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

Unidade Curricular de Bases de Dados

Ano Letivo de 2019/2020

Health First

Beatriz Rocha, Filipe Guimarães, Gonçalo Ferreira

Janeiro, 2020

BD

Data de Receção	
Responsável	
Avaliação	
Observações	

Health First

Beatriz Rocha, Filipe Guimarães, Gonçalo Ferreira

Janeiro, 2020

Resumo

O presente relatório foi desenvolvido no âmbito da disciplina de Base de Dados com vista na criação de uma base de dados para a empresa Health First, da área de análises clínicas em desportistas.

Neste relatório são apresentadas todas as etapas de desenvolvimento da base de dados bem como a implementação física da mesma.

Começamos por estudar o meio em que o nosso software vai ser inserido. Analisamos o motivo pelo qual surgiu a necessidade de uma base de dados e que objetivos esta precisava de cumprir.

Através de contacto com o cliente levantamos requisitos para determinar o seguimento do processo de desenvolvimento. A partir destes, e usando a ferramenta "TerraER", desenvolvemos um modelo conceptual adequado. Com recurso ao "MySQL Workbench", criamos então o modelo lógico, também este de conformidade com os requisitos do cliente. Com o modelo lógico definido e através da mesma ferramenta geramos a base de dados.

Com a implementação da base de dados no cliente demos por terminado este projeto.

Área de Aplicação: Desenho, Arquitetura, Desenvolvimento e Implementação de Sistemas de Bases de Dados.

Palavras-Chave: MySQL Workbench, SQL, Bases de Dados, Bases de Dados Relacionais**,** Modelo Conceptual, Modelo Lógico, Modelo Físico, Triggers, Neo4J.

Índice

Resumo	i
Índice	ii
Índice de Figuras	v
Índice de Tabelas	vii
1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	1
1.3. Motivação e Objetivos	1
1.4. Estrutura do Relatório	2
2. Levantamento e Análise dos Requisitos	3
2.1. Método para o Levantamento dos Requisitos	3
2.2. Levantamento de Requisitos	3
2.2.1 Requisitos de descrição	3
2.2.2 Requisitos de exploração	4
2.2.3 Requisitos de Controlo	5
2.3. Análise de Requisitos	5
3. Modelação Conceptual	6
3.1. Abordagem Adotada para a Modelação Conceptual	6
3.1.1 Modelo Conceptual	6
3.1.2 Entidades	6
3.1.3 Relacionamentos	9
3.1.4 Caracterização dos atributos de cada entidade	10
4. Modelação Lógica	12
4.1. Criação do modelo lógico de dados	12
4.1.1 Atleta	12
4.1.2 Competicao	13
4.1.3 Modalidade	13
4.1.4 Clube	14
4.1.5 Teste Clinico	14
4.1.6 Recurso	14
4.1.7 Staff	15
4.1.8 Clinica	15
4.2. Modelo Lógico	16
4.3. Validação do Modelo com as Transações Estabelecidas	16
4.3.1 Adicionar uma clínica	17
4.3.2 Adicionar Atleta	17

4.4. Avaliação do modelo lógico 18
5. Implementação Física 19
5.1. Seleção do Sistema de Gestão de Base de Dados 19
5.2. Tradução do Esquema Lógico para o Sistema de Gestão de Bases de Dados escolhido em SQL 19
5.3. Tradução das Interrogações do Utilizador para SQL 19
5.3.1 Alguns exemplos de interrogações implementadas 20
5.4. Tradução das transações estabelecidas para SQL 22
5.5. Triggers SQL 24
5.6. Estimativa do Espaço em Disco da Base de Dados e Taxa de Crescimento Anual 25
5.7. Definição e caraterização das vistas de utilização em SQL 27
5.8. Definição e caraterização dos Mecanismos de Segurança em SQL 28
5.9. Revisão do Sistema Implementado com o Utilizador 29
6. NoSQL – Neo4j 30
6.1. Motivo para adotar uma base de dados NoSQL 30
6.2. Objetivo desta implementação 30
6.3. Estrutura Base 30
6.4. Migração de dados 31
6.5. Questões anteriores usando NoSQL 31
7. Conclusões e Trabalho Futuro 33
Referências 34
Lista de Siglas e Acrónimos 35

Anexos

I. Anexo 1 – Script de Inicialização da Base de Dados 37
II. Anexo 2 – Script para carregar CSVs para povoamento inicial 44
III. Anexo 3 – Ficheiros CSV para povoamento inicial 46
IV. Anexo 4 - Script de implementação de functions 59
V. Anexo 5 - Script de implementação de triggers 61
VI. Anexo 6 - Script de implementação de view's 63
VII. Anexo 7 - Script de implementação de procedures para atletas 64
VIII. Anexo 8 - Script de implementação de procedures para diretores das clínicas 66
IX. Anexo 9 - Script de implementação de procedures para responsável de teste 69
X. Anexo 10 - Script de implementação de transactions 72
XI. Anexo 11 - Script de implementação de criação de user e atribuição de premições 75
XII. Anexo 12 - Script de load de csv's para Neo4j 77
XIII. Anexo 13 - Script de implementação de functions em Neo4j 82
XIV. Anexo 14 - Script de implementação de triggers em Neo4j 83

XV. Anexo 15 Script de implementação de view's em Neo4j 84
XVI. Anexo 16 - Script de implementação de procedures para atletas 85
XVII. Anexo 17 - Script de implementação de procedures para diretores das clínicas 86
XVIII. Anexo 18 - Script de implementação de procedures para responsável de teste 87
XIX. Anexo 19 - Script de implementação de criação de user e atribuição de premições 88

Índice de Figuras

Figura 1 - Modelo Conceptual Completo. 6
Figura 2 - Entidade Atleta 7
Figura 3 - Entidade clube 7
Figura 4 - Entidade Competição 7
Figura 5 - Entidade modalidade 8
Figura 6 - Entidade teste clinico 8
Figura 7 - Entidade recurso 8
Figura 8 - Entidade staff 8
Figura 9 - Entidade clinica 9
Figura 10 - Passagem da entidade atleta 12
Figura 11 - Passagem da entidade competicao 13
Figura 12 - Passagem da entidade modalidade 13
Figura 13 - Passagem da entidade clube 14
Figura 14 - Passagem da entidade teste clinico 14
Figura 15 - Passagem da entidade recurso 14
Figura 16 - Passagem da entidade staff 15
Figura 17 - Passagem da entidade clinica 15
Figura 18 - Desenho do Modelo Lógico 16
Figura 19 - Tabela Staff 17
Figura 20 - Tabela Clinica 17
Figura 21 - Tabela modalidade 17
Figura 22 - Tabela Clube 18
Figura 23 - Tabela Atleta 18
Figura 24 - Imagem do código obtido para a query 1 20
Figura 25 - Imagem do código obtido para a query 2 21
Figura 26 - Imagem do código obtido para a query 3 21
Figura 27 - Imagem do código obtido para a transaction 1 22
Figura 28 - Imagem do código obtido para a transaction 2 23
Figura 29 - Imagem do código obtido para o trigger 1 24
Figura 30 - Imagem do código obtido para o trigger 2 24
Figura 31 - Imagem do código obtido para o trigger 3 25
Figura 32 - Atletas ordenados por ordem alfabética 27
Figura 33 - Funcionários ordenados por ordem alfabética 27
Figura 34 - Clínicas ordenadas por ordem alfabética 27
Figura 35 - Testes clínicos ordenados por ordem cronológica 28

Figura 36 - Criação e atribuição de permissões aos utilizadores e administrador 29
Figura 32 - Legenda Neo4j 30
Figura 33 – Grafo Exemplo Neo4j 31

Índice de Tabelas

Tabela 1 - Caracterização dos Atributos 11
Tabela 2 - Bytes necessários por tabela 25

1. Introdução

1.1. Contextualização

O sistema imunitário é afetado de forma altamente específica pelo exercício físico por diversas formas, e por isso é necessário que atletas sejam constantemente obrigados a fazer testes clínicos.

Por exemplo, para a UEFA todos os jogadores que pertençam à equipa principal de um clube têm de ter nos seus registos médicos pessoais exames médicos. Com base nos resultados destes exames e, mediante a opinião profissional do médico, podem ser indicados outros testes de modo a garantir um acompanhamento médico adequado do jogador.

Assim sendo, a Health First nasceu com o objetivo de permitir que qualquer organização seja capaz de ver resolvidas as suas necessidades dentro dos seus serviços prestados. O seu principal foco passa pela capacidade de trabalhar com diversas empresas em simultâneo e sem conflitos, facilitando tanto as empresas como os atletas em competição.

1.2. Apresentação do Caso de Estudo

A Health First foi fundada em janeiro de 2020 por quatro estudantes da Universidade do Minho, onde está sediada, e tem como principal objetivo o agendamento e realização de Testes Clínicos de atletas de diferentes modalidades.

Quando foi contactada, a Health First mostrou-se interessada por suportar um projeto como este. Com interesse e experiência na área compromete-se a usar métodos necessários para que seja criada uma base de dados capaz de satisfazer o cliente.

1.3. Motivação e Objetivos

O que nos motivou a aceitar este projeto foi o facto de a base de dados anterior estar muito desatualizada, não ter sido bem conseguida e já não cumprir os objetivos que o cliente tinha em mente. Seguimos então por um caminho metódico. Queremos que a base de dados desenvolvida esteja em conformidade com todos os requisitos do cliente.

Tomamos então como o nosso primeiro objetivo o levantamento e análise de requisitos, para que pudéssemos, de maneira incremental, conceber uma solução que favorecesse o cliente.

1.4. Estrutura do Relatório

O relatório está dividido por 7 capítulos.

Começamos, como referido anteriormente, por explicar o nosso método de **Levantamento e Análise dos Requisitos**. Apresentamos e analisamos os requisitos que o cliente nos propôs a desenvolver.

No terceiro capítulo, com atenção aos requisitos, criamos um **Modelo Conceptual** para nos guiarmos na conceção da nossa base de dados.

No quarto capítulo a partir do modelo conceptual apresentamos o desenvolvimento do **Modelo Lógico** bem como as respetivas justificações para o mesmo.

No quinto capítulo descrevemos a **Implementação Física** da nossa base de dados. Demos também exemplos daquilo que o nosso sistema é capaz de realizar.

No sexto capítulo explicamos o motivo de passarmos de uma base de dados relacional para uma base de dados não relacional usando o Neo4j.

No sétimo e último capítulo do presente relatório apresentamos as nossas **conclusões** sobre o desenvolvimento deste projeto bem como futuras funcionalidades que este possa vir a ter.

Por fim, este relatório, possui também uma parte de anexos. Nos anexos estão presentes todos os scripts desenvolvidos, tanto de SQL como de NoSQL, para a base de dados.

2. Levantamento e Análise dos Requisitos

2.1. Método para o Levantamento dos Requisitos

A maneira de abordar esta fase passou por entrevistas com os principais clientes da Health First. Com estas conseguimos perceber o que os clientes desejavam conseguir obter e como.

Observamos posteriormente o funcionamento da empresa com quem estamos a trabalhar, percebemos os seus objetivos, princípios e o seu método de trabalho. Vimos como diferentes utilizadores interagiam com o sistema.

Foram também criados clientes falsos que nos mostravam como interagiam com o sistema atual, os seus comportamentos e quais eram as suas necessidades.

Com estes métodos conseguimos agregar e particionar todos os requisitos que serão apresentados de seguida.

2.2. Levantamento de Requisitos

2.2.1 Requisitos de descrição

1. O administrador deverá poder adicionar uma clínica à base de dados e, consequentemente, será necessário fornecer o respetivo código, nome e localidade.
2. O administrador deverá poder adicionar um atleta à base de dados e, consequentemente, deverá ser fornecido o seu NIF, o seu código-postal, a sua data de nascimento, o seu sexo, o seu peso e a sua altura.
3. Cada atleta deverá estar associado à sua modalidade e ao seu clube e poderá estar associado a um ou mais contactos.
4. Aquando do registo do clube deverá ser fornecido o respetivo código e designação.
5. O mesmo terá de se verificar para a modalidade que, por sua vez, deverá estar associada a uma competição.
6. Aquando do registo de uma competição deverá ser fornecido o respetivo identificador, designação, nome, data, local e hora e a ela estará associada uma inscrição numa determinada data e a uma determinada hora.
7. Aquando do registo de um teste clínico deverá ser fornecido o identificador do teste, a data, o preço e a hora.
8. Cada teste clínico deverá estar associado aos recursos utilizados, aos funcionários que nele participaram (EquipaEnvolvente) e à clínica onde é realizado que, por sua vez, deve estar

associada ao funcionário que a dirige (Staff) e ainda aos funcionários que nela trabalham (Staff_Clinica).

9. Para adicionar um novo recurso será necessário fornecer o respetivo código e designação.
10. Para adicionar um novo funcionário será necessário fornecer o seu identificado, nome, cargo, especialidade e data de início de serviço.

2.2.2 Requisitos de exploração

1. O atleta deverá poder ver todas as informações relativas a todos os testes clínicos que realizou.
2. O atleta deverá poder ver todas as informações relativas a todos os testes clínicos que realizou numa determinada clínica
3. O atleta deverá poder ver todas as suas informações bem como todas as informações relativas a todos os testes clínicos que realizou entre duas datas.
4. O atleta deverá poder ver o identificador e o preço relativos aos testes clínicos que realizou ordenados por preço (ascendente).
5. O atleta deverá poder ver o código da clínica e o respetivo montante gasto na clínica em que gastou mais dinheiro.
6. O diretor do teste clínico deverá poder ver o identificador do teste clínico e a designação do recurso usado no teste clínico que dirigiu.
7. O diretor do teste clínico deverá poder ver o nome da clínica e o número de testes clínicos que dirigiu na clínica em que dirigiu mais testes clínicos.
8. O diretor do teste clínico deverá poder ver o identificador do teste clínico e o nome do funcionário com quem trabalhou no teste clínico que dirigiu.
9. O diretor do teste clínico deverá poder ver o identificador do teste clínico bem como todas as informações relativas ao atleta que examinou no teste clínico que dirigiu.
10. O diretor do teste clínico deverá poder ver o identificador do teste clínico bem como o nome e contacto do atleta que examinou no teste clínico que dirigiu.
11. O diretor da clínica deverá poder ver o nome da clínica, o número de atletas examinados e a faturação da clínica que dirige entre duas datas.
12. O diretor da clínica deverá poder ver a designação e o número de testes clínicos do clube que realizou mais testes clínicos entre duas datas na clínica que dirige.
13. O diretor da clínica deverá poder ver a designação do clube que nunca realizou testes na clínica que dirige.
14. O diretor da clínica deverá poder ver quantos funcionários de cada especialidade tem na clínica que dirige.
15. O diretor da clínica deverá poder ver os meses e o número de testes realizados em cada um deles na clínica que dirige ordenados por ordem crescente.

16. O administrador deverá poder ver todas as informações dos atletas ordenados por ordem alfabética.
17. O administrador deverá poder ver todas as informações dos funcionários ordenados por ordem alfabética.
18. O administrador deverá poder ver todas as informações das clínicas ordenadas por ordem alfabética.
19. O administrador deverá poder ver todas as informações dos testes clínicos ordenados por ordem cronológica (do mais recente para o mais antigo).

2.2.3 Requisitos de Controlo

1. O atleta apenas pode consultar os aspetos do sistema que a este estão diretamente ou indiretamente associados.
2. O diretor do teste clínico apenas pode consultar os aspetos que a este estão diretamente ou indiretamente associados.
3. O diretor da clínica apenas pode consultar os aspetos que a este estão diretamente ou indiretamente associados.
4. O administrador pode consultar e alterar todos os aspetos do sistema.

2.3. Análise de Requisitos

Após várias entrevistas com o cliente conseguimos finalmente chegar aos requisitos que tinha em mente. Com a análise dos mesmos e recorrendo a um modelo conceptual que será apresentado no próximo capítulo estamos agora mais perto da implementação física da base de dados.

3. Modelação Conceptual

3.1. Abordagem Adotada para a Modelação Conceptual

Após a análise dos requisitos obtidos no cliente achamos que o mais natural a seguir passava pela criação de um modelo conceptual para a nossa base de dados.

Este modelo foi construído com recurso à ferramenta *TerraER* sempre recorrendo ao cliente a cada iteração, de modo a chegar a um modelo válido e de acordo com os requisitos.

3.1.1 Modelo Conceptual

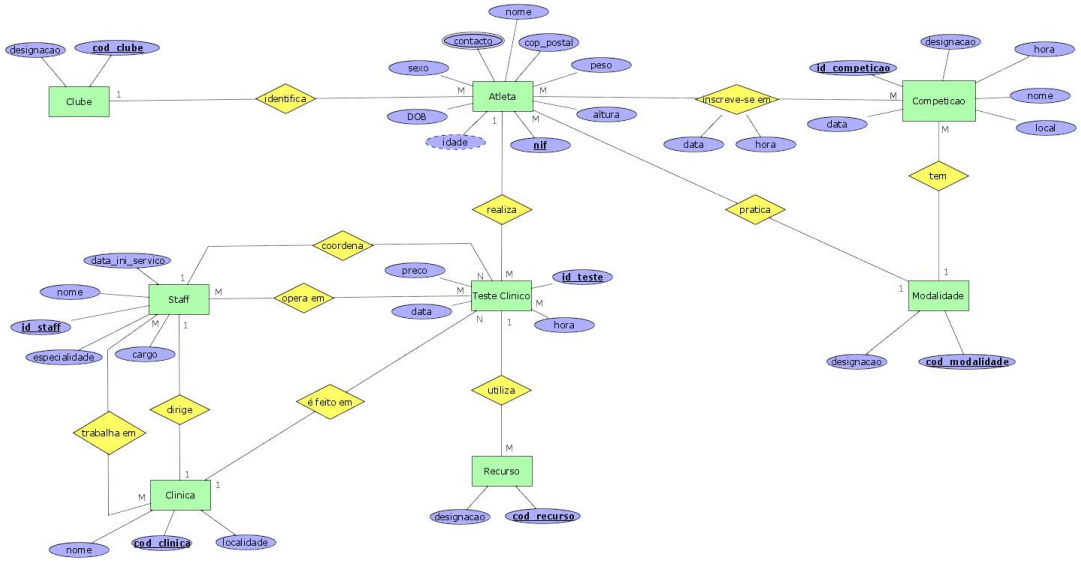

Figura 1 - Modelo Conceptual Completo.

A imagem acima mostra o nosso modelo conceptual de acordo com os requisitos do cliente. Nos pontos seguintes vamos explicar o processo até chegarmos a este modelo.

3.1.2 Entidades

Começamos por identificar as entidades que tinham de estar presentes no nosso modelo:

- **Atleta -** representa um atleta. Cada um dos atletas apresenta **nome**, **contacto** (pode conter mais que um), **sexo** (masculino ou feminino), **DOB** (data de nascimento), **idade** (calculado a partir da DOB), **NIF**, **altura**, **peso** e **código postal**.

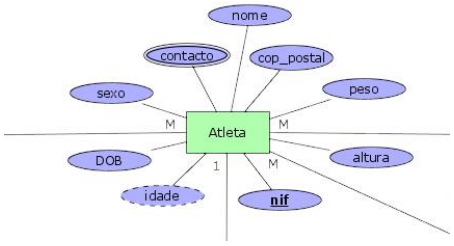

Figura 2 - Entidade Atleta

- **Clube** - representa um clube ou associação. Cada clube apresenta um **código de clube** e uma designação.

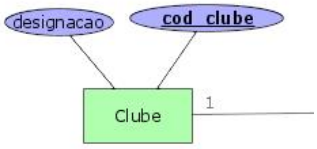

Figura 3 - Entidade clube

- **Competicao** – representa uma competição ou jogo. Cada competição contem um **ID**, uma **designação**, **data**, **hora**, **nome** e **local**.

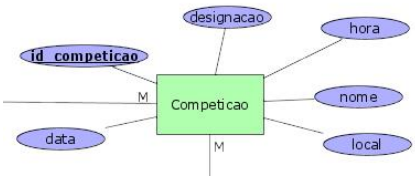

Figura 4 - Entidade Competição

- **Modalidade** – representa uma modalidade. Cada modalidade contém um **código de modalidade** e a sua respetiva **designação**.

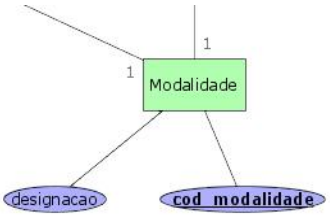

Figura 5 - Entidade modalidade

- **Teste Clinico –** representa um teste clínico. Cada teste clínico contém um **ID**, uma **hora**, uma **data** e um **preço** associado.

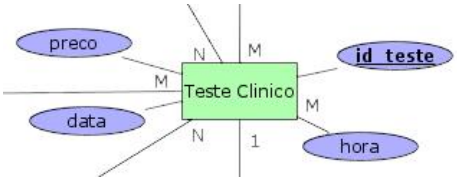

Figura 6 - Entidade teste clinico

- **Recurso** – representa um recurso médico. Cada recurso tem associado o seu **código** e **designação**.

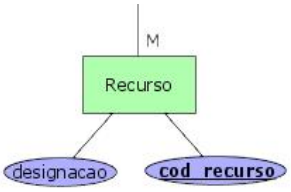

Figura 7 - Entidade recurso

- **Staff** – representa os funcionários. Cada staff tem associado um **ID**, a **data** em que começou a exercer, o seu **nome**, o **cargo** que ocupa e a sua **especialidade**.

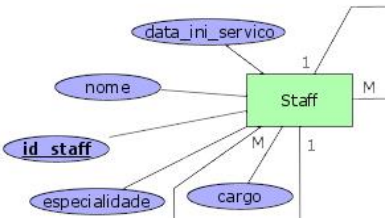

Figura 8 - Entidade staff

- **Clinica** – representa uma clínica. Cada clínica contém associado um **código**, a **localidade** onde está sediada e o seu **nome**.

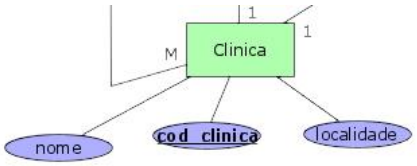

Figura 9 - Entidade clinica

3.1.3 Relacionamentos

Relacionamentos presentes no nosso modelo conceptual:

- **Clube – Atleta**
  - **Atributos –** Nenhum
  - **Relação –** Um clube identifica Atleta
  - **Cardinalidade –** Clube (1) – Atleta (N)

    Um clube é composto pelos seus atletas, enquanto um atleta só pode ser identificado exclusivamente por um clube.

- **Atleta - Competição**
  - **Atributos – data** e **hora** em que o atleta se inscreveu numa competição
  - **Relação –** Um atleta inscreve-se numa competição.
  - **Cardinalidade –** Atleta (N) – Competição (N)

    Um atleta pode inscrever-se em muitas competições, assim como uma competição pode ter muitos atletas inscritos.

- **Atleta - Modalidade**
  - **Atributos –** Nenhum
  - **Relação –** Um atleta pratica uma certa modalidade
  - **Cardinalidade –** Atleta (N) – Modalidade (1)

    Um atleta pratica uma certa modalidade e uma modalidade é praticada por muitos atletas.

- **Competição - Modalidade**
  - **Atributos –** Nenhum
  - **Relação –** Uma competição tem uma modalidade.
  - **Cardinalidade –** Competição (N) – Modalidade (1)

    Uma competição tem uma modalidade associada e existem várias competições para uma certa modalidade.

- **Atleta – Teste Clinico**

- **Atributos –** Nenhum
- **Relação –** Um atleta realiza testes clínicos
- **Cardinalidade –** Atleta (1) – Teste Clinico (N)

  Um atleta realiza vários testes clínicos e vários testes clínicos são realizados a atletas.

- **Staff - Teste Clinico**
  - **Atributos –** Nenhum
  - **Relação –** Vários membros do Staff operam num teste clínico / Um membro do staff coordena os Testes Clínicos
  - **Cardinalidade –** Staff (N) – Teste Clinico (N) / Staff (1) – Teste Clinico (N)

    Muitos membros do Staff operam num teste clinico em que um coordena o mesmo.

- **Staff - Clinica**
  - **Atributos –** Nenhum
  - **Relação –** Vários membros do Staff trabalham na clínica / Um membro do Staff dirige a clinica
  - **Cardinalidade –** Staff (N) – Clinica (N) / Staff (1) – Clinica (1)

    Muitos membros do staff operam num teste clinico em que um coordena o mesmo.

- **Teste Clinico - Recurso**
  - **Atributos –** Nenhum
  - **Relação –** Um teste clínico utiliza recursos
  - **Cardinalidade –** Teste Clinico (1) – Recurso (N)

    Um teste clinico pode utilizar vários recursos diferentes.

- **Teste Clinico - Clinica**
  - **Atributos –** Nenhum
  - **Relação –** Um teste clínico é feito numa clínica.
  - **Cardinalidade –** Teste Clinico (N) – Clinica (1)

    Um teste clinico tem de ser realizado numa determinada clinica, clinica esta que terá sempre vários testes clínicos a decorrer.

3.1.4 Caracterização dos atributos de cada entidade

Entidade	Atributos	Tipo de Dados	Nulo	Composto	Multivalor	Derivado	Candidato
Atleta	nome	VARCHAR(45)	Não	Não	Não	Não	Não

	contacto	VARCHAR(45)	Não	Não	Sim	Não	Não
	sexo	VARCHAR(1)	Não	Não	Não	Não	Não
	DOB	DATETIME	Não	Não	Não	Não	Não
	idade	INT	Não	Não	Não	Sim	Não
	nif	INT	Não	Não	Não	Não	Sim
	altura	DECIMAL(3,2)	Não	Não	Não	Não	Não
	Peso	INT	Não	Não	Não	Não	Não
	cod_postal	VARCHAR(45)	Não	Não	Não	Não	Não
	nome	VARCHAR(45)	Não	Não	Não	Não	Não
Clube	cod_clube	INT	Não	Não	Não	Não	Sim
	designação	VARCHAR(45)	Não	Não	Não	Não	Não
Competição	id_competição	INT	Não	Não	Não	Não	Sim
	data	DATE	Não	Não	Não	Não	Não
	hora	TIME	Não	Não	Não	Não	Não
	nome	VARCHAR(45)	Não	Não	Não	Não	Não
	designação	VARCHAR(45)	Não	Não	Não	Não	Não
	local	VARCHAR(45)	Não	Não	Não	Não	Não
Modalidade	cod_modalidade	INT	Não	Não	Não	Não	Sim
	designação	VARCHAR(45)	Não	Não	Não	Não	Não
Teste Clinico	id_teste	INT	Não	Não	Não	Não	Sim
	hora	TIME	Não	Não	Não	Não	Não
	data	DATE	Não	Não	Não	Não	Não
	preco	DOUBLE	Não	Não	Não	Não	Não
Recurso	cod_recurso	INT	Não	Não	Não	Não	Sim
	designacao	VARCHAR(45)	Não	Não	Não	Não	Não
staff	id_staff	INT	Não	Não	Não	Não	Sim
	data_ini_servico	DATETIME	Não	Não	Não	Não	Não
	nome	VARCHAR(45)	Não	Não	Não	Não	Não
	especialidade	VARCHAR(45)	Não	Não	Não	Não	Não
	cargo	VARCHAR(45)	Não	Não	Não	Não	Não
Clinica	cod_clinica	INT	Não	Não	Não	Não	Sim
	nome	VARCHAR(45)	Não	Não	Não	Não	Não
	localidade	VARCHAR(45)	Não	Não	Não	Não	Não

Tabela 1 - Caracterização dos Atributos

Grande parte dos dados presentes na base de dados são os atributos, são estes que guardam valores referentes às entidades.

Com exceção do **contacto** e **idade** da entidade atleta, são todos atributos simples.

O **contacto** é um atributo multivalor, contém todas as formas de contacto que o atleta possui.

A **idade** é um atributo derivado, podendo ser obtido a partir da subtração da data de nascimento (**DOB**) à data atual.

Podemos observar na coluna **Candidato** a chave candidata a ser chave primária de cada uma das entidades.

4. Modelação Lógica

4.1. Criação do modelo lógico de dados

Após darmos como terminada a modelação conceptual e esta estar de acordo com o que o cliente deseja passamos então para o desenvolvimento de um modelo lógico recorrendo ao *MySQL Workbench* (de acordo com o modelo anteriormente descrito).

4.1.1 Atleta

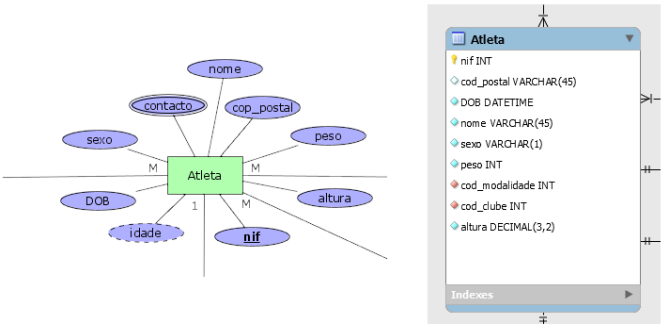

Figura 10 - Passagem da entidade atleta

O nif é um inteiro e vai ser usado como chave primária, ou seja, não pode ser nulo. O código postal e o nome são VARCHAR(45). Para a data de nascimento (DOB) optamos por um DATETIME pois assim conseguimos armazenar a data e a hora de nascimento. O sexo é um VARCHAR(1) que poderá conter M (Mulher) e H (Homem). O peso será armazenado num inteiro. A altura do atleta é um número com duas casas decimais.

4.1.2 Competicao

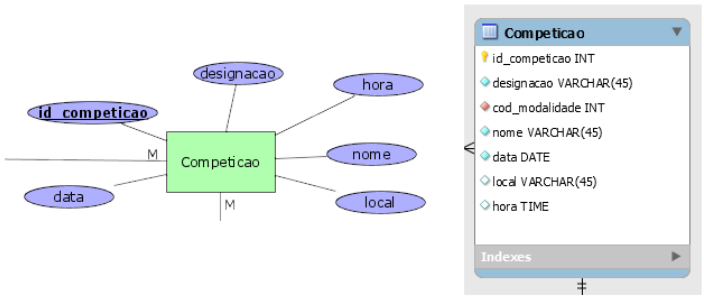

Figura 11 - Passagem da entidade competicao

No caso da competição decidimos que o id deveria ser a chave primária, um inteiro não nulo. A designação da competição é um VARCHAR(45) bem como o nome e o local da mesma. A data, neste caso, é um DATE e a hora um TIME.

4.1.3 Modalidade

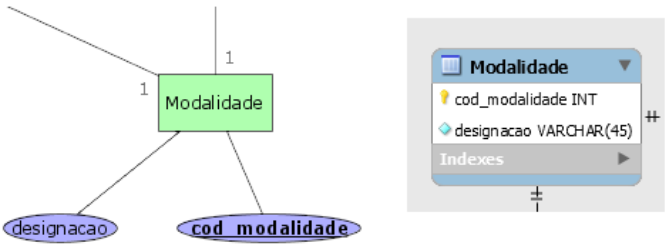

Figura 12 - Passagem da entidade modalidade

Na modalidade temos o código da mesma como chave primária, um inteiro não nulo, e a designação que é guardada num VARCHAR(45).

4.1.4 Clube

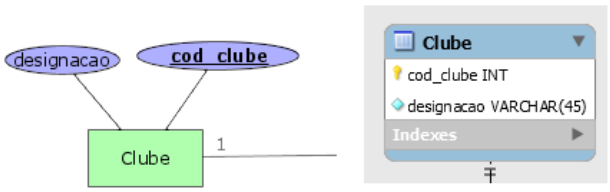

Figura 13 - Passagem da entidade clube

À semelhança da modalidade, no clube temos o código do mesmo como chave primária, um inteiro não nulo, e a designação que é guardada num VARCHAR(45).

4.1.5 Teste Clinico

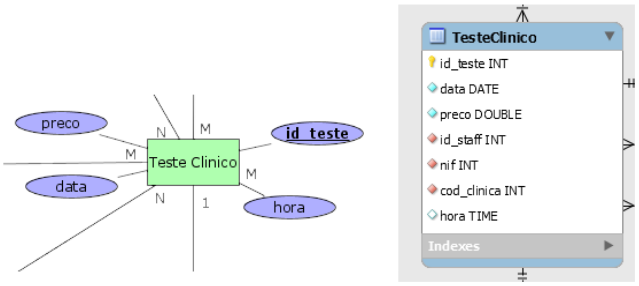

Figura 14 - Passagem da entidade teste clinico

No teste clinico vemos que tem como chave primária um id de teste clinico, inteiro e não nulo. A data e a hora deste é guardada, respetivamente, num DATE e num TIME e o preço é um DOUBLE.

4.1.6 Recurso

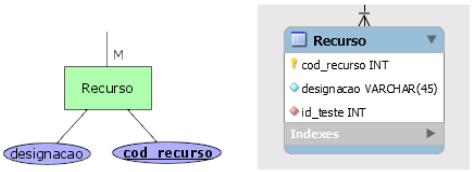

Figura 15 - Passagem da entidade recurso

Na tabela Recurso apenas se guarda o seu código que é a chave primária, um inteiro não nulo, e a descrição do mesmo num VARCHAR(45).

4.1.7 Staff

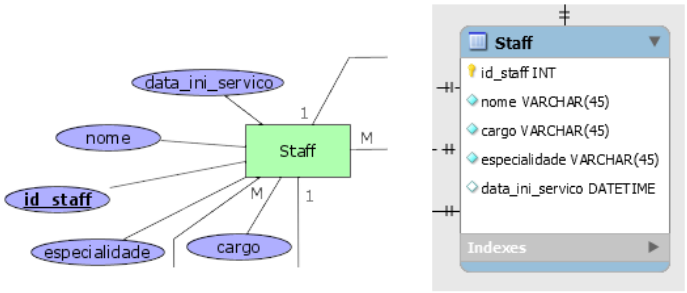

Figura 16 - Passagem da entidade staff

A nossa tabela de Staff tem o id deste como chave primária. Guarda-se também o nome de cada funcionário, o seu cargo e a sua especialidade num VARCHAR(45). Esta contém ainda a data em que começou a exercer num DATETIME.

4.1.8 Clinica

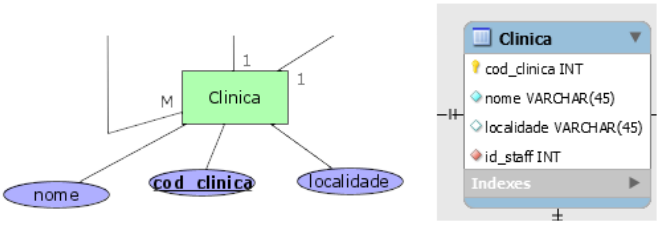

Figura 17 - Passagem da entidade clinica

A clinica tem como chave primária o seu código, um inteiro não nulo. Tem o seu nome e local da sede armazenados num VARCHAR(45).

Após termos definido o tipo de dados que íamos usar olhamos para as ligações presentes no modelo conceptual e criamos as tabelas “auxiliares” em ligações de N para N. Também ligamos as tabelas que tinham ligações neste modelo. Podemos ver tanto as tabelas criadas como as ligações no desenho do modelo seguinte.

4.2. Modelo Lógico

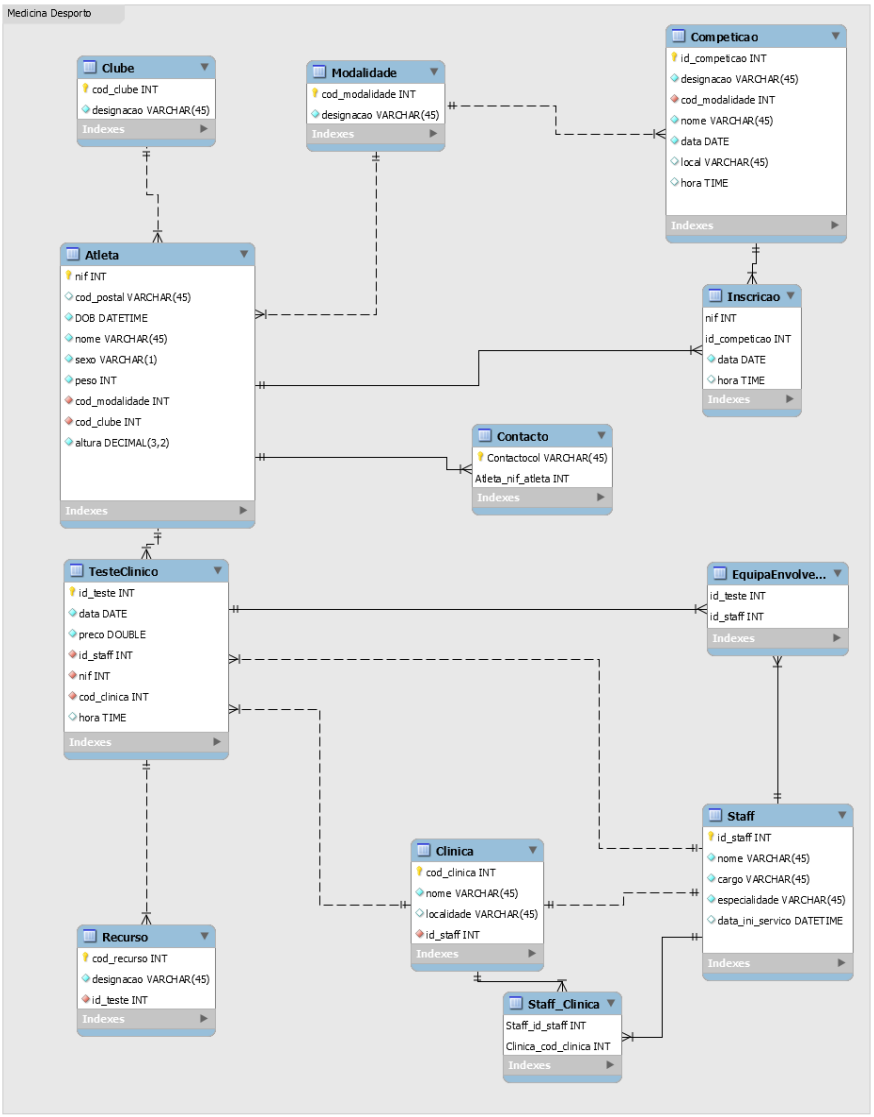

Figura 18 - Desenho do Modelo Lógico

4.3. Validação do Modelo com as Transações Estabelecidas

Foram estabelecidas transações que serão utilizadas pelos utilizadores (com as devidas permissões) da base de dados. Ao serem transações, fica assegurado que as operações dentro de cada transação vão ser efetuadas sequencialmente nos dados, permitindo que as tabelas envolvidas na transação não sejam editadas entre cada operação no bloco por uma outra operação que interaja com esses mesmos dados.

A transação permite-nos reverter para o estado inicial o que nos facilita no caso em que algo corra mal.

4.3.1 Adicionar uma clínica

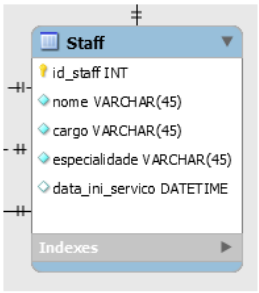

Figura 19 - Tabela Staff

Para adicionar uma clínica tem de, inicialmente, existir um diretor da clínica (Staff) na base de dados. Por isso criamos este primeiro e só depois a clínica.

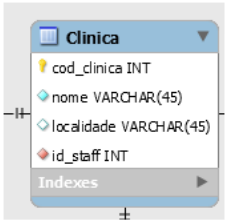

Figura 20 - Tabela Clinica

4.3.2 Adicionar Atleta

Quando se tenciona adicionar um atleta é necessário verificar se a sua modalidade e o seu clube já existem na base de dados. Caso não existam precisam de ser criadas as tabelas.

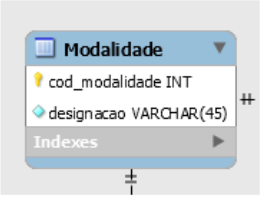

Figura 21 - Tabela modalidade

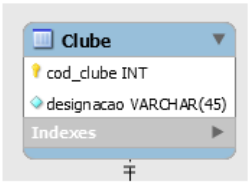

Figura 22 - Tabela Clube

Depois cria-se o atleta adicionando primeiro a modalidade e/ou o clube à base de dados e só depois o atleta.

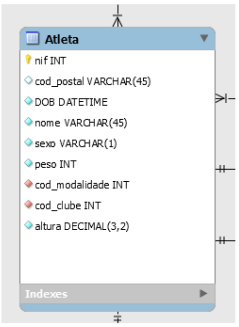

Figura 23 - Tabela Atleta

4.4. Avaliação do modelo lógico

Dando por terminada a conceção do modelo lógico voltamos a contactar o cliente, não querendo avançar sem perceber se estava tudo de acordo com o nosso desenvolvimento.

Com “luz verde” por parte do cliente passamos então para a próxima fase (**Implementação Física**) que será abordada no próximo capítulo.

5. Implementação Física

5.1. Seleção do Sistema de Gestão de Base de Dados

O Sistema de Gestão de Base de Dados Relacional escolhido para implementar a base de dados concebida foi o MySQL. O MySQL é open-source, fácil de usar e contém uma grande comunidade ativa. Para além disto, o facto de o sistema de armazenamento e gestão de transações do MySQL (InnoDB) seguir os princípios ACID (Atomicidade, Consistência, Isolamento e Durabilidade) também constitui um forte motivador para a adoção deste sistema de gestão de base de dados relacional.

5.2. Tradução do Esquema Lógico para o Sistema de Gestão de Bases de Dados escolhido em SQL

Decidimos usar a ferramenta de desenho, desenvolvimento e administração de bases de dados MySQL Workbench de maneira a usufruirmos dos mecanismos que o MySQL oferece da melhor forma.

Dado que o nosso modelo lógico foi criado nesta ferramenta, pudemos usufruir do Forward Engineering que esta oferece. Este mecanismo, tal como o seu nome indica, permite automatizar o processo top-down de construção de componentes de baixo nível (implementação no sistema de gestão de base de dados) através de uma abstração de alto nível (modelo Lógico). Obtendo o código resultante deste processo, apenas tivemos que o executar para que a base de dados fosse criada.

5.3. Tradução das Interrogações do Utilizador para SQL

Na tradução da álgebra relacional, em que são resolvidas as diferentes interrogações, para SQL apenas tivemos de especificar quais as propriedades de correspondência das entradas das relações intervenientes. Para este efeito usamos predominantemente o comando where.

Para a implementação das interrogações parametrizadas foram usados stored procedures. Stored procedures permitem guardar blocos de código SQL na base de dados para que mais tarde estes possam ser chamados, usando o seu nome, em vez de serem escritos na sua totalidade. Para além desta vantagem na interpretabilidade, os stored procedures ao serem guardados são pré-processados, podendo assim ser otimizados para uso futuro. Adicionalmente, estes permitem centralizar a lógica de acesso à base de dados, assim como o acesso controlado a dados restritos a um dado perfil de utilização.

Suplementarmente, para as interrogações não parametrizadas foram usadas views. Views são para os utilizadores apenas outras tabelas, no entanto, ao contrário das tabelas definidas na criação da

base de dados, estas não se encontram guardadas como linhas e colunas de dados. Apenas quando são selecionadas é que são obtidos os tuplos correspondentes a cada uma das entradas.

5.3.1 Alguns exemplos de interrogações implementadas

1. Um atleta deverá ser capaz de consultar o código da clínica em que gastou mais dinheiro e o respetivo montante gasto.

```
delimiter $$
CREATE PROCEDURE AClinicaEmQueMaisGasta(In nif_atleta int)
    begin
        Select
                        T.cod_clinica,
            sum(T.preco) as Gasto
        From
                Atleta as A,
            TesteClinico as T
        where
            A.nif = T.nif
            and A.nif = nif_atleta
                group by T.cod_clinica
        order by sum(T.preco) DESC
        limit 1;
        end $$
delimiter ;
```

Figura 24 - Imagem do código obtido para a query 1

2. Um diretor de um teste clínico deverá ser capaz de consultar o nome do funcionário com quem trabalhou nesse teste.

```
delimiter $$
CREATE PROCEDURE DTComQuemTrabalhouTestes(In n_staff int)
    begin
        Select
                E.id_teste,
            S.nome
        From
                EquipaEnvolvente as E,
            Staff as S
        where
                S.id_staff = E.id_staff
                and E.id_teste in
                (select
                        T.id_teste
                from
                        TesteClinico as T,
            EquipaEnvolvente as E
                where T.id_teste = E.id_teste
        and T.id_staff = n_staff);
        end $$
delimiter ;
```

Figura 25 - Imagem do código obtido para a query 2

3. Um diretor de uma clínica deverá ser capaz de consultar quantos funcionários de cada especialidade tem nessa clínica.

```
delimiter $$
create procedure DCStaffAreaClinica (IN n_staff int)
begin
select S.especialidade ,
count(S.especialidade) as NrEspecialistas
from Staff as S
join Staff_Clinica as SC
on SC.Staff_id_staff = S.id_staff
where SC.Staff_id_staff
in (select SC.Staff_id_staff
from Clinica as C
 join Staff_Clinica as SC
 on C.cod_clinica = SC.Clinica_cod_clinica where C.id_staff = n_staff)
group by S.especialidade order by count(S.especialidade);
end $$
delimiter ;
```

Figura 26 - Imagem do código obtido para a query 3

5.4. Tradução das transações estabelecidas para SQL

As operações que devem ser efetuadas na implementação de cada umas das transações já haviam sido especificadas na validação do modelo com as transações estabelecidas (Secção 4.3). Nesta fase, traduzimos essas operações para SQL.

A par das interrogações, também foram utilizados stored procedures, para permitir que estes procedimentos possam ser mais tarde utilizados. Assim, depois de criado o procedure inicia-se a transaction e, depois, efetuam-se as várias operações pretendidas.

1. Adicionar Clínica

```
DELIMITER $$
CREATE PROCEDURE addClinica(IN cod_clinica VARCHAR(45), IN clinica_nome VARCHAR(45),
                                                        IN localidade VARCHAR(45), IN id_staff INT,
                            IN staff_nome VARCHAR(45), IN cargo VARCHAR(45),
                            IN especialidade VARCHAR(45), IN data_ini_servico DATETIME)
BEGIN
        DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;

        START TRANSACTION;
    SET autocommit = 0;
    -- criar Staff
    INSERT INTO Staff(id_staff,nome,cargo,especialidade,data_ini_servico)
    values (id_staff,staff_nome,cargo,especialidade,data_ini_servico);
    -- criar clinica
    INSERT INTO Clinica(cod_clinica,nome,localidade,id_staff)
    values (cod_clinica,clinica_nome,localidade,id_staff);

    COMMIT;
END $$
DELIMITER ;
```

Figura 27 - Imagem do código obtido para a transaction 1

Quando se tenciona adicionar uma clínica sem se ter inicialmente o diretor da clínica (Staff) na base de dados é necessário primeiro adicionar o diretor da clínica e só depois a clínica.

O resultado será uma nova entrada na tabela Clinica e na tabela Staff, caso ambas as operações executem com êxito, ou nenhuma nova entrada na base de dados, caso contrário.

2. Adicionar Atleta

```
DELIMITER $$
CREATE PROCEDURE addAtleta(IN nif INT, IN cod_postal VARCHAR(45), IN DOB DATETIME,
                           IN nome VARCHAR(45), IN sexo VARCHAR(1), IN peso INT,
                           IN modalidade_designacao VARCHAR(45), IN clube_designacao VARCHAR(45),
                           IN altura DECIMAL(3, 2))
BEGIN

    DECLARE codigo_clube INT;
    DECLARE maximo_clube INT;
    DECLARE quantos_clube INT;
    DECLARE proximo_clube INT;
    DECLARE codigo_modalidade INT;
    DECLARE maximo_modalidade INT;
    DECLARE quantos_modalidade INT;
    DECLARE proximo_modalidade INT;

        DECLARE EXIT HANDLER FOR SQLEXCEPTION
        BEGIN
                ROLLBACK;
        END;

    START TRANSACTION;
    SET autocommit = 0;
    SET codigo_clube = (select cod_clube from Clube where Clube.designacao = clube_designacao);
    SET quantos_clube = (select count(*) from Clube where cod_clube = codigo_clube);
    SET maximo_clube = (select cod_clube from Clube order by cod_clube DESC limit 1);
    SET proximo_clube = maximo_clube + 1;
    SET codigo_modalidade = (select cod_modalidade from Modalidade where Modalidade.designacao = modalidade_designacao);
    SET quantos_modalidade = (select count(*) from Modalidade where cod_modalidade = codigo_modalidade);
    SET maximo_modalidade = (select cod_modalidade from Modalidade order by cod_modalidade DESC limit 1);
    SET proximo_modalidade = maximo_modalidade + 1;

    IF (quantos_modalidade = 0) THEN
        INSERT INTO Modalidade(cod_modalidade, designacao)
        values (proximo_modalidade, modalidade_designacao);
        SET codigo_modalidade = proximo_modalidade;
    END IF;

    IF (quantos_clube = 0) THEN
        INSERT INTO Clube(cod_clube, designacao)
        values (proximo_clube, clube_designacao);
        SET codigo_clube = proximo_clube;
    END IF;

    INSERT INTO Atleta(nif, cod_postal, DOB, nome, sexo, peso, cod_modalidade, cod_clube, altura)
    values (nif, cod_postal, DOB, nome, sexo, peso, codigo_modalidade, codigo_clube, altura);

    COMMIT;
END $$
DELIMITER ;
```

Figura 28 - Imagem do código obtido para a transaction 2

Quando se tenciona adicionar um atleta é necessário verificar se a sua modalidade e o seu clube já existem na base de dados. Caso ambos existam, apenas é necessário adicionar o atleta e, nesse caso, o resultado será uma nova entrada na tabela Atleta, caso a operação corra bem ou nenhuma nova entrada na base de dados, caso contrário. Caso a modalidade e/ou o clube não exista(m), primeiro será necessário adicionar a modalidade e/ou o clube à base de dados e

só depois o Atleta e, nesse caso, o resultado será uma nova entrada na tabela Modalidade e/ou Clube e na tabela Atleta caso todas as operações executem com êxito ou nenhuma nova entrada na base de dados, caso contrário.

5.5. Triggers SQL

Por forma a otimizar o espaço utilizado, decidimos criar triggers que eliminassem entradas de tabelas assim que já não estivessem a ser utilizadas.

1. Começamos por criar um trigger que eliminasse um clube da tabela Clube que já não estivesse a ser utilizado, ou seja, se o único atleta desse clube for eliminado, este trigger elimina o seu clube da base de dados.

```
DELIMITER $$
CREATE TRIGGER deleteClube
        AFTER DELETE
    ON Atleta
    FOR EACH ROW
BEGIN
    DECLARE num INT;
    SET num = (select count(*) from Atleta where cod_clube = old.cod_clube);
        IF (num = 0) THEN DELETE from Clube where cod_clube = old.cod_clube;
    END IF;
END $$
DELIMITER ;
```

Figura 29 - Imagem do código obtido para o trigger 1

2. De seguida, criamos um trigger que elimina uma modalidade da tabela Modalidade que já não esteja a ser utilizada, ou seja, caso o único atleta dessa modalidade for eliminado e não exista nenhuma competição dessa modalidade.

```
DELIMITER $$
CREATE TRIGGER deleteModalidade
        AFTER DELETE
    ON Atleta
    FOR EACH ROW
BEGIN
    DECLARE num INT;
    DECLARE num_competicao INT;
    SET num_competicao = (select count(*) from Competicao where cod_modalidade = old.cod_modalidade);
    SET num = (select count(*) from Atleta where cod_modalidade = old.cod_modalidade);
        IF (num = 0 and num_competicao = 0) THEN
    DELETE from Modalidade where cod_modalidade = old.cod_modalidade;
    END IF;
END $$
DELIMITER ;
```

Figura 30 - Imagem do código obtido para o trigger 2

3. Por último, criamos um trigger que elimina um recurso da tabela Recurso que já não esteja a ser utilizado, ou seja, caso o único teste clínico que utilizasse esse recurso for eliminado.

```
DELIMITER $$
CREATE TRIGGER deleteRecurso
        BEFORE DELETE
    ON TesteClinico
    FOR EACH ROW
BEGIN
    DECLARE num INT;
    SET num = (select count(*) from Recurso where id_teste = old.id_teste);
        IF (num = 1) THEN
    DELETE from Recurso where id_teste = old.id_teste;
    END IF;
END $$
DELIMITER ;
```

Figura 31 - Imagem do código obtido para o trigger 3

5.6. Estimativa do Espaço em Disco da Base de Dados e Taxa de Crescimento Anual

O cálculo das estimativas de espaço baseia-se no capítulo 11.7 do MySQL 5.7 Reference Manual.

Os valores foram calculados somando o número de bytes necessários para cada atributo.

Tabela	Espaço de uma entrada
Clube	50
Modalidade	50
Contacto	50
Atleta	120
Competição	152
Inscrição	14
TesteClinico	30
Recurso	54
Clinica	100
Staff	150
Staff_Clinica	8
EquipaEnvolvente	8

Tabela 2 - Bytes necessários por tabela

Comecemos por assumir que a Health First irá começar esta base de dados com cerca de 10 clubes, 10 modalidades, 30 contactos, 100 atletas, 30 competições, 35 inscrições, 100 testes clínicos, 25 recursos, 3 clínicas, 60 funcionários (Staff), 44 funcionários que trabalham numa clínica (Staff_Clinica) e 28 funcionários que operam num teste clínico (EquipaEnvolvente).

Daqui, podemos concluir que o tamanho inicial da base de dados presente na Health First tem o seguinte tamanho:

Clube = 10 * 50 = 500 Bytes
Modalidade = 10 * 50 = 500 Bytes
Contacto = 30 * 50 = 1500 Bytes
Atleta = 100 * 120 = 12000 Bytes
Competicao = 30 * 152 = 4560 Bytes
Inscricao = 35 * 14 = 490 Bytes
TesteClinico = 100 * 30 = 3000 Bytes
Recurso = 25 * 54 = 1350 Bytes
Clinica = 3 * 100 = 300 Bytes
Staff = 60 * 150 = 9000 Bytes
Staff_Clinica = 44 * 8 = 352 Bytes
EquipaEnvolvente = 28 * 8 = 224 Bytes

500 + 500 + 1500 + 12000 + 4560 + 490 + 3000 + 1350 + 300 + 9000 + 352 + 224 = 33776 Bytes = 33.776 KBytes

Contudo, a Health First espera aumentar bastante o público alvo e aumentar cerca de 50 clubes, 50 modalidades, 150 contactos, 500 atletas, 150 competições, 175 inscrições, 500 testes clínicos, 125 recursos, 15 clínicas, 300 funcionários (Staff), 220 funcionários que trabalham numa clínica (Staff_Clinica) e 140 funcionários que operam num teste clínico (EquipaEnvolvente) por ano.

Daqui, estimamos que o tamanho da base de dados presente na Health First irá aumentar o seguinte valor por ano:

Clube = 50 * 50 = 2500 Bytes
Modalidade = 50 * 50 = 2500 Bytes
Contacto = 150 * 50 = 7500 Bytes
Atleta = 500 * 120 = 60000 Bytes
Competicao = 150 * 152 = 22800 Bytes
Inscricao = 175 * 14 = 2450 Bytes
TesteClinico = 500 * 30 = 15000 Bytes

Recurso = 125 * 54 = 6750 Bytes

Clinica = 15 * 100 = 1500 Bytes

Staff = 300 * 150 = 45000 Bytes

Staff_Clinica = 220 * 8 = 1760 Byte

EquipaEnvolvente = 140 * 8 = 1120 Byte

2500 + 2500 + 7500 + 60000 + 22800 + 2450 + 15000 + 6750 + 1500 + 45000 + 1760 + 1120 = 168880 Bytes = 168.880 KBytes

5.7. Definição e caraterização das vistas de utilização em SQL

Começamos por definir uma vista que apresenta os atletas por ordem alfabética.

```
create view atletasAlfabeticamente as
        select A.*
        from Atleta A
        order by A.nome asc;
```

Figura 32 - Atletas ordenados por ordem alfabética

De seguida, definimos uma vista que apresenta os funcionários por ordem alfabética.

```
create view staffAlfabeticamente as
        select S.*
    from Staff S
    order by S.nome asc;
```

Figura 33 - Funcionários ordenados por ordem alfabética

Depois, definimos uma vista que apresenta as clínicas por ordem alfabética.

```
create view clinicasAlfabeticamente as
        select C.*
    from Clinica C
    order by C.nome asc;
```

Figura 34 - Clínicas ordenadas por ordem alfabética

Por último, definimos uma vista que apresenta os testes clínicos por ordem cronológica (do mais recente para o mais antigo).

```
create view testesData as
    select T.*
from TesteClinico T
order by T.data desc;
```

Figura 35 - Testes clínicos ordenados por ordem cronológica

5.8. Definição e caraterização dos Mecanismos de Segurança em SQL

Em qualquer projeto é necessário garantir que sejam cumpridos todos os requisitos de segurança necessários para assegurar a proteção dos dados. Assim, é essencial que a base de dados seja ela própria o primeiro mecanismo de defesa contra qualquer tipo de ações mal-intencionadas.

Deste modo, recorremos à criação de vários utilizadores que irão interagir de forma regular com a base de dados com permissões segundo a seguinte imagem. Existirá ainda um administrador que terá todos os privilégios.

```
Use MedicinaDesporto;

CREATE ROLE DIRETORCLINICA;
CREATE ROLE DIRETORTESTE;
CREATE ROLE ATLETA;
CREATE ROLE ADMINISTRADOR;

GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DCFaturacaoClinica TO DIRETORCLINICA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DCClubeMaisTestesEntreDatasClinica TO DIRETORCLINICA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DCClubesInexistentesClinica TO DIRETORCLINICA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DCStaffAreaClinica TO DIRETORCLINICA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DCMesesOrdemCrescenteClinica TO DIRETORCLINICA;

GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTRecursosTeste TO DIRETORTESTE;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTClinicaRealizouMaisTestes TO DIRETORTESTE;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTComQuemTrabalhouTestes TO DIRETORTESTE;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTAtletasTestes TO DIRETORTESTE;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTContactoAtletasTestes TO DIRETORTESTE;
```

```
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.AClinicaEmQueMaisGasta TO ATLETA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.ATestesEmClinica TO ATLETA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.ATestesEntreDatas TO ATLETA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.ATestesPrecoCrescente TO ATLETA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.ATodosTestes TO ATLETA;

GRANT ALL PRIVILEGES ON MedicinaDesporto.* TO ADMINISTRADOR WITH GRANT OPTION;

CREATE USER 'diretorclinica2'@'%' IDENTIFIED BY 'diretorclinicapassword';
GRANT DIRETORCLINICA to 'diretorclinica2'@'%';

CREATE USER 'diretorteste14'@'%' IDENTIFIED BY 'diretorteste';
GRANT DIRETORTESTE to 'diretorteste14'@'%';

CREATE USER 'atleta50'@'%' IDENTIFIED BY 'atletapassword';
GRANT ATLETA to 'atleta50'@'%';

CREATE USER 'administrador'@'%' IDENTIFIED BY 'adminpassword';
GRANT ADMINISTRADOR to 'administrador'@'%';
```

Figura 36 - Criação e atribuição de permissões aos utilizadores e administrador

5.9. Revisão do Sistema Implementado com o Utilizador

Após a conclusão da implementação da base de dados foi agendada uma reunião final com a Health First de forma a validar a base de dados e possivelmente dar como concluído o desenvolvimento do sistema.

A revisão da implementação consistiu principalmente em 5 fases.

Na primeira fase mostramos que a base de dados conseguia responder às várias interrogações às quais nos tínhamos comprometido a poder responder.

Na segunda fase demonstramos as várias operações de inserção de dados no sistema e como é que o sistema lidava com inserções incorretas de dados.

Na terceira fase mostramos ao cliente as estimativas de espaço ocupado pela base de dados e a taxa de crescimento esperada.

A quarta fase consistiu em explicar ao cliente os vários mecanismos de segurança implementados ao nível da base de dados.

A fase final consistiu numa discussão com o cliente sobre as várias dúvidas que a apresentação poderia ter deixado acerca da implementação.

Esclarecidas todas as dúvidas que poderiam existir o cliente deu autorização para avançar com a instalação do sistema.

6. NoSQL – Neo4j

6.1. Motivo para adotar uma base de dados NoSQL

A Health First nasceu este ano, mas já tem muitos clientes e muitas empresas já a estão a contactar. Isto traz problemas - cada vez mais testes clínicos, cada vez mais atletas... A informação presente tem escalado a nível exponencial. Isto levou-nos a pensar numa mudança na maneira como trabalhamos, precisávamos de uma melhor performance e queries com respostas mais eficazes.

Adotamos assim o sistema NoSQL, usando o Neo4j para o efeito. Deste modo queremos dar uma resposta mais rápida e segura às empresas com quem trabalhamos e assegurar a viabilidade das mesmas.

6.2. Objetivo desta implementação

Como referido anteriormente o objetivo desta mudança passa pela performance. Queremos agora disponibilizar as mesmas interrogações do modelo anterior, mas de forma extremamente rápida mesmo com mais informação presente. Vamos aproveitar o sistema de travessia de grafos que o Neo4j disponibiliza para o efeito.

6.3. Estrutura Base

Para ir de encontro aos requisitos do sistema de dados NoSQL, e com o intuito de melhorar a performance e viabilidade do sistema, utilizou-se a estrutura abaixo apresentada:

Figura 37 - Legenda Neo4j

Os nodos que podemos ver na legenda foram selecionados para representar as tabelas anteriormente usadas na implementação física do modelo SQL que possuem chaves primárias, assim estes nodos retêm a informação útil respetiva. As relações que vemos surgem para substituir as chaves estrangeiras que representavam as ligações do modelo anterior. Neste novo modelo as relações representam a ligação entre dois nodos, assim permitindo que seja possível a travessia entre os nodos de modo a obter os resultados necessários. Por exemplo, para o nodo do teste clinico (:TesteClinico) foram atribuídas as seguintes variáveis: id_teste (int), data (date), hora (time) e preço (float). Com estes atributos podemos manter o mesmo tipo de informação fornecida pelo modelo SQL. Para a presença deste teste clinico foi criada uma relação (:TesteClinico):-[:MARCADO_PARA]->(:Atleta), sendo que o teste clinico é marcado para um determinado atleta. O atleta pode ter N ligações, já o TesteClinico pode ter apenas uma

destas relações. Já a abordagem para a relação entre o TesteClinico e a EquipaEnvolvente no mesmo é representada por uma ligação sem direção com os nodos Staff, pois vem substituir um relacionamento N para N do modelo de conceptual. Abaixo pode ser observado um grafo exemplo com o tipo de informação descritas acima.

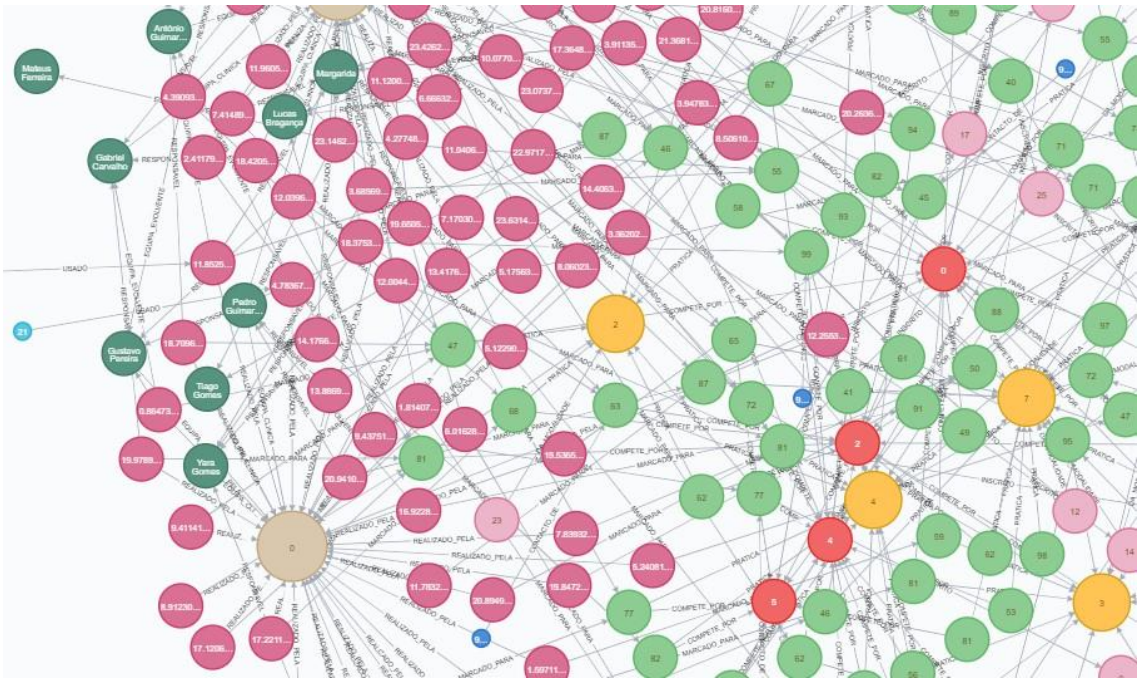

Figura 38 – Grafo Exemplo Neo4j

6.4. Migração de dados

De modo a efetuar a migração de dados do sistema *SQL* para o sistema Neo4j foi utilizada novamente a manipulação de ficheiros *csv*. Começando por usar a ferramenta do *MySQL Workbench* para exportar os respetivos ficheiros *csv* de cada tabela com os cabeçalhos apropriados para facilitar o parse desses ficheiros com as ferramentas disponibilizadas pela linguagem *cypher*. Na leitura dos ficheiros csv foi utilizado o código cypher que pode ser consultado no *Anexo 12.* Este faz o parse de acordo com o ficheiro, criando por sua vez as relações e nodos respetivos. Optamos por esta solução, pois é uma das mais genéricas e tem boa compatibilidade com a exportação de dados proveniente de outros sistemas similares.

6.5. Questões anteriores usando NoSQL

Vamos apresentar três exemplos de implementações de funcionalidades transferidas para o NoSQL, as restantes encontram-se nos respetivos anexos de Neo4j.

Para replicar o procedure 'ATestesEmClinica' implementado no sistema SQL, foi escrita a query em cypher. Substituindo '$n_clinica' e '$n_atleta' como input devolve os testes clínicos efetuados pelo atleta(n_atleta) na clínica(n_clinica), respetivamente.

```
//CREATE PROCEDURE ATestesEmClinica(IN n_clinica int, IN n_atleta INT)
MATCH  (e:Clinica)<-[:REALIZADO_PELA]-(c:TesteClinico)-[:MARCADO_PARA]->(a:Atleta) WHERE a.nif = $n_atleta and e.cod_clinica = $n_clinica RETURN c as TesteClinico
```

Para a implementação deste procedure usa-se o MATCH para efetuar as relações necessárias para aplicar as condições que permitem restringir a que atleta e clinica se referem os testes clínicos, que são posteriormente devolvidos.

Para replicar o procedure 'DTAtletasTestes' implementado no sistema SQL, foi escrita a query em cypher. Substituindo '$n_staff' como input devolve os identificadores dos testes clínicos efetuados para os respetivos atletas, na clínica onde o responsável tem como id_staff = n_staff.

```
//CREATE PROCEDURE DTAtletasTestes(In n_staff int)
MATCH      (s:Staff)<-[:RESPONSAVEL]-(tc:TesteClinico)-[:MARCADO_PARA]->(a:Atleta) WHERE s.id_staff = $n_staff WITH tc.id_teste as id_teste, a as atleta RETURN id_teste, atleta
```

Para a implementação deste procedure usa-se um MATCH para efetuar as relações que permitem restringir a solução ao responsável que tenha o id_staff igual ao input, de modo a fornecer a lista devida como output.

Para replicar o procedure 'DCClubesInexistentesClinica' implementado no sistema SQL, foi escrita a query em cypher que alterando '$n_staff' como input devolve os clubes que não têm atletas com testes clínicos marcados na clínica onde o diretor tem id_staff = n_staff.

```
//CREATE PROCEDURE DCClubesInexistentesClinica(IN n_staff int)
MATCH             (s:Staff)<-[:DIRETOR]-(c:Clinica)<-[:REALIZADO_PELA]-(:TesteClinico)-[:MARCADO_PARA]->(:Atleta)-[:COMPETE_POR]->(e:Clube)
WHERE s.id_staff = $n_staff WITH collect(DISTINCT e.designacao) as ci
MATCH (cu:Clube) WHERE not cu.designacao in ci
RETURN cu.designacao as nome_clube ORDER BY nome_clube
```

Para a implementação deste procedure usa-se o MATCH para realizar as relações que possibilitam aplicar a condição de que o staff realmente é diretor dessa clínica, tirando assim uma lista com todos os clubes que efetuam testes clínicos na clinica. Já no segundo MATCH verifica-se quais os clubes existentes que não pertencem a essa lista, por fim devolvendo os clubes inexistentes nessa clínica.

7. Conclusões e Trabalho Futuro

Com o desenvolvimento desta base de dados acreditamos que a Health First se impôs no mercado. Acreditamos que conseguimos ver os clientes satisfeitos. Estudamos intensivamente os requisitos e fizemos tudo o que estava ao nosso alcance para apresentar ao cliente o que ele desejava.

Com este projeto sentimo-nos muito mais capazes de desenvolver uma base de dados quer usando SQL como NoSQL.

Queremos num futuro próximo implementar mais funcionalidades na nossa base de dados, pois acreditamos que agora que temos uma base estruturada podemos ter confiança e segurança ao fazê-lo.

Em suma, este projeto foi crucial para a aplicar diversos conceitos adquiridos nas aulas da unidade curricular de Bases de Dados. É um projeto bastante complexo, mas acreditamos que abordamos com sucesso a maioria dos conceitos presentes na disciplina.

Referências

Connolly, Thomas and Begg, Carolyn (2005). Database Systems, A Practical Approach to Design, Implementation, and Management. 6th edition.

Mysql [Online]. Website: https://dev.mysql.com/doc/refman/5.7/en/.

Lista de Siglas e Acrónimos

BD Bases de Dados

DW Data Warehouse

OLTP *On-Line Analytical Processing*

UEFA *União das Associações Europeias de Futebol*

ACID *Atomicidade, Consistência, Isolamento e Durabilidade*

NIF *Número de Identificação Fiscal*

Anexos

Nesta parte do “extra-relatório” estão presentes todos os scripts desenvolvidos para a base de dados, tanto os de SQL como os de NoSQL.

I. Anexo 1 – Script de Inicialização da Base de Dados

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET                                          @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZE
RO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema MedicinaDesporto
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema MedicinaDesporto
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `MedicinaDesporto`;
CREATE SCHEMA IF NOT EXISTS `MedicinaDesporto` DEFAULT CHARACTER SET
utf8 ;
USE `MedicinaDesporto` ;

-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Modalidade`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Modalidade` (
  `cod_modalidade` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`cod_modalidade`),
  UNIQUE INDEX `idModalidade_UNIQUE` (`cod_modalidade` ASC) VISIBLE)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Clube`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Clube` (
  `cod_clube` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
```

```
  PRIMARY KEY (`cod_clube`),
  UNIQUE INDEX `cod_clube_UNIQUE` (`cod_clube` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Atleta`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Atleta` (
  `nif` INT NOT NULL,
  `cod_postal` VARCHAR(45) NULL,
  `DOB` DATETIME NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `sexo` VARCHAR(1) NOT NULL,
  `peso` INT NOT NULL,
  `cod_modalidade` INT NOT NULL,
  `cod_clube` INT NOT NULL,
  `altura` DECIMAL(3,2) NOT NULL,
  PRIMARY KEY (`nif`),
  UNIQUE INDEX `idAtleta_UNIQUE` (`nif` ASC) VISIBLE,
  INDEX `fk_Atleta_Modalidade1_idx` (`cod_modalidade` ASC) VISIBLE,
  INDEX `fk_Atleta_Clube1_idx` (`cod_clube` ASC) VISIBLE,
  CONSTRAINT `fk_Atleta_Modalidade1`
    FOREIGN KEY (`cod_modalidade`)
    REFERENCES `MedicinaDesporto`.`Modalidade` (`cod_modalidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Atleta_Clube1`
    FOREIGN KEY (`cod_clube`)
    REFERENCES `MedicinaDesporto`.`Clube` (`cod_clube`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Competicao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Competicao` (
  `id_competicao` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
```

```
  `cod_modalidade` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `data` DATE NOT NULL,
  `local` VARCHAR(45) NULL,
  `hora` TIME NULL,
  PRIMARY KEY (`id_competicao`),
  UNIQUE INDEX `id_competicao_UNIQUE` (`id_competicao` ASC) VISIBLE,
  INDEX `fk_Competicao_Modalidade1_idx` (`cod_modalidade` ASC) VISIBLE,
  CONSTRAINT `fk_Competicao_Modalidade1`
    FOREIGN KEY (`cod_modalidade`)
    REFERENCES `MedicinaDesporto`.`Modalidade` (`cod_modalidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Inscricao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Inscricao` (
  `nif` INT NOT NULL,
  `id_competicao` INT NOT NULL,
  `data` DATE NOT NULL,
  `hora` TIME NULL,
  PRIMARY KEY (`nif`, `id_competicao`),
  INDEX   `fk_Atleta_has_Competicao_Competicao1_idx`   (`id_competicao`
ASC) VISIBLE,
  INDEX `fk_Atleta_has_Competicao_Atleta_idx` (`nif` ASC) VISIBLE,
  CONSTRAINT `fk_Atleta_has_Competicao_Atleta`
    FOREIGN KEY (`nif`)
    REFERENCES `MedicinaDesporto`.`Atleta` (`nif`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Atleta_has_Competicao_Competicao1`
    FOREIGN KEY (`id_competicao`)
    REFERENCES `MedicinaDesporto`.`Competicao` (`id_competicao`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Staff`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Staff` (
  `id_staff` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `cargo` VARCHAR(45) NOT NULL,
  `especialidade` VARCHAR(45) NOT NULL,
  `data_ini_servico` DATETIME NULL,
  PRIMARY KEY (`id_staff`),
  UNIQUE INDEX `id_staff_UNIQUE` (`id_staff` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Clinica`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Clinica` (
  `cod_clinica` INT NOT NULL,
  `nome` VARCHAR(100) NOT NULL,
  `localidade` VARCHAR(45) NULL,
  `id_staff` INT NOT NULL,
  PRIMARY KEY (`cod_clinica`),
  UNIQUE INDEX `cod_clinica_UNIQUE` (`cod_clinica` ASC) VISIBLE,
  INDEX `fk_Clinica_Staff1_idx` (`id_staff` ASC) VISIBLE,
  CONSTRAINT `fk_Clinica_Staff1`
    FOREIGN KEY (`id_staff`)
    REFERENCES `MedicinaDesporto`.`Staff` (`id_staff`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`TesteClinico`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`TesteClinico` (
  `id_teste` INT NOT NULL,
  `data` DATE NOT NULL,
  `preco` DOUBLE NOT NULL,
  `id_staff` INT NOT NULL,
```

```
  `nif` INT NOT NULL,
  `cod_clinica` INT NOT NULL,
  `hora` TIME NULL,
  PRIMARY KEY (`id_teste`),
  UNIQUE INDEX `id_teste_UNIQUE` (`id_teste` ASC) VISIBLE,
  INDEX `fk_TesteClinico_Staff1_idx` (`id_staff` ASC) VISIBLE,
  INDEX `fk_TesteClinico_Atleta1_idx` (`nif` ASC) VISIBLE,
  INDEX `fk_TesteClinico_Clinica1_idx` (`cod_clinica` ASC) VISIBLE,
  CONSTRAINT `fk_TesteClinico_Staff1`
    FOREIGN KEY (`id_staff`)
    REFERENCES `MedicinaDesporto`.`Staff` (`id_staff`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TesteClinico_Atleta1`
    FOREIGN KEY (`nif`)
    REFERENCES `MedicinaDesporto`.`Atleta` (`nif`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TesteClinico_Clinica1`
    FOREIGN KEY (`cod_clinica`)
    REFERENCES `MedicinaDesporto`.`Clinica` (`cod_clinica`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Recurso`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Recurso` (
  `cod_recurso` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
  `id_teste` INT NOT NULL,
  PRIMARY KEY (`cod_recurso`),
  UNIQUE INDEX `cod_recurso_UNIQUE` (`cod_recurso` ASC) VISIBLE,
  INDEX `fk_Recurso_TesteClinico1_idx` (`id_teste` ASC) VISIBLE,
  CONSTRAINT `fk_Recurso_TesteClinico1`
    FOREIGN KEY (`id_teste`)
    REFERENCES `MedicinaDesporto`.`TesteClinico` (`id_teste`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`EquipaEvolvente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`EquipaEnvolvente` (
  `id_teste` INT NOT NULL,
  `id_staff` INT NOT NULL,
  PRIMARY KEY (`id_teste`, `id_staff`),
  INDEX    `fk_TesteClinico_has_Staff_Staff1_idx`    (`id_staff`    ASC)
VISIBLE,
  INDEX `fk_TesteClinico_has_Staff_TesteClinico1_idx` (`id_teste` ASC)
VISIBLE,
  CONSTRAINT `fk_TesteClinico_has_Staff_TesteClinico1`
    FOREIGN KEY (`id_teste`)
    REFERENCES `MedicinaDesporto`.`TesteClinico` (`id_teste`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TesteClinico_has_Staff_Staff1`
    FOREIGN KEY (`id_staff`)
    REFERENCES `MedicinaDesporto`.`Staff` (`id_staff`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Contacto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Contacto` (
  `Contactocol` VARCHAR(45) NOT NULL,
  `Atleta_nif_atleta` INT NOT NULL,
  PRIMARY KEY (`Contactocol`, `Atleta_nif_atleta`),
  INDEX `fk_Contacto_Atleta1_idx` (`Atleta_nif_atleta` ASC) VISIBLE,
  UNIQUE INDEX `Contactocol_UNIQUE` (`Contactocol` ASC) VISIBLE,
  CONSTRAINT `fk_Contacto_Atleta1`
    FOREIGN KEY (`Atleta_nif_atleta`)
    REFERENCES `MedicinaDesporto`.`Atleta` (`nif`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `MedicinaDesporto`.`Staff_Clinica`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `MedicinaDesporto`.`Staff_Clinica` (
  `Staff_id_staff` INT NOT NULL,
  `Clinica_cod_clinica` INT NOT NULL,
  PRIMARY KEY (`Staff_id_staff`, `Clinica_cod_clinica`),
  INDEX `fk_Staff_Clinica_Staff1_idx` (`Staff_id_staff` ASC) VISIBLE,
  INDEX  `fk_Staff_Clinica_Clinica1_idx`  (`Clinica_cod_clinica`  ASC)
VISIBLE,
  CONSTRAINT `fk_Staff_Clinica_Staff1`
    FOREIGN KEY (`Staff_id_staff`)
    REFERENCES `MedicinaDesporto`.`Staff` (`id_staff`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Staff_Clinica_Clinica1`
    FOREIGN KEY (`Clinica_cod_clinica`)
    REFERENCES `MedicinaDesporto`.`Clinica` (`cod_clinica`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


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

II. Anexo 2 – Script para carregar CSVs para povoamento inicial

```
LOAD DATA INFILE 'csv/modalidade.csv'
    INTO TABLE Modalidade
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/clube.csv'
    INTO TABLE Clube
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/atleta.csv'
    INTO TABLE Atleta
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/competicao.csv'
    INTO TABLE Competicao
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/staff.csv'
    INTO TABLE Staff
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/clinica.csv'
    INTO TABLE Clinica
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';
```

```
LOAD DATA INFILE 'csv/staff_clinica.csv'
    INTO TABLE Staff_Clinica
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/contacto.csv'
    INTO TABLE Contacto
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/inscricao.csv'
    INTO TABLE Inscricao
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/testeclinico.csv'
    INTO TABLE TesteClinico
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/equipa_envolvente.csv'
    INTO TABLE EquipaEnvolvente
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

LOAD DATA INFILE 'csv/recurso.csv'
    INTO TABLE Recurso
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';
```

III. Anexo 3 – Ficheiros CSV para povoamento inicial

a. Clínicas

0,Mediterapia-Consultórios Médicos Lda,Guimarães,14

1,Clínica Dermatológica Doutor A Mayer da Silva Lda,Castelo Branco,47

2,Mediterapia-Consultórios Médicos Lda,Marinha Grande,57

b. Clube

0,"Belenenses"

1,"FCPorto"

2,"SCBraga"

3,"Rio Ave"

4,"SCP"

5,"GDChaves"

6,"Marítimo"

7,"SLB"

8,"Boavista"

9,"Moreirense"

c. Competição

0,Supercup,0,Olympic Tournament,2017-12-3,Viana do Castelo,9:53:30

1,Exhibition,0,Worlds Championship,2018-9-25,Braga,13:15:35

2,Grand Slam,3,Grand Prix,2018-11-3,Setúbal,11:39:4

3,Hopman Cup,3,SEHA League,2015-11-15,Vila Nova de Famalicão,10:33:38

4,European Championship,5,Tournament of Champions,2015-8-20,Odivelas,9:55:27

5,Hopman Cup,1,Supercup,2016-4-17,Setúbal,11:45:6

6,Grand Slam,0,Grand Prix,2017-9-14,Amadora,16:42:33

7,SEHA League,1,Masters,2017-3-14,Leiria,17:55:34

8,Grand Slam,1,Hopman Cup,2018-5-17,Setúbal,14:45:15

9,European Championship,4,Supercup,2018-7-2,Portimão,16:58:16

10,Tournament of Champions,1,Grand Slam,2016-3-12,Queluz,17:0:52

11,Exhibition,1,Grand Prix,2015-4-11,Funchal,12:46:31

12,Tournoi de Paris,7,Worlds Championship,2015-2-22,Vila do Conde,16:44:31

13,Super Globe,0,Fed Cup,2016-12-24,Coimbra,11:48:23

14,Grand Prix,7,Supercup,2018-1-6,Aveiro,9:23:36

15,Tournament of Champions,7,Masters,2016-9-12,Portimão,11:40:26
16,Adriatic League,0,SEHA League,2018-4-6,Guarda,10:20:51
17,Exhibition,6,Olympic Tournament,2015-12-4,Viana do Castelo,11:7:1
18,Grand Prix,5,Fed Cup,2016-6-26,Porto,16:14:39
19,Super Globe,5,Fed Cup,2018-8-26,Figueira da Foz,11:45:41
20,Grand Slam,5,Tournament of Champions,2016-11-4,Guarda,12:29:26
21,Tournament of Champions,8,Hopman Cup,2018-4-20,Setúbal,9:43:9
22,Grand Prix,8,Worlds Championship,2016-11-19,Figueira da Foz,12:39:1
23,Olympic Tournament,2,Worlds Championship,2018-7-9,Faro,11:13:33
24,Masters,5,Masters,2015-2-3,Amadora,16:17:36
25,Super Globe,5,Adriatic League,2015-2-6,Agualva-Cacém,15:48:44
26,Exhibition,0,European Championship,2016-8-1,Queluz,13:50:39
27,Tournoi de Paris,9,European Championship,2017-4-4,Almada,11:35:18
28,Tournoi de Paris,1,Masters,2018-9-27,Évora,8:50:48
29,Tournoi de Paris,5,Supercup,2016-6-21,Vila Nova de Gaia,9:7:9

d. Contacto

96023152,10000020
96476182,10000011
96257165,10000012
96914465,10000065
96602170,10000045
96572395,10000097
96075711,10000014
96558128,10000088
96789828,10000020
96333474,10000058
96780993,10000083
96484455,10000007
96022467,10000072
96566954,10000088
96240208,10000025
96897518,10000055
96300861,10000097
96335295,10000014
96771937,10000027
96669364,10000024
96056035,10000078
96815464,10000049

96037639,10000096
96252361,10000043
96869366,10000087
96909057,10000099
96386311,10000073
96807252,10000078
96202516,10000052
96748295,10000039

e. Equipa Envolvente

95,4
56,30
32,5
30,48
91,43
48,52
10,6
51,42
3,22
67,54
80,43
46,34
69,17
52,52
58,18
67,56
42,29
25,14
65,43
27,3
81,44
15,6
84,0
34,48
13,6
29,45
16,31
33,2

f. Modalidade

0,Futsal
1,Ginastica Artistica
2,Ciclismo
3,Badminton
4,Esgrima
5,Box
6,Xadrez
7,Basquete
8,Futebol
9,Ginastica Ritmica

g. Inscrição

10000096,18,2017-7-1,15:8:17
10000096,7,2017-1-13,14:17:14
10000087,17,2018-5-22,16:21:22
10000055,24,2018-10-2,10:11:33
10000076,16,2016-3-25,9:41:10
10000020,27,2017-2-20,8:17:34
10000058,11,2016-1-25,12:33:9
10000057,12,2018-3-27,17:20:2
10000000,9,2016-3-8,17:45:51
10000073,3,2016-1-21,9:48:15
10000012,6,2017-2-2,10:34:58
10000075,17,2016-12-1,15:11:20
10000012,27,2015-7-1,14:52:26
10000032,11,2017-9-24,10:36:30
10000076,0,2017-2-24,10:56:46
10000003,25,2017-1-23,13:16:57
10000097,18,2016-4-23,11:21:9
10000023,7,2017-3-10,8:59:47
10000010,19,2015-9-3,13:6:33
10000039,17,2015-12-15,13:23:3
10000061,14,2017-9-23,17:58:28
10000076,1,2016-9-19,10:23:37
10000006,15,2015-1-10,17:42:59
10000006,24,2016-3-3,10:40:56
10000074,27,2018-2-26,11:45:13

10000050,19,2015-1-26,9:14:30
10000000,8,2017-8-1,11:50:35
10000072,22,2018-5-16,16:7:8
10000039,0,2015-4-13,10:24:22
10000089,25,2015-2-4,16:41:28
10000012,4,2017-9-17,9:0:50
10000089,3,2017-2-27,12:20:29
10000040,7,2017-5-23,17:1:48
10000013,3,2015-9-9,10:45:40
10000003,6,2016-1-10,8:14:16

h. Recurso

0,Inalador de pó seco,76
1,Monitor Holter,24
2,Otoscópio,82
3,Abaixador de língua,59
4,Esparadrapo,58
5,Escova dental ultrassônica,52
6,Abaixador de língua,57
7,Seringa,73
8,Seringa,94
9,Escova dental ultrassônica,74
10,Oftalmoscópio,26
11,Aparelho de anestesia,17
12,Oxigenação por membrana extracorpórea,46
13,Aparelho de anestesia,19
14,Purificador de ar,51
15,Purificador de ar,98
16,Oftalmoscópio,45
17,Oftalmoscópio,8
18,Esparadrapo,19
19,Seringa,31
20,Monitor Holter,25
21,Otoscópio,15
22,Oxigenação por membrana extracorpórea,13
23,Oxigenação por membrana extracorpórea,56
24,Lanceta de Franke,66

i. Staff

0,Pedro Guimarães,Médico,Neurologia,2017-7-6 11:37:58
1,Laura Sampaio,Cirurgião,Anatomia Patológica,2018-3-11 14:53:22
2,Rodrigo de Caminha,Médico,Neurologia,2016-1-21 12:19:34
3,Vicente Marques,Enfermeiro,Cirurgia Geral,2016-4-8 17:47:44
4,Bianca Nunes,Enfermeiro,Cirurgia Pediátrica,2016-6-5 17:46:6
5,Vasco Fernandes,Enfermeiro estagiario,Urologia,2015-9-6 14:23:42
6,António Guimarães,Auxiliar,Pneumologia,2017-5-15 14:42:14
7,Camila Teixeira,Enfermeiro,Patologia Clínica,2017-1-8 16:12:42
8,Catarina Álvares,Auxiliar estagiario,Cirurgia Pediátrica,2016-6-21 8:40:51
9,Maria Rodrigues,Enfermeiro estagiario,Medicina Desportiva,2018-9-14 12:35:49
10,Mateus Correia,Enfermeiro estagiario,Oncologia Médica,2017-9-15 16:55:17
11,Henrique Costa,Médico,Farmacologia Clínica,2018-5-12 8:16:15
12,Benedita Álvares,Médico estagiario,Otorrinolaringologia,2018-8-22 15:59:0
13,Bruna Gomes,Médico estagiario,Pediatria,2017-3-15 14:47:3
14,Bernardo Braga,Enfermeiro,Urologia,2016-12-22 16:16:37
15,Matias Nunes,Médico,Genética Médica,2016-5-17 15:1:55
16,Martim Álvares,Auxiliar estagiario,Estomatologia,2016-2-19 16:4:4
17,Joana Fernandes,Auxiliar,Angiologia e Cirurgia Vascular,2016-5-8 9:21:1
18,Tiago Gomes,Enfermeiro,Gastrenterologia,2018-5-10 11:36:34
19,Yara Gomes,Médico,Cirurgia Maxilo-facial,2017-8-17 11:38:26
20,Ema da Gama,Auxiliar estagiario,Cirurgia Pediátrica,2015-5-16 8:42:50
21,Eva Albuquerque,Médico estagiario,Cardiologia,2017-6-20 15:32:50
22,Mateus Ferreira,Enfermeiro,Cirurgia Plástica e Reconstrutiva e Estética,2017-10-10 10:21:0
23,Yara Ribeiro,Médico,Cirurgia Maxilo-facial,2017-1-12 10:35:3
24,Íris Braga,Auxiliar,Radioterapia,2016-12-16 8:39:50
25,Vicente Alves,Enfermeiro,Radiodiagnóstico,2018-2-24 13:38:13
26,Madalena Carvalho,Médico estagiario,Cardiologia,2018-3-3 8:19:26
27,Joana Marques,Médico estagiario,Anestesiologia,2018-5-14 14:9:10
28,Leonardo Fernandes,Cirurgião,Psiquiatria,2016-11-21 10:6:39
29,Matias Bragança,Auxiliar estagiario,Medicina Desportiva,2018-9-27 10:2:38
30,Margarida Costa,Enfermeiro,Medicina do trabalho,2018-4-9 8:34:40
31,Gabriel Carvalho,Enfermeiro estagiario,Medicina Legal,2017-7-2 10:20:58
32,Mafalda Gonçalves,Cirurgião,Gastrenterologia,2016-12-12 15:13:17
33,Samuel Mendes,Enfermeiro estagiario,Otorrinolaringologia,2017-1-26 12:34:57
34,Rita Jesus,Médico estagiario,Pneumologia,2018-12-11 13:0:14
35,Constança Nunes,Cirurgião,Estomatologia,2018-10-16 14:5:14
36,Iara Guimarães,Médico,Otorrinolaringologia,2015-8-20 13:44:6
37,Alice Rodrigues,Enfermeiro estagiario,Reumatologia,2015-12-27 12:50:7
38,Maria Almeida,Cirurgião,Estomatologia,2018-2-25 8:55:40

39,Nicole Gomes,Médico estagiario,Oncologia Médica,2017-10-14 14:56:34
40,Simão Carvalho,Enfermeiro,Farmacologia Clínica,2017-12-3 14:50:21
41,Henrique Ferreira,Cirurgião,Dermato-venereologia,2018-10-10 12:20:21
42,Sara Albuquerque,Enfermeiro estagiario,Estomatologia,2018-6-1 8:41:5
43,Eva Sampaio,Cirurgião,Endocrinologia-Nutrição,2017-7-14 9:43:25
44,Gustavo Pereira,Enfermeiro,Radiodiagnóstico,2017-10-1 14:38:33
45,Alexandre Moreira,Cirurgião,Angiologia e Cirurgia Vascular,2015-5-17 15:23:39
46,Inês Martins,Auxiliar estagiario,Genética Médica,2017-4-27 14:38:30
47,Filipe Pereira,Auxiliar estagiario,Pediatria,2016-11-1 8:43:7
48,Dinis da Veiga,Médico estagiario,Pneumologia,2017-3-11 13:18:8
49,Lucas Bragança,Auxiliar,Dermato-venereologia,2016-3-17 16:35:35
50,Rúben Alves,Enfermeiro estagiario,Oftalmologia,2017-5-18 15:34:12
51,Bernardo Albuquerque,Médico,Imuno-alergologia,2018-10-23 13:56:6
52,Filipe Rodrigues,Médico,Pneumologia,2018-6-21 13:49:40
53,Inês Ferreira,Médico estagiario,Medicina Geral e Familiar,2018-5-18 12:7:22
54,José Carvalho,Cirurgião,Nefrologia,2016-7-27 10:50:14
55,Bruna Braga,Médico,Otorrinolaringologia,2018-4-19 10:23:58
56,Lucas Almeida,Enfermeiro,Anatomia Patológica,2017-11-3 14:35:36
57,Yara Santos,Auxiliar estagiario,Dermato-venereologia,2016-6-1 16:6:47
58,Pedro Moreira,Médico estagiario,Medicina Geral e Familiar,2017-3-19 9:45:54
59,Sofia Mendes,Auxiliar,Medicina Geral e Familiar,2018-11-12 10:30:33

j. Staff Clínica

18,0
51,2
23,0
41,0
45,1
32,0
40,0
1,2
31,1
0,2
0,0
23,2
14,2
37,2
58,1
36,1

46,1
33,2
51,1
3,0
8,0
57,1
42,0
10,1
53,1
27,0
30,1
49,1
59,1
13,2
9,1
39,1
7,1
2,2
15,1
21,0
44,0
19,0
4,1
6,1
50,0
35,0
56,1
28,0

k. Atleta

10000000,"1663-165","1994-12-12 16:22:20","Alexandre Gomes","H",65,1,6,1.61
10000001,"1813-312","1984-7-12 8:30:27","Enzo Álvares","H",97,7,0,1.53
10000002,"1645-741","1980-7-19 9:15:36","Carlota Castro","M",53,3,3,1.69
10000003,"1669-750","1986-4-1 12:57:54","Tomás Vaz","H",81,4,4,1.38
10000004,"1188-816","1988-11-6 8:16:22","Vasco Marques","H",98,7,2,1.86
10000005,"1175-836","1993-6-12 12:56:47","Mateus Sousa","H",82,6,4,1.86
10000006,"1472-570","1983-10-11 15:17:19","Ariana Álvares","M",71,7,3,1.7
10000007,"1478-97","1999-9-21 15:46:59","Bianca Mendes","M",55,9,0,1.54
10000008,"1502-125","1986-9-2 9:32:51","Rodrigo Fernandes","H",71,4,4,1.63

10000009,"1664-114","1990-7-7 9:14:19","Lourenço Teixeira","H",58,2,0,1.3
10000010,"1703-772","1992-7-7 11:32:34","Bianca Nunes","M",69,8,1,1.37
10000011,"1475-751","1983-2-10 8:39:14","Rodrigo Albuquerque","H",89,6,1,1.48
10000012,"1576-879","1994-8-22 9:48:14","Beatriz Mendes","M",88,0,3,1.55
10000013,"1124-439","1982-11-17 10:50:14","Ema Vaz","M",50,5,3,1.41
10000014,"1043-825","1981-8-1 17:31:55","Nicole Ferreira","M",95,8,5,1.51
10000015,"1635-591","1988-9-19 10:34:2","Núria da Gama","M",89,6,0,1.55
10000016,"1569-820","1990-8-2 10:50:2","Diana Álvares","M",60,9,0,1.7
10000017,"1393-403","1988-3-20 16:51:1","Alexandre Bragança","H",62,2,2,1.47
10000018,"1543-477","1986-11-17 15:36:52","Diana Coimbra","M",91,9,5,1.7
10000019,"1283-491","1991-5-25 9:22:24","José Fernandes","H",80,8,4,1.66
10000020,"1841-939","1989-6-13 10:36:15","Afonso Braga","H",88,5,1,1.3
10000021,"1217-614","1987-4-22 14:13:23","André Albuquerque","H",64,6,1,1.65
10000022,"1192-674","1997-5-14 12:28:55","Luís Mendes","H",71,1,4,1.64
10000023,"1115-673","1983-9-1 15:58:34","Alice Jesus","M",55,7,0,1.77
10000024,"1951-733","1981-9-20 17:23:35","Gabriela Gomes","M",45,5,2,1.87
10000025,"1871-254","1989-8-1 9:54:16","Sofia Oliveira","M",68,3,4,1.42
10000026,"1193-184","1983-11-1 16:15:24","Matilde Ribeiro","M",51,6,6,1.81
10000027,"1220-232","1992-10-11 16:16:37","Lucas Moreira","H",42,7,7,1.71
10000028,"1885-81","1988-4-14 12:5:48","Diogo Fernandes","H",41,7,4,1.77
10000029,"1144-740","1992-10-25 17:44:28","Íris Sampaio","M",46,7,2,1.83
10000030,"1703-820","1994-7-25 17:45:9","Benedita Pinto","M",97,3,5,1.36
10000031,"1530-614","1997-12-1 9:36:24","Catarina Fernandes","M",56,7,6,1.35
10000032,"1413-17","1996-12-6 17:10:32","Afonso Ribeiro","H",62,8,6,1.82
10000033,"1230-427","1992-1-17 8:2:51","Rafael Costa","H",93,6,5,1.36
10000034,"1021-773","1982-6-20 17:31:39","Nicole Almeida","M",94,0,3,1.57
10000035,"1804-12","1994-11-23 11:47:18","Bruna Jesus","M",77,4,5,1.72
10000036,"1778-573","1980-4-19 15:44:38","Enzo Alves","H",96,6,7,1.5
10000037,"1063-913","1983-8-19 9:51:29","Manuel Costa","H",62,6,1,1.72
10000038,"1771-604","1994-6-2 17:8:34","Margarida Carvalho","M",50,9,2,1.45
10000039,"1505-700","1995-10-20 8:45:6","Constança Sampaio","M",77,2,4,1.4
10000040,"1751-129","1981-3-9 13:8:38","Vicente Rodrigues","H",95,5,7,1.38
10000041,"1067-839","1993-10-17 15:26:10","Inês Alves","M",81,2,2,1.44
10000042,"1289-614","1980-4-6 9:17:0","Mara Carvalho","M",68,2,2,1.71
10000043,"1249-606","1987-10-9 13:0:28","Vasco Vaz","H",99,4,4,1.42
10000044,"1059-318","1989-11-24 9:39:20","Filipe Pinto","H",50,1,6,1.73
10000045,"1009-70","1994-6-7 15:45:5","Daniela Marques","M",81,9,6,1.65
10000046,"1120-43","1984-6-13 16:27:0","Gonçalo Gonçalves","H",69,3,3,1.83
10000047,"1750-267","1999-6-4 10:46:14","Duarte Marques","H",67,6,4,1.43
10000048,"1285-668","1980-7-22 9:30:22","Constança Carvalho","M",87,6,2,1.38

10000049,"1556-463","1980-10-8 14:14:20","Inês Vaz","M",55,8,2,1.84
10000050,"1092-951","1982-8-17 8:48:15","Camila Alves","M",97,0,2,1.35
10000051,"1671-260","1998-11-6 8:4:26","Bianca Ribeiro","M",81,0,3,1.43
10000052,"1774-330","1995-5-23 10:16:33","Ema Jesus","M",82,6,1,1.41
10000053,"1509-615","1983-10-13 17:42:17","Gabriela Sampaio","M",88,2,1,1.87
10000054,"1391-650","1996-10-1 14:54:56","Tomás Sampaio","H",81,5,0,1.78
10000055,"1775-547","1999-12-24 9:6:56","Yara Fernandes","M",72,4,0,1.81
10000056,"1787-924","1992-7-18 12:27:59","Sofia Alves","M",62,7,2,1.87
10000057,"1061-636","1987-4-3 10:43:21","Francisco Sampaio","H",61,0,5,1.81
10000058,"1486-715","1983-7-1 10:28:34","Diana Teixeira","M",92,3,1,1.3
10000059,"1139-387","1998-6-25 17:17:7","Catarina Fernandes","M",60,3,6,1.76
10000060,"1143-944","1990-4-20 10:31:9","António Guimarães","H",67,5,6,1.76
10000061,"1108-36","1994-1-17 12:58:43","Margarida Almeida","M",49,4,0,1.61
10000062,"1982-405","1995-9-19 12:59:31","Rafael Silva","H",46,6,4,1.89
10000063,"1163-973","1991-4-7 9:54:39","Matilde Guimarães","M",65,7,5,1.4
10000064,"1421-567","1998-3-9 8:35:33","Núria Teixeira","M",59,3,0,1.71
10000065,"1656-609","1995-9-3 10:13:44","Sara Castro","M",63,4,2,1.71
10000066,"1827-648","1982-10-21 13:27:2","Bruna Sousa","M",56,4,4,1.69
10000067,"1244-721","1982-5-9 16:12:48","Francisco Bragança","H",86,4,4,1.41
10000068,"1309-672","1995-4-22 12:43:31","Rodrigo de Caminha","H",81,3,0,1.71
10000069,"1435-868","1997-1-9 17:10:39","Carlota Almeida","M",90,9,3,1.64
10000070,"1419-653","1986-4-6 15:12:21","Alice Gonçalves","M",67,5,7,1.39
10000071,"1029-949","1999-4-21 16:30:16","Lara de Caminha","M",68,0,7,1.7
10000072,"1168-56","1992-9-20 12:16:27","Francisca Pinto","M",58,7,3,1.85
10000073,"1814-479","1994-7-25 15:35:1","Henrique Correia","H",95,3,0,1.69
10000074,"1772-168","1982-6-1 10:14:14","Vasco Alves","H",74,5,3,1.32
10000075,"1759-230","1992-1-14 11:4:12","Ariana Mendes","M",73,0,3,1.41
10000076,"1257-654","1985-6-3 15:44:51","Beatriz Mendes","M",64,0,2,1.53
10000077,"1934-707","1994-7-27 9:29:0","Alexandre Oliveira","H",47,0,3,1.31
10000078,"1856-799","1990-1-9 10:58:16","Letícia Moreira","M",76,1,1,1.86
10000079,"1745-710","1990-12-1 11:50:59","Francisco Costa","H",46,1,7,1.58
10000080,"1986-692","1998-6-16 12:28:32","Beatriz da Veiga","M",46,4,4,1.3
10000081,"1502-152","1986-3-26 9:23:47","Lucas Santos","H",47,2,0,1.31
10000082,"1963-168","1993-7-20 17:27:9","Marta Nunes","M",72,3,4,1.63
10000083,"1436-363","1980-2-15 10:52:9","Yara Sousa","M",82,4,5,1.45
10000084,"1419-280","1989-11-12 11:41:21","Santiago Almeida","H",63,6,6,1.42
10000085,"1348-659","1982-2-18 8:52:51","Santiago Castro","H",87,7,0,1.38
10000086,"1507-59","1993-8-13 15:51:52","Gustavo Sampaio","H",53,7,4,1.55
10000087,"1889-247","1991-8-11 15:1:17","António da Gama","H",85,5,0,1.58
10000088,"1243-983","1983-4-7 13:0:0","Leonardo Martins","H",47,3,4,1.73

10000089,"1287-446","1996-11-21 16:0:13","Guilherme da Veiga","H",40,1,2,1.79
10000090,"1712-41","1995-8-25 15:36:23","Manuel da Veiga","H",94,4,7,1.62
10000091,"1805-239","1985-1-19 16:52:8","Rafael Carvalho","H",82,2,1,1.4
10000092,"1152-223","1983-12-2 12:3:7","André Castro","H",81,7,5,1.49
10000093,"1535-710","1994-9-8 10:12:34","Bernardo Vaz","H",94,6,1,1.45
10000094,"1806-762","1989-1-16 14:12:9","Matias Teixeira","H",64,1,0,1.31
10000095,"1728-312","1999-7-22 15:58:16","Rúben Silva","H",62,8,4,1.45
10000096,"1346-841","1992-8-8 11:6:55","Constança Nunes","M",88,4,2,1.65
10000097,"1673-414","1997-3-25 13:50:20","Nicole Alves","M",68,9,6,1.83
10000098,"1175-298","1991-3-20 10:26:28","Filipe Coimbra","H",64,0,0,1.69
10000099,"1342-316","1980-4-11 17:44:25","Santiago Nunes","H",80,9,1,1.58

l. Teste Clínico

10000000,"1663-165","1994-12-12 16:22:20","Alexandre Gomes","H",65,1,6,1.61
10000001,"1813-312","1984-7-12 8:30:27","Enzo Álvares","H",97,7,0,1.53
10000002,"1645-741","1980-7-19 9:15:36","Carlota Castro","M",53,3,3,1.69
10000003,"1669-750","1986-4-1 12:57:54","Tomás Vaz","H",81,4,4,1.38
10000004,"1188-816","1988-11-6 8:16:22","Vasco Marques","H",98,7,2,1.86
10000005,"1175-836","1993-6-12 12:56:47","Mateus Sousa","H",82,6,4,1.86
10000006,"1472-570","1983-10-11 15:17:19","Ariana Álvares","M",71,7,3,1.7
10000007,"1478-97","1999-9-21 15:46:59","Bianca Mendes","M",55,9,0,1.54
10000008,"1502-125","1986-9-2 9:32:51","Rodrigo Fernandes","H",71,4,4,1.63
10000009,"1664-114","1990-7-7 9:14:19","Lourenço Teixeira","H",58,2,0,1.3
10000010,"1703-772","1992-7-7 11:32:34","Bianca Nunes","M",69,8,1,1.37
10000011,"1475-751","1983-2-10 8:39:14","Rodrigo Albuquerque","H",89,6,1,1.48
10000012,"1576-879","1994-8-22 9:48:14","Beatriz Mendes","M",88,0,3,1.55
10000013,"1124-439","1982-11-17 10:50:14","Ema Vaz","M",50,5,3,1.41
10000014,"1043-825","1981-8-1 17:31:55","Nicole Ferreira","M",95,8,5,1.51
10000015,"1635-591","1988-9-19 10:34:2","Núria da Gama","M",89,6,0,1.55
10000016,"1569-820","1990-8-2 10:50:2","Diana Álvares","M",60,9,0,1.7
10000017,"1393-403","1988-3-20 16:51:1","Alexandre Bragança","H",62,2,2,1.47
10000018,"1543-477","1986-11-17 15:36:52","Diana Coimbra","M",91,9,5,1.7
10000019,"1283-491","1991-5-25 9:22:24","José Fernandes","H",80,8,4,1.66
10000020,"1841-939","1989-6-13 10:36:15","Afonso Braga","H",88,5,1,1.3
10000021,"1217-614","1987-4-22 14:13:23","André Albuquerque","H",64,6,1,1.65
10000022,"1192-674","1997-5-14 12:28:55","Luís Mendes","H",71,1,4,1.64
10000023,"1115-673","1983-9-1 15:58:34","Alice Jesus","M",55,7,0,1.77
10000024,"1951-733","1981-9-20 17:23:35","Gabriela Gomes","M",45,5,2,1.87
10000025,"1871-254","1989-8-1 9:54:16","Sofia Oliveira","M",68,3,4,1.42

10000026,"1193-184","1983-11-1 16:15:24","Matilde Ribeiro","M",51,6,6,1.81
10000027,"1220-232","1992-10-11 16:16:37","Lucas Moreira","H",42,7,7,1.71
10000028,"1885-81","1988-4-14 12:5:48","Diogo Fernandes","H",41,7,4,1.77
10000029,"1144-740","1992-10-25 17:44:28","Íris Sampaio","M",46,7,2,1.83
10000030,"1703-820","1994-7-25 17:45:9","Benedita Pinto","M",97,3,5,1.36
10000031,"1530-614","1997-12-1 9:36:24","Catarina Fernandes","M",56,7,6,1.35
10000032,"1413-17","1996-12-6 17:10:32","Afonso Ribeiro","H",62,8,6,1.82
10000033,"1230-427","1992-1-17 8:2:51","Rafael Costa","H",93,6,5,1.36
10000034,"1021-773","1982-6-20 17:31:39","Nicole Almeida","M",94,0,3,1.57
10000035,"1804-12","1994-11-23 11:47:18","Bruna Jesus","M",77,4,5,1.72
10000036,"1778-573","1980-4-19 15:44:38","Enzo Alves","H",96,6,7,1.5
10000037,"1063-913","1983-8-19 9:51:29","Manuel Costa","H",62,6,1,1.72
10000038,"1771-604","1994-6-2 17:8:34","Margarida Carvalho","M",50,9,2,1.45
10000039,"1505-700","1995-10-20 8:45:6","Constança Sampaio","M",77,2,4,1.4
10000040,"1751-129","1981-3-9 13:8:38","Vicente Rodrigues","H",95,5,7,1.38
10000041,"1067-839","1993-10-17 15:26:10","Inês Alves","M",81,2,2,1.44
10000042,"1289-614","1980-4-6 9:17:0","Mara Carvalho","M",68,2,2,1.71
10000043,"1249-606","1987-10-9 13:0:28","Vasco Vaz","H",99,4,4,1.42
10000044,"1059-318","1989-11-24 9:39:20","Filipe Pinto","H",50,1,6,1.73
10000045,"1009-70","1994-6-7 15:45:5","Daniela Marques","M",81,9,6,1.65
10000046,"1120-43","1984-6-13 16:27:0","Gonçalo Gonçalves","H",69,3,3,1.83
10000047,"1750-267","1999-6-4 10:46:14","Duarte Marques","H",67,6,4,1.43
10000048,"1285-668","1980-7-22 9:30:22","Constança Carvalho","M",87,6,2,1.38
10000049,"1556-463","1980-10-8 14:14:20","Inês Vaz","M",55,8,2,1.84
10000050,"1092-951","1982-8-17 8:48:15","Camila Alves","M",97,0,2,1.35
10000051,"1671-260","1998-11-6 8:4:26","Bianca Ribeiro","M",81,0,3,1.43
10000052,"1774-330","1995-5-23 10:16:33","Ema Jesus","M",82,6,1,1.41
10000053,"1509-615","1983-10-13 17:42:17","Gabriela Sampaio","M",88,2,1,1.87
10000054,"1391-650","1996-10-1 14:54:56","Tomás Sampaio","H",81,5,0,1.78
10000055,"1775-547","1999-12-24 9:6:56","Yara Fernandes","M",72,4,0,1.81
10000056,"1787-924","1992-7-18 12:27:59","Sofia Alves","M",62,7,2,1.87
10000057,"1061-636","1987-4-3 10:43:21","Francisco Sampaio","H",61,0,5,1.81
10000058,"1486-715","1983-7-1 10:28:34","Diana Teixeira","M",92,3,1,1.3
10000059,"1139-387","1998-6-25 17:17:7","Catarina Fernandes","M",60,3,6,1.76
10000060,"1143-944","1990-4-20 10:31:9","António Guimarães","H",67,5,6,1.76
10000061,"1108-36","1994-1-17 12:58:43","Margarida Almeida","M",49,4,0,1.61
10000062,"1982-405","1995-9-19 12:59:31","Rafael Silva","H",46,6,4,1.89
10000063,"1163-973","1991-4-7 9:54:39","Matilde Guimarães","M",65,7,5,1.4
10000064,"1421-567","1998-3-9 8:35:33","Núria Teixeira","M",59,3,0,1.71
10000065,"1656-609","1995-9-3 10:13:44","Sara Castro","M",63,4,2,1.71

10000066,"1827-648","1982-10-21 13:27:2","Bruna Sousa","M",56,4,4,1.69
10000067,"1244-721","1982-5-9 16:12:48","Francisco Bragança","H",86,4,4,1.41
10000068,"1309-672","1995-4-22 12:43:31","Rodrigo de Caminha","H",81,3,0,1.71
10000069,"1435-868","1997-1-9 17:10:39","Carlota Almeida","M",90,9,3,1.64
10000070,"1419-653","1986-4-6 15:12:21","Alice Gonçalves","M",67,5,7,1.39
10000071,"1029-949","1999-4-21 16:30:16","Lara de Caminha","M",68,0,7,1.7
10000072,"1168-56","1992-9-20 12:16:27","Francisca Pinto","M",58,7,3,1.85
10000073,"1814-479","1994-7-25 15:35:1","Henrique Correia","H",95,3,0,1.69
10000074,"1772-168","1982-6-1 10:14:14","Vasco Alves","H",74,5,3,1.32
10000075,"1759-230","1992-1-14 11:4:12","Ariana Mendes","M",73,0,3,1.41
10000076,"1257-654","1985-6-3 15:44:51","Beatriz Mendes","M",64,0,2,1.53
10000077,"1934-707","1994-7-27 9:29:0","Alexandre Oliveira","H",47,0,3,1.31
10000078,"1856-799","1990-1-9 10:58:16","Letícia Moreira","M",76,1,1,1.86
10000079,"1745-710","1990-12-1 11:50:59","Francisco Costa","H",46,1,7,1.58
10000080,"1986-692","1998-6-16 12:28:32","Beatriz da Veiga","M",46,4,4,1.3
10000081,"1502-152","1986-3-26 9:23:47","Lucas Santos","H",47,2,0,1.31
10000082,"1963-168","1993-7-20 17:27:9","Marta Nunes","M",72,3,4,1.63
10000083,"1436-363","1980-2-15 10:52:9","Yara Sousa","M",82,4,5,1.45
10000084,"1419-280","1989-11-12 11:41:21","Santiago Almeida","H",63,6,6,1.42
10000085,"1348-659","1982-2-18 8:52:51","Santiago Castro","H",87,7,0,1.38
10000086,"1507-59","1993-8-13 15:51:52","Gustavo Sampaio","H",53,7,4,1.55
10000087,"1889-247","1991-8-11 15:1:17","António da Gama","H",85,5,0,1.58
10000088,"1243-983","1983-4-7 13:0:0","Leonardo Martins","H",47,3,4,1.73
10000089,"1287-446","1996-11-21 16:0:13","Guilherme da Veiga","H",40,1,2,1.79
10000090,"1712-41","1995-8-25 15:36:23","Manuel da Veiga","H",94,4,7,1.62
10000091,"1805-239","1985-1-19 16:52:8","Rafael Carvalho","H",82,2,1,1.4
10000092,"1152-223","1983-12-2 12:3:7","André Castro","H",81,7,5,1.49
10000093,"1535-710","1994-9-8 10:12:34","Bernardo Vaz","H",94,6,1,1.45
10000094,"1806-762","1989-1-16 14:12:9","Matias Teixeira","H",64,1,0,1.31
10000095,"1728-312","1999-7-22 15:58:16","Rúben Silva","H",62,8,4,1.45
10000096,"1346-841","1992-8-8 11:6:55","Constança Nunes","M",88,4,2,1.65
10000097,"1673-414","1997-3-25 13:50:20","Nicole Alves","M",68,9,6,1.83
10000098,"1175-298","1991-3-20 10:26:28","Filipe Coimbra","H",64,0,0,1.69
10000099,"1342-316","1980-4-11 17:44:25","Santiago Nunes","H",80,9,1,1.58

IV. Anexo 4 - Script de implementação de functions

```
#Quantos recursos gastou uma clinica entre duas datas
DELIMITER $$
CREATE FUNCTION recursosGastos (cod_clinica int, di date, df date)
returns int deterministic
begin
      declare num int;
    set num = (select count(*) from Recurso join TesteClinico on
TesteClinico.id_teste = Recurso.id_teste
    where TesteClinico.cod_clinica = cod_clinica and TesteClinico.data
between di and df);
      return num;
end $$
DELIMITER ;

#Hora em que há mais testes numa clínica (em média)
DELIMITER $$
CREATE FUNCTION horaMaisTestes (cod_clinica int) returns int
deterministic
begin
      declare num int;
    set num = (select round(sum(hour(TesteClinico.hora))/count(*)) from
TesteClinico where TesteClinico.cod_clinica = cod_clinica);
      return num;
end $$
DELIMITER ;

#Quantos funcionários com um determinado cargo tem uma clínica
DELIMITER $$
CREATE FUNCTION quantosFuncionariosCargo (cod_clinica int, cargo
varchar(45)) returns int deterministic
begin
      declare num int;
    set num = (select count(*) from Staff where Staff.cargo = cargo and
Staff.id_staff in
    (select Staff_id_Staff from Staff_Clinica where Clinica_cod_clinica
= cod_clinica));
      return num;
end $$
```

```
DELIMITER ;

#Quantos anos de serviço tem um determinado funcionário
DELIMITER $$
CREATE FUNCTION quantosAnosServico (id_staff int) returns int
deterministic
begin
      declare num int;
    set num = (select year(now())-year(data_ini_servico) from Staff
where Staff.id_staff = id_staff);
      return num;
end $$
DELIMITER ;

#Quanto custa realizar um teste numa clínica (em média)
DELIMITER $$
CREATE FUNCTION quantoCustaTeste (cod_clinica int) returns double
deterministic
begin
      declare num double;
    set num = (select round(sum(TesteClinico.preco)/count(*),2) from
TesteClinico where TesteClinico.cod_clinica = cod_clinica);
      return num;
end $$
DELIMITER ;
```

V. Anexo 5 - Script de implementação de triggers

```
DELIMITER $$
CREATE TRIGGER deleteClube
      AFTER DELETE
    ON Atleta
    FOR EACH ROW
BEGIN
    DECLARE num INT;
    SET num = (select count(*) from Atleta where cod_clube =
old.cod_clube);
      IF (num = 0) THEN DELETE from Clube where cod_clube =
old.cod_clube;
    END IF;
END $$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER deleteModalidade
      AFTER DELETE
    ON Atleta
    FOR EACH ROW
BEGIN
    DECLARE num INT;
    DECLARE num_competicao INT;
    SET num_competicao = (select count(*) from Competicao where
cod_modalidade = old.cod_modalidade);
    SET num = (select count(*) from Atleta where cod_modalidade =
old.cod_modalidade);
      IF (num = 0 and num_competicao = 0) THEN
    DELETE from Modalidade where cod_modalidade = old.cod_modalidade;
    END IF;
END $$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER deleteRecurso
      BEFORE DELETE
    ON TesteClinico
    FOR EACH ROW
```

```
BEGIN
    DECLARE num INT;
    SET num = (select count(*) from Recurso where id_teste =
old.id_teste);
      IF (num = 1) THEN
    DELETE from Recurso where id_teste = old.id_teste;
    END IF;
END $$
DELIMITER ;
```

VI. Anexo 6 - Script de implementação de view's

```
create view atletasAlfabeticamente as
      select A.*
      from Atleta A
      order by A.nome asc;

create view staffAlfabeticamente as
      select S.*
    from Staff S
    order by S.nome asc;

create view clinicasAlfabeticamente as
      select C.*
    from Clinica C
    order by C.nome asc;

create view testesData as
      select T.*
    from TesteClinico T
    order by T.data desc;
```

VII. Anexo 7 - Script de implementação de procedures para atletas

```
delimiter $$
CREATE PROCEDURE ATodosTestes(IN nif_atleta int)
    begin
        Select
            T.*
        From
            TesteClinico as T,
            Atleta as A
        where
              A.nif = T.nif
            and A.nif = nif_atleta;
      end $$
delimiter ;

delimiter $$
CREATE PROCEDURE ATestesEmClinica(IN n_clinica int, IN n_atleta INT)
begin
      Select
            T.*
      From
            TesteClinico as T,
            Atleta as A
      where
            T.nif = A.nif
        and A.nif = n_atleta
            and T.cod_clinica = n_clinica;
end$$
delimiter ;

delimiter $$
CREATE PROCEDURE ATestesEntreDatas(IN d_start DATE, IN d_end DATE, IN
n_atleta INT)
    begin
        Select *
        From
              TesteClinico as T,
```

```
            Atleta as A
        where
                  T.nif = A.nif
            and A.nif = n_atleta
              and T.data between d_start and d_end;
      end $$
delimiter ;

delimiter $$
CREATE PROCEDURE ATestesPrecoCrescente(In nif_atleta int)
    begin
        Select
                  T.id_teste,
                  T.preco
        From
              Atleta as A,
            TesteClinico as T
        where
            A.nif = T.nif
            and A.nif = nif_atleta
        order by T.preco;
      end $$
delimiter ;

delimiter $$
CREATE PROCEDURE AClinicaEmQueMaisGasta(In nif_atleta int)
    begin
        Select
                  T.cod_clinica,
            sum(T.preco) as Gasto
        From
              Atleta as A,
            TesteClinico as T
        where
            A.nif = T.nif
            and A.nif = nif_atleta
            group by T.cod_clinica
        order by sum(T.preco) DESC
        limit 1;
      end $$
delimiter ;
```

VIII. Anexo 8 - Script de implementação de procedures para diretores das clínicas

```
delimiter $$
CREATE PROCEDURE DCFaturacaoClinica(IN n_staff int, IN di DATE, IN df
DATE)
    begin
        Select
                  C.nome,
            count(*) as NrAtletas,
                  sum(T.preco) as Faturacao
        From
                  Clinica as C,
            TesteClinico as T
        where
            C.cod_clinica = T.cod_clinica
            and C.id_staff = n_staff
            and T.data between di and df
            group by C.nome;
      end $$
delimiter ;


delimiter $$
CREATE PROCEDURE DCClubeMaisTestesEntreDatasClinica(IN n_staff int, IN
di DATE, IN df DATE)
      begin
      SELECT Clube.designacao,
    count(Clube.designacao) as NrTestes
      FROM Clinica
      INNER JOIN TesteClinico
      ON Clinica.cod_clinica = TesteClinico.cod_clinica
      INNER JOIN Atleta
      ON Atleta.nif = TesteClinico.nif
      INNER JOIN Clube
      ON Atleta.cod_clube = Clube.cod_clube where Clinica.id_staff =
n_staff and TesteClinico.data between di and df group by Clube.designacao
order by count(Clube.designacao) DESC limit 1;
      end $$
```

```
delimiter ;


delimiter $$
CREATE PROCEDURE DCClubesInexistentesClinica(IN n_staff int)
begin
Select Cl.designacao from Clube as Cl where Cl.cod_clube not in
(SELECT Cl.cod_clube
      FROM Clinica as C
      INNER JOIN TesteClinico as T
      ON C.cod_clinica = T.cod_clinica
      INNER JOIN Atleta as A
      ON A.nif = T.nif
      INNER JOIN Clube as Cl
      ON A.cod_clube = Cl.cod_clube where C.id_staff = n_staff);
end $$
delimiter ;


delimiter $$
create procedure DCStaffAreaClinica (IN n_staff int)
begin
select S.especialidade ,
count(S.especialidade) as NrEspecialistas
from Staff as S
join Staff_Clinica as SC
on SC.Staff_id_staff = S.id_staff
where SC.Staff_id_staff
in (select SC.Staff_id_staff
from Clinica as C
 join Staff_Clinica as SC
 on C.cod_clinica = SC.Clinica_cod_clinica where C.id_staff = n_staff)
group by S.especialidade order by count(S.especialidade);
end $$
delimiter ;


delimiter $$
create procedure DCMesesOrdemCrescenteClinica(IN n_staff int)
begin
select
```

```
month(T.data) as Mes,
count(month(T.data)) as NrTestes
from Clinica as C,
TesteClinico as T
where T.cod_clinica = C.cod_clinica
and C.id_staff = n_staff
group by month(T.data)
order by count(month(T.data));
end $$
delimiter ;
```

IX. Anexo 9 - Script de implementação de procedures para responsável de teste

```
delimiter $$
CREATE PROCEDURE DTRecursosTeste(IN n_staff int)
    begin
        Select
                  T.id_teste,
            R.designacao
        From
                  Recurso as R,
            TesteClinico as T
        where
            R.id_teste = T.id_teste
            and T.id_staff = n_staff;
      end $$
delimiter ;

delimiter $$
CREATE PROCEDURE DTClinicaRealizouMaisTestes(In n_staff int)
    begin
        Select
              C.nome,
            count(C.cod_clinica) as NrTestes
        From
              Clinica as C,
            TesteClinico as T
        where
              T.cod_clinica = C.cod_clinica
              and T.id_staff = n_staff
            group by C.cod_clinica
            order by count(C.cod_clinica) DESC
            limit 1;
      end $$
delimiter ;

delimiter $$
CREATE PROCEDURE DTComQuemTrabalhouTestes(In n_staff int)
    begin
```

```
        Select
              E.id_teste,
            S.nome
        From
              EquipaEnvolvente as E,
            Staff as S
        where
              S.id_staff = E.id_staff
              and E.id_teste in
            (select
                  T.id_teste
            from
                  TesteClinico as T,
            EquipaEnvolvente as E
            where T.id_teste = E.id_teste
        and T.id_staff = n_staff);
      end $$
delimiter ;

delimiter $$
CREATE PROCEDURE DTAtletasTestes(In n_staff int)
    begin
        Select
              T.id_teste,
            A.*
        From
              Atleta as A,
            TesteClinico as T
        where
              A.nif = T.nif
            and T.id_staff = n_staff;
      end $$
delimiter ;

delimiter $$
CREATE PROCEDURE DTContactoAtletasTestes(In n_staff int)
      begin
            SELECT
        T.id_teste,
        A.nome,
        C.ContactoCol
```

```
        FROM TesteClinico as T
        join Atleta as A
        ON T.nif = A.nif
            join Contacto as C
        ON A.nif = C.Atleta_nif_atleta
        where T.id_staff = n_staff;
      end $$
delimiter ;
```

X. Anexo 10 - Script de implementação de transactions

```
DELIMITER $$
CREATE PROCEDURE addClinica(IN cod_clinica VARCHAR(45), IN clinica_nome
VARCHAR(45),
                                        IN localidade VARCHAR(45), IN
id_staff INT,
                            IN   staff_nome   VARCHAR(45),   IN   cargo
VARCHAR(45),
                            IN    especialidade    VARCHAR(45),    IN
data_ini_servico DATETIME)
BEGIN
      DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;

      START TRANSACTION;
    SET autocommit = 0;
    -- criar Staff
    INSERT                                                          INTO
Staff(id_staff,nome,cargo,especialidade,data_ini_servico)
    values (id_staff,staff_nome,cargo,especialidade,data_ini_servico);
    -- criar clinica
    INSERT INTO Clinica(cod_clinica,nome,localidade,id_staff)
    values (cod_clinica,clinica_nome,localidade,id_staff);

    COMMIT;
END $$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE addAtleta(IN nif INT, IN cod_postal VARCHAR(45), IN DOB
DATETIME,
                           IN nome VARCHAR(45), IN sexo VARCHAR(1), IN
peso INT,
                           IN  modalidade_designacao  VARCHAR(45),  IN
clube_designacao VARCHAR(45),
                           IN altura DECIMAL(3, 2))
BEGIN
```

```
    DECLARE codigo_clube INT;
    DECLARE maximo_clube INT;
    DECLARE quantos_clube INT;
    DECLARE proximo_clube INT;
    DECLARE codigo_modalidade INT;
    DECLARE maximo_modalidade INT;
    DECLARE quantos_modalidade INT;
    DECLARE proximo_modalidade INT;

      DECLARE EXIT HANDLER FOR SQLEXCEPTION
      BEGIN
            ROLLBACK;
      END;

    START TRANSACTION;
    SET autocommit = 0;
    SET codigo_clube = (select cod_clube from Clube where
Clube.designacao = clube_designacao);
    SET quantos_clube = (select count(*) from Clube where cod_clube =
codigo_clube);
    SET maximo_clube = (select cod_clube from Clube order by cod_clube
DESC limit 1);
    SET proximo_clube = maximo_clube + 1;
    SET codigo_modalidade = (select cod_modalidade from Modalidade where
Modalidade.designacao = modalidade_designacao);
    SET quantos_modalidade = (select count(*) from Modalidade where
cod_modalidade = codigo_modalidade);
    SET maximo_modalidade = (select cod_modalidade from Modalidade order
by cod_modalidade DESC limit 1);
    SET proximo_modalidade = maximo_modalidade + 1;

    IF (quantos_modalidade = 0) THEN
        INSERT INTO Modalidade(cod_modalidade, designacao)
        values (proximo_modalidade, modalidade_designacao);
        SET codigo_modalidade = proximo_modalidade;
    END IF;

    IF (quantos_clube = 0) THEN
        INSERT INTO Clube(cod_clube, designacao)
        values (proximo_clube, clube_designacao);
```

```
        SET codigo_clube = proximo_clube;
    END IF;

    INSERT INTO Atleta(nif, cod_postal, DOB, nome, sexo, peso,
cod_modalidade, cod_clube, altura)
    values (nif, cod_postal, DOB, nome, sexo, peso, codigo_modalidade,
codigo_clube, altura);

    COMMIT;
END $$
DELIMITER ;
```

XI. Anexo 11 - Script de implementação de criação de user e atribuição de premições

```
CREATE ROLE DIRETORCLINICA;
CREATE ROLE DIRETORTESTE;
CREATE ROLE ATLETA;
CREATE ROLE ADMINISTRADOR;

GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DCFaturacaoClinica TO
DIRETORCLINICA;
GRANT EXECUTE ON PROCEDURE
MedicinaDesporto.DCClubeMaisTestesEntreDatasClinica TO DIRETORCLINICA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DCClubesInexistentesClinica
TO DIRETORCLINICA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DCStaffAreaClinica TO
DIRETORCLINICA;
GRANT EXECUTE ON PROCEDURE
MedicinaDesporto.DCMesesOrdemCrescenteClinica TO DIRETORCLINICA;

GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTRecursosTeste TO
DIRETORTESTE;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTClinicaRealizouMaisTestes
TO DIRETORTESTE;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTComQuemTrabalhouTestes TO
DIRETORTESTE;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTAtletasTestes TO
DIRETORTESTE;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.DTContactoAtletasTestes TO
DIRETORTESTE;

GRANT EXECUTE ON PROCEDURE MedicinaDesporto.AClinicaEmQueMaisGasta TO
ATLETA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.ATestesEmClinica TO ATLETA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.ATestesEntreDatas TO ATLETA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.ATestesPrecoCrescente TO
ATLETA;
GRANT EXECUTE ON PROCEDURE MedicinaDesporto.ATodosTestes TO ATLETA;
```

```
GRANT ALL PRIVILEGES ON MedicinaDesporto.* TO ADMINISTRADOR WITH GRANT
OPTION;

CREATE USER 'diretorclinica2'@'%' IDENTIFIED BY
'diretorclinicapassword';
GRANT DIRETORCLINICA to 'diretorclinica2'@'%';

CREATE USER 'diretorteste14'@'%' IDENTIFIED BY 'diretorteste';
GRANT DIRETORTESTE to 'diretorteste14'@'%';

CREATE USER 'atleta50'@'%' IDENTIFIED BY 'atletapassword';
GRANT ATLETA to 'atleta50'@'%';

CREATE USER 'administrador'@'%' IDENTIFIED BY 'adminpassword';
GRANT ADMINISTRADOR to 'administrador'@'%';
```

XII. Anexo 12 - Script de load de csv's para Neo4j

```
// Create Clubes
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///clube.csv" AS row
CREATE (:Clube {cod_clube: toInteger(row.cod_clube), designacao:
row.designacao});

CREATE INDEX ON :Clube(cod_clube);

// Create Atletas
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///atleta.csv" AS row
CREATE (:Atleta {nif: toInteger(row.nif), cod_postal: row.cod_postal,
DOB: datetime(replace(row.DOB,' ','T')), nome: row.nome, sexo: row.sexo,
peso: toInteger(row.peso), altura: toFloat(row.altura)});

CREATE INDEX ON :Atleta(nif);

// Create Modalidade
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///modalidade.csv" AS row
CREATE (:Modalidade {cod_modalidade: toInteger(row.cod_modalidade),
designacao: row.designacao});

CREATE INDEX ON :Modalidade(cod_modalidade);

// Create Competicao
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///competicao.csv" AS row
CREATE (:Competicao {id_competicao: toInteger(row.id_competicao),
designacao: row.designacao, nome: row.nome, data: date(row.data), local:
row.local, hora: time(row.hora)});

CREATE INDEX ON :Competicao(id_competicao);

// Create Contacto
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///contacto.csv" AS row
CREATE (:Contacto {contactocol: row.Contactocol});
```

```
CREATE INDEX ON :Contacto(contactocol);

// Create TesteClinico
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///testeclinico.csv" AS row
CREATE (:TesteClinico {id_teste: toInteger(row.id_teste), data:
date(row.data), preco: toFloat(row.preco), hora: time(row.hora)});

CREATE INDEX ON :TesteClinico(id_teste);

// Create Recurso
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///recurso.csv" AS row
CREATE (:Recurso {cod_recurso: toInteger(row.cod_recurso), designacao:
row.designacao});

CREATE INDEX ON :Recurso(cod_recurso);

// Create Clinica
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///clinica.csv" AS row
CREATE (:Clinica {cod_clinica: toInteger(row.cod_clinica), nome:
row.nome, localidade: row.localidade});

CREATE INDEX ON :Clinica(cod_clinica);

// Create Staff
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///staff.csv" AS row
CREATE (:Staff {id_staff: toInteger(row.id_staff), nome: row.nome,
cargo: row.cargo, especialidade: row.especialidade, data_ini_servico:
datetime(replace(row.data_ini_servico,' ','T'))});

CREATE INDEX ON :Staff(id_staff);

//RelationShip PRATICA
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///atleta.csv" AS row
MATCH (a:Atleta{nif: toInteger(row.nif)})
MATCH (m:Modalidade {cod_modalidade: toInteger(row.cod_modalidade)})
```

```
MERGE (a)-[:PRATICA]->(m);


//RelationShip COMPETE_POR
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///atleta.csv" AS row
MATCH (a:Atleta{nif: toInteger(row.nif)})
MATCH (c:Clube {cod_clube: toInteger(row.cod_clube)})
MERGE (a)-[:COMPETE_POR]->(c);


//RelationShip INSCRITO
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///inscricao.csv" AS row
MATCH (a:Atleta{nif: toInteger(row.nif)})
MATCH (c:Competicao {id_competicao: toInteger(row.id_competicao)})
MERGE (a)-[:INSCRITO{data : date(row.data), hora: time(row.hora)}]-(c);


//RelationShip CONTACTO_DE
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///contacto.csv" AS row
MATCH (a:Atleta{nif: toInteger(row.Atleta_nif_atleta)})
MATCH (c:Contacto {contactocol: row.Contactocol})
MERGE (c)-[:CONTACTO_DE]->(a);


//RelationShip DA_MODALIDADE
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///competicao.csv" AS row
MATCH (m:Modalidade {cod_modalidade: toInteger(row.cod_modalidade)})
MATCH (c:Competicao {id_competicao: toInteger(row.id_competicao)})
MERGE (c)-[:DA_MODALIDADE]->(m);


//RelationShip MARCADO_PARA
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///testeclinico.csv" AS row
MATCH (tc:TesteClinico {id_teste: toInteger(row.id_teste)})
MATCH (a:Atleta{nif: toInteger(row.nif)})
MERGE (tc)-[:MARCADO_PARA]->(a);
```

```
//RelationShip RESPONSAVEL
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///testeclinico.csv" AS row
MATCH (tc:TesteClinico {id_teste: toInteger(row.id_teste)})
MATCH (s:Staff{id_staff: toInteger(row.id_staff)})
MERGE (tc)-[:RESPONSAVEL]->(s);


//RelationShip EQUIPA_EVOLVENTE
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///equipaevolvente.csv" AS row
MATCH (tc:TesteClinico {id_teste: toInteger(row.id_teste)})
MATCH (s:Staff{id_staff: toInteger(row.id_staff)})
MERGE (tc)-[:EQUIPA_EVOLVENTE]-(s);


//RelationShip USADO
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///recurso.csv" AS row
MATCH (tc:TesteClinico {id_teste: toInteger(row.id_teste)})
MATCH (r:Recurso{cod_recurso: toInteger(row.cod_recurso)})
MERGE (r)-[:USADO]->(tc);


//RelationShip REALIZADO_PELA
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///testeclinico.csv" AS row
MATCH (tc:TesteClinico {id_teste: toInteger(row.id_teste)})
MATCH (c:Clinica{cod_clinica: toInteger(row.cod_clinica)})
MERGE (tc)-[:REALIZADO_PELA]->(c);


//RelationShip EQUIPA_CLINICA
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///staff_clinica.csv" AS row
MATCH (s:Staff{id_staff: toInteger(row.Staff_id_staff)})
MATCH (c:Clinica{cod_clinica: toInteger(row.Clinica_cod_clinica)})
MERGE (s)-[:EQUIPA_CLINICA]-(c);
```

```
//RelationShip DIRETOR
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///clinica.csv" AS row
MATCH (s:Staff{id_staff: toInteger(row.id_staff)})
MATCH (c:Clinica{cod_clinica: toInteger(row.cod_clinica)})
MERGE (c)-[:DIRETOR]->(s);
```

XIII. Anexo 13 - Script de implementação de functions em Neo4j

```
// CREATE FUNCTION recursosGastos (cod_clinica int, di date, df date) returns int deterministic
MATCH (r:Recurso)-[:USADO]->(tc:TesteClinico)-[:REALIZADO_PELA]->(c:Clinica) WHERE c.cod_clinica = $cod_clinica and date($di) <= tc.data and date($df) >= tc.data RETURN count(r) as recursos_gastos

// CREATE FUNCTION horaMaisTestes (cod_clinica int) returns int deterministic
MATCH (tc:TesteClinico)-[:REALIZADO_PELA]->(c:Clinica) WHERE c.cod_clinica = $cod_clinica WITH toInteger(round(toFloat(sum(tc.hora.hour))/ toFloat(count(tc)))) as hora_mais_testes RETURN hora_mais_testes

// CREATE FUNCTION quantosFuncionariosCargo (cod_clinica int, cargo varchar(45)) returns int deterministic
MATCH (s:Staff)-[:EQUIPA_CLINICA]-(c:Clinica) WHERE c.cod_clinica = $cod_clinica and s.cargo = $cargo RETURN count(s) as quantos_funcionarios_cargo

// CREATE FUNCTION quantosAnosServico (id_staff int) returns int deterministic
MATCH (s:Staff) WHERE s.id_staff = $id_staff WITH (toInteger(datetime().year) - toInteger(s.data_ini_servico.year)) as anos_de_servico RETURN anos_de_servico

// CREATE FUNCTION quantoCustaTeste (cod_clinica int) returns double deterministic
MATCH (tc:TesteClinico)-[:REALIZADO_PELA]->(c:Clinica) WHERE c.cod_clinica = $cod_clinica WITH round(toFloat(sum(tc.preco)))/count(tc) as media_custo_testes RETURN media_custo_testes
```

XIV. Anexo 14 - Script de implementação de triggers em Neo4j

```
// CREATE TRIGGER deleteClube
MATCH (c:Clube)<-[:COMPETE_POR]-(a:Atleta) WHERE a.nif = $nif_atleta_apagado WITH collect(c.cod_clube) as clubes
MATCH (c2:Clube)-[rl]-() WITH clubes as clubes2, count(rl) as nao_usados
MATCH (c3:Clube) WHERE c3.cod_clube in clubes2 and nao_usados = 0
DELETE c3


// CREATE TRIGGER deleteModalidade
MATCH (m:Modalidade)<-[:DA_MODALIDADE]-(a:Atleta) WHERE a.nif = $nif_atleta_apagado WITH collect(m.cod_modalidade) as modalidades
MATCH (m2:Modalidade)-[rl]-() WITH modalidades as modalidades2, count(rl) as nao_usados
MATCH (m3:Modalidade) WHERE m3.cod_modalidade in modalidades2 and nao_usados = 0
DELETE m3


// CREATE TRIGGER deleteRecurso
MATCH (r:Recurso)-[]-(tc:TesteClinico) WHERE tc.id_teste = $id_teste_apagado WITH collect(r.cod_recurso) as recursos
MATCH (r2:Recurso)-[rl]-() WITH recursos as usados, count(rl) as nao_usados
MATCH (r3:Recurso) WHERE r3.cod_clube in usados and nao_usados = 0
DELETE r3
```

XV. Anexo 15 Script de implementação de view's em Neo4j

```
// create view atletasAlfabeticamente
MATCH (a:Atleta) RETURN a as atleta ORDER BY a.nome

// create view staffAlfabeticamente
MATCH (s:Staff) RETURN s as staff ORDER BY s.nome

// create view clinicasAlfabeticamente
MATCH (c:clinica) RETURN c as clinica ORDER BY c.nome

// create view testesData
MATCH (tc:TesteClinico) RETURN tc as teste_clinico ORDER BY tc.data
```

XVI. Anexo 16 - Script de implementação de procedures para atletas

```
// CREATE PROCEDURE ATodosTestes(IN nif_atleta int)
MATCH (c:TesteClinico)-[:MARCADO_PARA]->(a:Atleta) WHERE a.nif = $nif_atleta RETURN c as TesteClinico

// CREATE PROCEDURE ATestesEmClinica(IN n_clinica int, IN n_atleta INT)
MATCH (e:Clinica)<-[:REALIZADO_PELA]-(c:TesteClinico)-[:MARCADO_PARA]->(a:Atleta) WHERE a.nif = $n_atleta and e.cod_clinica = $n_clinica RETURN c as TesteClinico

// CREATE PROCEDURE ATestesEntreDatas(IN d_start DATE, IN d_end DATE, IN n_atleta INT)
MATCH (c:TesteClinico)-[:MARCADO_PARA]->(a:Atleta) WHERE a.nif = $n_atleta and c.data>= date($d_start) and c.data <=date($d_end) RETURN c as TesteClinico

// CREATE PROCEDURE ATestesPrecoCrescente(In nif_atleta int)
MATCH (c:TesteClinico)-[:MARCADO_PARA]->(a:Atleta) WHERE a.nif = $nif_atleta RETURN c.id_teste as id_teste,c.preco as preco ORDER BY preco

// CREATE PROCEDURE AClinicaEmQueMaisGasta(In nif_atleta int)
MATCH (e:Clinica)<-[:REALIZADO_PELA]-(c:TesteClinico)-[:MARCADO_PARA]->(a:Atleta) WHERE a.nif = $nif_atleta WITH e.cod_clinica as clinica, sum(c.preco) as gastos RETURN clinica,gastos ORDER BY gastos DESC LIMIT 1
```

XVII. Anexo 17 - Script de implementação de procedures para diretores das clínicas

```
// CREATE PROCEDURE DCFaturacaoClinica(IN n_staff int, IN di DATE, IN df DATE)
MATCH (s:Staff)<-[:DIRETOR]-(c:Clinica)<-[:REALIZADO_PELA]-(tc:TesteClinico) WHERE s.id_staff = $n_staff and tc.data>= date($di) and tc.data <= date($df) WITH c.cod_clinica as cod_clinica, count(tc) as numero_atletas, sum(tc.preco) as faturacao_total RETURN cod_clinica, numero_atletas, faturacao_total;

// CREATE PROCEDURE DCClubeMaisTestesEntreDatasClinica(IN n_staff int, IN di DATE, IN df DATE)
MATCH (s:Staff)<-[:DIRETOR]-(c:Clinica)<-[:REALIZADO_PELA]-(tc:TesteClinico)-[:MARCADO_PARA]->(:Atleta)-[:COMPETE_POR]->(e:Clube) WHERE s.id_staff = $n_staff and tc.data>= date($di) and tc.data <= date($df) WITH c.cod_clinica as cod_clinica, count(tc) as numero_testes_clinicos, e.designacao as nome_clube RETURN cod_clinica, nome_clube ,numero_testes_clinicos ORDER BY numero_testes_clinicos DESC, nome_clube LIMIT 1;

// CREATE PROCEDURE DCClubesInexistentesClinica(IN n_staff int)
MATCH (s:Staff)<-[:DIRETOR]-(c:Clinica)<-[:REALIZADO_PELA]-(:TesteClinico)-[:MARCADO_PARA]->(:Atleta)-[:COMPETE_POR]->(e:Clube)
WHERE s.id_staff = $n_staff WITH collect(DISTINCT e.designacao) as ci
MATCH (cu:Clube) WHERE not cu.designacao in ci
RETURN cu.designacao as nome_clube ORDER BY nome_clube

// CREATE PROCEDURE DCStaffAreaClinica (IN n_staff int)
MATCH (s1:Staff)<-[:DIRETOR]-(c:Clinica)<-[:EQUIPA_CLINICA]-(s2:Staff) WHERE s1.id_staff = 47 WITH count(s2.especialidade) as num_staff, s2.especialidade as especialidade RETURN especialidade, num_staff ORDER BY num_staff

// CREATE PROCEDURE DCMesesOrdemCrescenteClinica(IN n_staff int)
MATCH (s:Staff)<-[:DIRETOR]-(:Clinica)<-[:REALIZADO_PELA]-(tc:TesteClinico) WHERE s.id_staff = 47 WITH tc.data.month as mes, count(tc) as num_tc RETURN mes, num_tc ORDER BY num_tc
```

XVIII. Anexo 18 - Script de implementação de procedures para responsável de teste

```
// CREATE PROCEDURE DTRecursosTeste(IN n_staff int)
MATCH (s:Staff)<-[:RESPONSAVEL]-(tc:TesteClinico)<-[:USADO]-(r:Recurso)
WHERE s.id_staff = $n_staff WITH tc.id_teste as id_teste, r.designacao
as designacao RETURN id_teste, designacao

// CREATE PROCEDURE DTClinicaRealizouMaisTestes(In n_staff int)
MATCH    (s:Staff)<-[:RESPONSAVEL]-(tc:TesteClinico)-[:REALIZADO_PELA]-
>(c:Clinica) WHERE s.id_staff = $n_staff WITH c.cod_clinica as cod ,
c.nome as nome, count(tc.id_teste) as num_teste RETURN nome, num_teste
ORDER BY num_teste DESC LIMIT 1

// CREATE PROCEDURE DTComQuemTrabalhouTestes(In n_staff int)
MATCH  (s:Staff)<-[:RESPONSAVEL]-(tc:TesteClinico)-[:EQUIPA_EVOLVENTE]-
>(s2:Staff) WHERE s.id_staff = $n_staff WITH tc.id_teste as id_teste,
s2.nome as nome RETURN id_teste, nome

// CREATE PROCEDURE DTAtletasTestes(In n_staff int)
MATCH      (s:Staff)<-[:RESPONSAVEL]-(tc:TesteClinico)-[:MARCADO_PARA]-
>(a:Atleta) WHERE s.id_staff = $n_staff WITH tc.id_teste as id_teste, a
as atleta RETURN id_teste, atleta

// CREATE PROCEDURE DTContactoAtletasTestes(In n_staff int)
MATCH      (s:Staff)<-[:RESPONSAVEL]-(tc:TesteClinico)-[:MARCADO_PARA]-
>(a:Atleta)<-[:CONTACTO_DE]-(c:Contacto)  WHERE  s.id_staff  =  $n_staff
WITH tc.id_teste as id_teste, a.nome as nome, c.contactocol as contacto
RETURN id_teste,nome,contacto
```

XIX. Anexo 19 - Script de implementação de criação de user e atribuição de premições

```
// CREATE ROLES

CALL dbms.security.createRole('ATLETA');
CALL dbms.security.createRole('DIRETORCLINICA');
CALL dbms.security.createRole('DIRETORTESTE');


// CREATE USERS

CALL dbms.security.createUser('atleta50', 'atletapassword', true);
CALL                        dbms.security.createUser('diretorclinica2',
'diretorclinicapassword', true);
CALL dbms.security.createUser('diretorteste14', 'diretorteste', true);


// ADD ROLES TO USERS

CALL dbms.security.addRoleToUser('ATLETA','atleta50');
CALL dbms.security.addRoleToUser('DIRETORCLINICA','diretorclinica2');
CALL dbms.security.addRoleToUser('DIRETORTESTE','diretorteste14');

CALL dbms.security.addRoleToUser('reader','atleta50');
CALL dbms.security.addRoleToUser('reader','diretorclinica2');
CALL dbms.security.addRoleToUser('reader','diretorteste14');
```