agendamento}

Chave Primária: a chave primária escolhida é a única chave candidata, o n_agendamento, pois é o único atributo capaz de identificar de forma única um agendamento.

Chaves Alternadas: nenhuma

Teste Clínico:

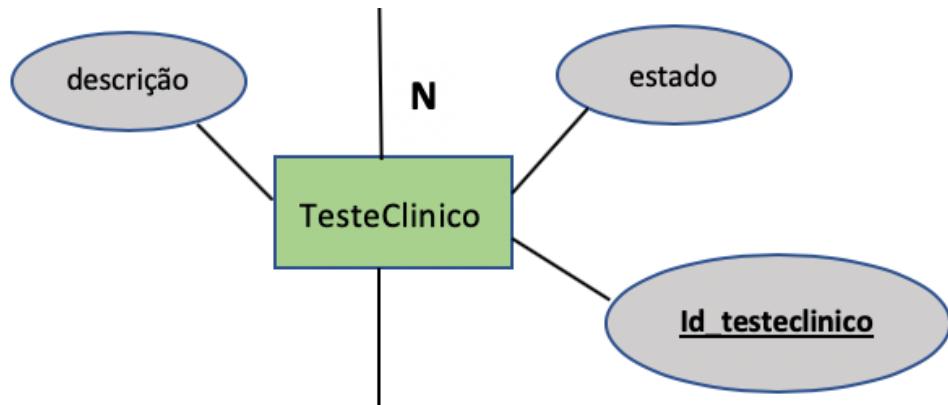


Figura 17. Entidade Agendamento no modelo conceptual

Chaves Candidatas: {id_testeclinico}

Chave Primária: a chave primária escolhida é a única chave candidata, o id_testeclinico, pois é o único atributo capaz de identificar de forma única um teste clínico.

Chaves Alternadas: nenhuma

Utensílio:

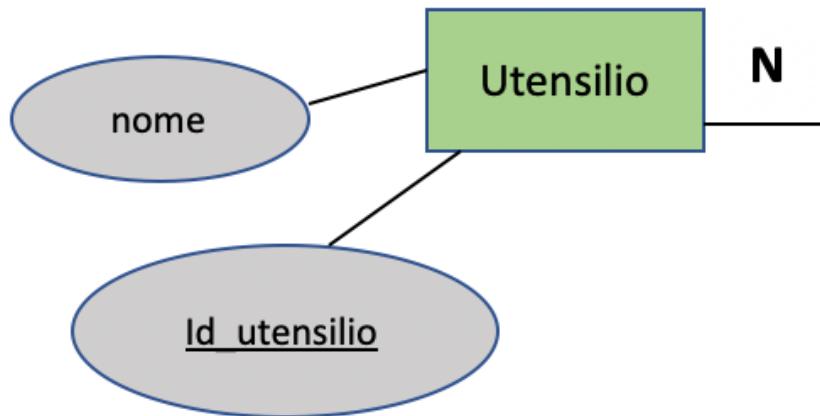


Figura 18. Entidade Utensílio no modelo conceptual

Chaves Candidatas: {id_utensilio, nome}

Chave Primária: tanto o nome como o id_utensilio são capazes de identificar inequivocamente a entidade, porém optamos por escolher o atributo id_utensilio como chave primária.

Chaves Alternadas: nome

3.5. Validação do modelo

Depois de finalizado o modelo conceptual chegou a altura de fazermos a sua validação. Sendo assim, definimos algumas perguntas de forma a verificar a sua viabilidade através da comparação com o modelo por nós definido.

1. Quais os nomes dos atletas que correm os 1200m barreiras?
2. Quantos testes clínicos foram agendados por atletas do Benfica?
3. Quantos testes clínicos foram supervisionados por médicos com mais de 15 anos de serviço?
4. Quais os nomes dos utensílios utilizados no teste clínico 123456?

Validação

1. Para termos acesso aos nomes dos atletas que correm os 1200m barreiras são necessárias as entidades *Atleta*, *Modalidade* e *Categoria*. Assim, a entidade *Atleta* fornece informação acerca de todos os atletas, a entidade *Modalidade* fornece informação sobre todas as modalidades possíveis, e a *Categoria* permite saber qual o tipo de prova. Juntas, estas três entidades dão-nos os nomes de todos os atletas que correm os 1200m barreiras.
2. Para responder a esta pergunta é necessário reunir informação das entidades *Atleta*, *Agendamento* e *Clube*. À entidade *Atleta* está associada a entidade *Clube* e a entidade *Agendamento* fornece informação sobre todos os agendamentos realizados. Ao reunir a informação presente no relacionamento das três entidades consegue-se calcular o número de testes clínicos realizados pelo Benfica.
3. Para efetuar esta verificação serão envolvidas as entidades *TesteClinico* e *Medico*, pois através do relacionamento entre estas entidades (e o atributo *data inicio serviço*) conseguimos saber o número de médicos com mais de 15 anos de serviço que supervisionaram testes clínicos.
4. As entidades envolvidas para a resolução desta pergunta são *TesteClinico* e *Utensílio*. Ao procurarmos pelo número do teste clínico na entidade *TesteClinico* conseguimos ver os utensílios que foram usados nesse teste.

4. MODELO LÓGICO

Depois de finalizada a conceptualização do problema, é agora momento de traduzir o modelo conceptual num modelo lógico. Esta transição permite-nos derivar relações e captar todos os requisitos definidos, fazendo com que no fim, estando o modelo lógico validado, ele seja capaz de dar suporte ao problema em causa. Esta transformação de modelo conceptual em relacional/lógico segue 3 passos fundamentais:

1. Tradução das entidades e respetivos atributos;
2. Tradução dos relacionamentos e respetivos atributos;
3. Tradução de generalizações/agregações;

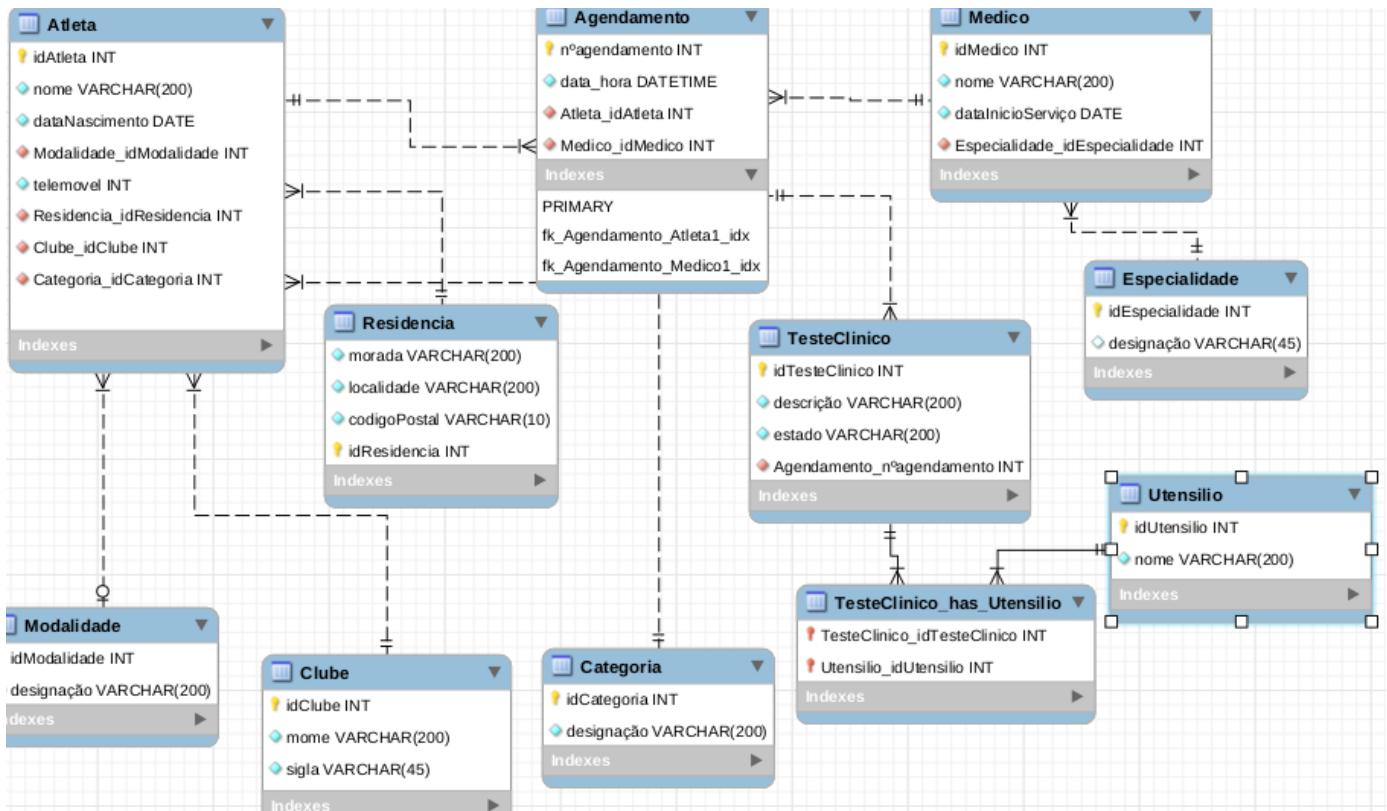


Figura 19. Modelo Lógico

4.1. Derivação das relações para modelo lógico

Na transição de modelo conceptual para modelo lógico, o primeiro ponto que requer a nossa atenção consiste na criação de todas as relações no modelo lógico que representam as entidades, relacionamentos e atributos previamente identificados no modelo conceptual. Para

que tal derivação se concretize, foquemos a nossa atenção nos elementos do modelo conceptual, tendo por base o seguinte:

1. Entidades Fortes;
2. Entidades Fracas;
3. Relacionamentos binários um para muitos (1-N);
4. Relacionamentos binários um para um (1-1);
5. Relacionamentos recursivos um para um (1-1);
6. Relacionamentos superclasse/subclasse;
7. Relacionamentos binários muitos para muitos (N-M);
8. Relacionamentos complexos;
9. Atributos multivalor;

4.1.1. Entidades Fortes

Uma entidade forte é uma entidade cuja existência é independente de outras entidades, ou seja, por si só ela possui total sentido de existir. Cada entidade forte existente no modelo conceptual será traduzida numa tabela no modelo lógico, onde cada um dos atributos da entidade torna-se uma coluna dessa tabela e cada linha representa um registo. Assim, temos que a entidade:

- **Atleta** (idAtleta, nome, dataNascimento, telemovel)

Possui como chave primária: idAtleta

idAtleta	nome	dataNascimento	telemovel
75119	Rosa Moreira	25/12/1985	919293949
(...)	(...)	(...)	(...)

Tabela 3. Representação da tabela da entidade *Atleta*

- **Modalidade** (idModalidade, designação)

Possui como chave primária: idModalidade

idModalidade	designação
360	Corrida
(...)	(...)

Tabela 4. Representação da tabela da entidade *Modalidade*

- **Clube** (idClube, nome, sigla)

Possui como chave primária: idClube

idClube	nome	sigla
112	Grupo Desportivo de Chaves	GDC
(...)	(...)	(...)

Tabela 5. Representação da tabela da entidade *Clube*

- **Categoria** (idCategoria, designação)

Possui como chave primária: idCategoria

idCategoria	designação
453	100m barreiras
(...)	(...)

Tabela 6. Representação da tabela da entidade *Categoria*

- **Residencia** (idResidencia, morada, localidade, codigoPostal)

Possui como chave primária: idResidencia

idResidencia	Morada	Localidade	codigoPostal
2421	Rua de Baixo	Braga	4700-392
(...)	(...)	(...)	(...)

Tabela 7. Representação da tabela da entidade *Residencia*

- **Agendamento** (nºagendamento, data_hora)

Possui como chave primária: nºagendamento

nºagendamento	data_hora
11092	06/06/2016 16:16h
(...)	(...)

Tabela 8. Representação da tabela da entidade *Agendamento*

- **Medico** (idMedico, nome, dataInicioServiço)

Possui como chave primária: idMedico

idMedico	nome	dataInicioServiço
75119	Roberto Carlos Antunes	02/01/1982
(...)	(...)	(...)

Tabela 9. Representação da tabela da entidade *Medico*

- **TesteClinico** (idTesteClinico, descrição, estado)

Possui como chave primária: idTesteClinico

idTesteClinico	descrição	estado
9834	Sangue	A
(...)	(...)	(...)

Tabela 10. Representação da tabela da entidade *TesteClinico*

- **Especialidade** (idEspecialidade, designação)

Possui como chave primária: idEspecialidade

idEspecialidade	designação
118	Cardiologia
(...)	(...)

Tabela 11. Representação da tabela da entidade *Especialidade*

- **Utensílio** (idUtensilio, nome)

Possui como chave primária: idUtensilio

idUtensilio	nome
0235	Seringa
(...)	(...)

Tabela 12. Representação da tabela da entidade *Utensílio*

4.1.2. Entidades Fracas

No nosso modelo conceptual não identificamos nenhuma entidade fraca, isto é, nenhuma das entidades depende de outra para que a sua existência faça total sentido.

4.1.3. Relacionamentos binários um para muitos (1-N)

Relacionamentos do tipo 1:N (um para muitos) ocorrem quando uma ocorrência de uma entidade se pode relacionar com várias ocorrências de outra entidade, isto é, na tabela do lado N cria-se uma chave estrangeira que aponta para a chave primária da tabela 1.

- **Atleta** (idAtleta, nome, dataNascimento, telemovel)
Possui como chave primária: idAtleta
Possui como chaves estrangeiras: Modalidade_idModalidade (vindo da entidade *Modalidade*), Clube_idClube (vindo da entidade *Clube*), Residencia_idResidencia (vindo da entidade *Residencia*) e Categoria_idCategoria (vindo da entidade *Categoria*).
- **Agendamento** (nºagendamento, data_hora)
Possui como chave primária: nºagendamento
Possui como chaves estrangeiras: Atleta_idAtleta (vindo da entidade *Atleta*) e Medico_idMedico (vindo da entidade *Medico*).
- **Medico** (idMedico, nome, dataInicioServiço)
Possui como chave primária: idMedico
Possui como chaves estrangeiras: Especialidade_idEspecialidade (vindo da entidade *Especialidade*).
- **TesteClinico** (idTesteClinico, descrição, estado)
Possui como chave primária: idTesteClinico
Possui como chaves estrangeiras: Agendamento_nºAgendamento (proveniente da entidade *Agendamento*).

4.1.4. Relacionamentos binários de um para um (1-1)

No caso do nosso modelo em particular não consideramos nenhum relacionamento binário de um para um (1:1).

4.1.5. Relacionamentos recursivos um para um (1-1)

Não identificamos nenhum relacionamento recursivo de um para um (1:1) no nosso modelo conceptual.

4.1.6. Relacionamentos superclasse/subclasse

Não consideramos a existência de superclasses nem de subclasses no nosso modelo conceptual.

4.1.7. Relacionamentos binários muitos para muitos (N-M)

Um relacionamento muitos para muitos ocorre quando vários registo de uma tabela são associados a vários registo de outra tabela. Assim, é comum dividir os relacionamentos N:M em dois relacionamentos 1:N usando uma terceira tabela (tabela de associação). Cada registo numa tabela de associação inclui um campo de correspondência que contém o valor das chaves primárias das duas tabelas que ela associa (sendo que na tabela de associação, esses valores correspondem a chaves estrangeiras).

- **TesteClinico_has_Utensilio:**(TesteClinico_idTesteClinico, Utensilio_idUtensilio)
Possui como chave primária: nenhuma
Possui como chaves estrangeiras: TesteClinico_idTesteClinico e Utensilio_idUtensilio

4.1.8. Relacionamentos complexos

Não consideramos a existência de nenhum relacionamento complexo no nosso modelo conceptual.

4.1.9. Atributos multivalor

Não consideramos a existência de nenhum relacionamento complexo no nosso modelo conceptual.

4.2. Validação das relações através da normalização

Este processo denominado normalização é um conjunto de regras que tem como principal objetivo reduzir a redundância dos dados e diminuir anomalias associadas às operações que alteram os dados, conduzindo os dados a sucessivas formas normais.

4.2.1. 1FN - 1^a Forma Normal

A 1^a forma normal (1FN) consiste em eliminar subconjuntos de dados, ou seja, eliminar todos os casos em que a interseção de uma linha com uma coluna corresponde a múltiplos valores. Assim, visto que o nosso modelo não contém valores multivalor nem grupos repetidos podemos afirmar que este está de acordo com a 1^a forma normal.

4.2.2. 2FN - 2^a Forma Normal

Uma relação está na 2^a forma normal (2FN) quando a chave primária é constituída por um só atributo (chave elementar) e não existem atributos dependentes de uma parte da chave, visto que a chave não é composta por partes. Para além disso, a tabela tem de estar em 1FN. Como não existem chaves primárias compostas no nosso modelo e este está em 1FN, então o nosso modelo respeita a 2^a forma normal.

4.2.3. 3FN - 3^a Forma Normal

A 3^a forma normal (3FN) diz que a tabela tem de estar na 2FN e que nenhum atributo não chave pode depender funcionalmente de outro atributo que não seja a chave primária. Como no nosso sistema está em 2FN e após verificarmos todas as relações do nosso modelo, concluímos que este se encontra de acordo com a 3^a forma normal.

4.3. Validação das relações

Finalizado o modelo lógico chegou a altura de fazermos a sua validação. Para tal é necessário responder às mesmas perguntas que foram respondidas na parte anterior de forma a verificar a sua viabilidade através da comparação com o modelo definido.

1. Quais os nomes dos atletas que correm os 100m barreiras?
2. Quantos testes clínicos foram agendados por atletas do Benfica?
3. Quantos testes clínicos foram supervisionados por médicos com mais de 15 anos de serviço?
4. Quais os nomes dos utensílios utilizados no teste clínico 123456?

Validação:

1. Para descobrir o nome dos atletas que correm os 100 metros em barreira, é necessário saber o nome dos atletas e saber o id da categoria dos 100 metros em barreira. Para tal, basta fazer um cruzamento entre a informação da tabela Atleta e a tabela Categoria, e depois buscar os nomes dos atletas que tem como chave estrangeira o id da Categoria 100 metros em barreira.
2. Para saber quantos testes clínicos foram agendados para atletas do clube Benfica, é preciso primeiro saber o id dos clubes, o id dos atletas e o id dos agendamentos. Primeiro descobre-se o id do Benfica na tabela Clube e cruza-se essa informação com a da Tabela Atleta. Depois de saber quais os atletas que pertencem a esse clube, com cada id dos Atletas, vai-se a tabela de Agendamentos e ver o qual o id de cada Agendamento que o atleta teve. Com o id desse agendamento, vai se a tabela de Testes Clínicos e conta-se por cada agendamento o nº de testes clínicos. O resultado final dará o total de testes clínicos que os atletas do Clube Benfica realizaram.
3. Para resolver, é necessário saber o id dos médicos, o id dos agendamentos e o id dos testes clínicos. Primeiro, é preciso filtrar da tabela Médico, os ids dos médicos com mais de 15 anos de serviço. Depois de obter esses ids, é preciso cruzar essa informação com a tabela de Agendamento, de maneira a obter os ids dos agendamentos que esses médicos realizaram. Com o id desses agendamentos, cruza-se a informação com a tabela de TestesClinicos de maneira a obter o número total de testes que esses médicos realizaram.
4. Para responder a esta questão, vamos precisar de informação da tabela de Utensílio e informação da tabela TesteClinico_has_Utensilio. Já tendo o id do Teste clínico, basta ir a tabela TesteClinico_has_Utensilio buscar o id dos utensílios que foram usados nesse Teste Clínico. Com esses ids, vai-se buscar a tabela Utensilio o nome respetivo pra cada id, de maneira a saber o nome dos utensílios que foram usados nesse teste clínico.

4.4. Análise das restrições de integridade

Neste capítulo vamos aplicar as restrições de integridade ao nosso modelo lógico, para que este represente de forma fidedigna todos os dados. Assim, temos as seguintes restrições:

- **Necessidade de valores:**

Esta restrição obriga que alguns atributos têm de ter um valor não-nulo. Para o nosso modelo, assumimos que não existem valores nulos e que todos os atributos são valorados.

- **Integridade da entidade:**

Esta restrição obriga a existência de atributos que fazem com que uma entidade seja inequivocamente identificada e garante que nenhuma entidade é repetida. A validação desta regra é conseguida pelo uso de chaves primárias, declaradas especificamente como não nulas como está representado na seguinte tabela:

Tabela	Chave Primária
Atleta	idAtleta
Residencia	idResidencia
Clube	idClube
Modalidade	idModalidade
Categoria	idCategoria
Agendamento	nºagendamento
TesteClinico	idTesteClinico
Medico	idMedico
Especialidade	idEspecialidade
Utensílio	idUtensilio

Tabela 13. Tabela de integridade das entidades

- **Restrições de domínio do atributo:**

Esta restrição diz respeito aos atributos das entidades, fazendo com que os atributos se enquadrem numa gama de valores próprios, permitindo que cada um "ganhe significado" e não aceite valores aleatórios. Na tabela 2 encontra-se especificado para cada atributo o seu domínio de valores.

- **Multiplicidade:**

A multiplicidade define o relacionamento entre as duas entidades, sendo representada pela cardinalidade entre elas. Na tabela 1 encontra-se especificado os relacionamentos do sistema.

- **Integridade referencial:**

Esta restrição diz respeito à relação entre chaves estrangeiras e chaves primárias, preservando as relações definidas entre as várias tabelas da nossa base de dados quando registo são alterados ou eliminados. Esta restrição permite que os dados permaneçam consistentes em todas as tabelas visto que a alteração numa chave primária se reflete em todas as chaves estrangeiras. Considerando como relação “filho” a entidade que contém a chave estrangeira e como relação “pai” a entidade que contém a chave primária, a integridade referencial deve ser verificada sempre que uma relação “pai” sofra uma atualização, inserção ou remoção. A integridade referencial evita que hajam registo na relação “filho” sem um registo na relação “pai”, sendo que a exclusão/alteração da chave primária de um registo na relação “pai” que possui registo na relação “filho” a ele associados designa-se por violação da chave estrangeira.

Resumindo:

Quando uma operação de INSERT ou UPDATE recai sobre uma chave primária de uma tabela "pai" que é referenciada como chave estrangeira noutra tabela "filho", o resultado depende da ação referencial especificada nas opções ON UPDATE e ON DELETE da chave estrangeira. Independentemente da ação tomada (INSERT ou UPDATE), o MySQL suporta quatro opções: RESTRICT, CASCADE, SET NULL, NO ACTION. No nosso caso em particular, considerámos que qualquer UPDATE de uma chave primária da relação pai siga a opção CASCADE, propagando o UPDATE pelas restantes relações filho. Por outro lado, qualquer DELETE da relação pai segue a opção NO ACTION, na medida em que não é possível efetuar a remoção de um elemento da tabela pai caso haja um valor de chave estrangeira relacionado em alguma tabela filho.

No nosso caso em particular:

A tabela **Atleta** possui como chave primária ***idAtleta*** e como chaves estrangeiras:

1. **Modalidade_idModalidade** referente à tabela **Modalidade(idModalidade)** ON UPDATE: CASCADE, ON DELETE: NO ACTION;
2. **Residencia_idResidencia** referente à tabela **Residencia(idResidencia)** ON UPDATE: CASCADE, ON DELETE: NO ACTION;
3. **Clube_idClube** referente à tabela **Clube(idClube)** ON UPDATE: CASCADE, ON DELETE: NO ACTION;
4. **Categoria_idCategoria** referente à tabela **Categoria(idCategoria)** ON UPDATE: CASCADE, ON DELETE: NO ACTION;

A tabela *Agendamento* possui como chave primária ***nºagendamento*** e como chaves estrangeiras:

1. ***Atleta_idAtleta*** referente à tabela ***Atleta(idAtleta)*** ON UPDATE: CASCADE, ON DELETE: NO ACTION;
2. ***Medico_idMedico*** referente à tabela ***Medico(idMedico)*** ON UPDATE: CASCADE, ON DELETE: NO ACTION;

A tabela *Medico* possui como chave primária o ***idMedico*** e como chaves estrangeiras:

1. ***Especialidade_idEspecialidade*** referente à tabela ***Especialidade(idEspecialidade)*** ON UPDATE: CASCADE, ON DELETE: NO ACTION;

A tabela *TesteClinico* possui como chave primária ***idTesteClinico*** e como chaves estrangeiras:

1. ***Agendamento_nºagendamento*** referente à tabela ***Agendamento(nºagendamento)*** ON UPDATE: CASCADE, ON DELETE: NO ACTION;

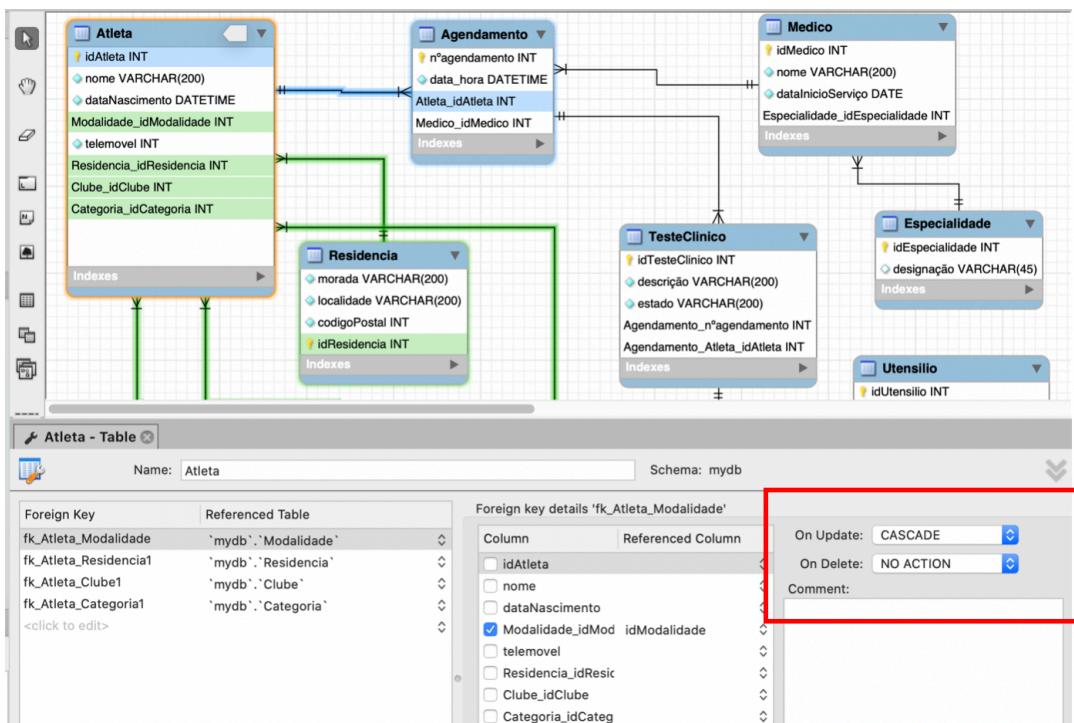


Figura 20. Exemplo de integridade relacional

5. MODELO FÍSICO

Nesta fase do projeto, iremos realizar a tradução do nosso modelo lógico num modelo físico implementado num SGBD, de maneira a poder otimizar o acesso aos dados e a maneira como são guardados, tendo em conta que as estruturas de armazenamento são condicionadas pelo SGBD.

5.1 Transcrição do modelo lógico para SGBD

Este tópico consiste na representação das relações base provenientes do modelo lógico de maneira a que estas e as suas restrições sejam suportadas pelo SGDB.

5.1.1. Desenho das relações base

Nesta secção serão especificadas as relações base existentes no modelo. Para tal, iremos fornecer os domínios, valores por defeito, valores nulos e todo o tipo de restrições existentes.

Relação Atleta

Domínio idAtleta	Inteiro
Domínio nome	String de tamanho variável, de tamanho 200
Domínio dataNascimento	Data e hora
Domínio idModalidade	Inteiro
Domínio telemovel	Inteiro
Domínio idResidencia	Inteiro
Domínio idClube	Inteiro
Domínio idCategoria	Inteiro

Atleta (

idAtleta	Id do Atleta	NOT NULL,
nome	Nome do Atleta	NOT NULL,
dataNascimento	Data de Nascimento	NOT NULL,
idModalidade	Id da Modalidade	NOT NULL,
telemovel	Telemóvel	NOT NULL,
idResidencia	Id da residência	NOT NULL,
idClube	Id do Clube do Atleta	NOT NULL,
idCategoria	id da categoria	NOT NULL,

PRIMARY KEY (idAtleta)

FOREIGN KEY (Modalidade_idModalidade) REFERENCES Modalidade (idModalidade)

FOREIGN KEY (Residencia_idResidencia) REFERENCES Residencia (idResidencia)
 FOREIGN KEY (Clube_idClube) REFERENCES Clube (idClube)
 FOREIGN KEY (Categoria_idCategoria) REFERENCES Categoria(idCategoria);

Relação Residencia

Domínio morada	String de tamanho variável, tamanho 200
Dominio localidade	String de Tamanho variável, tamanho 200
Dominio codigoPostal	String de Tamanho variável, tamanho 10
Dominio idResidencia	Inteiro

Residencia (
morada	Morada do atleta	NOT NULL,
localidade	Localidade da Morada	NOT NULL,
codigoPostal	Código Postal	NOT NULL,
idResidencia	Id da Residência	NOT NULL,

PRIMARY KEY (idResidencia);

Relação Modalidade

Domínio idModalidade	Inteiro	
Domínio designação	String de tamanho variável,tamanho 200	
Modalidade (
idModalidade	Id da Modalidade	NOT NULL,
designação	Designação modalidade	NOT NULL,

PRIMARY KEY (idModalidade);

Relação Categoria

Domínio idCategoria	Inteiro	
Domínio designação	String de tamanho variável, tamanho 200	
Categoria (
idCategoria	Id da Categoria	NOT NULL,
designação	Designação Categoria	NOT NULL,

PRIMARY KEY (idCategoria);

Relação Clube

Domínio idClube	Inteiro	
Domínio nome	String de tamanho variável, tamanho 200	
Domínio sigla	String de tamanho variável, tamanho 45	
Clube (
idClube	Id do Clube	NOT NULL,
nome	Nome do Clube	NOT NULL,
sigla	Sigla do Clube	NOT NULL,

PRIMARY KEY (idClube);

Relação Agendamento

Dominio nºagendamento	Inteiro	
Dominio data_hora	DateTime	
Dominio idAtleta	Inteiro	
Dominio idMedico	Inteiro	
Agendamento (
nºagendamento	Número da consulta	NOT NULL ,
data_hora	Data e hora da consulta	NOT NULL,
Atleta_idAtleta	Id do atleta	NOT NULL,
Medico_idMedico	Id do Médico	NOT NULL,

PRIMARY KEY (nºagendamento)

FOREIGN KEY (Atleta_idAtleta) REFERENCES Atleta (idAtleta)

FOREIGN KEY (Medico_idMedico)REFERENCES Medico (idMedico);

Relação Teste Clínico

Domínio idTesteClinico	Inteiro	
Domínio descrição	String de tamanho variável, tamanho 200	
Domínio estado	String de tamanho variável, tamanho 200	
Domínio nºagendamento	Inteiro	
TesteClinico (
idTesteClinico	Id teste clínico	NOT NULL,
descrição	Descrição	NOT NULL,

estado	Estado do teste	NOT NULL,
Agendamento_nºagendamento	Nºagendamento	NOT NULL,

PRIMARY KEY (idTesteClinico)

FOREIGN KEY (Agendamento_nºagendamento) REFERENCES Agendamento (nºagendamento);

Relação Médico

Domínio idMedico	Inteiro	
Domínio nome	String de tamanho variável, tamanho 200	
Domínio dataInicioSreviço	Data	
Domínio idEspecialidade	Inteiro	

Medico (

idMedico	Id médico	NOT NULL,
nome	Nome do Médico	NOT NULL,
dataInicioServiço	Data do início de Serviço	NOT NULL,
Especialidade_idEspecialidade	Id especialidade	NOT NULL,

PRIMARY KEY (idMedico)

FOREIGN KEY (Especialidade_idEspecialidade) REFERENCES Especialidade (idEspecialidade);

Relação Especialidade

Domínio idEspecialidade	Inteiro	
Domínio designação	String tamanho variável, tamanho 200	

Especialidade (

idEspecialidade	Id da especialidade	NOT NULL,
designação	Designação da especialidade	NOT NULL,

PRIMARY KEY (idEspecialidade);

Relação Utensílio

Domínio idUtensilio	Inteiro	
Domínio nome	String de tamanho variável, tamanho 200	
Utensilio (
idUtensilio	Id do Utensílio	NOT NULL,
nome	Nome do Utensílio	NOT NULL,
 PRIMARY KEY (idUtensilio);		

Relação TesteClinico_has_Utensilio

Domínio idTesteClinico	Inteiro	
Domínio idMedico	Inteiro	
Domínio idUtensilio	Inteiro	
TesteClinico_has_Utensilio (
TesteClinico_idTesteClinico	Id do teste Clínico	NOT NULL,
TesteClinico_Medico_idMedico	Id do Médico	NOT NULL,
Utensilio_idUtensilio	id do utensílio	NOT NULL,
 PRIMARY KEY (TesteClinico_idTesteClinico, Utensilio_idUtensilio, TesteClinico_Medico_idMedico), FOREIGN KEY (TesteClinico_idTesteClinico) REFERENCES TesteClinico (idTesteClinico) FOREIGN KEY (Utensilio_idUtensilio) REFERENCES Utensilio (idUtensilio);		

5.1.2 Desenho das representações dos dados derivados

Como no nosso sistema não temos nenhum atributo cujo valor depende de outros, não abordaremos este tópico.

5.1.3 Desenho das restrições gerais

Nesta fase vamos apresentar as condições restritivas gerais referentes ao problema. Para tal, iremos apresentar para cada relação as respetivas restrições. Irão ser demonstradas pela sua definição no script para a criação das tabelas.

Atleta

```
-- Table `mydb`.`Atleta`  
-----  
DROP TABLE IF EXISTS `mydb`.`Atleta` ;  
  
CREATE TABLE IF NOT EXISTS `mydb`.`Atleta` (  
    `idAtleta` INT NOT NULL AUTO_INCREMENT,  
    `nome` VARCHAR(200) NOT NULL,  
    `dataNascimento` DATETIME NOT NULL,  
    `Modalidade_idModalidade` INT NOT NULL,  
    `telemovel` INT NOT NULL,  
    `Residencia_idResidencia` INT NOT NULL,  
    `Clube_idClube` INT NOT NULL,  
    `Categoria_idCategoria` INT NOT NULL,  
    PRIMARY KEY (`idAtleta`),  
    CONSTRAINT `fk_Atleta_Modalidade`  
        FOREIGN KEY (`Modalidade_idModalidade`)  
            REFERENCES `mydb`.`Modalidade` (`idModalidade`)  
            ON DELETE NO ACTION  
            ON UPDATE NO ACTION,  
    CONSTRAINT `fk_Atleta_Residencial`  
        FOREIGN KEY (`Residencia_idResidencia`)  
            REFERENCES `mydb`.`Residencia` (`idResidencia`)  
            ON DELETE NO ACTION  
            ON UPDATE NO ACTION,  
    CONSTRAINT `fk_Atleta_Clube1`  
        FOREIGN KEY (`Clube_idClube`)  
            REFERENCES `mydb`.`Clube` (`idClube`)  
            ON DELETE NO ACTION  
            ON UPDATE NO ACTION,  
    CONSTRAINT `fk_Atleta_Categorial`  
        FOREIGN KEY (`Categoria_idCategoria`)  
            REFERENCES `mydb`.`Categoria` (`idCategoria`)  
            ON DELETE NO ACTION  
            ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

Figura 21. Criação da tabela Atleta

Modalidade

```
-- Table `mydb`.`Modalidade`  
--  
DROP TABLE IF EXISTS `mydb`.`Modalidade` ;  
  
CREATE TABLE IF NOT EXISTS `mydb`.`Modalidade` (  
  `idModalidade` INT NOT NULL AUTO_INCREMENT,  
  `designação` VARCHAR(200) NOT NULL,  
  PRIMARY KEY (`idModalidade`))  
ENGINE = InnoDB;
```

Figura 22. Criação da tabela Modalidade

Residência

```
CREATE TABLE IF NOT EXISTS `mydb`.`Residencia` (  
  `morada` VARCHAR(200) NOT NULL,  
  `localidade` VARCHAR(200) NOT NULL,  
  `codigoPostal` VARCHAR(10) NOT NULL,  
  `idResidencia` INT NOT NULL AUTO_INCREMENT,  
  PRIMARY KEY (`idResidencia`))  
ENGINE = InnoDB;
```

Figura 23. Criação da tabela Residência

Clube

```
-- Table `mydb`.`Clube`  
--  
DROP TABLE IF EXISTS `mydb`.`Clube` ;  
  
CREATE TABLE IF NOT EXISTS `mydb`.`Clube` (  
  `idClube` INT NOT NULL AUTO_INCREMENT,  
  `mome` VARCHAR(200) NOT NULL,  
  `sigla` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idClube`))  
ENGINE = InnoDB;
```

Figura 24. Criação da tabela Clube

Categoría

```
-- Table `mydb`.`Categoria`  
---  
DROP TABLE IF EXISTS `mydb`.`Categoria` ;  
  
CREATE TABLE IF NOT EXISTS `mydb`.`Categoria` (  
    `idCategoria` INT NOT NULL AUTO_INCREMENT,  
    `designação` VARCHAR(200) NOT NULL,  
    PRIMARY KEY (`idCategoria`))  
ENGINE = InnoDB;
```

Figura 25. Criação da tabela Categoria

Especialidade

```
-- Table `mydb`.`Especialidade`  
---  
DROP TABLE IF EXISTS `mydb`.`Especialidade` ;  
  
CREATE TABLE IF NOT EXISTS `mydb`.`Especialidade` (  
    `idEspecialidade` INT NOT NULL AUTO_INCREMENT,  
    `designação` VARCHAR(45) NULL,  
    PRIMARY KEY (`idEspecialidade`))  
ENGINE = InnoDB;
```

Figura 26. Criação da tabela Especialidade

Médico

```
-- Table `mydb`.`Medico`  
---  
DROP TABLE IF EXISTS `mydb`.`Medico` ;  
  
CREATE TABLE IF NOT EXISTS `mydb`.`Medico` (  
    `idMedico` INT NOT NULL AUTO_INCREMENT,  
    `nome` VARCHAR(200) NOT NULL,  
    `dataInicioServiço` DATE NOT NULL,  
    `Especialidade_idEspecialidade` INT NOT NULL,  
    PRIMARY KEY (`idMedico`),  
    CONSTRAINT `fk_Medico_Especialidadel`  
        FOREIGN KEY (`Especialidade_idEspecialidade`)  
            REFERENCES `mydb`.`Especialidade`(`idEspecialidade`)  
            ON DELETE NO ACTION  
            ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

Figura 27. Criação da tabela Medico

Agendamento

```
-- Table `mydb`.`Agendamento`

DROP TABLE IF EXISTS `mydb`.`Agendamento` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Agendamento` (
  `nºagendamento` INT NOT NULL AUTO_INCREMENT,
  `data_hora` DATETIME NOT NULL,
  `Atleta_idAtleta` INT NOT NULL,
  `Medico_idMedico` INT NOT NULL,
  PRIMARY KEY (`nºagendamento`),
  CONSTRAINT `fk_Agendamento_Atletal`
    FOREIGN KEY (`Atleta_idAtleta`)
    REFERENCES `mydb`.`Atleta` (`idAtleta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Agendamento_Medicol`
    FOREIGN KEY (`Medico_idMedico`)
    REFERENCES `mydb`.`Medico` (`idMedico`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 28. Criação da tabela Agendamento

TesteClínico

```
CREATE TABLE IF NOT EXISTS `mydb`.`TesteClinico` (
  `idTesteClinico` INT NOT NULL AUTO_INCREMENT,
  `descrição` VARCHAR(200) NOT NULL,
  `estado` VARCHAR(200) NOT NULL,
  `Agendamento_nºagendamento` INT NOT NULL,
  PRIMARY KEY (`idTesteClinico`),
  CONSTRAINT `fk_TesteClinico_Agendamento1`
    FOREIGN KEY (`Agendamento_nºagendamento`)
    REFERENCES `mydb`.`Agendamento` (`nºagendamento`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 29. Criação da tabela TesteClinico

Utensílio

```
-- Table `mydb`.`Utensilio`  
  
DROP TABLE IF EXISTS `mydb`.`Utensilio` ;  
  
CREATE TABLE IF NOT EXISTS `mydb`.`Utensilio` (  
  `idUtensilio` INT NOT NULL AUTO_INCREMENT,  
  `nome` VARCHAR(200) NOT NULL,  
  PRIMARY KEY (`idUtensilio`))  
ENGINE = InnoDB;
```

Figura 30. Criação da tabela Utensílio

TesteClinicoHasUtensilio

```
CREATE TABLE IF NOT EXISTS `mydb`.`TesteClinico_has_Utensilio` (  
  `TesteClinico_idTesteClinico` INT NOT NULL,  
  `Utensilio_idUtensilio` INT NOT NULL,  
  PRIMARY KEY (`TesteClinico_idTesteClinico`, `Utensilio_idUtensilio`),  
  CONSTRAINT `fk_TestClinico_has_Utensilio_TestClinico`  
    FOREIGN KEY (`TesteClinico_idTesteClinico`)  
    REFERENCES `mydb`.`TesteClinico` (`idTesteClinico`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_TestClinico_has_Utensilio_Utensilio`  
    FOREIGN KEY (`Utensilio_idUtensilio`)  
    REFERENCES `mydb`.`Utensilio` (`idUtensilio`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

Figura 31. Criação da tabela TesteClinicoHasUtensilio

5.2 Organização dos ficheiros e índices

A nossa base de dados consiste num conjunto de tabelas que se relacionam entre si e que nos permite visualizar os dados que se encontram armazenados na base de dados. Os dados encontram-se armazenados na memória física do dispositivo e, como tal, eles são convertidos para o formato binário e guardados em “blocos”. Assim, o facto dos dados da nossa base de dados estarem a ser guardados na memória física faz com que o tipo de pesquisa que lhes fazemos tenha impacto na obtenção dos resultados. Depois de pesquisarmos acerca de índices, concluímos que são estruturas de dados que permitem ao sistema de gestão de base de dados localizar registos num ficheiro/arquivo, de forma a minimizar o tempo de resposta a queries feitas pelo utilizador. O *InnoDB* usa índice especial para otimizar as operações de DML (*Data Manipulation Language*) mais comuns: SELECT, INSERT, UPDATE e DELETE.

5.3. Validação das relações

Finalizado o modelo físico chegou a altura de fazermos a sua validação. Para tal é necessário responder às mesmas perguntas que foram respondidas na parte anterior de forma a verificar a sua viabilidade através da comparação com o modelo definido.

1. Quais os nomes dos atletas que correm os 1200m barreiras?
2. Quantos testes clínicos foram agendados por atletas do Benfica?
3. Quantos testes clínicos foram supervisionados por médicos com mais de 15 anos de serviço?
4. Quais os nomes dos utensílios utilizados no teste clínico 123456?

Validação:

Query 1:

```
DELIMITER //
#Primeira query: Quantos atletas correram 1200M em Barreiras
CREATE PROCEDURE atletaModCat(in modalidade varchar(200), in Cate varchar(200))
begin
    SELECT Atleta.nome, Modalidade.designação, Categoria.designação
    FROM Atleta
    INNER JOIN Modalidade
    ON Atleta.Modalidade_idModalidade = Modalidade.idModalidade and Modalidade.designação= modalidade
    INNER JOIN Categoria ON Atleta.Categoria_idCategoria=Categoria.idCategoria
    WHERE Categoria.designação= Cate;
end//
DELIMITER ;
call atletaModCat('Corrida de Obstaculos', '1200M');
```

Figura 32. Resolução da Query 1 em SQL

Query 2:

```
DELIMITER //
#Segunda query: Quantos testes clinicos foram agendados por atletas do clube XXXX?
CREATE PROCEDURE testePorClube(in cl VARCHAR(200))
begin
    SELECT count(*) idTesteclinico
    FROM Atleta
    INNER JOIN Clube
    ON Atleta.Clube_idClube = Clube.idClube and Clube.mome= cl
    INNER JOIN Agendamento
    ON Agendamento.Atleta_idAtleta=Atleta.idAtleta
    INNER JOIN TesteClinico
    ON TesteClinico.Agendamento_nºagendamento= Agendamento.nºagendamento;
END//
```



```
call testePorClube('Sport Lisboa e Benfica')
```

Figura 33. Resolução da Query 2 em SQL

Query 3:

```
#Terceira Query: Quantos testes clinicos forma supervisionados por médicos com mais de 15 anos de serviço?  
CREATE PROCEDURE testesPorMedicoAS(in anos int)  
BEGIN  
SELECT distinct TesteClinico.idTesteclinico, Medico.nome  
FROM Medico  
INNER JOIN Agendamento  
ON Agendamento.Medico_idMedico=Medico.idMedico and TIMESTAMPDIFF(YEAR, Medico.dataInicioServiço, now()) > anos  
INNER JOIN TesteClinico  
ON TesteClinico.Agendamento_nºagendamento=Agendamento_nºagendamento;  
END//  
  
call testesPorMedicoAS('15')
```

Figura 34. Resolução da Query 3 em SQL

Query 4:

```
#Quarta QUery: Quais os nomes dos utensilios utilizados no teste clinico xxxx?  
CREATE PROCEDURE utensilioPorTeste(in testealvo int)  
BEGIN  
SELECT distinct Utensilio.nome,  
TesteClinico_has_Utensilio.TesteClinico_idTesteClinico  
FROM Utensilio  
INNER JOIN TesteClinico_has_Utensilio  
ON TesteClinico_has_Utensilio.TesteClinico_idTesteClinico= testealvo;  
END//  
  
call utensilioPorTeste('1')
```

Figura 35. Resolução da Query 4 em SQL

MODELO NÃO RELACIONAL

6. Base de dados NoSQL

O termo NoSQL (Not Only SQL) descreve soluções de armazenamento de base de dados não relacionais. Pode dizer-se que uma base de dados NoSQL é open source e completamente distinta do modelo relacional tradicional, isto porque as bases de dados NoSQL foram criadas a partir de necessidades que as bases de dados tradicionais relacionais não satisfaziam, como alta performance e capacidade de expansão. Existem quatro tipos diferentes de base de dados NoSQL:

- Chave-Valor: Armazena os seus dados num padrão chave-valor, fazendo lembrar as tabelas de hash.
- Grafos: Armazena os seus dados na forma de grafo, com vértices e arestas (ex: Neo4j).
- Colunas: Armazena os seus dados em linhas particulares de uma tabela no disco.
- Documento: armazena os seus dados como documentos, onde um documento pode ser um dado num formato chave-valor (ex: MongoDB).

6.1. Principais características

- Alta performance e escalabilidade horizontal:

Enquanto que os servidores de base de dados tradicionais permitiam a adição de um maior número de recursos ao servidor (como memória e disco para suportar mais dados) as bases de dados NoSQL redistribuem-se horizontalmente, ou seja, funcionam como um sistema pouco conectado, o que a torna mais económica e escalável. Devido a esse mecanismo, a sua performance provém de vários servidores, que não necessitam de ser de alta performance, conectados e a trabalhar em conjunto.

- Replicação:

A replicação é uma estratégia que se encaixa na arquitetura do NoSQL, já que segue o conceito de sistema pouco conectado. Essa ferramenta possibilita o armazenamento de dados e backups independentemente do local físico onde os dados estão armazenados.

- Alta disponibilidade
- API simples para acesso aos dados
- Raízes Open Source:

Muitas bases de dados NoSQL tem raízes na comunidade open source. Talvez tenha sido isso o motivo para o rápido crescimento do seu uso e popularidade.

- Baixo custo operacional:

Devido ao peso do open source no NoSQL, o custo para iniciar a utilização dessas bases de dados torna-se muito baixo ou zero.

- Armazenamento de dados estruturados ou não estruturados:

Permite uma fácil aplicação da escalabilidade e também um aumento na disponibilidade de dados. Na ausência de dados estruturados não existe garantia de haver integridade de dados.

6.2. SQL vs NoSQL

	<i>SQL</i>	<i>NoSQL</i>
Armazenamento de dados	O armazenamento é num modelo relacional, em tabelas, em que as linhas são as entradas/entidades e as colunas os atributos das respetivas entidades.	O NoSQL abrange uma série de base de dados, cada uma com a sua estrutura de armazenamento de dados.
Estrutura e Flexibilidade	Alterações que sejam feitas a nível de estrutura implicam alterar toda a base de dados.	Estruturas dinâmicas. Informações podem ser adicionadas facilmente e permite haver entradas em colunas da tabela vazias.
Escalabilidade	A escalabilidade é vertical. Assim, mais dados implica haver um servidor maior o que poderá ser dispendioso. É possível escalar uma base de dados em vários servidores, mas isso é um processo difícil e demorado.	A escalabilidade é horizontal, como já referido anteriormente. Mais económico.
Propriedades ACID	A grande maioria dos bancos de dados relacionais são compatíveis com ACID.	Varia de acordo com as tecnologias, mas muitas soluções NoSQL sacrificam a compatibilidade ACID para desempenho e escalabilidade.

Tabela 14. Tabela das principais diferenças entre SQL e NoSQL

6.3. Vantagens

- Dados sempre disponíveis;
- Custo mais reduzido que uma base de dados relacional;
- Base de dados orientada a objetos flexíveis;
- Facilidade em introduzir novos dados;
- Excelente ao lidar com o problema de dados em massa.

6.4. Limitações

- Mais recentes que as bases de dados relacionais, não contendo tantas alternativas;
- Não resolve problemas de escalabilidade de um website;
- Situações onde o modo como os dados estão estruturados é importante.

6.5. Modelo de Dados NoSQL baseado em grafos

As bases de dados relacionais não permitem que os dados sejam representados através de grafos, fazendo com que algumas pesquisas se tornem extremamente complexas ou

impossíveis de realizar. Uma base de dados baseada em grafos utiliza o modelo de grafos para representar o esquema. Esse modelo apresenta três componentes: nodos (vértices), relacionamentos entre nodos (arestas) e as propriedades dos nodos ou dos relacionamentos existentes entre eles. Assim, neste tipo de base de dados, basta navegarmos pela estrutura de dados desse grafo para conseguirmos obter os resultados esperados.

6.6. Novo sistema de base de dados

Na figura que se segue podemos ter uma melhor percepção de como vai ficar implementada a nossa base de dados em Neo4j, identificando todos os tipos de nodos e todos os relacionamentos entre os nodos. A figura vai funcionar como um esboço de modo a facilitar a percepção do conteúdo de dados que a nossa base de dados vai suportar.

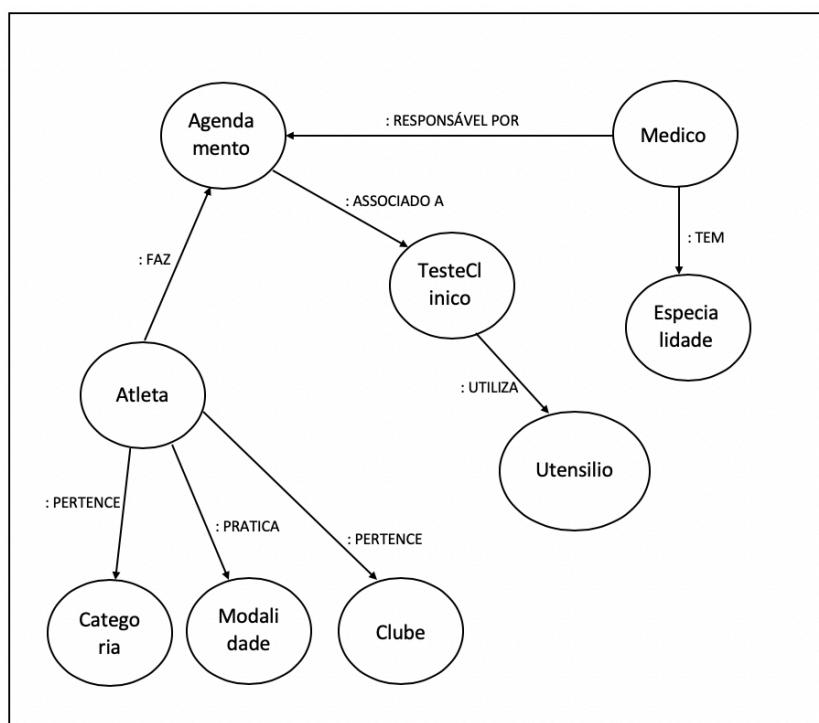


Figura 36. Esquema tipo do novo sistema de base de dados

6.7. Migração de dados

Esta migração consiste no processo de exportação dos dados que temos na nossa base de dados em MySQL, previamente povoada na primeira parte do projeto, para ficheiros com extensão .csv. Cada tabela vai ser exportada para um ficheiro diferente com o respetivo nome.

6.7.1. Importação de ficheiros csv para Neo4j

Para cada tabela, usamos a ferramenta “table data export wizard” do Workbench, para fazer o respetivo ficheiro csv. Depois de termos os ficheiros .csv, fizemos vários scripts na linguagem Cypher para criar os nodos e os respetivos relacionamentos. Utilizamos um ‘LOAD’ para carregarmos o ficheiro .csv com as informações da tabela, um ‘CREATE’ para criarmos os respetivos nodos e com um ‘MERGE’ os relacionamentos presentes entre nodos.

6.7.2. Criação de nodos

```
//Criação nodos Atleta
CREATE CONSTRAINT ON(a:Alteta) ASSERT a.idAtleta IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///atle.csv" AS line
CREATE(a:Atleta {idAtleta:TOINTEGER(line.idAtleta),
Nome:line.nome,
DataNascimento:line.dataNascimento,
idModalidade: TOINTEGER(line.Modalidade_idModalidade),
Telemovel: line.telemovel,
Residencia: TOINTEGER(line.Residencia_idResidencia)
Clube: TOINTEGER(line.Clube_idclube),
Categoria: TOINTEGER(line.Categoria_idCategoria)});

//Criação Nodos Modalidade
CREATE CONSTRAINT ON(m:Modalidade) ASSERT m.idModalidade IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///mod.csv" AS line
CREATE(m:Modalidade {idModalidade:TOINTEGER(line.idModalidade),
Designação: line.designação});

//Criação nodos Residencia
CREATE CONSTRAINT ON(r:Residencia) ASSERT r.idResidencia IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///res.csv" AS line
CREATE(r:Residencia {idResidencia:TOINTEGER(line.idResidencia),
Morada:line.morada,
Localidade:line.localidade,
CodigoPostal: line.codigoPostal});

//Criação nodos Clube
CREATE CONSTRAINT ON(clu:Clube) ASSERT clu.idClube IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///clu.csv" AS line
CREATE(clu:Clube {idClube:TOINTEGER(line.idClube),
Nome:line.mome,
Sigla:line.sigla});

//Criação Nodos Categoria
CREATE CONSTRAINT ON(cat:Categoria) ASSERT cat.idCategoria IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///cat.csv" AS line
CREATE(cat:Categoria {idCategoria:TOINTEGER(line.idCategoria),
Designação:line.designação});
```

```

//Criação Nodos Especialidade
CREATE CONSTRAINT ON(esp:Especialidade) ASSERT esp.idEspecialidade IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///esp.csv" AS line
CREATE(esp:Especialidade {idEspecialidade:TOINTEGER(line.idEspecialidade),
Designação:line.designação});

//Criação Nodos Medico
CREATE CONSTRAINT ON(med:Medico) ASSERT med.idMedico IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///med.csv" AS line
CREATE(med:Medico {idMedico:TOINTEGER(line.idMedico),
Nome:line.nome,
DataInicioServiço:line.dataInicioServiço,
idEspecialidade: TOINTEGER(line.Especialidade_idEspecialidade)});

//Criação Nodos Agendamento
CREATE CONSTRAINT ON(age:Agendamento) ASSERT age.nºagendamento IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///agen.csv" AS line
CREATE(age:Agendamento {nºagendamento:TOINTEGER(line.nºagendamento),
DataHora:line.data hora,
idAtleta: TOINTEGER(line.Atleta_idAtleta),
idMedico: TOTNTEGER(line.Medico_idMedico)}).
//Criação Nodos Teste Clinico

CREATE CONSTRAINT ON(test:TesteClinico) ASSERT test.idTesteClinico IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///tc.csv" AS line
CREATE(test:TesteClinico {idTesteClinico:TOINTEGER(line.idTesteClinico),
Descrição:line.descrição,
Estado:line.estado,
Agendamento: TOINTEGER(line.Agendamento_nºagendamento)});

//Criação Nodos Utensilio
CREATE CONSTRAINT ON(uten:Utensilio) ASSERT uten.idUtensilio IS UNIQUE;
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///ut.csv" AS line
CREATE(uten:Utensilio {idUtensilio:TOINTEGER(line.idUtensilio),
Nome:line.nome});

```

Figura 37. Script de criação de nodos

Como podemos observar nas imagens acima, temos os scripts que são responsáveis por efetuarem a criação dos nodos com base nos valores contidos no ficheiro.csv. Para além de criarmos os respetivos nodos inserimos toda a informação que o ficheiro .csv continha sobre esse nodo em específico.

6.7.3. Criação de indexes

```
//Start dos indices
CREATE INDEX ON :Atleta(idAtleta);
CREATE INDEX ON :Modalidade(idModalidade);
CREATE INDEX ON :Residencia(idResidencia);
CREATE INDEX ON :Clube(idClube);
CREATE INDEX ON :Categoria(idCategoria);
CREATE INDEX ON :Especialidade(idEspecialidade);
CREATE INDEX ON :Medico(idMedico);
CREATE INDEX ON :Agendamento(nºagendamento);
CREATE INDEX ON :Utensilio(idUtensilio);
CREATE INDEX ON :TesteClinico(idTesteClinico);
```

Figura 38. Script de criação de indexes

Estes Indexes ou índices vão ser uma cópia redundante das informações da nossa base de dados e tem como objetivo tornar a pesquisa de dados mais eficiente. Isto requer um espaço extra de armazenamento e gravações mais lentas, pelo que este processo desempenha um papel importante. A linguagem Cypher permite a criação de índices sobre uma dada propriedade para todos os nodos que têm um rótulo em comum. Uma vez criado o índice, ele será automaticamente gerenciado e atualizado pela base de dados, sempre que o gráfico for alterado.

6.7.4. Criação de relacionamentos

```
//Relacionamento
MATCH (a:Atleta),(r:Residencia)
WHERE a.Residencia=r.idResidencia
CREATE (a) -[sm:Tem]-> (r);

MATCH (a:Atleta),(cat:Categoria)
WHERE a.Categoria=cat.idCategoria
CREATE (a) -[sm:Pertence]-> (cat);

MATCH (a:Atleta),(m:Modalidade)
WHERE a.idModalidade=m.idModalidade
CREATE (a) -[sm:Pratica]-> (m);

MATCH (a:Atleta),(clu:Clube)
WHERE a.Clube=clu.idClube
CREATE (a) -[sm:Pertence]-> (clu);

MATCH (a:Atleta),(age:Agendamento)
WHERE a.idAtleta=age.idAtleta
CREATE (a) -[sm:Tem]-> (r);

MATCH (med:Medico),(age:Agendamento)
WHERE med.idMedico=age.idMedico
CREATE (med) -[sm:responsavel]-> (age);

MATCH (med:Medico),(esp:Especialidade)
WHERE med.Especialidade=esp.idEspecialidade
CREATE (med) -[sm:Tem]-> (esp);

MATCH (age:Agendamento),(test:TesteClinico)
WHERE age.nºagendamento=test.Agendamento
CREATE (age) -[sm:realizado]-> (test);
```

Figura 39. Script da criação de relacionamentos

Como podemos verificar nas imagens anexas em cima, possuímos oito diferentes tipos de relacionamentos entre os nodos. Estes são:

1. Pertence

É o relacionamento entre o Atleta e a Categoria a que o mesmo pertence.

2. Pratica

É o relacionamento entre o Atleta e a Modalidade que este pratica.

3. Pertence

É o relacionamento entre o Atleta e o Clube a que este pertence.

4. Faz

É o relacionamento entre a Atleta e Agendamento, ou seja, o atleta é responsável por fazer a marcação de um agendamento.

5. Associado a

É o relacionamento entre Agendamento e TesteClinico, ou seja, a cada teste clínico está associado um dado agendamento.

6. Utiliza

É o relacionamento entre TesteClinico e Utensilio na medida que para se realizar um teste clínico é necessário a utilização de diversos utensílios.

7. Responsável por

É o relacionamento entre Medico e Agendamento, ou seja, um médico é responsável por um dado agendamento.

8. Tem

É o relacionamento entre o Medico e Especialidade que este tem.

Os relacionamentos mencionados em cima vão desempenhar um papel importante na elaboração de queries futuras, pois vai ser através deles que vamos navegar no grafo de modo a obtermos a informação desejada.

6.8. Queries

De seguida apresentamos as queries e a sua respetiva explicação que efetuamos na nossa base de dados criada nesta segunda fase do projeto, desta vez utilizando Neo4j.

1. Quais os nomes dos atletas que correm os 1200m barreiras?
 2. Quantos testes clínicos foram agendados por atletas do Benfica?
 3. Quantos testes clínicos foram supervisionados por médicos com mais de 15 anos de serviço?
 4. Quais os nomes dos utensílios utilizados no teste clínico 123456?

Validação:

6.9. Versão final da base de dados em Neo4j

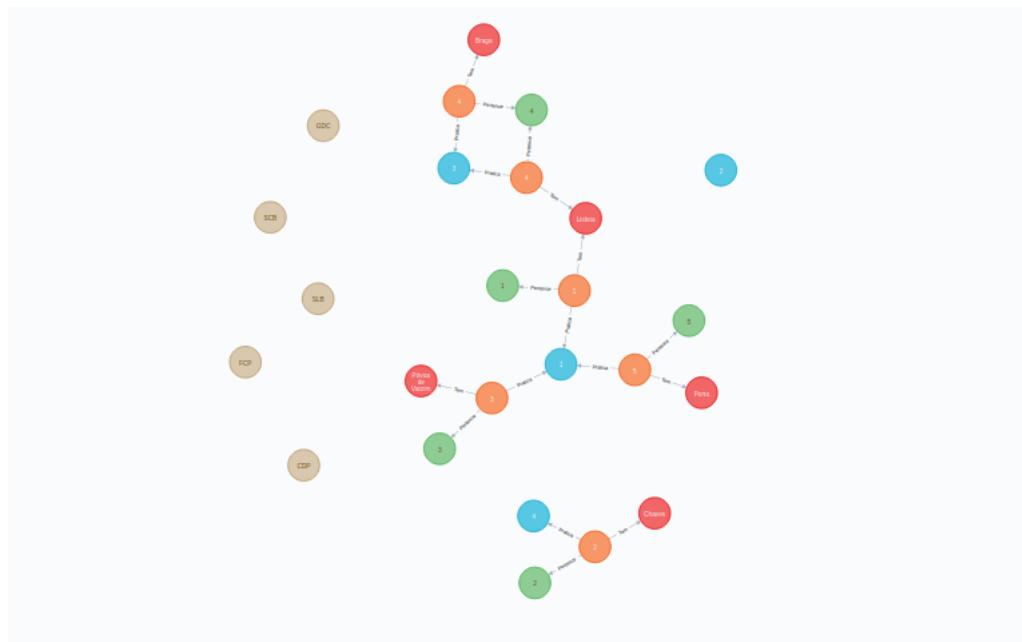


Figura 40. Modelo final da nova base de dados

2. Conclusões e Trabalho Futuro

A implementação e elaboração de um sistema de bases de dados é uma tarefa que requer diferentes cuidados ao longo da mesma. Desta maneira, é preciso seguir certos passos para o seu desenvolvimento, nomeadamente a análise dos requisitos do problema, onde são definidas as entidades e relacionamentos do projeto etc de forma a obtermos uma melhor performance possível para o sistema. Para a elaboração deste projeto, seguimos a metodologia sugerida pelo livro Database Systems- Thomas Connolly.Carolyn Begg.

Numa fase inicial do projeto foi elaborado o modelo conceptual. Seguidamente foram elaborados os modelos lógico e esquema físico. Na segunda parte do trabalho tivemos de adaptar os modelos anteriormente feitos, passando por várias etapas e percebendo o que é necessário para a passagem de uma base de dados relacional a uma não relacional. O uso de NoSQL e nomeadamente do Neo4J deu-nos outra perspetiva sobre as diferentes maneiras da criação de uma base de dados, pois esta é constituída por grafos e nodos, algo completamente diferente daquilo que tínhamos feito numa primeira parte do projeto. Além disso tivemos de nos adaptar a outra linguagem na criação da nova base de dados, que serviu para criar nodos, criar relacionamentos e realizar as queries para obtermos a informação necessária da nossa base de dados.

Ao realizarmos este trabalho que consistia em realizarmos a mesma base de dados num modelo relacional e num modelo não relacional, adquirimos uma ideia de como ambas funcionam e respetiva importância que cada uma têm no mercado. As bases de dados relacionais são as mais usadas no mundo de trabalho, mas à medida que o tempo vai avançando as não relacionais também vão ganhando o seu espaço no mercado devido às características referidas acima. Ou seja, ao realizarmos este trabalho ficamos também com um sentido crítico para num projeto futuro decidirmos qual o modelo de base de dados a implementar consoante os requisitos que nos vão ser apresentados.

Referências

Thomas M. Connolly, Carolyn E. Begg - Database Systems: A Practical Approach to Design, Implementation and Management - 4th Edition.

Website do Neo4j. <https://neo4j.com/>

Lista de Siglas e Acrónimos

BD	Base de Dados
SGBD.....	Sistema de Gestão de Base de Dados
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
ER	Entidade-Relacionamento