



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

Unidade Curricular de Bases de Dados

Ano Letivo de 2019/2020

Agendamento e realização de testes clínicos de atletas

Grupo 42

Hugo Cunha A84656

Maria Pires A86268

Susana Marques A84167

4 de Janeiro de 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Agendamento e realização de testes clínicos de atletas

Grupo 42

Hugo Cunha A84656

Maria Pires A86268

Susana Marques A84167

4 de Janeiro de 2020

Resumo

Este relatório foi realizado no âmbito do desenvolvimento de uma Base de Dados para a *Clinica Alta Performance*, que foca a sua atividade na área da saúde principalmente na gestão da realização de testes clínicos a atletas de alta competição. Desde o agendamento destes testes até ao resultado dos mesmos, passando pelas provas realizadas pelos atletas que se submetem aos testes, este software facilita e permite de forma eficiente organizar a realização destes exames.

Neste relatório são apresentadas todas as etapas realizadas para o desenvolvimento desta base de dados até à implementação física da mesma.

Inicialmente, estudou-se o meio onde o software implementado será inserido. Analisou-se as motivações e os objetivos da clínica mencionada, como esta mesma surgiu e de que modo a implementação de uma base de dados será economicamente rentável para a mesma.

A seguir, por métodos de análise, levantou-se e aprovou-se os requisitos essenciais para o sistema que guiaram todo o processo que foi desenvolvido. Com estes requisitos, iniciou-se a modulação conceptual que depois, em conformidade com as regras do mapeamento ER, serviu para desenvolver o modelo lógico na ferramenta *MySQL Workbench*.

Concluído o modelo lógico, implementou-se fisicamente a Base de Dados através também do *MySQL Workbench*, garantindo a validação de todo o trabalho realizado anteriormente e permitindo implementar uma solução correta para o problema apresentado.

Recorreu-se também a um sistema NoSQL quando houve necessidade de uma Base de Dados complementar com maior escalabilidade e “performance” para alcançar a solução para as dificuldades sentidas com o aumento de agendamentos de atletas.

Com a resolução deste problema, depois de satisfeitas as necessidades encontradas, demos por concluído este projeto.

Área de Aplicação: Desenho, arquitetura, desenvolvimento e implementação de Sistemas de Bases de Dados.

Palavras-Chave: Bases de Dados, Bases de Dados Relacionais, Análise de Requisitos, Entidades, Atributos, Relacionamentos, Modelo Conceptual, Modelo Lógico, Modelo Físico, Normalização, Interrogações, Transações, Triggers, Índices, Vistas de Utilização, Backup, MySQL WorkBench, MySQL.

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. Fundamentação da Implementação da Base de Dados	2
1.4. Análise da viabilidade do projeto	2
1.5. Estrutura do Relatório	2
2. Levantamento e Análise de requisitos	4
2.1. Método de Levantamento e de Análise de Requisitos Adotados	4
2.2. Requisitos Levantados	4
2.2.1 Requisitos de Descrição	4
2.2.2 Requisitos de Exploração	5
2.2.3 Requisitos de Controlo	5
2.3. Análise de Requisitos	6
2.4. Caraterização dos perfis de utilização	6
3. Modelação Conceptual	7
3.1. Apresentação da Abordagem de Modelação Realizada	7
3.2. Identificação e Caracterização das Entidades	7
3.3. Identificação e Caracterização dos Relacionamentos	8
3.3.1 Dicionário de relacionamentos	11
3.4. Identificação e Caracterização da Associação dos Atributos com as Entidades e Relacionamentos	12
3.5. Chaves Candidatas, Primárias e Alternativas	13
3.6. Detalhe ou Generalização de Entidades	13
3.7. Apresentação e Explicação do Diagrama ER	14
3.8. Validação do Modelo de Dados com o Utilizador	14
4. Modelação Lógica	16
4.1. Construção e Validação do Modelo de Dados Lógico	16
4.1.1 Entidades fortes	16
4.1.2 Relacionamentos 1:N	18
4.1.3 Relacionamentos N:M	19
4.1.4 Atributos Multivvalorados	19
4.2. Desenho do Modelo Lógico	20
4.3. Validação do Modelo através da Normalização	20
4.3.1 Primeira Forma Normal (1FN)	20

4.3.2 Segunda Forma Normal (2FN)	21
4.3.3 Terceira Forma Normal (3FN)	21
4.4. Validação do Modelo com as Interrogações do Utilizador	21
4.5. Validação do Modelo com as Transações Estabelecidas	24
4.6. Verificação das Restrições de Integralidade	26
4.7. Revisão do Modelo Lógico com o Utilizador	26
5. Implementação Física	27
5.1. Seleção do Sistema de Gestão de Base de Dados	27
5.2. Tradução do Esquema Lógico para o Sistema de Gestão de Bases de Dados escolhido em SQL	27
5.2.1 Desenho das relações base	27
5.2.2 Desenho das restrições	31
5.3. Tradução das Interrogações do Utilizador para SQL	34
5.4. Tradução das transações estabelecidas para SQL	35
5.5. Triggers SQL	38
5.6. Escolha, Definição e Caracterização de Índices em SQL	38
5.7. Estimativa do Espaço em Disco da Base de Dados e Taxa de Crescimento Anual	38
5.8. Definição e Caracterização das vistas de utilização em SQL	42
5.9. Definição e Caracterização dos Mecanismos de Segurança em SQL	43
5.10. Revisão do Sistema Implementado com o Utilizador	44
6. Sistema <i>NoSQL</i> : <i>Neo4j</i>	45
6.1. Justificação da utilização de um sistema <i>NoSQL</i>	45
6.2. Identificação dos objetivos da base de dados	45
6.3. Identificação e explicação do tipo de questões (necessidades) que serão realizadas sobre o sistema de dados <i>NoSQL</i>	45
6.4. Definição da estrutura base para o sistema de dados <i>NoSQL</i> que satisfaça os requisitos e as questões apresentadas anteriormente	46
6.5. Identificação dos objetos de dados no sistema <i>SQL</i> que serão utilizados para alimentar o novo sistema	48
6.6. Mapeamento do processo de migração de dados, descrevendo o processo de conversão dos vários objetos de dados	48
6.7. Explicação do processo de migração de dados, explicando de forma detalhada as suas principais etapas - extração, transformação e carregamento	50
6.8. Apresentação e descrição da implementação do processo de migração de dados	50
6.9. Apresentação da forma como as questões identificadas anteriormente podem ser satisfeitas com o novo sistema, utilizando a linguagem de interrogação do sistema <i>NoSQL</i>	54
8. Conclusões e Trabalho Futuro	58

Referências	60
Lista de Siglas e Acrónimos	61
Anexos	62
I. Anexo 1 – Script Completo de Criação	63
II. Anexo 2 – Script de Povoamento	67
III. Anexo 3 – Queries em SQL	78

Índice de Figuras

Figura 1 - Relacionamento Atleta - Teste Clínico	8
Figura 2 - Relacionamento Teste Clínico – Médico	9
Figura 3 - Relacionamento Médico - Clínica	10
Figura 4 - Relacionamento Atleta - Prova	11
Figura 5 - Modelo Conceptual	14
Figura 6 - Modelo Lógico	20
Figura 7 - Árvore representativa da Interrogação nº1	22
Figura 8 - Árvore representativa da Interrogação nº2	22
Figura 9 - Árvore representativa da Interrogação nº3	23
Figura 10 - Árvore representativa da Interrogação nº4	23
Figura 11 - Árvore representativa da Interrogação nº5	24
Figura 12 - Mapa de Transações para registar um atleta	25
Figura 13 - Tabela TesteClinico	25
Figura 14 - Mapa de Transações para adicionar uma prova	25
Figura 15 - Criação da tabela Atleta	31
Figura 16 - Criação da tabela TesteClinico	31
Figura 17 - Criação da Tabela Médico	32
Figura 18 - Criação da tabela Clínica	32
Figura 19 - Criação da tabela Contacto	32
Figura 20 - Criação da Tabela Prova_TipoTeste	33
Figura 21 - Criação da tabela Prova	33
Figura 22 - Criação da tabela Atleta_Prova	33
Figura 23 - Criação da tabela Morada	33
Figura 24 - Resolução da interrogação nº1	34
Figura 25 - Resolução da interrogação nº2	34
Figura 26 - Resolução da interrogação nº3	35
Figura 27 - Resolução da interrogação nº4	35
Figura 28 - Resolução da interrogação nº5	35
Figura 29 - Inserir um novo atleta na base de dados	36
Figura 30 - Inserir provas do atleta	36
Figura 31 - Inserir um agendamento na base de dados	37
Figura 32 - Inserir o registo de um novo tipo de prova	37
Figura 33 - Imagem do código obtido para o trigger	38
Figura 34 - View de agendamentos	42
Figura 35 – View dos exames realizados	42
Figura 36 – View dos medicos e as suas especialidades	43
Figura 37 - View dos dados dos pacientes	43

Figura 38 - Permissões dos diferentes utilizadores	44
Figura 39 - Estrutura Neo4j	47
Figura 40 - Ilustração da base de dados NoSQL criada em Neo4j	47
Figura 41 - Ilustração da base de dados NoSQL criada em Neo4j	48

Índice de Tabelas

Tabela 1 – Dicionário de dados das entidades	8
Tabela 2 - Dicionário de relacionamentos	11
Tabela 3 - Caraterização de todos os atributos existentes	12
Tabela 4 - Representação da entidade Atleta	16
Tabela 5 - Representação da entidade TesteClinico	17
Tabela 6 - Representação da entidade Medico	17
Tabela 7- Representação da entidade Clínica	17
Tabela 8 - Representação da entidade Prova	17
Tabela 9 - Espaço ocupado em disco por cada tipo de dados	39
Tabela 10 - Espaço ocupado pela tabela Atleta	39
Tabela 11 - Espaço ocupado pela tabela Clínica	39
Tabela 12 - Espaço ocupado pela tabela Médico	39
Tabela 13 - Espaço ocupado pela tabela Teste Clínico	40
Tabela 14 - Espaço ocupado pela tabela Prova	40
Tabela 15 - Espaço ocupado pela tabela Morada	40
Tabela 16 - Espaço ocupado pela tabela Contacto	40
Tabela 17 - Espaço ocupado pela tabela Atleta_Prova	40
Tabela 18 - Espaço ocupado pela tabela Prova_TipoTeste	41
Tabela 19 - Espaço ocupado em disco pela população atual	41
Tabela 20 - Espaço ocupado em disco pela população no futuro	42

1. Introdução

1.1. Contextualização

Antes de iniciar a prática de qualquer atividade física é importante realizar uma avaliação médica detalhada, especialmente quando um indivíduo deseja praticar desporto a nível competitivo, no qual terá que realizar um esforço físico intenso ou contínuo.

Atletas de alta competição devem realizar regularmente exames médicos como uma medida preventiva essencial à proteção da sua saúde, através da avaliação da sua pré-disposição a lesões e deste modo contribuir para reduzir o risco de ocorrência. Estes testes são também fundamentais para avaliar a evolução das suas diferentes capacidades de adaptação e resposta do aparelho cardiorrespiratório a um trabalho muscular do tipo aeróbio ou de resistência, necessário na maioria dos desportos de alta competição, mas especialmente no Atletismo.

Existe uma grande variedade de exames de alto rendimento para avaliar as capacidades e saúde do atleta que dependem da modalidade de competição.

É também necessário realizar, por vezes, testes para a deteção do uso de substâncias de melhoria de performance, cujo uso por desportistas de alta competição é proibido por lei.

Assim sendo, a *Clínica Alta Performance* nasceu com o objetivo de permitir a qualquer atleta de alta competição realizar todos os seus exames médicos periódicos com acompanhamento especializado face às suas necessidades desportivas. É obviamente uma prioridade empresarial automatizar todos os processos, manter um histórico da condição física e saúde dos atletas promovendo uma prática desportiva segura oferecendo orientação e preparação física de modo a que os pacientes possam obter o máximo rendimento das suas qualidades.

1.2. Apresentação do Caso de Estudo

A *Clínica Alta Performance* é uma empresa composta por um conjunto de clínicas que se encontram em 3 pontos do país. A clínica disponibiliza todo o tipo de exames médicos necessários para atestar a saúde de um atleta, para tal são contratados médicos de diferentes áreas clínicas e especializados em desporto. É de destacar o fator notável que distingue esta

empresa da sua concorrência, o seu acompanhamento personalizado, que permite aos atletas efetuarem todos os testes requeridos pelas provas em que pretendem participar e testes complementares para poderem obter um diagnóstico real do seu estado de saúde e das suas competências físicas de modo a poder alcançar os seus objetivos.

1.3. Fundamentação da Implementação da Base de Dados

Com o aumento da procura dos serviços da clínica verificou-se uma necessidade de evoluir o sistema tecnológico, pois o seu sistema atual baseava-se em folhas de cálculo. Este é método é rudimentar e bastante limitado. Sendo que a empresa pretende ser capaz de manipular e estudar os dados que vai recolhendo para melhor servir os seus clientes, é fundamental organizar e tratar estes dados de forma objetiva e eficiente. Para responder a esta necessidade, a melhor alternativa é a implementação de uma Base de Dados que facilite o armazenamento de toda a informação.

1.4. Análise da viabilidade do projeto

Este projeto consiste na implementação de uma Base de Dados Relacional que possibilita estudar e relacionar os dados que a empresa possui de modo a melhorar o processo de gestão de agendamentos e de dados dos pacientes. Possuir uma Base de Dados Relacional é essencial pois permite a manipulação e a extração de conhecimento de uma forma metodológica, podendo assim assegurar a consistência dos dados uma vez que, em larga escala, a necessidade de redundância desaparece. A nova base de dados, como será implementada num sistema de gestão de base de dados relacional baseado no uso de transações, permitirá efetuar inúmeras sequências de operações em simultâneo sem que a consistência e integridade dos dados seja sacrificada.

O investimento na criação de uma Base de Dados Relacional permitirá à empresa aumentar a qualidade do seu serviço, uma vez que permite uma maior facilidade no acesso à informação de um cliente de modo a conhecer as suas limitações físicas e todo o historial de doenças e lesões e sobretudo ser capaz de avaliar o estado de saúde do atleta antes de competir.

1.5. Estrutura do Relatório

O relatório do projeto é constituído por 5 capítulos.

O primeiro e atual capítulo é tratada a **Definição do sistema**. É apresentada a contextualização do problema, os objetivos e a motivação para a realização do projeto, incluindo também a análise da sua viabilidade.

No segundo capítulo, é feito o **Levantamento e Análise de Requisitos**, é apresentado o método utilizado e posteriormente descritos os vários tipos de requisitos e a sua análise.

No terceiro capítulo, é demonstrado todo o processo a realizar para a implementação da nossa **Base de Dados Relacional**. É apresentada a **Modelação Conceptual**, onde são identificadas as entidades, os seus relacionamentos e atributos. Seguidamente, é feita a **Modelação Lógica**, nomeadamente, a derivação de relações do modelo lógico e validação destas através da normalização e verificação das restrições de integridade. Por último é exposto o processo de **Implementação física**.

No quarto capítulo, **Base de Dados Não Relacional** tem como base a implementada no capítulo anterior. Para tal, demonstramos a realização da migração dos dados para o Neo4j, assim como a realização das queries definidas anteriormente.

Finalmente, é apresentada a **Conclusão** do projeto.

2. Levantamento e Análise de requisitos

2.1. Método de Levantamento e de Análise de Requisitos Adotados

Quanto à abordagem no levantamento dos requisitos, foram feitas entrevistas a médicos, rececionistas da Clínica Alta Performance e a atletas que são pacientes da mesma, de forma a identificar as diferentes dificuldades encontradas quando estes interagem com o sistema atual. Observou-se como a Clinica funciona em dias normais e como os funcionários interagem com o sistema atual percebendo assim como gerir e explorar o sistema.

Com a análise do sistema de informação utilizado, conseguiu-se obter os tipos de dados armazenados, a forma como esses vários tipos de dados se encontram conectados e tudo o que se deve ter em consideração para a migração de dados já existentes para o novo sistema.

Depois de levantados os requisitos, os mesmos foram categorizados em requisitos de descrição, controlo e de exploração.

2.2. Requisitos Levantados

2.2.1 Requisitos de Descrição

1. Quando um médico se regista no sistema é atribuído um ID sequencial pela aplicação e devem ser indicados os seguintes campos: o seu nome, a data com que iniciou a carreira, a data de nascimento, a especialidade e a sua morada;
2. Aquando do registo de um atleta é-lhe atribuído um ID sequencialmente e deverá ser indicado o seu nome, a data de nascimento, o sexo, a morada e um ou mais contactos telefónicos e o seu email;
3. Quando se agenda um teste clínico é atribuído um ID e deverá ser indicado a sua designação, a data e hora do teste e o preço;
4. Cada médico deverá estar associado obrigatoriamente a um chefe e a uma clínica;
5. Cada teste clínico deverá estar associado a um atleta;
6. Aquando do registo de uma prova deverá ser indicado a sua designação, a sua modalidade e a sua categoria e um ID gerado pelo sistema sequencialmente;
7. Para cada atleta são registadas as provas que realizou e a data destas;
8. O contacto terá de ser um número de telemóvel/telefone e um email;
9. A morada terá que ser um endereço e uma localidade;
10. Para cada teste clínico não poderá haver mais do que um atleta a realizá-lo;

11. Para cada prova não poderá haver mais do que um atleta igual, ou seja, não poderá haver atletas repetidos na lista de atletas da prova naquela data;
12. Após a realização do teste clínico o médico pode indicar o resultado do mesmo;

2.2.2 Requisitos de Exploração

1. Obter o número de pacientes de todas as clínicas;
2. Consultar os testes realizados antes de determinada prova para um dado atleta ordenados do mais recente para o mais antigo;
3. Consultar os nomes dos atletas de determinada modalidade que acusaram positivo nos testes de *doping*;
4. Consultar os nomes dos pacientes/atletas que foram submetidos a determinado teste;
5. Consultar o número de testes clínicos realizados por um dado médico num determinado ano e o montante faturado;
6. Consultar agendamentos para um determinado tipo de exame numa determinada clínica;
7. Obter o número de médicos de uma determinada especialidade numa determinada clínica;
8. Obter o top 3 atletas que gastaram mais dinheiro em exames;
9. Obter o top 5 atletas realizaram mais exames;
10. Obter o top 3 médicos de determinada especialidade que mais faturaram;
11. Obter a designação e resultado dos últimos 5 testes a que um atleta foi submetido;
12. Consultar a lista de exames obrigatórios para poder concorrer em determinada prova;
13. Consultar a designação das provas realizadas por determinado atleta;
14. Obter os atletas de uma determinada localidade;
15. Obter o número de testes clínicos realizados numa determinada clínica no ano 2019;

2.2.3 Requisitos de Controlo

1. O médico apenas poderá consultar os aspetos do sistema que lhe estão direta ou indiretamente associados;
2. O médico poderá inserir e modificar o resultado dos testes clínicos por ele realizados;
3. O médico poderá inserir, modificar e apagar tipos de testes aos quais determinado atleta terá que ser submetido realizar para que possa participar numa determinada prova;
4. A rececionista poderá agendar testes clínicos;
5. A rececionista poderá adicionar, editar e apagar atletas, testes clínicos agendados, contactos, moradas e provas de atletas do sistema;
6. O médico e a rececionista não poderão alterar propriedades do esquema da base de dados.

2.3. Análise de Requisitos

Após o levantamento e categorização dos requisitos foi necessária uma análise dos mesmos. Este processo incluiu várias discussões entre os elementos da nossa equipa para que desta forma fossem solucionados os conflitos entre os requisitos das diferentes entidades.

Com base na satisfação da utilização do novo sistema, nestas discussões foi abordado o grau de importância a dar aos diferentes níveis de detalhe em determinados aspectos do sistema a desenvolver.

Esta fase serviu principalmente para ajudar a solucionar complicações futuras relativas a eventuais diferenças nas interpretações dos requisitos do sistema de armazenamento de dados a desenvolver.

2.4. Caraterização dos perfis de utilização

Atendendo aos requisitos apresentados é possível identificar três perfis de utilização: o médico, o rececionista e por último o administrador que tem a possibilidade de fazer alterações diretas no sistema no que toca à atualização, inserção, consulta de dados e remoção de dados.

3. Modelação Conceptual

3.1. Apresentação da Abordagem de Modelação Realizada

Após o levantamento e análise de requisitos do sistema começa a fase do planeamento do design da Base de Dados. Para que a base de dados que será construída esteja de acordo com as necessidades dos seus utilizadores finais recorremos ao **diagrama ER** para efetuar a modelação conceptual. O objetivo do modelo conceptual é representar os dados presentes nos requisitos do utilizador.

O diagrama ER utiliza uma abordagem *top-down*, isto é, começamos por identificar os dados relevantes ao problema, as entidades, os relacionamentos entre esses dados, seguidamente completamos a informação do modelo com os seus atributos, quer nas entidades, que nos relacionamentos entre as mesmas e respetivos domínios. Finalmente, adicionamos a cardinalidade dos relacionamentos. Ao longo da sua construção é fundamental que o modelo seja testado e validado com o utilizador e com os seus requisitos, assegurando assim, a coerência e capacidade de resolução de problemas da futura base de dados.

Um aspeto muito importante desta metodologia é também o facto de ser completamente independente dos futuros detalhes de implementação.

3.2. Identificação e Caracterização das Entidades

Atendendo aos requisitos levantados foram identificadas as entidades apresentadas em seguida.

Atleta

O atleta é paciente da clínica que será submetido a diversos testes. Cada atleta possui atributos próprios, tem uma existência autónoma e pode ser identificado univocamente, sendo assim, uma entidade. É representado pelo seu **id, nome, sexo, data de nascimento, morada e contacto**.

Teste Clínico

O teste clínico refere-se a qualquer tipo de teste ou exame realizado pelo atleta. É representado pelo seu **id, designação (nome do exame), resultado, data e hora** em que foi ou será realizado, **preço, id do médico** que realizou ou realizará o exame e o **id do paciente** que se refere ao atleta sujeito ao teste.

Médico

A entidade Médico representa o funcionário responsável pela realização dos testes clínicos. É caracterizado pelo seu **id, nome, data de nascimento, a sua morada, data de início de serviço, especialidade, id da Clínica** em que trabalha, e ainda o **id do chefe**.

Clínica

A entidade Clínica representa cada uma das clínicas da empresa. É caracterizada pelo seu **id**, **nome** e **morada**.

Prova

A entidade prova refere-se às provas em que os atletas competem. É caracterizada pelo **id**, **designação** (nome da prova), **modalidade** (tipo de prova), **categoria**, que se refere ao intervalo de idades dos atletas para essa prova e ainda **tipoTesteClinico** que representa os vários tipos de testes clínicos que são obrigatórios para um atleta poder competir nesta.

3.2.1 Dicionário de dados das entidades

Entidade	Descrição	Alcunha	Ocorrência
Atleta	Entidade que recorre aos serviços da clínica	Paciente	O atleta é uma entidade fundamental uma vez que é este que recorre à clínica.
Teste Clínico	Entidade que contém toda a informação sobre um exame médico a que determinado atleta é submetido	Exame	O teste clínico é o motivo pelo qual os pacientes recorrem aos serviços da clínica.
Médico	Entidade que representa o responsável pela realização de testes clínicos	-	O médico é a base do funcionamento dos exames realizados na clínica.
Clínica	Entidade que representa o local em que um atleta é submetido a um determinado exame	-	A clínica é o local onde os pacientes são submetidos aos testes clínicos.
Prova	Entidade que contém a informação das provas em que o atleta compete	-	A prova é a atividade que conduz à necessidade de efetuar testes clínicos.

Tabela 1 – Dicionário de dados das entidades

3.3. Identificação e Caracterização dos Relacionamentos

Relacionamento: Atleta – Teste Clínico

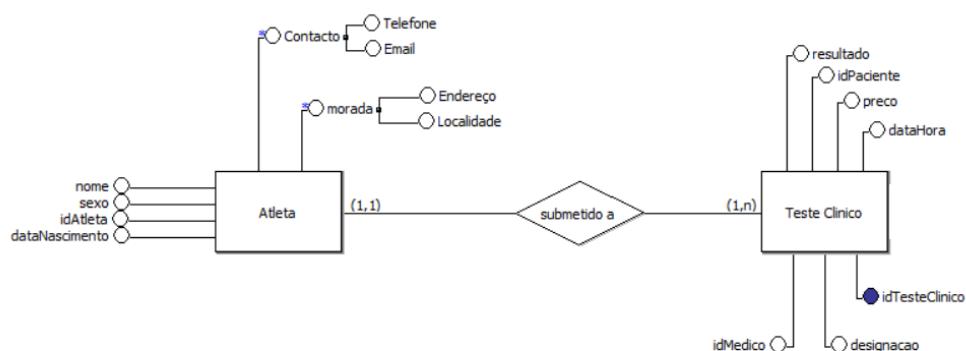


Figura 1 - Relacionamento Atleta - Teste Clínico

Relacionamento: Atleta é submetido a um Teste Clínico

Descrição: Os atletas recorrem à Clínica Alta Performance para realizar exames médicos. Assim, é importante preservar a informação dos testes médicos anteriormente realizados, para além de salvaguardar a informação de agendamentos futuros.

Cardinalidade: 1 para N

Um atleta tem que realizar pelo menos um teste clínico ou ter algum teste agendado para ser uma paciente da clínica. Contudo apenas um atleta pode realizar esse teste.

Atributos: Este relacionamento não possui atributos.

Relacionamento: Teste Clínico – Médico

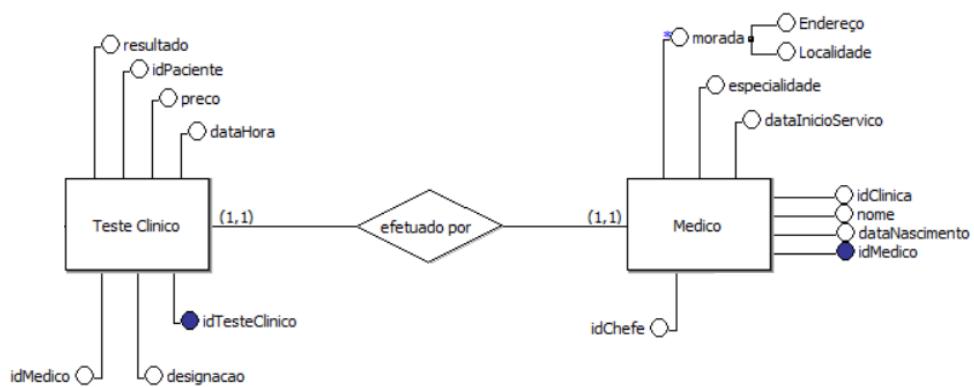


Figura 2 - Relacionamento Teste Clínico – Médico

Relacionamento: Teste Clínico é efetuado pelo Médico

Descrição: O medico é a entidade responsável pela supervisão e realização do exame do atleta. É importante ser capaz de identificar a identidade do medico que realizou determinado exame não só para o acompanhamento ser consistente e permitir que o paciente seja examinado pelo mesmo médico, mas também para em caso de negligencia poder apurar os responsáveis.

Cardinalidade: 1 para N

Um teste clínico é realizado somente por um médico. Um médico, por sua vez, pode realizar vários exames.

Atributos: Este relacionamento não possui atributos.

Relacionamento: Médico – Clínica

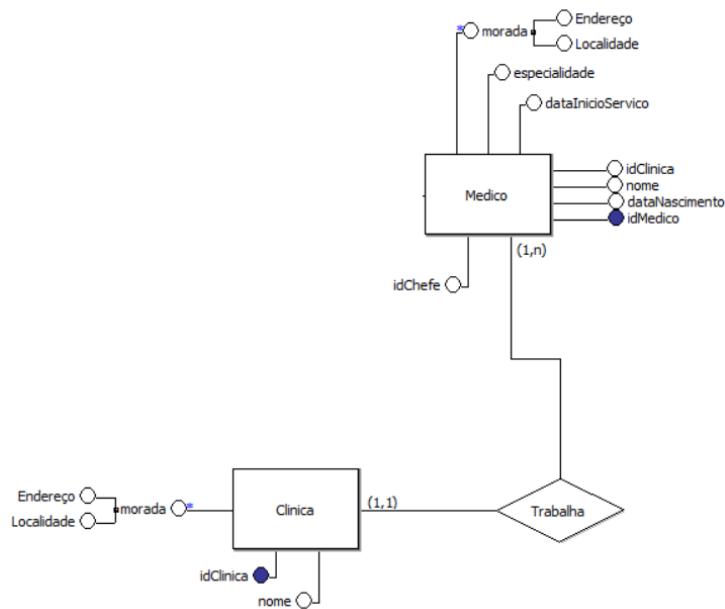


Figura 3 - Relacionamento Médico - Clínica

Relacionamento: Médico trabalha na Clínica

Descrição: O médico é a entidade que trabalha na clínica. Uma vez que a Clínica Alta Performance é uma empresa constituída por 3 clínicas diferentes é importante identificar o local de trabalho de cada médico.

Cardinalidade: N para 1

Vários médicos trabalham numa clínica, contudo cada médico apenas pode trabalhar numa clínica.

Atributos: Este relacionamento não possui atributos.

Relacionamento: Atleta – Prova

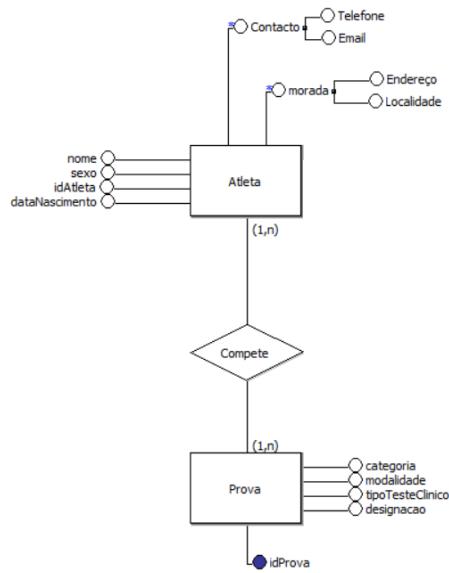


Figura 4 - Relacionamento Atleta - Prova

Relacionamento: Atleta compete numa Prova

Descrição: Um Atleta pode competir em várias provas, sendo esta a entidade que leva o atleta a recorrer aos serviços da clínica pois as provas exigem a realização de teste clínicos anteriormente.

Cardinalidade: N para N

Várias atletas podem competir em várias provas.

Atributos: Este relacionamento não possui atributos.

3.3.1 Dicionário de relacionamentos

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Atleta	1..N	Compete	1..N	Prova
Atleta	1..1	Submetido a	1..N	Teste
Teste	1..1	Realizado por	1..1	Médico
Médico	1..N	Trabalha	1..1	Clínica

Tabela 2 - Dicionário de relacionamentos

3.4. Identificação e Caracterização da Associação dos Atributos com as Entidades e Relacionamentos

ENTIDADE	ATRIBUTOS	DESCRIÇÃO	TIPO DO ATRIBUTO	TIPO DE DADOS E TAMANHO
Atleta	Id	Identificador do atleta	Chave primária	INT
	Nome	Nome do atleta	Simples	VARCHAR(100)
	Sexo	Género do atleta	Simples	VARCHAR(5)
	Data de Nascimento	Data de Nascimento do atleta	Simples	DATE
	Contacto:		Composto	
	Email	Email do atleta		VARCHAR(45)
	Telefone	Número de telefone do atleta		INT
	Morada:		Composto	
	Endereço	Endereço postal do atleta		VARCHAR(100)
	Localidade	Localidade do atleta		VARCHAR(100)
Teste Clínico	IdTesteClínico	Identificador do teste clínico	Chave primária	INT
	DataHora	Data e hora da realização do teste clínico	Simples	DATE
	Preço	Preço do exame	Simples	DECIMAL(6,2)
	Designação	Nome do exame	Simples	VARCHAR(100)
	IdAtleta	Identificador do atleta submetido ao exame	Simples	INT
	IdMedico	Identificador do médico que realiza o exame	Simples	INT
Médico	IdMédico	Identificador do médico	Chave primária	INT
	Nome	Nome do médico	Simples	VARCHAR(100)
	Data de Nascimento	Data de Nascimento do médico	Simples	DATE
	Morada		Composto	
	Endereço	Endereço postal do médico		VARCHAR(100)
	Localidade	Localidade do médico		VARCHAR(100)
	Especialidade	Área de especialização do médico	Simples	
Clínica	Data de Inicio de Serviço	Data de inicio de serviço do médico na clínica	Simples	DATE
	IdHospital	Identificador do hospital	Simples	INT
	IdClinica	Identificador da clínica	Chave primária	INT
Prova	Nome	Nome da Clínica	Simples	
	Morada		Composto	
	Endereço	Endereço postal da clínica		VARCHAR(100)
	Localidade	Localidade da clínica		VARCHAR(100)
	IdProva	Identificador da prova	Chave primária	INT
	tipoTesteClínico	Tipos de testes clínicos necessários para poder realizar a prova	Multivalorado	VARCHAR(100)
	Categoria	Intervalo de idades dos atletas que podem competir numa prova	Simples	VARCHAR(100)
	Designação	Nome da prova	Simples	VARCHAR(100)
	Modalidade	Tipo de prova	Simples	VARCHAR(100)

Tabela 3 - Caraterização de todos os atributos existentes

3.5. Chaves Candidatas, Primárias e Alternativas

Uma vez identificadas as entidades, os seus atributos e relacionamentos, podemos agora verificar quais desses atributos poderão ser chaves primárias. É importante referir que uma chave primária corresponde a um atributo, que pode identificar a identidade da entidade inequivocamente. As chaves candidatas são todos os atributos da entidade que a identificam univocamente e os que não forem escolhidos para chave primária denominam-se chaves alternativas.

Atleta

Chaves candidatas: IdAtleta, Contacto

Chave Primária: IdAtleta

Chave Alternativa: Contacto

Teste

Chaves candidatas: IdTesteClínico

Chave Primária: IdTesteClinico

Chave Alternativa: Nenhuma

Médico

Chaves candidatas: IdMédico

Chave Primária: IdMédico

Chave Alternativa: Nenhuma

Clínica

Chaves candidatas: IdClínica, Morada

Chave Primária: IdClínica

Chave Alternativa: Morada

Prova

Chaves candidatas: IdProva

Chave Primária: IdProva

Chave Alternativa: Nenhuma

3.6. Detalhe ou Generalização de Entidades

No nosso projeto não utilizamos generalização nem especialização de entidades no Modelo Conceptual.

3.7. Apresentação e Explicação do Diagrama ER

Depois de explicada a importância e o significado de cada uma das entidades existentes no nosso sistema, cada um dos relacionamentos entre elas e os respetivos atributos, tanto das entidades, como dos relacionamentos, apresentamos o modelo conceptual:

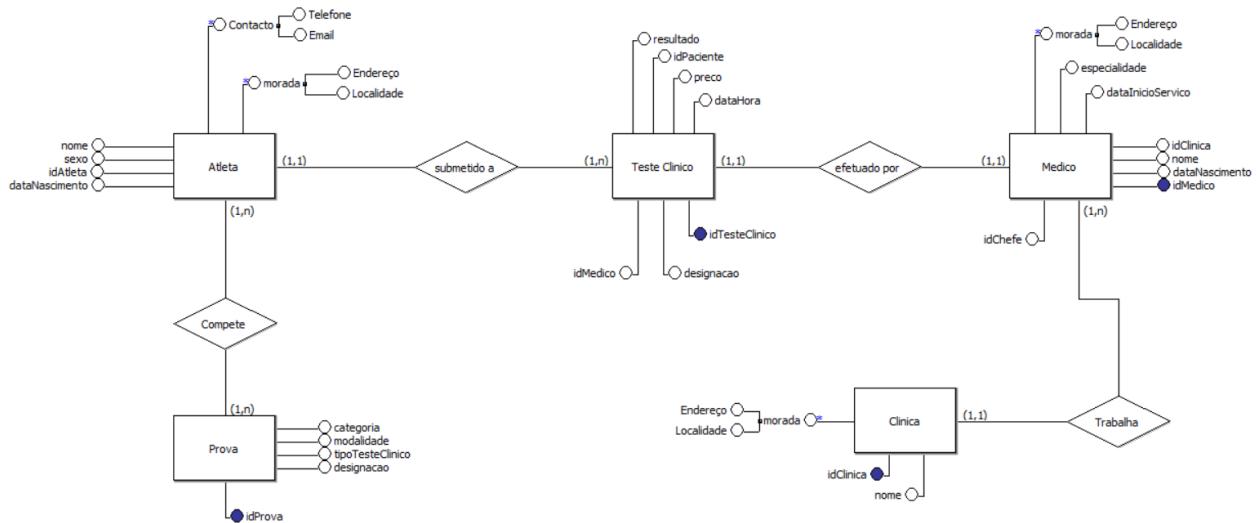


Figura 5 - Modelo Conceptual

O funcionamento do nosso sistema começa a ganhar textura. Traduzindo o Modelo Conceptual ilustrado anteriormente para texto temos que: O Atleta que compete em provas é submetido a Testes Clínicos. Os Testes Clínicos são efetuados por um Médico que trabalha numa Clínica.

3.8. Validação do Modelo de Dados com o Utilizador

Uma vez terminado o Modelo Conceptual, é necessário efetuar a sua validação. Para tal, é fundamental ser possível responder a todas as perguntas que o utilizador possa elaborar. Assim, escolhemos as queries posteriormente expostas, de modo a verificar a viabilidade com o Modelo Conceptual.

1. **Consultar os testes realizados antes de determinada prova para um dado atleta ordenados do mais recente para o mais antigo (Requisito de Exploração 2)**

Para obter a informação necessária para a resposta recorremos às entidades **Prova**, **Atleta** e **Teste Clínico**. Tendo em conta o relacionamento entre o **Atleta** e a **Prova**, temos acesso a todas as provas em que o atleta competiu e, por conseguinte, à sua data. Através do relacionamento entre o **Atleta** e o **Teste Clínico** obtemos todos os exames a

que o atleta foi submetido e as suas datas. Assim é possível determinar quais os exames realizados antes da data da prova.

2. Consultar os nomes dos atletas de determinada modalidade que acusaram positivo nos testes de doping (Requisito de Exploração 3)

Através das entidades *Atleta*, *Prova* e *Teste Clínico* conseguimos dar resposta a esta interrogação. Do relacionamento *Atleta* e *Prova* obtemos o nome e modalidade em que os atletas concorrem e da relação entre *Atleta* e *Teste Clínico* conseguimos descobrir se o resultado dos exames de *doping* realizados alguma vez deu positivo.

3. Consultar os nomes dos pacientes/atletas que foram submetidos a determinado teste (Requisito de Exploração 4)

Através das entidades *Teste Clínico* e *Atleta* conseguimos identificar todos os atletas submetidos a determinado teste.

4. Consultar o número de testes clínicos realizados por um dado médico num determinado ano e o montante faturado (Requisito de Exploração 5)

Através das entidades *Teste Clínico* e *Medico* é possível determinar o médico que realizou cada exame, a sua data e preço.

5. Consultar agendamentos para um determinado tipo de exame numa determinada clínica (Requisito de Exploração 6)

Através da entidade *Médico* conseguimos determinar a clínica em que determinado médico trabalha, podendo assim determinar através da junção deste com a entidade *TesteClinico* os tipos de exames efetuados.

4. Modelação Lógica

Terminada a conceptualização do problema apresentado através do modelo anteriormente elaborado, torna-se agora necessária a construção do mesmo num Modelo Lógico. Esta será uma etapa fundamental para o desenvolvimento do nosso sistema de gestão da base de dados uma vez que nos permitirá derivar os relacionamentos. Assim, quando o modelo lógico estiver validado, tendo em conta as formas normais, este irá suportar o nosso problema.

4.1. Construção e Validação do Modelo de Dados Lógico

Para a elaboração do Modelo Lógico, começamos por derivar/criar todas as relações que retratam as entidades, os relacionamentos e os atributos previamente identificados no Modelo Conceptual.

4.1.1 Entidades fortes

Uma entidade forte identifica-se pela posse de uma de uma chave primária que a identifica e pela independência de outras entidades. No Modelo Logico, cada entidade forte será representada numa tabela, em que cada um dos seus atributos constitui uma coluna. Para os atributos compostos, na tabela apenas serão apresentados os atributos simples que o constituem.

Entidades fortes resultantes:

Atleta (idAtleta, nome, dataNascimento, sexo, morada)

Chave primária: idAtleta

idAtleta	Nome	Data de Nascimento	Sexo	Morada
1	Maria José Borges Pires	27-10-1998	F	123
(...)	(...)	(...)	(...)	(...)

Tabela 4 - Representação da entidade Atleta

TesteClinico (idTesteClinico, designação, dataHora, preço, resultado, idMedico, idAtleta)

Chave primária: idTesteClinico

idTesteClinico	Designação	Data e Hora	Preço	Resultado	idMedico	idAtleta
1123	Eletrocardiograma	10-12-19 10h30	50,00	Ok	250	12
(...)	(...)	(...)	(...)	(...)	(...)	(...)

Tabela 5 - Representação da entidade TesteClinico

Médico (idMedico, nome, dataNascimento, morada, especialidade, dataInicioServico, idClinica, idChefe)

Chave Primária: idMedico

idMedico	Nome	Data de Nascimento	Morada	Especialidade	Data de Inicio de Serviço	idClinica	idChefe
201	Susana Marques	01-04-1986	987	Cardiologia	03-10-2002	2	100
(...)	(...)	(...)	(...)	(...)	(...)	(...)	(...)

Tabela 6 - Representação da entidade Medico

Clínica (idClinica, nome, morada)

Chave primária: idClinica

idClinica	Nome	morada
1	Clínica Alta Performance - Braga	111
(...)	(...)	(...)

Tabela 7- Representação da entidade Clínica

Prova (idProva, designação, modalidade, categoria)

Chave primária: idProva

idProva	Designação	Modalidade	Categoria
1	Corrida de Obstáculos: 100m	Corrida de Obstáculos	18-21
(...)	(...)	(...)	(...)

Tabela 8 - Representação da entidade Prova

4.1.2 Relacionamentos 1:N

Neste tipo de relacionamento, a entidade que apresenta multiplicidade N possui como atributo a chave primária da entidade com multiplicidade 1. Este atributo é considerado uma chave estrangeira.

TesteClinico (idTesteClinico, designação, data, hora, preço, resultado, idMedico, idAtleta)

Chave primária: idTesteClinico

Chaves estrangeiras: idMedico, idAtleta

Médico (idMedico, nome, dataNascimento, morada, especialidade, dataInicioServico, idClinica, idChefe)

Chave primária: idMedico

Chaves estrangeiras: idClinica, morada

Como os atletas, médicos e cada clínica possuem uma morada, o atributo morada irá repetir-se algumas vezes, pelo que sentimos a necessidade de criar uma tabela *Morada*. A nova tabela terá como atributos o endereço, a localidade e o idMorada, este último corresponde à chave primária e que será uma chave estrangeira em Médico, como visto anteriormente e nas tabelas *Clinica* e *Atleta*, como se pode verificar de seguida.

Clínica (idClinica, nome, morada)

Chave primária: idClinica

Chaves estrangeiras: morada

Atleta (idAtleta, nome, dataNascimento, sexo, morada)

Chave primária: idAtleta

Chaves estrangeiras: morada

4.1.3 Relacionamentos N:M

Este tipo de multiplicidade gera um novo relacionamento, onde a chave primária é composta pelas chaves primárias de cada uma das entidades envolvidas.

Atleta_Prova (id_Atleta, id_Prova, data)

Chave primária: id_Atleta, id_Prova

4.1.4 Atributos Multivalorados

Este tipo de atributo é utilizado, quando para a mesma entidade, o atributo pode assumir múltiplos valores. Um atributo multivalorado cria um novo relacionamento, em que a chave primária é composta pela chave primária da entidade e o próprio atributo.

Contacto (idMorada, telefone, email, idAlteta)

Chave primária: idMorada

Chaves estrangeiras: idAtleta

Prova_TipoTeste (idProva, designação)

Chave primária: Não possui uma chave primária, a sua existência depende da entidade Prova

Chaves estrangeiras: idProva

4.2. Desenho do Modelo Lógico

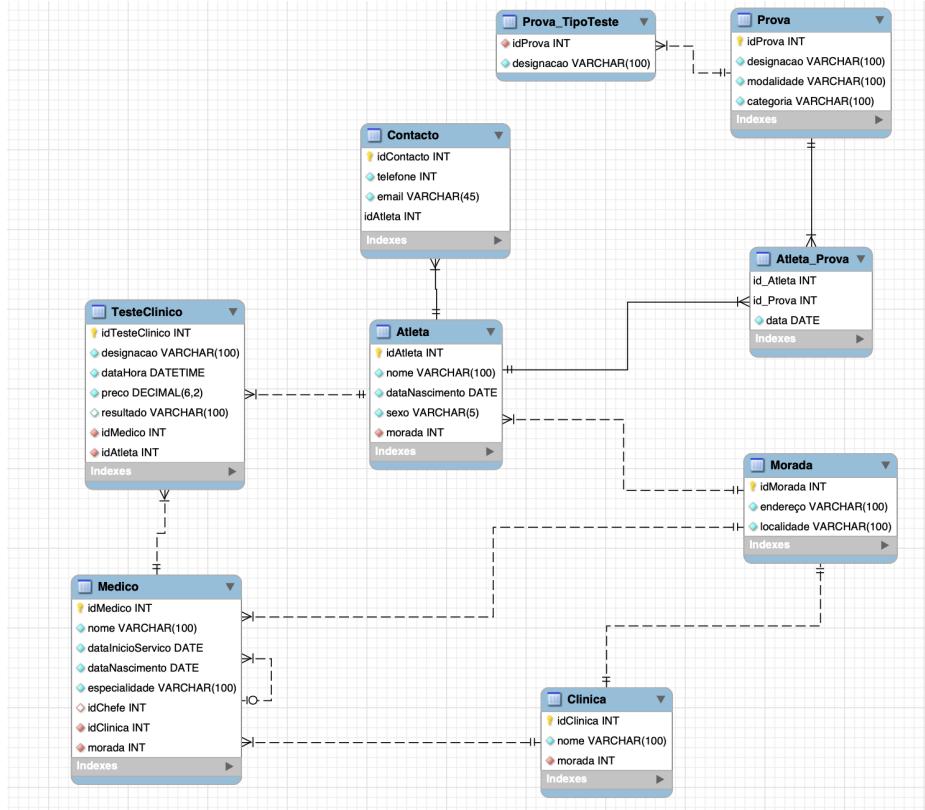


Figura 6 - Modelo Lógico

4.3. Validação do Modelo através da Normalização

No modelo relacional de uma base de dados o objetivo é agrupar os atributos em relações, uma vez que existem dependências entre estes. Assim, é possível construir um modelo lógico consistente em termos estruturais e com a menor redundância possível. Uma vez que a normalização tem como objetivo garantir que as relações têm um número mínimo, mas suficiente de atributos necessários para suportar as exigências relativas aos dados a guardar, podemos perceber melhor cada atributo e o que este representa na base de dados. Para proceder à validação do modelo através de normalização, começamos por identificar as dependências funcionais entre os atributos.

4.3.1 Primeira Forma Normal (1FN)

Um relacionamento encontra-se na Primeira Forma Normal se todos os atributos forem atómicos, isto é, não ser possível decompô-los em subconjuntos de atributos e não existirem diferentes atributos que descrevem o mesmo. No nosso esquema conceptual temos atributos compostos e multivalorados, os atributos Morada e Contacto. Assim, nas tabelas exclui-se a

Morada e o *Contacto*, de modo a não violar a primeira forma normal. Podemos, assim, concluir que o nosso modelo está de acordo com a 1FN.

4.3.2 Segunda Forma Normal (2FN)

Uma relação está na Segunda Forma Normal se pertencer à 1FN e se todos os atributos não chave dependerem totalmente da chave primária. Na nossa base de dados não existe qualquer dependência funcional total em relação à chave primária. Assim, o nosso modelo está de acordo com a 2FN.

4.3.3 Terceira Forma Normal (3FN)

Um relacionamento encontra-se na Terceira Forma Normal se está na 2FN e se todos os atributos não chave apenas dependerem totalmente da chave primária e não dependerem de qualquer outro atributo que por sua vez dependesse da chave primária. Como podemos verificar o nosso modelo não apresenta dependências transitivas então está de acordo com a 3FN.

4.4. Validação do Modelo com as Interrogações do Utilizador

1. Consultar os testes realizados antes de determinada prova para um dado atleta, ordenados do mais recente para o mais antigo (Requisito de exploração 2)

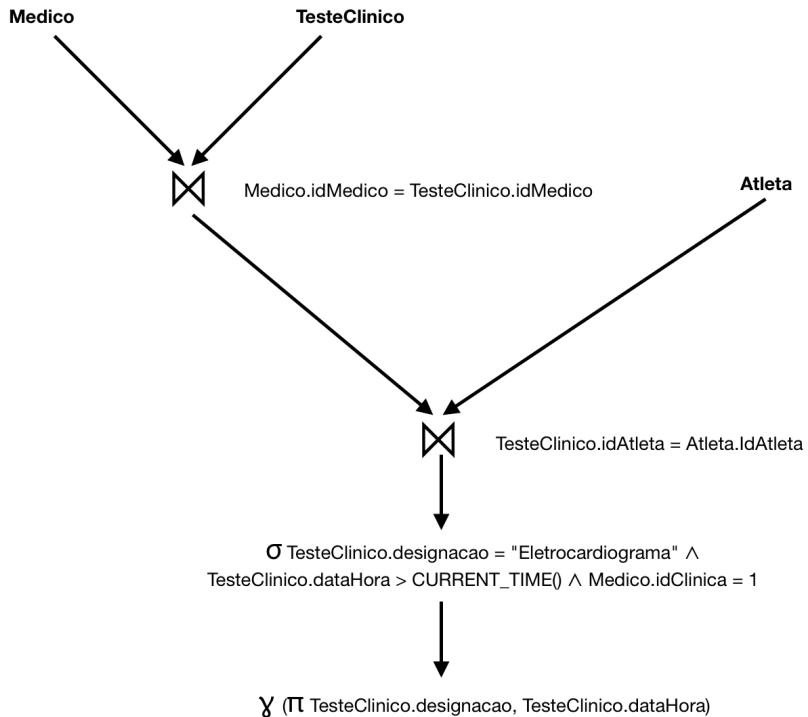


Figura 7 - Árvore representativa da Interrogação nº1

2. Consultar os nomes dos atletas de determinada modalidade que acusaram positivo nos testes de doping (Requisito de exploração 3)

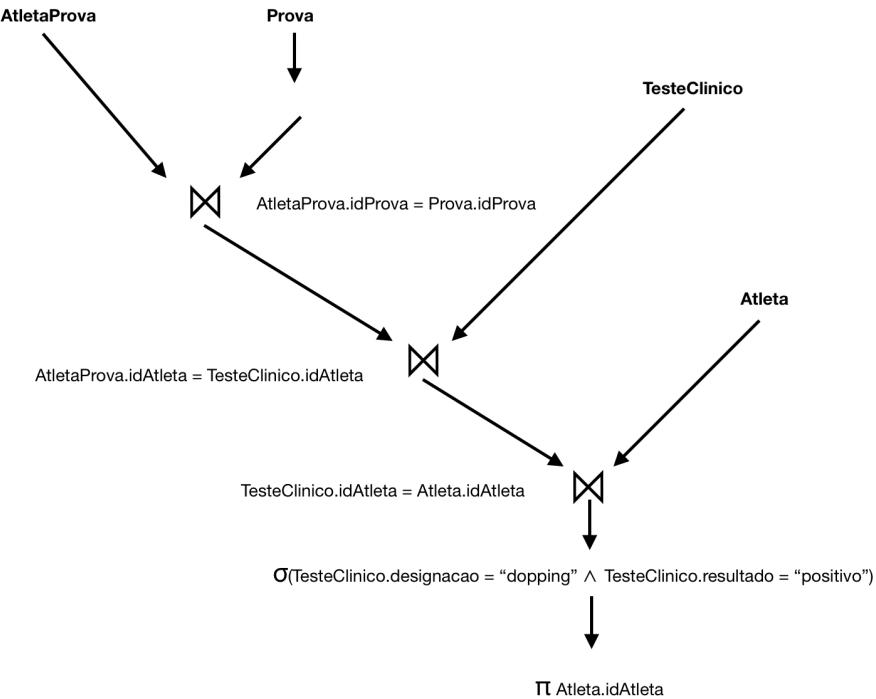


Figura 8 - Árvore representativa da Interrogação nº2

- 3. Consultar os nomes dos pacientes/atletas que foram submetidos a determinado teste (Requisito de exploração 4)**

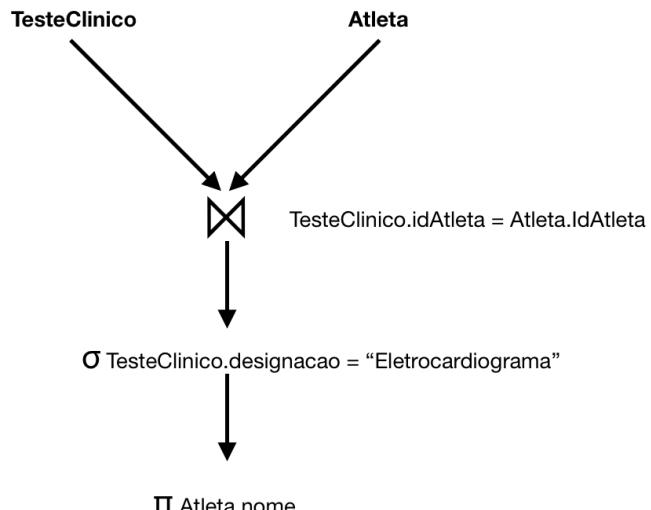


Figura 9 - Árvore representativa da Interrogação nº3

- 4. Consultar o número de testes clínicos realizados por um dado médico num determinado ano e o montante faturado (Requisito de exploração 5)**

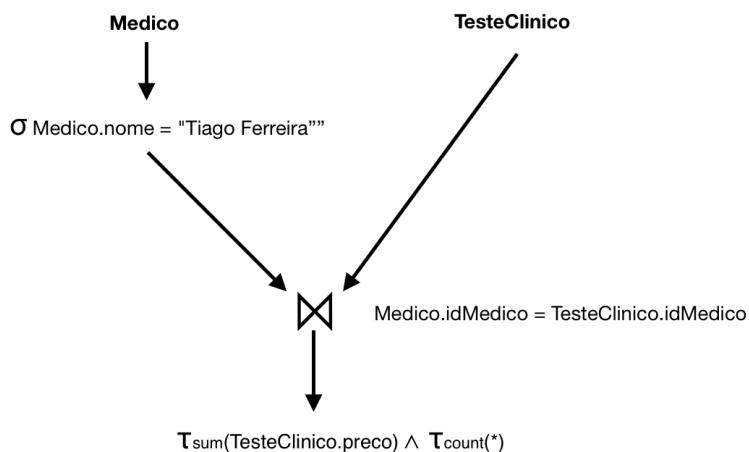


Figura 10 - Árvore representativa da Interrogação nº4

- 5. Consultar agendamentos para um determinado tipo de exame numa determinada clínica (Requisito de exploração 6)**

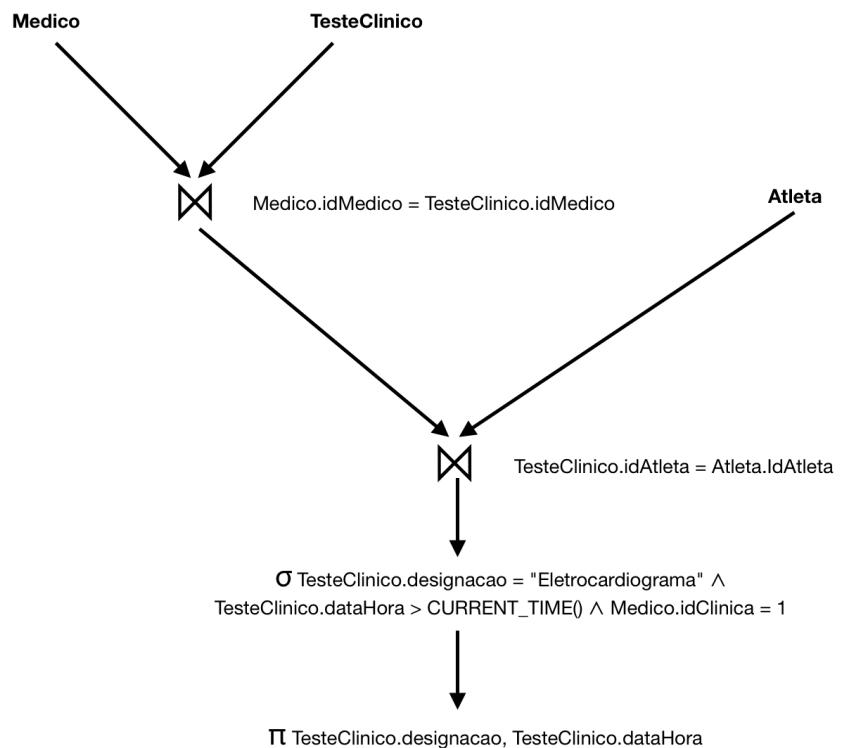


Figura 11 - Árvore representativa da Interrogação nº5

4.5. Validação do Modelo com as Transações Estabelecidas

Foram estabelecidas algumas transações que serão posteriormente utilizadas pelos utilizadores (com as devidas permissões) da base de dados. Por se tratarem de transações, fica assegurado que cada uma destas é interpretada pelo sistema de base de dados como um bloco linear e atómico, pois as operações dentro de cada transação serão efetuadas sequencialmente nos dados. Deste modo, fica garantido que as tabelas envolvidas na transação não serão editadas entre cada operação no bloco da transação por uma outra operação que interaja com esses mesmos dados.

- **Registrar um atleta:** Para registrar um atleta começa-se por elaborar um novo registo na tabela *Atleta*, adicionando a morada na tabela *Morada* caso essa ainda não exista. De seguida, é necessário inserir os contactos telefónicos e email do mesmo, efetuando um novo registo na tabela *Contactos*. Finalmente, é preenchida a tabela *Atleta_Prova*, com as provas futuras que levaram o atleta a recorrer aos serviços da clínica.

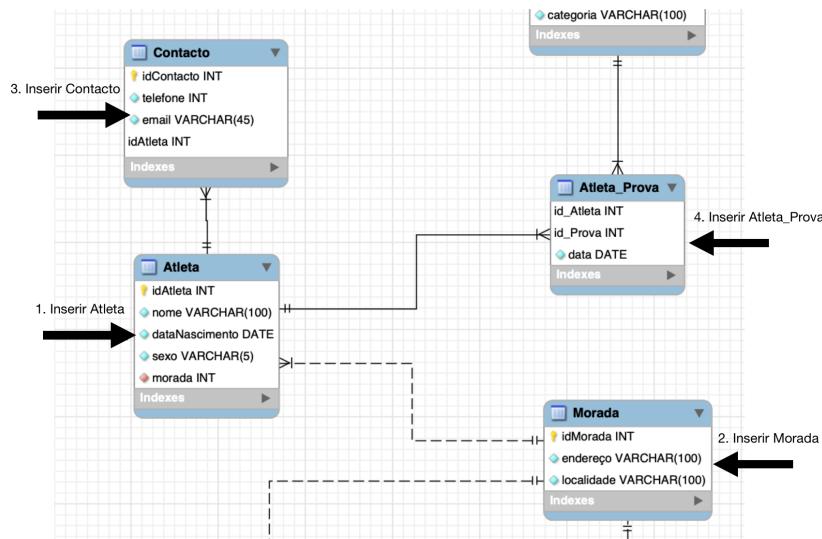


Figura 12 - Mapa de Transações para registar um atleta

- Efetuar um agendamento:** Para efetuar um agendamento é necessário adicionar uma nova entrada na tabela *TesteClinico*. Esta entrada possui uma data futura e o resultado do teste é *null*.

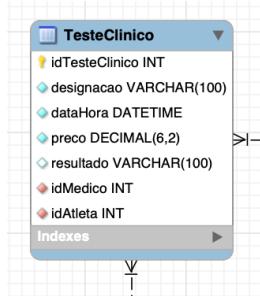


Figura 13 - Tabela TesteClinico

- Adicionar uma prova:** Para adicionar um novo tipo de prova ao sistema, adiciona-se primeiro uma entrada na tabela *Prova* e de seguida são registados os exames obrigatórios para competir nessa prova na tabela *Prova_TipoTeste*.

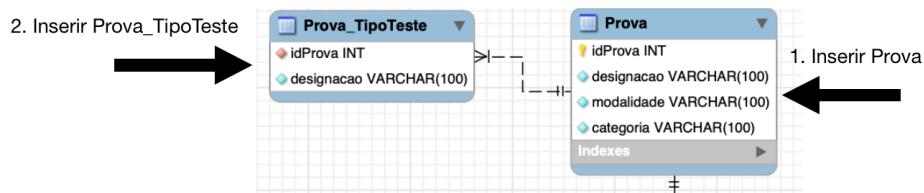


Figura 14 - Mapa de Transações para adicionar uma prova

4.6. Verificação das Restrições de Integralidade

Para garantir que o modelo lógico representa os dados de forma fiável, é necessário a aplicar algumas regras, nomeadamente, as restrições de integridade, sendo possível assim garantir, que a base de dados é consistente e completa. As restrições aplicadas são as seguintes:

Dados Necessários: Numa Base de Dados existem campos de preenchimento obrigatório e que devem possuir um valor válido, como é o caso das chaves primárias. No SGBD que estamos a implementar, todos os campos são de preenchimento obrigatório.

Integridade das Entidades: Para que todas as entidades sejam identificadas e que não exista repetição das mesmas, é necessário garantir a existência de uma chave primária, tendo esta de ser única e não nula. Como as chaves primárias definidas no Modelo Conceptual transitam para o modelo lógico, é possível verificar as que identificam cada entidade na seção 3.5.

Integridade do Domínio: Esta restrição refere-se aos atributos, dado que todos possuem um domínio de valores admissíveis. É possível observar o tipo de dados de cada atributo no dicionário de dados dos atributos das entidades e relacionamentos.

Integridade Referencial: Esta restrição tem por base a relação entre chaves estrangeiras e primárias, de modo a salvaguardar os relacionamentos entre as tabelas quando é efetuado uma inserção, atualização ou remoção de um registo. Esta garante que os dados permanecem consistentes em todas as tabelas.

Restrições da Multiplicidade: As multiplicidades são definidas na criação do modelo conceitual, e como tal devem transitar para o modelo lógico. Contudo, existem casos particulares como os atributos multivvalorados que devem ser analisados, como é retratado na seção 4.1, onde é possível encontrar os relacionamentos estabelecidos.

4.7. Revisão do Modelo Lógico com o Utilizador

Após concluído o desenvolvimento do modelo lógico foi agendada nova reunião com a *Clínica Alta Performance* de modo a apresentar todo o trabalho desenvolvido e obter autorização para avançar para a próxima fase do projeto. Uma vez que este modelo é significativamente mais complexo que o modelo conceptual, o primeiro passo foi apresentar em traços gerais como analisar e interpretar este tipo de modelos, para que o cliente possa, de maneira fiável, confirmar que o modelo atende às necessidades da empresa. Cumpridas todas as exigências do cliente, foi dada autorização para avançar para a próxima fase do projeto.

5. Implementação Física

5.1. Seleção do Sistema de Gestão de Base de Dados

O sistema de gestão de bases de dados utilizado para o nosso projeto foi o *MySQL*. A nossa escolha baseou-se essencialmente no facto de este ser de fácil utilização, possuir um bom desempenho e ser *open-source*. Adicionalmente, o facto de o sistema de armazenamento e gestão de transações do *MySQL* seguir os princípios ACID (Atomicidade, Consistência, Isolamento e Durabilidade) também contribuiu positivamente para a nossa escolha.

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

Durante todo o processo decidimos usar a ferramenta de desenho, desenvolvimento e administração de base de dados, *MySQL Workbench*. Para gerar o modelo físico através do modelo lógico recorremos à ferramenta *Forward Engineer*. Este mecanismo, tal como o seu nome elucida permite automatizar o processo *top-down* de construção de componentes de baixo nível, neste caso a implementação no sistema de gestão de base de dados, através de uma abstração de alto nível, o modelo lógico. Tendo obtido o código resultante deste processo apenas tivemos que o executar para que a base de dados fosse criada.

5.2.1 Desenho das relações base

Relação Atleta

Domínio IdAtleta: Inteiro

Domínio nome: String de tamanho variável

Domínio dataNascimento: Data

Domínio sexo: String de tamanho fixo

Domínio morada: Inteiro

Atleta(

idAtleta	INT	NOT NULL,
nome	VARCHAR(100)	NOT NULL,
dataNascimento	DATE	NOT NULL,
sexo	VARCHAR(5)	NOT NULL,
morada	INT	NOT NULL,

PRIMARY KEY (idAtleta),

FOREIGN KEY (morada)

REFERENCES Morada(idMorada));

Relação Médico

Domínio idMedico: Inteiro
Domínio nome: String de tamanho variável
Domínio dataInicioServico: data
Domínio dataNascimento: data
Domínio especialidade: String de tamanho variável
Domínio idChefe: Inteiro
Domínio idClinica: Inteiro
Domínio morada: Inteiro

Medico(

idMedico	INT	NOT NULL,
nome	VARCHAR(100)	NOT NULL,
dataInicioServico	DATE	NOT NULL,
dataNascimento	DATE	NOT NULL,
especialidade	VARCHAR(100)	NOT NULL,
idChefe	INT	NULL,
idClinica	INT	NOT NULL,
morada	INT	NOT NULL,

PRIMARY KEY(idMedico),

FOREIGN KEY(idChefe) **REFERENCES** Medico (idMedico)

FOREIGN KEY(idClinica) **REFERENCES** Clinica(idClinica)

FOREIGN KEY(morada) **REFERENCES** Morada(idMorada));

Relação TesteClinico

Domínio idTesteClinico: Inteiro
Domínio designação: String de tamanho variável
Domínio dataHora: Data e hora
Domínio preco: Número decimal
Domínio resultado: String de tamanho variável
Domínio idMedico: Inteiro
Domínio idAtleta: Inteiro

TesteClinico(

idTesteClinico	INT	NOT NULL,
designacao	VARCHAR(100)	NOT NULL,
dataHora	DATETIME	NOT NULL,
preco	DECIMAL(6,2)	NOT NULL,

```

resultado      VARCHAR(100)  NULL,
idMedico       INT          NOT NULL,
idAtleta        INT          NOT NULL,
PRIMARY KEY(idTesteClinico),
FOREIGN KEY(idMedico) REFERENCES Medico(idMedico)
FOREIGN KEY(idAtleta) REFERENCES Atleta(idAtleta));

```

Relação Clínica

Domínio idClinica: Inteiro

Domínio nome: String de tamanho variável

Domínio morada: Inteiro

Clinica(

```

idClinica      INT          NOT NULL,
nome           VARCHAR(100) NOT NULL,
morada         INT          NOT NULL,
PRIMARY KEY (idClinica),
FOREIGN KEY (morada) REFERENCES Morada(idMorada));

```

Relação Morada

Domínio idMorada: Inteiro

Domínio endereço: String de tamanho variável

Domínio localidade: String de tamanho variável

Morada(

```

idMorada       INT          NOT NULL,
endereço      VARCHAR(100) NOT NULL,
localidade    VARCHAR(100) NOT NULL,
PRIMARY KEY (idMorada));

```

Relação Prova

Domínio idProva: Inteiro

Domínio designação: String de tamanho variável

Domínio modalidade: String de tamanho variável

Domínio categoria: String de tamanho variável

Prova(

```

idProva        INT          NOT NULL,
designacao    VARCHAR(100) NOT NULL,
modalidade    VARCHAR(100) NOT NULL,
categoria     VARCHAR(100) NOT NULL,

```

PRIMARY KEY (idProva);

Relação Atleta_Prova

Domínio idAtleta: Inteiro

Domínio idProva: Inteiro

Domínio data: Data

Atleta_Prova(

 id_Atleta INT NOT NULL,

 id_Prova INT NOT NULL,

 data DATE NOT NULL,

 PRIMARY KEY (id_Prova, id_Atleta),

FOREIGN KEY (id_Atleta) **REFERENCES** Atleta (idAtleta)

FOREIGN KEY (id_Prova) **REFERENCES** Prova(idProva));

Relação Prova_TipoTeste

Domínio idProva: inteiro

Domínio designacão: String de tamanho variável

Prova_TipoTeste(

 idProva INT NOT NULL,

 designacao VARCHAR(100) NOT NULL,

FOREIGN KEY (idProva) **REFERENCES** Prova(idProva));

Relação Contacto

Domínio idContacto: Inteiro

Domínio telefone: Inteiro

Domínio email: String de tamanho variável

Domínio idAtleta: Inteiro

Contacto(

 idContacto INT NOT NULL,

 telefone INT NOT NULL,

 email VARCHAR(45) NOT NULL,

 idAtleta INT NOT NULL,

 PRIMARY KEY (idContacto, idAtleta),

FOREIGN KEY (idAtleta) **REFERENCES** Atleta(idAtleta));

5.2.2 Desenho das restrições

Nesta parte serão expostas as restrições gerais do problema agregadas por relação.

```
-- -----
-- Table `ClinicaAltaPerformance`.`Atleta`
-----
CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Atleta` (
    `idAtleta` INT NOT NULL AUTO_INCREMENT,
    `nome` VARCHAR(100) NOT NULL,
    `dataNascimento` DATE NOT NULL,
    `sexo` VARCHAR(5) NOT NULL,
    `morada` INT NOT NULL,
    PRIMARY KEY (`idAtleta`),
    INDEX `fk_Atleta_Morada_idx` (`morada` ASC) VISIBLE,
    CONSTRAINT `fk_Atleta_Morada`
        FOREIGN KEY (`morada`)
            REFERENCES `ClinicaAltaPerformance`.`Morada` (`idMorada`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 15 - Criação da tabela Atleta

```
-- -----
-- Table `ClinicaAltaPerformance`.`TesteClinico`
-----
CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`TesteClinico` (
    `idTesteClinico` INT NOT NULL AUTO_INCREMENT,
    `designacao` VARCHAR(100) NOT NULL,
    `dataHora` DATETIME NOT NULL,
    `preco` DECIMAL(6,2) NOT NULL,
    `resultado` VARCHAR(100) NULL,
    `idMedico` INT NOT NULL,
    `idAtleta` INT NOT NULL,
    PRIMARY KEY (`idTesteClinico`),
    INDEX `fk_TestClinico_Tecnico_idx` (`idMedico` ASC) VISIBLE,
    INDEX `fk_TestClinico_Atleta_idx` (`idAtleta` ASC) VISIBLE,
    CONSTRAINT `fk_TestClinico_Tecnico`
        FOREIGN KEY (`idMedico`)
            REFERENCES `ClinicaAltaPerformance`.`Medico` (`idMedico`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION,
    CONSTRAINT `fk_TestClinico_Atleta`
        FOREIGN KEY (`idAtleta`)
            REFERENCES `ClinicaAltaPerformance`.`Atleta` (`idAtleta`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 16 - Criação da tabela TesteClinico

```

-- Table `ClinicaAltaPerformance`.`Medico`
CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Medico` (
  `idMedico` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(100) NOT NULL,
  `dataInicioServico` DATE NOT NULL,
  `dataNascimento` DATE NOT NULL,
  `especialidade` VARCHAR(100) NOT NULL,
  `idChefe` INT NULL,
  `idClinica` INT NOT NULL,
  `morada` INT NOT NULL,
  PRIMARY KEY (`idMedico`),
  INDEX `fk_Tecnico_Hospital_idx` (`idClinica` ASC) VISIBLE,
  INDEX `fk_Medico_Chefe_idx` (`idChefe` ASC) VISIBLE,
  INDEX `fk_Medico_Morada1_idx` (`morada` ASC) VISIBLE,
  CONSTRAINT `fk_Medico_Chefe`
    FOREIGN KEY (`idChefe`)
      REFERENCES `ClinicaAltaPerformance`.`Medico` (`idMedico`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Tecnico_Hospital`
    FOREIGN KEY (`idClinica`)
      REFERENCES `ClinicaAltaPerformance`.`Clinica` (`idClinica`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Medico_Morada1`
    FOREIGN KEY (`morada`)
      REFERENCES `ClinicaAltaPerformance`.`Morada` (`idMorada`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 17 - Criação da Tabela Médico

```

-- Table `ClinicaAltaPerformance`.`Clinica`
CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Clinica` (
  `idClinica` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(100) NOT NULL,
  `morada` INT NOT NULL,
  PRIMARY KEY (`idClinica`),
  INDEX `fk_Clinica_Morada1_idx` (`morada` ASC) VISIBLE,
  CONSTRAINT `fk_Clinica_Morada1`
    FOREIGN KEY (`morada`)
      REFERENCES `ClinicaAltaPerformance`.`Morada` (`idMorada`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 18 - Criação da tabela Clínica

```

-- Table `ClinicaAltaPerformance`.`Contacto`
CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Contacto` (
  `idContacto` INT NOT NULL AUTO_INCREMENT,
  `telefone` INT NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `idAtleta` INT NOT NULL,
  PRIMARY KEY (`idContacto`, `idAtleta`),
  INDEX `fk_Contacto_Atleta1_idx` (`idAtleta` ASC) VISIBLE,
  CONSTRAINT `fk_Contacto_Atleta1`
    FOREIGN KEY (`idAtleta`)
      REFERENCES `ClinicaAltaPerformance`.`Atleta` (`idAtleta`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 19 - Criação da tabela Contacto

```

-- -----
-- Table `ClinicaAltaPerformance`.`Prova_TipoTeste` 

CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Prova_TipoTeste` (
  `idProva` INT NOT NULL,
  `designacao` VARCHAR(100) NOT NULL,
  CONSTRAINT `fk_Prova_Teste_Prova`
    FOREIGN KEY (`idProva`)
      REFERENCES `ClinicaAltaPerformance`.`Prova`(`idProva`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
ENGINE = InnoDB;

```

Figura 20 - Criação da Tabela Prova_TipoTeste

```

-- -----
-- Table `ClinicaAltaPerformance`.`Prova` 

CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Prova` (
  `idProva` INT NOT NULL AUTO_INCREMENT,
  `designacao` VARCHAR(100) NOT NULL,
  `modalidade` VARCHAR(100) NOT NULL,
  `categoria` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idProva`)
ENGINE = InnoDB;

```

Figura 21 - Criação da tabela Prova

```

-- -----
-- Table `ClinicaAltaPerformance`.`Atleta_Prova` 

CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Atleta_Prova` (
  `id_Atleta` INT NOT NULL,
  `id_Prova` INT NOT NULL,
  `data` DATE NOT NULL,
  INDEX `fk_Atleta_Prova_Modalidade_prova_idx`(`id_Prova` ASC) VISIBLE,
  PRIMARY KEY (`id_Prova`, `id_Atleta`),
  CONSTRAINT `fk_Atleta_Prova_Modalidade_atleta`
    FOREIGN KEY (`id_Atleta`)
      REFERENCES `ClinicaAltaPerformance`.`Atleta`(`idAtleta`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Atleta_Prova_Modalidade_prova`
    FOREIGN KEY (`id_Prova`)
      REFERENCES `ClinicaAltaPerformance`.`Prova`(`idProva`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
ENGINE = InnoDB;

```

Figura 22 - Criação da tabela Atleta_Prova

```

-- -----
-- Table `ClinicaAltaPerformance`.`Morada` 

CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Morada` (
  `idMorada` INT NOT NULL AUTO_INCREMENT,
  `endereco` VARCHAR(100) NOT NULL,
  `localidade` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idMorada`)
ENGINE = InnoDB;

```

Figura 23 - Criação da tabela Morada

5.3. Tradução das Interrogações do Utilizador para SQL

De seguida serão apresentadas as respostas a algumas das interrogações realizadas pelo utilizador.

1. Consultar os testes realizados antes de determinada prova para um dado atleta ordenados do mais recente para o mais antigo (Requisito de Exploração 2)

```
DROP PROCEDURE IF EXISTS testes_anteriores_prova
DELIMITER $$

CREATE PROCEDURE testes_anteriores_prova (IN testes_atleta VARCHAR(100))
BEGIN
    SELECT t.designacao AS Designação, t.dataHora AS "Data e Hora" FROM TesteClinico t
    INNER JOIN Atleta_Prova ap ON ap.data > t.dataHora
    WHERE ap.id_Atleta = t.idAtleta AND id_Atleta = 1
    ORDER BY t.dataHora;
END $$

DELIMITER ;
```

Figura 24 - Resolução da interrogação nº1

2. Consultar os nomes dos atletas de determinada modalidade que acusaram positivo nos testes de doping (Requisito de Exploração 3)

```
DROP PROCEDURE IF EXISTS doping_positivo
DELIMITER $$

CREATE PROCEDURE doping_positivo (IN nome_atleta VARCHAR(100))
BEGIN
    SELECT DISTINCT a.nome
    FROM (SELECT ap.id_Atleta
          FROM Atleta_Prova ap
          INNER JOIN Prova p ON p.idProva = ap.id_Prova WHERE p.modalidade = "velocidade") AS partial_table
    INNER JOIN TesteClinico t ON t.idAtleta = partial_table.id_Atleta
    INNER JOIN Atleta a ON a.idAtleta = partial_table.id_Atleta
    WHERE designacao = "doping" AND resultado = "positivo" ;
END $$

DELIMITER ;
```

Figura 25 - Resolução da interrogação nº2

3. Consultar os nomes dos pacientes/atletas que foram submetidos a determinado teste (Requisito de Exploração 4)

```

DROP PROCEDURE IF EXISTS pacientes_testes
DELIMITER $$ 
CREATE PROCEDURE pacientes_testes (IN nome_atleta VARCHAR(100))
BEGIN
SELECT a.nome as Nome FROM Atleta a
INNER JOIN TesteClinico t ON t.idAtleta = a.idAtleta
WHERE t.designacao = "Analise_sanguinea";
END $$ 
DELIMITER ;

```

Figura 26 - Resolução da interrogação nº3

4. Consultar o número de testes clínicos realizados por um dado médico num determinado ano e o montante faturado (Requisito de Exploração 5)

```

DROP PROCEDURE IF EXISTS nr_examens_medico
DELIMITER $$ 
CREATE PROCEDURE nr_examens_medico (IN nome_medico VARCHAR(100))
SELECT count(*), sum(preco) as Faturado FROM TesteClinico t
WHERE t.idMedico = (SELECT m.idMedico FROM Medico m where m.nome = "Tiago Ferreira")
AND YEAR(t.dataHora) = 2019;
END $$ 
DELIMITER ;

```

Figura 27 - Resolução da interrogação nº4

5. Consultar agendamentos para um determinado tipo de exame numa determinada clínica (Requisito de Exploração 6)

```

DROP PROCEDURE IF EXISTS agendamentos_exame_clinica
DELIMITER $$ 
CREATE PROCEDURE agendamentos_exame_clinica (IN designcao_testeclinico VARCHAR(100))
SELECT t.designacao as Designação, t.dataHora as "Data e Hora", a.nome as Nome FROM TesteClinico t
INNER JOIN Medico m ON m.idMedico = t.idMedico
INNER JOIN Atleta a ON a.idAtleta = t.idAtleta
WHERE t.designacao = "Eletrocardiograma" AND t.dataHora > CURRENT_TIME() AND m.idClinica = 1;
END $$ 
DELIMITER ;

```

Figura 28 - Resolução da interrogação nº5

5.4. Tradução das transações estabelecidas para SQL

De seguida será exposta a implementação de todas as transações consideradas importantes para o projeto.

Inserir Atleta:

```
DROP PROCEDURE IF EXISTS criar_atleta;
DELIMITER $$
CREATE PROCEDURE criar_atleta(IN idAtleta INT, IN nome VARCHAR(100), IN dataNascimento DATE,
                             IN sexo ENUM('F', 'M'), IN morada INT, IN idMorada INT, IN endereco VARCHAR(100),
                             IN localidade VARCHAR(100), IN idContacto INT, IN telefone INT, IN email VARCHAR(100))
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;
    START TRANSACTION;
    -- criar morada caso não exista
    SET @morada := 0;
    SELECT @morada := m.idMorada FROM Morada AS m WHERE m.idMorada = idMorada;
    IF(@morada = 0)
    THEN INSERT INTO Morada(idMorada, endereco, localidade) VALUE (idMorada, endereco, localidade);
    SELECT @morada := m.idMorada FROM Morada AS m WHERE m.idMorada = idMorada;
    END IF;
    -- criar atleta
    INSERT INTO Atleta(idAtleta, nome, dataNascimento, sexo, morada)
    VALUES (idAtleta, nome, dataNascimento, sexo, @morada);
    -- criar contacto
    INSERT INTO Contacto(idContacto, telefone, email, idAtleta)
    VALUES (idContacto, telefone, email, idAtleta);
    COMMIT;
END $$
DELIMITER ;
```

Figura 29 - Inserir um novo atleta na base de dados

Para inserir um atleta é necessário fazer primeiro o registo do mesmo na tabela *Atleta*. Note-se que a morada pode ainda não se encontrar na base de dados, caso tal aconteça é necessário fazer um novo registo na tabela *Morada*. Finalmente, é criado um novo registo na tabela *Contacto*. Opcionalmente, pode ser feita a inserção de provas associadas ao atleta na tabela *Atleta_Prova*.

```
DROP PROCEDURE IF EXISTS inserir_alteta_prova;
DELIMITER $$
CREATE PROCEDURE inserir_alteta_prova(IN idAtleta INT, IN idProva INT, IN data DATE)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;
    START TRANSACTION;
    -- criar relação atleta prova
    INSERT INTO Atleta_Prova(idAtleta, idProva, data)
    VALUES (idAtleta, idProva, data);
    COMMIT;
END $$
DELIMITER ;
```

Figura 30 - Inserir provas do atleta

Efetuar Agendamento:

```
DROP PROCEDURE IF EXISTS agendamento;
DELIMITER $$

CREATE PROCEDURE agendamento(IN idTesteClinico INT, IN designacao VARCHAR(100), IN dataHora DATETIME,
                            IN preco DECIMAL(6,2), IN resultado VARCHAR(100), IN idMedico INT, IN idAtleta INT)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;
    START TRANSACTION;
    -- criar agendamento
    INSERT INTO testeClinico(idTesteClinico, designacao, dataHora, preco, resultado, idMedico, idAtleta)
    VALUES(idTesteClinico, designacao, dataHora, preco, null, idMedico, idAtleta);
    COMMIT;
END $$

DELIMITER ;
```

Figura 31 - Inserir um agendamento na base de dados

Um agendamento não é nada mais que uma nova entrada na tabela *TesteClinico* que possui o resultado a *null* e uma data futura.

Adicionar Prova:

```
DROP PROCEDURE IF EXISTS criar_prova;
DELIMITER $$

CREATE PROCEDURE criar_prova(IN idProva INT, IN designacao VARCHAR(100), IN modalidade VARCHAR(100),
                            IN categoria VARCHAR(100), IN designacao ENUM('Dopping', 'Eletrocardiograma',
                            'Ecocardiograma', 'Raio X', 'Prova de Esforço'))
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;
    START TRANSACTION;
    -- criar prova
    INSERT INTO prova(idProva, designacao, modalidade, categoria)
    VALUES(idProva, designacao, modalidade, categoria);
    -- criar tabela de testes obrigatórios para a prova
    INSERT INTO prova_tipoteste(idProva, designacao)
    values(idProva, designacao);
    COMMIT;
END $$

DELIMITER ;
```

Figura 32 - Inserir o registo de um novo tipo de prova

5.5. Triggers SQL

Aquando da remoção de um atleta da base de dados existe a necessidade de atualizar as restantes tabelas com que está relacionada, isto implica apagar os seus contactos e, caso não exista mais nenhum atleta com a mesma morada, também apagar esta.

```
DELIMITER $$  
CREATE TRIGGER atleta_delete AFTER DELETE ON Atleta  
FOR EACH ROW  
BEGIN  
    DECLARE morada_count INT;  
    SET morada_count = (SELECT count(morada) FROM Atleta WHERE OLD.morada = morada);  
    IF morada_count = 1 THEN  
        DELETE FROM Morada m WHERE m.idMorada = OLD.morada;  
    END IF;  
    DELETE FROM Contacto c WHERE c.idAtleta = (SELECT idAtleta FROM Atleta);  
END $$  
DELIMITER ;
```

Figura 33 - Imagem do código obtido para o trigger

5.6. Escolha, Definição e Caracterização de Índices em SQL

Apesar de a opção de *Forward Engineering* criar índices por predefinição, para as chaves primárias, no nosso sistema de bases de dados optamos por não implementar índices, pois consideramos que este não dispõe de uma dimensão e complexidade significativa.

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

Ao elaborar a base de dados é necessário ter em atenção o espaço que ocupará em disco. Para efetuar uma previsão do espaço ocupado em disco, começamos por identificar o tamanho de cada tipo de dados que utilizamos. De seguida, para cada tabela do modelo lógico verificamos o espaço ocupado por esta.

Tipo de Dados	Tamanho (bytes)
INT	4
DATE	4
DATETIME	8

VARCHAR(N)	N+1
DECIMAL(I,D)	4+4*

Tabela 9 - Espaço ocupado em disco por cada tipo de dados

*4 para o pior caso pois varia consoante a parte decimal

	Atributos	Tipo de Dados	Espaço Ocupado
ATLETA	idAtleta	INT	4
	nome	VARCHAR(100)	101
	dataNascimento	DATE	4
	sexo	VARCHAR(5)	6
	morada	INT	4
TOTAL			119

Tabela 10 - Espaço ocupado pela tabela Atleta

	Atributos	Tipo de Dados	Espaço Ocupado
Clínica	idClinica	INT	4
	nome	VARCHAR(100)	101
	morada	INT	4
TOTAL			109

Tabela 11 - Espaço ocupado pela tabela Clínica

	Atributos	Tipo de Dados	Espaço Ocupado
Médico	idMedico	INT	4
	nome	VARCHAR(100)	101
	dataNascimento	DATE	4
	dataInicioServico	DATE	4
	especialidade	VARCHAR(100)	101
	idChefe	INT	4
	idClinica	INT	4
	morada	INT	4
TOTAL			226

Tabela 12 - Espaço ocupado pela tabela Médico

Atributos	Tipo de Dados	Espaço Ocupado
-----------	---------------	----------------

TesteClínico	idTesteClinico	INT	4
	designacao	VARCHAR(100)	101
	dataHora	DATE	4
	preco	DECIMAL(6,2)	4+1
	resultado	VARCHAR(100)	101
	idMedico	INT	4
	idAtleta	INT	4
TOTAL			223

Tabela 13 - Espaço ocupado pela tabela Teste Clínico

	Atributos	Tipo de Dados	Espaço Ocupado
Prova	idProva	INT	4
	designacao	VARCHAR(100)	101
	modalidade	VARCHAR(100)	101
	categoria	VARCHAR(100)	101
TOTAL			307

Tabela 14 - Espaço ocupado pela tabela Prova

	Atributos	Tipo de Dados	Espaço Ocupado
Morada	idMorada	INT	4
	endereco	VARCHAR(100)	101
	localidade	VARCHAR(100)	101
TOTAL			206

Tabela 15 - Espaço ocupado pela tabela Morada

	Atributos	Tipo de Dados	Espaço Ocupado
Contacto	idContacto	INT	4
	telefone	INT	4
	email	VARCHAR(100)	101
	idAtleta	INT	4
TOTAL			113

Tabela 16 - Espaço ocupado pela tabela Contacto

	Atributos	Tipo de Dados	Espaço Ocupado
Atleta_Prova	idAtleta	INT	4
	idProva	INT	4
	data	DATE	4
TOTAL			12

Tabela 17 - Espaço ocupado pela tabela Atleta_Prova

	Atributos	Tipo de Dados	Espaço Ocupado
Prova_TipoTeste	idProva	INT	4
	designacao	VARCHAR(100)	101
TOTAL			105

Tabela 18 - Espaço ocupado pela tabela Prova_TipoTeste

Considerando a dimensão da população presente na nossa base de dados, em seguida será ilustrado o espaço ocupado no disco por esta.

Tabela	Espaço Ocupado em Disco(bytes)
Atleta	$35*119 = 4165$
Clínica	$3*109 = 327$
Médico	$15*226 = 3390$
TesteClinico	$99*223 = 22077$
Prova	$39*307 = 11973$
Morada	$49*206 = 10094$
Contacto	$43*113 = 4859$
Atleta_Prova	$45*12 = 540$
Prova_TipoTeste	$47*105 = 4935$
Total	62360

Tabela 19 - Espaço ocupado em disco pela população atual

Como podemos observar na tabela acima, a estimativa para o tamanho da nossa base de dados com o povoamento final é de 62360 bytes. No entanto, é fundamental verificar como se comportará com uma quantidade de dados significante, ou seja, com uma escala real. Atendendo ao crescimento da procura dos serviços da clínica estima-se que o número de atletas quadruplica, o que implicará contratar mais médicos. O número de testes clínicos e provas vai aumentar numa escala aproximadamente proporcional ao aumento do número de atletas. Na tabela seguinte podemos observar a estimativa para o crescimento da população da base de dados no próximo ano.

Tabela	Espaço Ocupado em Disco(bytes)
Atleta	$140*119 = 16660$
Clínica	$3*109 = 327$
Médico	$35*226 = 7910$
TesteClinico	$400*223 = 89200$
Prova	$138*307 = 42366$
Morada	$190*206 = 39140$
Contacto	$215*113 = 24295$
Atleta_Prova	$135*12 = 1620$

Prova_TipoTeste	141*105 = 14805
Total	236323

Tabela 20 - Espaço ocupado em disco pela população no futuro

Assim, se a estimativa se mantiver, podemos concluir que anualmente o crescimento esperado para a base de dados é de 174 KBytes por ano.

5.8. Definição e Caracterização das vistas de utilização em SQL

Nesta seção, são apresentadas as views que consideramos fundamentais para os utilizadores da nossa base de dados.

Na figura seguinte apresenta-se a view dos agendamentos, acessível a qualquer médico e rececionista.

```
DROP VIEW IF EXISTS view_agendamento;
DELIMITER $$ 
CREATE VIEW view_agendamento AS
    SELECT t.designacao AS 'Exame', t.dataHora AS 'Data e Hora', a.nome AS 'Paciente', m.nome AS 'Medico'
    FROM TesteClinico t
    INNER JOIN Medico m ON m.idMedico = t.idMedico
    INNER JOIN Atleta a ON a.idAtleta = t.idAtleta
    WHERE t.dataHora > CURRENT_TIME();
    ORDER BY t.dataHora ASC
DELIMITER ;
```

Figura 34 - View de agendamentos

Na figura seguinte apresenta-se a view dos exames realizados, acessível a qualquer médico e rececionista.

```
DROP VIEW IF EXISTS view_exames_realizados;
DELIMITER $$ 
CREATE VIEW view_exames_realizados AS
    SELECT a.nome AS 'Nome', t.designacao AS 'Designação', t.dataHora AS 'Realizado'
    FROM Atleta a
    INNER JOIN TesteClinico t ON t.idAtleta = a.idAtleta
    WHERE t.dataHora < CURRENT_TIME();
    ORDER BY t.dataHora ASC
DELIMITER ;
```

Figura 35 – View dos exames realizados

Na figura seguinte apresenta-se a view dos agendamentos, acessível a qualquer utilizador do sistema.

```

DROP VIEW IF EXISTS view_medicos_especialidades;
DELIMITER $$ 
CREATE VIEW view_medicos_especialidades AS
    SELECT m.idMedico AS 'ID', m.nome AS 'Nome', m.especialidade AS 'Especialidade', c.nome AS 'Clinica'
    FROM Medico m
    INNER JOIN Clinica c ON m.idClinica = c.idClinica
    ORDER BY m.idMedico
DELIMITER ;

```

Figura 36 – View dos medicos e as suas especialidades

Na figura seguinte apresenta-se a view dos agendamentos, acessível aos médicos e rececionistas.

```

DROP VIEW IF EXISTS view_dados_pacientes;
DELIMITER $$ 
CREATE VIEW view_dados_pacientes AS
    SELECT a.idAtleta AS 'ID', a.nome AS 'Nome', a.sexo AS 'Género', a.dataNascimento AS 'Data de Nascimento',
    a.morada AS 'Morada' FROM Atleta a
    ORDER BY a.idAtleta
DELIMITER ;

```

Figura 37 - View dos dados dos pacientes

5.9. Definição e Caracterização 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. Assim a primeira medida foi, a criação de vários utilizadores que irão interagir de forma regular com a base de dados com permissões segundo a seguinte tabela. Existirá ainda um administrador que terá todos os privilégios.

Adicionalmente, como medida de segurança, serão regularmente backups dos dados do sistema recorrendo ao comando *mysqldump*. Este comando será agendado para ser executado todos os dias o que irá permitir manter um registo seguro dos dados.

```

CREATE USER 'admin'@'localhost' IDENTIFIED BY 'Admin';
CREATE USER 'medico'@'localhost' IDENTIFIED BY 'Medico';
CREATE USER 'receptionista'@'localhost' IDENTIFIED BY 'Receptionista';

GRANT SELECT, UPDATE, INSERT, DELETE ON * TO 'admin'@'localhost';

GRANT EXECUTE ON PROCEDURE testes_anter_prova TO 'admin'@'localhost';
GRANT EXECUTE ON PROCEDURE dopping_positivo TO 'admin'@'localhost';
GRANT EXECUTE ON PROCEDURE pacientes_testes TO 'admin'@'localhost';
GRANT EXECUTE ON PROCEDURE nr_examens_medico TO 'admin'@'localhost';
GRANT EXECUTE ON PROCEDURE agendamentos_exame_clinica TO 'admin'@'localhost';

GRANT EXECUTE ON PROCEDURE testes_anter_prova TO 'medico'@'localhost';
GRANT EXECUTE ON PROCEDURE dopping_positivo TO 'medico'@'localhost';
GRANT EXECUTE ON PROCEDURE pacientes_testes TO 'medico'@'localhost';
GRANT EXECUTE ON PROCEDURE agendamentos_exame_clinica TO 'medico'@'localhost';

GRANT EXECUTE ON PROCEDURE testes_anter_prova TO 'receptionista'@'localhost';
GRANT EXECUTE ON PROCEDURE pacientes_testes TO 'receptionista'@'localhost';
GRANT EXECUTE ON PROCEDURE agendamentos_exame_clinica TO 'receptionista'@'localhost';

GRANT SELECT ON view_agendamento TO 'admin'@'localhost';
GRANT SELECT ON view_agendamento TO 'medico'@'localhost';
GRANT SELECT ON view_agendamento TO 'receptionista'@'localhost';
GRANT SELECT ON view_dados_pacientes TO 'admin'@'localhost';
GRANT SELECT ON view_dados_pacientes TO 'medico'@'localhost';
GRANT SELECT ON view_dados_pacientes TO 'receptionista'@'localhost';
GRANT SELECT ON view_examens_especialidades TO 'admin'@'localhost';
GRANT SELECT ON view_examens_especialidades TO 'receptionista'@'localhost';
GRANT SELECT ON view_examens_realizados TO 'admin'@'localhost';
GRANT SELECT ON view_examens_realizados TO 'medico'@'localhost';
GRANT SELECT ON view_examens_realizados TO 'receptionista'@'localhost';

FLUSH PRIVILEGES;

```

Figura 38 - Permissões dos diferentes utilizadores

5.10. Revisão do Sistema Implementado com o Utilizador

Após a conclusão da implementação da base de dados foi agendada uma reunião final com a empresa de forma a validar a base de dados e possivelmente dar como concluído o desenvolvimento do sistema. Aqui, foram analisados todos os requisitos estabelecidos e foi revisto novamente todas as fases do trabalho já realizado, desde a elaboração do Modelo Conceptual até ao Modelo Físico. Esclarecidas todas as dúvidas que poderiam existir sobre a implementação o cliente deu autorização para avançar para a instalação do sistema.

6. Sistema NoSQL: *Neo4j*

6.1. Justificação da utilização de um sistema NoSQL

Com a criação de uma base de dados relacional, a valorização da clínica no mercado aumentou drasticamente (devido à sua gestão de qualidade) e mais atletas e responsáveis pelas competições têm contactado a clínica para a realização de testes.

Deste modo, alguns problemas têm ocorrido, com cada vez mais agendamentos, a informação tem crescido intensamente e cada vez mais, é necessário usar um sistema de base de dados com maior escalabilidade, e melhor capacitado à obtenção de respostas rápidas a “queries” (interrogações).

Surge assim a necessidade de utilização de um sistema NoSQL, no nosso caso o motor Neo4j que tem em vista bases de dados com uma enorme quantidade de informação (“Performance”), tal como a Clínica Alta Performance, que permite a sua alteração face às necessidades dos médicos e dos atletas e aos requisitos de negócio (Flexibilidade e Agilidade), permitindo assim corresponder de forma positiva e eficiente às dificuldades encontradas.

6.2. Identificação dos objetivos da base de dados

O objetivo da implementação deste sistema NoSQL é possibilitar a realização de interrogações que envolvem a junção de várias relações de uma forma muito mais rápida. Estas interrogações correspondem às interrogações anteriormente implementadas em SQL, que devido a necessidade de melhoria na qualidade do serviço serão agora realizadas em Neo4j, tirando proveito das suas funcionalidades de travessia em grafos.

6.3. Identificação e explicação do tipo de questões (necessidades) que serão realizadas sobre o sistema de dados NoSQL

De acordo com os objetivos antes mencionados, irá surgir um conjunto dos dados da base de dado relacional apresentada que serão utilizados para efetuar as questões (“queries”) que mais se destacam para a resolução do problema atual.

1. Obter o número de pacientes de todas as clínicas
2. Consultar os testes realizados antes de determinada prova para um dado atleta ordenados do mais recente para o mais antigo.
3. Consultar os nomes dos atletas de determinada modalidade que acusaram positivo nos testes de *doping*.
4. Consultar os nomes dos pacientes/atletas que foram submetidos a determinado teste.
5. Consultar o número de testes clínicos realizados por um dado médico num determinado ano e o montante faturado.
6. Consultar agendamentos para um determinado tipo de exame numa determinada clínica.
7. Obter o número de médicos de uma determinada especialidade numa determinada clínica.
8. Obter o top 3 atletas que gastaram mais dinheiro em exames.
9. Obter o top 5 atletas que realizaram mais exames.
10. Obter o top 3 médicos de determinada especialidade que mais faturaram.
11. Obter a designação e resultado dos últimos 5 testes a que um atleta foi submetido.
12. Consultar a lista de exames obrigatórios para poder concorrer em determinada prova.
13. Consultar a designação das provas realizadas por determinado atleta.
14. Obter os atletas de uma dada localidade
15. Obter o número de testes clínicos realizados numa determinada clínica no ano 2019.

6.4. Definição da estrutura base para o sistema de dados NoSQL que satisfaça os requisitos e as questões apresentadas anteriormente

De modo a satisfazer todos os requisitos apresentados na questão anterior surgiu a seguinte estrutura:

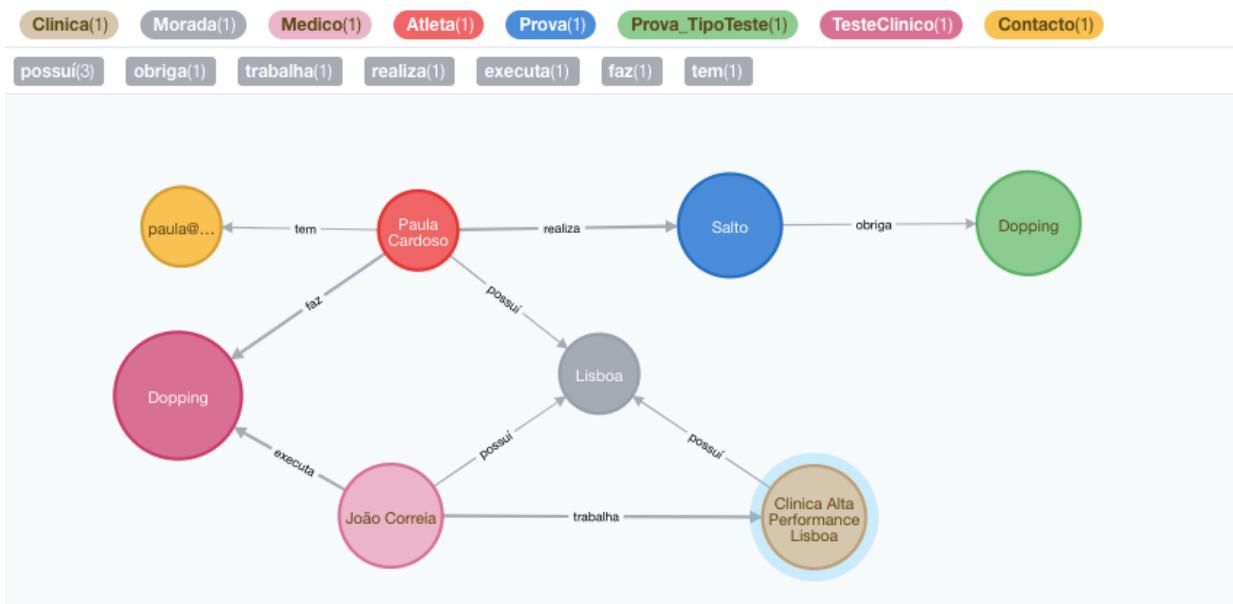


Figura 39 - Estrutura Neo4j

A cada nodo “Prova” estão ligados pelo relacionamento “realiza” desde 0 até N nodos “Atleta”. 0 até N nodos “Atleta” estão ligados ao nodo “TesteClinico” pelo relacionamento “faz”. A cada nodo “TesteClinico” está ligado 1 nodo “Medico” pela relação “executa” que por sua vez está ligado ao nodo “Clinica” pela relação “trabalha”. O nodo “Atleta”, o nodo “Clínica” e o nodo “Médico” estão ligados ao nodo “Morada” por uma morada respetiva e o relacionamento “possui”. O nodo “Atleta” está ligado ao nodo “Contacto” pelo relacionamento “tem” e o nodo “Prova” está ligado ao nodo “Prova_TipoTeste” pelo relacionamento “obriga”.

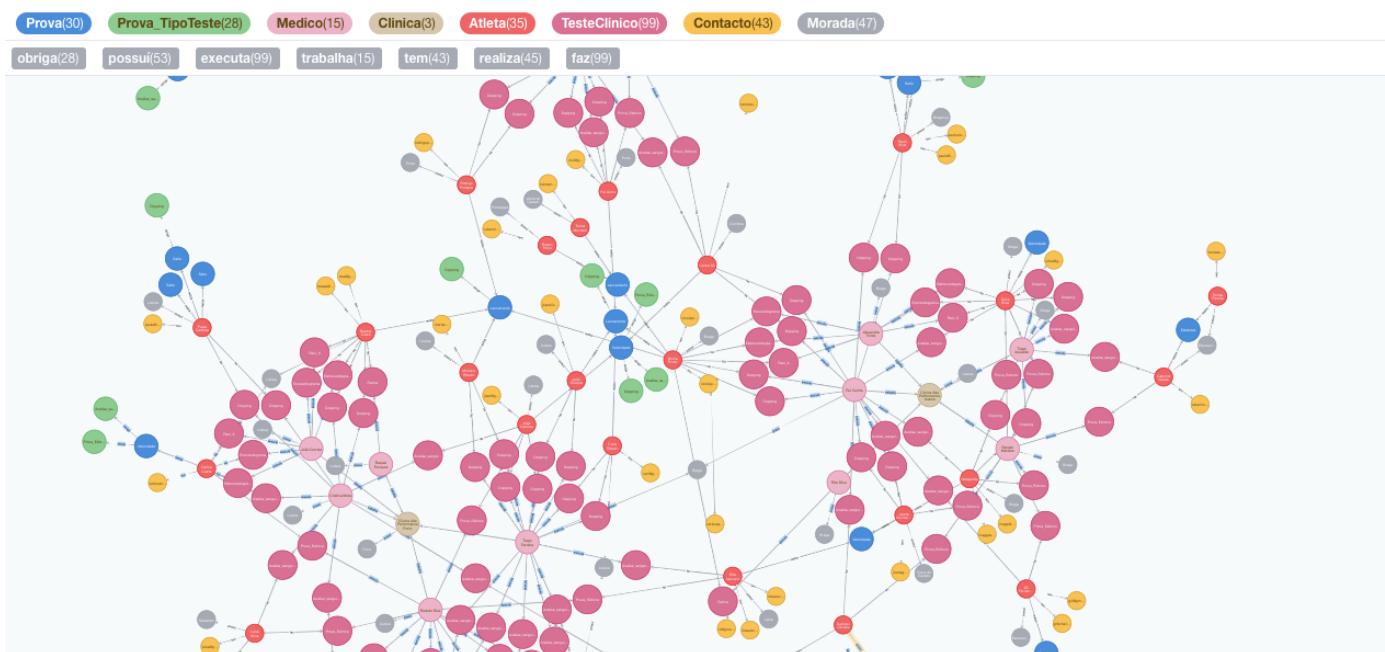


Figura 40 - Ilustração da base de dados NoSQL criada em Neo4j

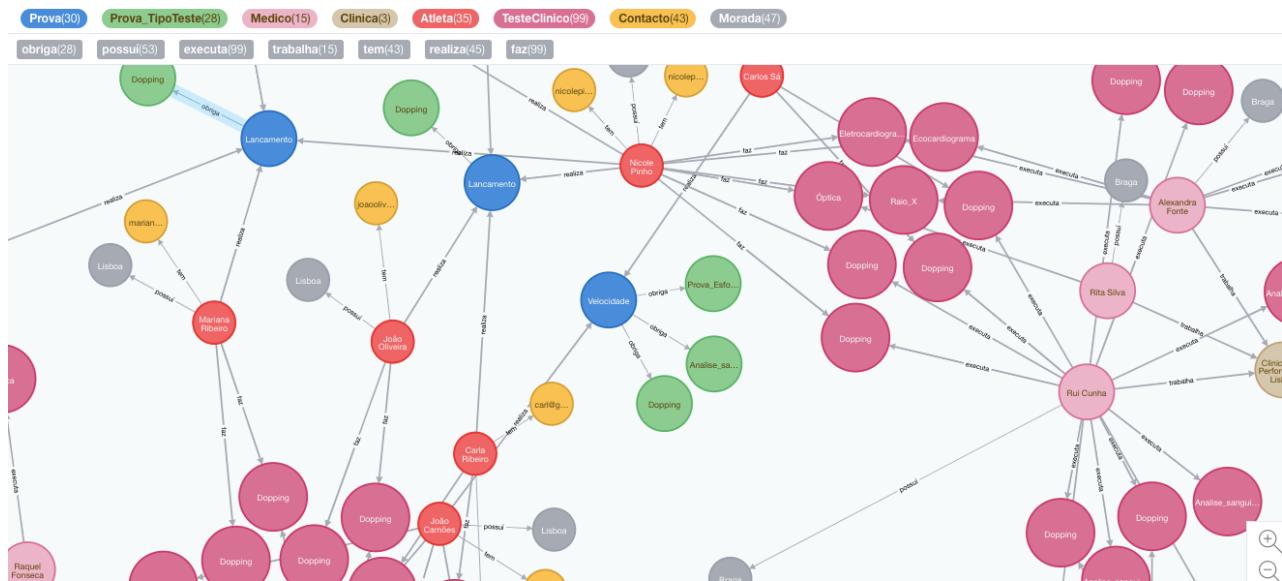


Figura 41 - Ilustração da base de dados NoSQL criada em Neo4j

6.5. Identificação dos objetos de dados no sistema SQL que serão utilizados para alimentar o novo sistema

Obtemos cada nodo através de uma tabela no modelo relacional. O nodo Atleta obtém-se através da tabela Atleta, o nodo Clinica obtém-se através da tabela Clinica, o nodo Prova obtém-se através da tabela Prova, o nodo TesteClinico através da tabela TesteClinico, o nodo Medico através da tabela Medico, o nodo Prova_TipoTeste através da tabela Prova_TipoTeste, o nodo Contacto através da tabela Contacto e o nodo Morada através da tabela Morada.

Os relacionamentos “realiza” são gerados a partir da tabela Atleta_Prova, os relacionamentos “faz” e “executa” são gerados a partir da tabela TesteClinico e o “trabalha” a partir da tabela Medico. O relacionamento “tem” é gerado a partir da tabela Contacto, o “obriga” a partir da tabela Prova_TipoTeste e o relacionamento possuí a partir da tabela Atleta, Médico e Clínica.

6.6. Mapeamento do processo de migração de dados, descrevendo o processo de conversão dos vários objetos de dados

O processo de conversão dos vários objetos de dados deu-se em duas fases: o carregamento dos nodos e o carregamento dos relacionamentos entre esses nodos.

Carregamento dos nodos:

- As entradas da tabela Clinica foram convertidas para nodos do tipo Clinica com os atributos idClinica e nome;
- As entradas da tabela Atleta foram convertidas para nodos do tipo Atleta com os atributos idAtleta, nome, dataNascimento e sexo;
- As entradas da tabela Prova foram convertidas para nodos do tipo Prova com os atributos idProva, designacao, modalidade e categoria;
- As entradas da tabela TesteClinico foram convertidas para nodos do tipo TesteClinico com os atributos idTesteClinico, designacao, dataHora, preco e resultado;
- As entradas da tabela Medico foram convertidas para nodos do tipo Medico com os atributos idMedico, nome, dataInicioServico, dataNascimento e especialidade;
- As entradas da tabela Morada foram convertidas para nodos do tipo Morada com os atributos idMorada, endereço e localidade;
- As entradas da tabela Contacto foram convertidas para nodos do tipo Contacto com os atributos idContacto, telefone e email;
- As entradas da tabela Prova_TipoTeste foram convertidas para nodos do tipo Prova_TipoTeste com os atributos idProva e designacao;

Carregamento dos relacionamentos:

- A tabela Atleta_Prova será utilizada para criar o relacionamento “realiza” com direção do Atleta para Prova e que irá guardar a data da prova;
- A tabela TesteClinico será utilizada para dois tipos de relacionamento:
 - O “executa” será criado com o idMedico e o idTesteClinico (Medico executa TesteClinico);
 - O relacionamento “faz” com direção Atleta para TesteClinico;
- A tabela Medico será utilizada para criar o relacionamento “trabalha” com direção do Medico para a Clinica;
- A tabela Contacto será utilizada para criar o relacionamento “tem” com direção do Atleta para o Contacto;
- A tabela Prova_TipoTeste será utilizada para criar o relacionamento “obriga” com direção Prova para Prova_TipoTeste (para realizar uma prova é necessário realizar um determinado teste clínico antes)
- Para a criação do relacionamento “possui” com direção Atleta para Morada usou-se a tabela Atleta, para a criação do relacionamento “possui” com direção Médico para Morada usou-se a tabela Médico e para a criação do relacionamento “possui” com direção Clínica para Morada usou-se a tabela Clínica.

6.7. Explicação do processo de migração de dados, explicando de forma detalhada as suas principais etapas - extração, transformação e carregamento

Para a conversão dos vários objetos de dados usamos as tabelas da base de dados relacional mencionadas acima (SELECT * FROM TABELADESEJADA) exportando-as e convertendo-as em ficheiros CSV que podem ser carregados na base de dados não relacional.

Para estes ficheiros serem carregados pelo método usado no neo4j com a linguagem cypher :“LOAD CSV from ‘file:///ficheiro.csv’ AS line” teremos que coloca-los na diretoria correta para poderem ser invocados com o “file:///”. Esta diretoria está predefinida como a diretoria import do neo4j, conseguindo-se chegar lá pelo path: usr/local/Cellar/neo4j/*3.5.12/libexec/import.

Todas as relações e nodos base que foram migradas para a nova base de dados foram alvo de projeções uma vez que nem todos os atributos constantes nestes eram necessárias para as interrogações criadas.

O processo é realizado sistematicamente e parcialmente. É sempre feita a extração parcial das entradas, a sua transformação, e, por fim,, o seu carregamento na base de dados. A extração é parcial para solucionar os problemas de memória que o respetivo programa pode ter, desta forma este funciona para bases de dados de “qualquer” dimensão.

*A versão pode ser diferente dependendo do driver instalado.

6.8. Apresentação e descrição da implementação do processo de migração de dados

Para a migração da base de dados foi criado o seguinte script em linguagem cypher que pode ser utilizado para o neo4j;

Como referido anteriormente, em primeiro lugar devemos iniciar a criação dos nodos principais:

Criação do nodo Clinica:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Clinica.csv' AS line
CREATE (:Clinica { idClinica: line.idClinica, nome: line.nome })
```

Criação do nodo Atleta:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Atleta.csv' AS line
CREATE (:Atleta { idAtleta: line.idAtleta, nome: line.nome, dataNascimento:
line.dataNascimento, sexo: line.sexo })
```

Criação do nodo Prova:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Prova.csv' AS line
CREATE (:Prova { idProva: line.idProva, designacao: line.designacao, modalidade:
line.modalidade ,categoria: line.categoria})
```

Criação do nodo TesteClinico:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Teste_Clinico.csv' AS line
CREATE (:TesteClinico { idTesteClinico: line.idTesteClinico, designacao:
line.designacao, dataHora: line.dataHora, preco: line.preco , resultado:
line.resultado})
```

Criação do nodo Medico:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Medico.csv' AS line
CREATE (:Medico { idMedico: line.idMedico, nome: line.nome, dataInicioServico:
line.dataInicioServico , dataNascimento: line.dataNascimento, especialidade:
line.especialidade})
```

Criação do nodo Morada:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Morada.csv' AS line
CREATE (:Morada { idMorada: line.idMorada, endereco: line.endereco, localidade:
line.localidade })
```

Criação do nodo Contacto:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Contacto.csv' AS line
CREATE (:Contacto { idContacto: line.idContacto, telefone: line.idtelefone, email:
line.email })
```

Criação do nodo Prova_TipoTeste:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Prova_TipoTeste.csv' AS line
CREATE (:Prova_TipoTeste{ idProva: line.idProva, designação: line.designacao})
```

Foram criados índices para assegurar uma procura mais eficiente quando for feita a criação de relacionamentos num passo seguinte:

```
CREATE INDEX ON : Clinica(idClinica);
CREATE INDEX ON : Atleta(idAtleta);
CREATE INDEX ON : Prova(idProva);
CREATE INDEX ON : TesteClinico(idTesteClinico);
CREATE INDEX ON : Medico(idMedico);
CREATE INDEX ON : Morada(idMorada);
CREATE INDEX ON : Contacto(idContacto);
CREATE INDEX ON : Prova_TipoTeste(idProva);
```

Após a criação dos índices e dos nodos, é feita a criação dos relacionamentos:

Criação do relacionamento entre o nodo Atleta e o nodo Prova:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Atleta_Prova.csv' AS line
MATCH (a:Atleta {idAtleta: line.id_Atleta})
MATCH (p:Prova {idProva: line.id_Prova})
MERGE (a)-[r:realiza {data: line.data}]->(p);
```

Criação do relacionamento entre o nodo Atleta e o nodo TesteClinico:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Teste_Clinico.csv' AS line
MATCH (a:Atleta {idAtleta: line.idAtleta })
MATCH (t:TesteClinico {idTesteClinico: line.idTesteClinico })
MERGE (a)-[r:faz]->(t);
```

Criação do relacionamento entre o nodo Medico e o nodo Clinica:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Medico.csv' AS line
MATCH (m:Medico {idMedico: line.idMedico})
MATCH (c:Clinica {idClinica: line.idClinica })
MERGE (m)-[r:trabalha]->(c);
```

Criação do relacionamento entre o nodo Medico e o nodo TesteClinico:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Teste_Clinico.csv' AS line
MATCH (m:Medico {idMedico: line.idMedico})
MATCH (t:TesteClinico {idTesteClinico: line.idTesteClinico })
MERGE (m)-[r:executa]->(t);
```

Criação do relacionamento entre o nodo Atleta e o nodo Morada:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Atleta.csv' AS line
MATCH (a:Atleta {idAtleta: line.idAtleta})
MATCH (m:Morada {idMorada: line.morada })
MERGE (a)-[r:possui]->(m);
```

Criação do relacionamento entre o nodo Médico e o nodo Morada:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Medico.csv' AS line
MATCH (me:Medico {idMedico: line.idMedico})
MATCH (m:Morada {idMorada: line.morada })
MERGE (me)-[r:possui]->(m);
```

Criação do relacionamento entre o nodo Clinica e o nodo Morada:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Clinica.csv' AS line
MATCH (c:Clinica {idClinica: line.idClinica})
MATCH (m:Morada {idMorada: line.morada })
MERGE (c)-[r:possui]->(m);
```

Criação do relacionamento entre o nodo Contacto e o nodo Atleta:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Contacto.csv' AS line
MATCH (a:Atleta {idAtleta: line.idAtleta})
MATCH (c:Contacto {idContacto: line.idContacto })
MERGE (a)-[r:tem]->(c);
```

Criação do relacionamento entre o Prova_TipoTeste o nodo Prova:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:///Prova_TipoTeste.csv' AS line
MATCH (p:Prova {idProva: line.idProva})
MATCH (ptt:Prova_TipoTeste {idProva: line.idProva})
MERGE (p)-[r:obriga]->(ptt);
```

É boa prática o uso de “USING PERIODIC COMMIT” quando criamos nodos ou relacionamentos dado a estarmos a trabalhar com bases de dados de grandes dimensões que devem ser atualizadas (commit) frequentemente ao carregarem os dados.

6.9. Apresentação da forma como as questões identificadas anteriormente podem ser satisfeitas com o novo sistema, utilizando a linguagem de interrogação do sistema NoSQL

A resolução para qualquer uma das questões levantadas anteriormente é obtida de forma eficiente fácil ao recorrermos ao Neo4j.

As queries que fazem apenas a leitura de atributos não sobrecarregam a base de dados MySQL pelo que a performance destas já é eficiente, mesmo perante uma enorme quantidade de dados.

1. Obter o número de pacientes de todas as clínicas

```
MATCH (:Atleta)
RETURN count(*) as count
```

2. Consultar os testes realizados antes de determinada prova para um dado atleta ordenados do mais recente para o mais antigo.

```
MATCH (t:TesteClinico) <- [:faz] - (a:Atleta{nome:'Silvia Braz'})
MATCH (a) - [r:realiza] ->(p:Prova)
WHERE datetime(r.data) > datetime(REPLACE(t.dataHora, " ","T"))
RETURN t.designacao, t.dataHora
ORDER BY t.dataHora DESC;
```

3. Consultar os nomes dos atletas de determinada modalidade que acusaram positivo nos testes de *doping*.

```
MATCH (a:Atleta) - [:realiza] ->
(p:Prova{modalidade:'Velocidade'})
MATCH (a) - [:faz] ->
(t:TesteClinico)
WHERE t.designacao='Dopping' AND t.resultado='Positivo'
RETURN DISTINCT a.nome;
```

4. Consultar os nomes dos pacientes/atletas que foram submetidos a determinado teste.

```
MATCH (a:Atleta) - [:faz] ->
(t:TesteClinico{designacao:'Analise_sanguinea'})
RETURN DISTINCT a.nome;
```

5. Consultar o número de testes clínicos realizados por um dado médico num determinado ano e o montante faturado.

```
MATCH (t:TesteClinico) <- [:executa] -(m:Medico{nome:'Tiago Ferreira'})
WHERE datetime(REPLACE(t.dataHora, " ","T")).year = 2019
RETURN count(t), sum(toFloat(t.preco));
```

6. Consultar agendamentos para um determinado tipo de exame numa determinada clínica.

```
MATCH (m:Medico) - [:trabalha] ->
(c:Clinica{nome:'Clinica Alta Performance Lisboa'})
MATCH (m) - [:executa] ->
(t:TesteClinico{designacao:'Dopping'})
WHERE datetime(REPLACE(t.dataHora, " ","T")) > datetime()
RETURN t.dataHora;
```

7. Obter o número de médicos de uma determinada especialidade numa determinada clínica.

```
MATCH (m:Medico) - [:trabalha] ->
(c:Clinica{nome:'Clinica Alta Performance Porto'})
WHERE m.especialidade = 'Cardiologia'
RETURN count(m);
```

8. Obter o top 3 atletas que gastaram mais dinheiro em exames.

```
MATCH (a:Atleta) - [:faz] ->(t:TesteClinico)
WITH a, sumtoFloat(t.preco) AS soma
ORDER BY soma DESC LIMIT 3
RETURN a.nome
```

9. Obter o top 5 atletas que realizaram mais exames.

```
MATCH (a:Atleta) - [:faz] ->(t:TesteClinico)
WITH a, count(t) AS conta
ORDER BY conta DESC LIMIT 5
RETURN a.nome;
```

10.Obter o top 3 médicos de determinada especialidade que mais faturaram.

```
MATCH (t:TesteClinico) <- [:executa] -(m:Medico{especialidade:'Cardiologia'})
WITH m, sumtoFloat(t.preco) AS fatura
ORDER BY fatura DESC LIMIT 3
RETURN m.nome;
```

11. Obter a designação e resultado dos últimos 5 testes a que um atleta foi submetido.

```
MATCH (t:TesteClinico) <- [:faz] - (a:Atleta{nome:'Paulo Silva'})  
WHERE datetime(REPLACE(t.dataHora, " ","T")) < datetime()  
RETURN t.designacao, t.resultado  
LIMIT 5;
```

12. Consultar a lista de exames obrigatórios para poder concorrer em determinada prova.

```
MATCH (ptt:Prova_TipoTeste) <- [:obriga] - (p:Prova{idProva:'1'})  
RETURN ptt.designação;
```

13. Consultar a designação das provas realizadas por determinado atleta.

```
MATCH (p:Prova)< - [:realiza] -  
(a:Atleta{nome:'Paulo Silva'})  
RETURN p.designacao;
```

14.Obter os atletas de uma dada localidade

```
MATCH (a:Atleta) - [:possui] ->(:Morada{localidade:'Braga'})  
RETURN a.nome;
```

15.Obter o número de testes clínicos realizados numa determinada clínica no ano 2019.

```
MATCH (m:Medico)-[:trabalha]->(c:Clinica{nome:'Clinica Alta Performance Porto'})  
MATCH (m) - [:executa] ->(t:TesteClinico)  
WHERE datetime(REPLACE(t.dataHora, " ","T")).year = 2019  
RETURN count(t);
```

8. Conclusões e Trabalho Futuro

A implementação da base de dados proposta neste relatório resulta de uma abordagem que nos pareceu adequada à resolução do problema, onde os principais objetivos foram alcançados. Desta forma, ao analisarmos cuidadosamente a nossa implementação da base de dados, é-nos possível apontar os pontos altos desta.

Primeiramente, aquilo que é mais visível é a simplicidade com que a base de dados está estruturada, o que a torna bastante intuitiva e de fácil alteração. Apesar da sua simplicidade, a consistência e a reduzida redundância de dados são igualmente um dos seus pontos fortes. Estas foram conseguidas através da modelação da base de dados que achamos mais assertiva e correta.

A realização deste trabalho permitiu-nos acompanhar todos os estados de desenvolvimento de uma base de dados, desde a análise de requisitos até, por fim, à criação de um modelo físico, capaz de suportar as transações de informações pretendidas.

O processo de levantamentos de requisitos permitiu desenvolver a capacidade de abstração do grupo, de maneira a que através de uma contextualização do problema foram concebidas diversas soluções. O desenvolvimento do modelo conceptual permitiu-nos chegar a uma solução viável para o problema, capaz de corresponder a todos os requisitos anteriormente apresentados, através da definição das entidades e correspondentes atributos e relacionamentos. De seguida, o desenvolvimento do modelo lógico aprofundou todos os conceitos do modelo anterior, aproximando-os de um modelo de programação orientada a SGBD. Esta parte do processo permitiu-nos aplicar regras de normalização de tabelas e integridade referencial entre as mesmas. Por último, a conversão para modelo físico possibilitou-nos aprofundar os conhecimentos SQL com a criação do esquema físico da base de dados, povoamento de tabelas e consulta de informação sob queries.

Pelo método adotado, e devido à crescente carga de informação que a clínica tem que armazenar, a migração dos dados, resultante da execução da solução desenvolvida, foi realizada de forma fácil e sem complicações. A sua existência introduziu novas possibilidades de escalabilidade, sem introduzir as limitações características associadas.

Esta base de dados criada vai impulsionar a ascensão da *Clinica Alta Performance* uma vez que conseguirá tirar proveito de uma arquitetura feita para o nível da clínica. Assim esta irá estar melhor preparada para resolver problemas que encontrará no dia-a-dia da sua utilização.

O modo como foi implementada a base de dados permitirá que esta a qualquer momento possa ser expandida com a implementação de novas funcionalidade de um modo simples e seguro, como por exemplo para guardar dados referentes à clínica, a

testes clínicos criados e até a implementação de novas outras clínicas em outros diferentes pontos do país.

Por fim, podemos com a realização deste trabalho concluir que as bases de dados são fundamentais para a gestão e tratamento da informação e constituem uma tecnologia sem a qual seria muito difícil o crescimento do mundo digital, principalmente nos dias de hoje com a quantidade de informação que é necessária gerar e armazenar a cada segundo.

Referências

Connolly, T. and Begg, C. (2005). Database Systems, A Practical Approach to Design, Implementation, and Management. 4th ed. Addison-Wesley.

Lista de Siglas e Acrónimos

BD	Base de Dados
SGBD	Sistema de Gestão de Base de Dados
SQL	<i>Structured Query Language</i>
ER	<i>Entity–Relationship</i>
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
NoSQL	Not only SQL

Anexos

- 1. Script Completo de Criação**
- 2. Script Parcial do Povoamento**
- 3. Queries em SQL**

I. Anexo 1 – Script Completo de Criação

```
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_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_EN
GINE_SUBSTITUTION';

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

-- -----
-- Table `ClinicaAltaPerformance`.`Morada`
-- -----
CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Morada` (
`idMorada` INT NOT NULL AUTO_INCREMENT,
`endereco` VARCHAR(100) NOT NULL,
`localidade` VARCHAR(100) NOT NULL,
PRIMARY KEY (`idMorada`))
ENGINE = InnoDB;

-- -----
-- Table `ClinicaAltaPerformance`.`Atleta`
-- -----
CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Atleta` (
`idAtleta` INT NOT NULL AUTO_INCREMENT,
`nome` VARCHAR(100) NOT NULL,
`dataNascimento` DATE NOT NULL,
`sexo` VARCHAR(5) NOT NULL,
`morada` INT NOT NULL,
PRIMARY KEY (`idAtleta`),
INDEX `fk_Atleta_Morada_idx` (`morada` ASC) VISIBLE,
CONSTRAINT `fk_Atleta_Morada`
FOREIGN KEY (`morada`)
REFERENCES `ClinicaAltaPerformance`.`Morada` (`idMorada`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `ClinicaAltaPerformance`.`Prova`
-- -----
CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Prova` (
`idProva` INT NOT NULL AUTO_INCREMENT,
`designacao` VARCHAR(100) NOT NULL,
`modalidade` VARCHAR(100) NOT NULL,
`categoria` VARCHAR(100) NOT NULL,
PRIMARY KEY (`idProva`))
```

```

ENGINE = InnoDB;

-- -----
-- Table `ClinicaAltaPerformance`.`Atleta_Prova`


CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Atleta_Prova` (
  `id_Atleta` INT NOT NULL,
  `id_Prova` INT NOT NULL,
  `data` DATE NOT NULL,
  INDEX `fk_Atleta_Prova_Modalidade_prova_idx` (`id_Prova` ASC) VISIBLE,
  PRIMARY KEY (`id_Prova`, `id_Atleta`),
  CONSTRAINT `fk_Atleta_Prova_Modalidade_atleta`
    FOREIGN KEY (`id_Atleta`)
      REFERENCES `ClinicaAltaPerformance`.`Atleta` (`idAtleta`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Atleta_Prova_Modalidade_prova`
    FOREIGN KEY (`id_Prova`)
      REFERENCES `ClinicaAltaPerformance`.`Prova` (`idProva`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `ClinicaAltaPerformance`.`Clinica`


CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Clinica` (
  `idClinica` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(100) NOT NULL,
  `morada` INT NOT NULL,
  PRIMARY KEY (`idClinica`),
  INDEX `fk_Clinica_Morada1_idx` (`morada` ASC) VISIBLE,
  CONSTRAINT `fk_Clinica_Morada1`
    FOREIGN KEY (`morada`)
      REFERENCES `ClinicaAltaPerformance`.`Morada` (`idMorada`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `ClinicaAltaPerformance`.`Medico`


CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Medico` (
  `idMedico` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(100) NOT NULL,
  `dataInicioServico` DATE NOT NULL,
  `dataNascimento` DATE NOT NULL,
  `especialidade` VARCHAR(100) NOT NULL,
  `idChefe` INT NULL,
  `idClinica` INT NOT NULL,
  `morada` INT NOT NULL,
  PRIMARY KEY (`idMedico`),
  INDEX `fk_Tecnico_Hospital_idx` (`idClinica` ASC) VISIBLE,
  INDEX `fk_Medico_Chefe_idx` (`idChefe` ASC) VISIBLE,

```

```

INDEX `fk_Medico_Morada1_idx` (`morada` ASC) VISIBLE,
CONSTRAINT `fk_Medico_Chefe`
FOREIGN KEY (`idChefe`)
REFERENCES `ClinicaAltaPerformance`.`Medico` (`idMedico`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Tecnico_Hospital`
FOREIGN KEY (`idClinica`)
REFERENCES `ClinicaAltaPerformance`.`Clinica` (`idClinica`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Medico_Morada1`
FOREIGN KEY (`morada`)
REFERENCES `ClinicaAltaPerformance`.`Morada` (`idMorada`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `ClinicaAltaPerformance`.`TesteClinico`

CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`TesteClinico` (
`idTesteClinico` INT NOT NULL AUTO_INCREMENT,
`designacao` VARCHAR(100) NOT NULL,
`dataHora` DATETIME NOT NULL,
`preco` DECIMAL(6,2) NOT NULL,
`resultado` VARCHAR(100) NULL,
`idMedico` INT NOT NULL,
`idAtleta` INT NOT NULL,
PRIMARY KEY (`idTesteClinico`),
INDEX `fk_TestClinico_Tecnico1_idx` (`idMedico` ASC) VISIBLE,
INDEX `fk_TestClinico_Atleta_idx` (`idAtleta` ASC) VISIBLE,
CONSTRAINT `fk_TestClinico_Tecnico1`
FOREIGN KEY (`idMedico`)
REFERENCES `ClinicaAltaPerformance`.`Medico` (`idMedico`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_TestClinico_Atleta`
FOREIGN KEY (`idAtleta`)
REFERENCES `ClinicaAltaPerformance`.`Atleta` (`idAtleta`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `ClinicaAltaPerformance`.`Prova_TipoTeste`

CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Prova_TipoTeste` (
`idProva` INT NOT NULL,
`designacao` VARCHAR(100) NOT NULL,
CONSTRAINT `fk_Prova_Teste_Prova`
FOREIGN KEY (`idProva`)
REFERENCES `ClinicaAltaPerformance`.`Prova` (`idProva`)

```

```

    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `ClinicaAltaPerformance`.`Contacto`


CREATE TABLE IF NOT EXISTS `ClinicaAltaPerformance`.`Contacto` (
  `idContacto` INT NOT NULL AUTO_INCREMENT,
  `telefono` INT NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `idAtleta` INT NOT NULL,
  PRIMARY KEY (`idContacto`, `idAtleta`),
  INDEX `fk_Contacto_Atleta1_idx` (`idAtleta` ASC) VISIBLE,
  CONSTRAINT `fk_Contacto_Atleta1`
    FOREIGN KEY (`idAtleta`)
    REFERENCES `ClinicaAltaPerformance`.`Atleta` (`idAtleta`)
    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 de Povoamento

-- Morada das clínicas

```
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (1,'Rua D.Afonso Henriques nº24','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (2,'Rua dos Pilares nº40','Braga');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (3,'Rua da Imaculada nº3','Porto');

-- Morada dos Atletas
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (4,'Rua Padre Andrade nº35','Bragança');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (5,'Rua D.Joao nº220','Esposende');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (6,'Rua Santo Antonio nº7','Algarve');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (7,'Rua Frei Luis nº133','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (8,'Rua dos Sete Andares nº1024','Coimbra');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (9,'Rua dos Leoes nº2','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (10,'Rua da Antiga Sociedade nº29','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (11,'Rua do Museu dos Congros nº44','Portimao');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (12,'Rua do Almada nº339','Porto');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (13,'Rua de Costa Cabral nº279','Porto');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (14,'Rua da Prata nº340','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (15,'Rua dos Moradores do Pascal nº75','Leiria');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (16,'Rua da Moderna História de Portugal nº42','Porto');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (17,'Rua do Loureiro nº62','Regua');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (18,'Rua Nova de S.Jumil nº165','Viana do Castelo');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (19,'Rua D.Jose I nº34','Braga');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (20,'Rua de São Mamede','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (21,'Rua Caetano de Oliveira nº22','Portalegre');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (22,'Rua Latino Coelho nº72','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (23,'Rua Santos Minho nº65','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (24,'Rua Primeiro de Maio nº349','Porto');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (25,'Rua da Carreira nº91','Braga');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (26,'Rua ddas Aranhas nº82','Fatima');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (27,'Rua dos Netos nº445','Portimao');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (28,'Rua Dr.Fernao de Ornelas nº245','Viana do Castelo');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (29,'Rua do Rosario nº256','Porto');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (30,'Rua das Taipas nº320','Santarem');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (31,'Rua da Esperanca nº56','Leiria');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (32,'Rua da Pereira nº243','Esposende');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (33,'Rua do Moinho nº26','Braga');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (34,'Rua do Outeiro nº561','Viana do Castelo');

-- Morada dos médicos
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (35,'Rua do Carmo nº65','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (36,'Rua dos Fanqueiros nº400','Braga');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (37,'Rua da Madalena nº302','Porto');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (38,'Rua da Oliveira ao Carmo nº359','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (39,'Rua Ferreira Lapa nº221','Braga');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (40,'Rua do Salitre nº74','Porto');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (41,'Rua Garrett nº3','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (42,'Rua de São Juliao nº124','Braga');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (43,'Rua da Betesga nº27','Porto');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (44,'Rua Cidade de Bolama nº621','Lisboa');
```

```

INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (45,'Rua do Guarda Mor nº442','Braga');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (46,'Rua do Comercio nº33','Porto');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (47,'Rua General Morais Sarmento nº89','Lisboa');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (48,'Rua da Bainharia nº40','Braga');
INSERT INTO ClinicaAltaPerformance.Morada (idMorada,endereço,localidade) VALUES (49,'Rua dos Clerigos nº88','Porto');

-- Atletas
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (1,'Paulo Silva','1992-12-13','M',4);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (2,'Mariana Ribeiro','1998-11-17','F',7);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (3,'Beatriz Costa','1999-12-04','F',20);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (4,'Rui Alves','1998-11-19','M',13);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (5,'Carla Ribeiro','1998-07-12','F',10);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (6,'Paula Cardoso','2003-07-21','F',9);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (7,'Margarida Soares','1994-07-05','F',19);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (8,'Hugo Silva','2000-03-27','M',12);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (9,'António Ribeiro','1999-03-08','M',10);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (10,'Joana Pinto','1998-03-14','F',17);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (11,'João Camões','1997-10-08','M',14);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (12,'Carlos Sá','2001-12-07','M',8);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (13,'Joana Guimarães','1993-01-10','F',18);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (14,'Rita Azevedo','1992-06-03','F',15);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (15,'Carlos Costa','2003-05-15','M',20);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (16,'Ricardo Silva','1997-01-19','M',16);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (17,'Rui Ramos','1993-02-25','M',6);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (18,'Catarina Neves','1992-07-13','F',32);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (19,'Marta Oliveira','1993-06-04','F',12);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (20,'Gil Fernandes','2003-06-22','M',5);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (21,'Rodrigo Pinheiro','1999-02-14','M',24);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (22,'João Oliveira','1999-03-08','M',22);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (23,'Ruben Mota','1999-11-07','M',21);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (24,'Rafaela Paz','1992-02-10','F',29);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (25,'Luisa Silva','1993-03-03','F',30);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (26,'Gustavo Oliveira','2001-08-05','M',34);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (27,'Raul Martins','1995-08-09','M',23);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (28,'Samuel Prata','1993-07-26','M',31);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (29,'Silvia Braz','1993-07-23','F',25);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (30,'Daniela dos Santos','1994-09-30','F',11);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (31,'Filipa Alves','2004-06-12','F',27);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (32,'João dos Santos','1999-03-31','M',11);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (33,'Tomás Pereira','1992-10-17','M',32);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (34,'Telma Monteiro','1997-06-13','F',28);
INSERT INTO ClinicaAltaPerformance.Atleta (idAtleta,nome,dataNascimento,sexo,morada) VALUES (35,'Nicole Pinho','1999-06-07','F',33);

-- Clínicas
INSERT INTO ClinicaAltaPerformance.Clinica (idClinica,nome,morada) VALUES (2,'Clinica Alta Performance Lisboa',1);
INSERT INTO ClinicaAltaPerformance.Clinica (idClinica,nome,morada) VALUES (3,'Clinica Alta Performance Braga',2);
INSERT INTO ClinicaAltaPerformance.Clinica (idClinica,nome,morada) VALUES (1,'Clinica Alta Performance Porto',3);

-- Médicos
-- LISBOA

```

```

INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES (1,'João Correia', '1992-04-12', '1966-05-02','Cardiologia',1,1,35);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES (2,'Tiago Ferreira', '1996-12-19', '1971-04-04','Analises Clinicas',1,1,39);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES (3,'Cristina Mota', '1995-07-15', '1974-09-08','Analises Clinicas',1,1,41);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES (4,'Raquel Fonseca', '1996-10-02', '1970-09-17','Oftamologia',1,1,44);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES (5,'Ricardo Reis', '1995-02-25', '1968-05-21','Clinica Geral',1,1,47);
-- BRAGA
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES (6,'Alexandra Fonte', '1993-07-10', '1965-03-09','Cardiologia',6,2,36);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES (7,'Rui Cunha', '1997-11-09', '1972-12-01','Analises Clinicas',6,2,39);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES (8,'Tiago Azevedo', '1996-03-11', '1973-09-16','Analises Clinicas',6,2,42);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES(9,'Rita Silva', '1994-11-22', '1971-08-08','Oftamologia',6,2,45);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES(10,'Renato Mendes', '1997-04-21', '1970-06-24','Clinica Geral',6,2,48);
-- PORTO
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES(11,'Diogo Neves', '1991-02-24', '1965-06-29','Cardiologia',11,3,37);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES(12,'Gustavo Santos', '1995-11-12', '1970-03-30','Analises Clinicas',11,3,40);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES(13,'Manuel Barros', '1994-03-29', '1970-08-17','Analises Clinicas',11,3,43);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES(14,'Catarina Tomás', '1994-11-03', '1968-05-10','Oftamologia',11,3,46);
INSERT INTO ClinicaAltaPerformance.Medico (idMedico,nome,dataInicioServico,dataNascimento,especialidade,idChefe,idClinica,morada)
VALUES(15,'Teresa Alves', '1998-12-04', '1971-06-22','Clinica Geral',11,3,49);
-- Contactos
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (1,937319002, 'paulo@gmail.com',1);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (2,22240671, 'paulosilva@gmail.com',1);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (3,917311232, 'marian@gmail.com',2);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (4,927001232, 'bea@gmail.com',3);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (5,926018248, 'beaaa@gmail.com',3);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (6,932366002, 'rruji@gmail.com',4);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (7,921749203, 'carl@gmail.com',5);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (8,938471100, 'paula@gmail.com',6);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (9,913334291, 'mags@gmail.com',7);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (10,229934244, 'maggss@gmail.com',7);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (11,918883454, 'hugo@gmail.com',8);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (12,912218586, 'antonio@gmail.com',9);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (13,929879871, 'joan@gmail.com',10);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (14,937786001, 'joao@gmail.com',11);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (15,923314987, 'carlossa@gmail.com',12);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (16,223143209, 'carlosss@gmail.com',12);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (17,932212218, 'joanag@gmail.com',13);
INSERT INTO ClinicaAltaPerformance.Contacto (idContacto,telefone,email,idAtleta) VALUES (18,911912762, 'ritaazeve@gmail.com',14);

```

```

INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (19,925611210, 'rit@gmail.com',14);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (20,223120098, 'ritaazevedo@gmail.com',14);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (21,919982345, 'carlocarlos@gmail.com',15);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (22,927612909, 'ricardoo@gmail.com',16);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (23,917892231, 'ruramatos@gmail.com',17);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (24,931173002, 'catarinn@gmail.com',18);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (25,926563494, 'martaoliveira@gmail.com',19);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (26,931212998, 'gill@gmail.com',20);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (27,224532123, 'gilfernandes@gmail.com',20);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (28,935740023, 'rodrigop@gmail.com',21);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (29,920203440, 'joaooliveira@gmail.com',22);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (30,931200009, 'rubenmota@gmail.com',23);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (31,929981202, 'rafael@gmail.com',24);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (32,919926712, 'luisa@gmail.com',25);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (33,930912177, 'gustav@gmail.com',26);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (34,229923531, 'raul@gmail.com',27);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (35,932657766, 'samuelp@gmail.com',28);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (36,924024434, 'silvia@gmail.com',29);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (37,920039911, 'danielaa@gmail.com',30);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (38,911231005, 'pips@gmail.com',31);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (39,939821023, 'joaoo@gmail.com',32);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (40,923546282, 'tomass@gmail.com',33);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (41,910923423, 'telmam@gmail.com',34);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (42,927761230, 'nicolepinto@gmail.com',35);
INSERT INTO ClinicaAltaPerformance.Contato (idContato,telefone,email,idAtleta) VALUES (43,934540912, 'nicolep@gmail.com',35);

-- TESTES CLÍNICOS REALIZADOS

-- LISBOA :DOPPING

INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(1,'Dopping','2019-12-12 15:30',5.0,'Negativo',2,22);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(2,'Dopping','2019-12-12 16:30',5.0,'Negativo',2,2);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(3,'Dopping','2019-12-12 16:30',5.0,'Negativo',3,3);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(4,'Dopping','2019-12-12 17:30',5.0,'Positivo',2,5);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(5,'Dopping','2019-12-12 17:30',5.0,'Negativo',3,6);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(6,'Dopping','2019-12-12 18:30',5.0,'Negativo',2,11);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(7,'Dopping','2019-06-12 09:30',5.0,'Negativo',2,22);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(8,'Dopping','2019-06-12 10:30',5.0,'Negativo',2,2);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(9,'Dopping','2019-06-12 10:30',5.0,'Negativo',3,3);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(10,'Dopping','2019-06-12 11:30',5.0,'Positivo',2,5);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(11,'Dopping','2019-06-12 11:30',5.0,'Negativo',3,6);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(12,'Dopping','2019-06-12 13:30',5.0,'Positivo',2,11);

```

```
-- BRAGA :DOPPING
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(13,'Dopping','2019-12-14 16:30',5.0,'Negativo',7,1);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(14,'Dopping','2019-12-14 16:30',5.0,'Negativo',8,7);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(15,'Dopping','2019-12-14 17:30',5.0,'Positivo',7,12);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(16,'Dopping','2019-12-14 17:30',5.0,'Positivo',8,29);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(17,'Dopping','2019-12-14 18:30',5.0,'Negativo',7,35);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(18,'Dopping','2019-12-14 19:30',5.0,'Negativo',7,13);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(19,'Dopping','2019-06-13 10:30',5.0,'Negativo',7,1);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(20,'Dopping','2019-06-13 10:30',5.0,'Positivo',8,7);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(21,'Dopping','2019-06-13 11:30',5.0,'Positivo',7,12);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(22,'Dopping','2019-06-13 11:30',5.0,'Negativo',8,29);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(23,'Dopping','2019-06-13 13:30',5.0,'Positivo',7,35);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(24,'Dopping','2019-06-13 14:30',5.0,'Negativo',7,13);

-- PORTO :DOPPING
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(25,'Dopping','2019-12-16 16:30',5.0,'Negativo',12,4);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(26,'Dopping','2019-12-16 16:30',5.0,'Negativo',13,8);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(27,'Dopping','2019-12-16 17:30',5.0,'Positivo',12,16);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(28,'Dopping','2019-12-16 17:30',5.0,'Positivo',13,19);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(29,'Dopping','2019-12-16 18:30',5.0,'Negativo',12,21);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(30,'Dopping','2019-06-16 10:30',5.0,'Negativo',12,4);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(31,'Dopping','2019-06-16 10:30',5.0,'Positivo',13,8);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(32,'Dopping','2019-06-16 11:30',5.0,'Positivo',12,16);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(33,'Dopping','2019-06-16 11:30',5.0,'Negativo',13,19);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(34,'Dopping','2019-06-16 13:30',5.0,'Positivo',12,21);

-- BRAGA :PROVA ESFORCO
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(35,'Prova_Esforco','2019-07-16 10:30',5.0,'Valores aceitáveis',10,7);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
```

```

(36,'Prova_Esforco','2019-07-15 10:30',5.0,'Valores a controlar',10,18);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(37,'Prova_Esforco','2019-07-16 12:30',5.0,'Valores aceitáveis',10,29);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(38,'Prova_Esforco','2019-07-16 09:30',5.0,'Valores aceitáveis',10,26);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(39,'Prova_Esforco','2019-07-16 08:30',5.0,'Valores aceitáveis',10,20);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(40,'Prova_Esforco','2019-07-16 09:00',5.0,'Valores aceitáveis',10,13);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
-- BRAGA :ANALISE SANGUINEA
(41,'Analise_sanguinea','2019-07-16 13:30',5.0,'Valores a controlar',7,7);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(42,'Analise_sanguinea','2019-07-15 13:30',5.0,'Valores a controlar',8,18);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(43,'Analise_sanguinea','2019-07-16 14:30',5.0,'Valores aceitáveis',7,29);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(44,'Analise_sanguinea','2019-07-16 15:30',5.0,'Valores aceitáveis',7,26);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(45,'Analise_sanguinea','2019-07-16 16:00',5.0,'Valores aceitáveis',7,20);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(46,'Analise_sanguinea','2019-07-17 16:30',5.0,'Valores aceitáveis',7,13);
-- LISBOA :PROVA ESFORCO
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(47,'Prova_Esforco','2019-07-16 10:30',5.0,'Valores aceitáveis',5,9);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(48,'Prova_Esforco','2019-07-16 11:30',5.0,'Valores aceitáveis',5,11);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(49,'Prova_Esforco','2019-07-16 12:30',5.0,'Valores aceitáveis',5,14);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(50,'Prova_Esforco','2019-07-16 13:00',5.0,'Valores a controlar',5,15);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(51,'Prova_Esforco','2019-07-17 12:00',5.0,'Valores a controlar',5,25);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(52,'Prova_Esforco','2019-07-17 13:00',5.0,'Valores aceitáveis',5,27);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(53,'Prova_Esforco','2019-07-17 14:00',5.0,'Valores a controlar',5,28);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(54,'Prova_Esforco','2019-07-18 13:00',5.0,'Valores aceitáveis',5,30);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(55,'Prova_Esforco','2019-07-19 14:00',5.0,'Valores aceitáveis',5,31);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(56,'Prova_Esforco','2019-07-20 15:00',5.0,'Valores aceitáveis',5,32);
-- LISBOA : ANALISE SANGUINEA
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(57,'Analise_sanguinea','2019-08-16 10:30',5.0,'Valores aceitáveis',2,9);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(58,'Analise_sanguinea','2019-08-16 11:30',5.0,'Valores aceitáveis',3,11);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(59,'Analise_sanguinea','2019-08-16 12:30',5.0,'Valores aceitáveis',2,14);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES

```

```

(60,'Analise_sanguinea','2019-08-16 13:00',5.0,'Valores a controlar',3,15);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(61,'Analise_sanguinea','2019-08-17 12:00',5.0,'Valores a controlar',3,25);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(62,'Analise_sanguinea','2019-08-17 13:00',5.0,'Valores aceitáveis',2,27);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(63,'Analise_sanguinea','2019-08-17 14:00',5.0,'Valores a controlar',3,28);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(64,'Analise_sanguinea','2019-08-18 13:00',5.0,'Valores aceitáveis',3,30);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(65,'Analise_sanguinea','2019-08-19 14:00',5.0,'Valores aceitáveis',2,31);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(66,'Analise_sanguinea','2019-08-20 15:00',5.0,'Valores aceitáveis',2,32);
-- PORTO :PROVA ESFORCO
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(67,'Prova_Esforco','2019-09-20 10:30',5.0,'Valores aceitáveis',15,10);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(68,'Prova_Esforco','2019-09-20 11:30',5.0,'Valores a controlar',15,12);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(69,'Prova_Esforco','2019-09-20 12:30',5.0,'Valores aceitáveis',15,16);
-- PORTO :ANALISE SANGUINEA
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(70,'Analise_sanguinea','2019-09-20 10:00',5.0,'Valores aceitáveis',12,10);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(71,'Analise_sanguinea','2019-09-20 11:00',5.0,'Valores aceitáveis',12,12);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(72,'Analise_sanguinea','2019-09-20 12:00',5.0,'Valores aceitáveis',13,16);
-- TESTES AGENDADOS
-- PORTO
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(73,'Eletrocardiograma','2020-04-16 10:30',10.0,'N\o',11,19);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(74,'Raio_X','2020-04-16 11:30',25.0,'N\o',11,19);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(75,'Ecocardiograma','2020-04-16 09:30',25.0,'N\o',11,19);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(76,'Eletrocardiograma','2020-04-17 10:30',10.0,'N\o',11,24);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(77,'Raio_X','2020-04-17 11:30',25.0,'N\o',11,24);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(78,'Ecocardiograma','2020-04-17 09:30',25.0,'N\o',11,24);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(79,'Óptica','2020-04-18 09:30',15.0,'N\o',14,24);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(80,'Analise_sanguinea','2020-04-16 12:30',5.0,'N\o',12,4);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(81,'Prova_Esforco','2020-04-16 11:30',5.0,'N\o',15,4);
-- BRAGA
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(82,'Eletrocardiograma','2020-04-16 10:30',10.0,'N\o',6,29);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES

```

```

(83,'Raio_X','2020-04-16 11:30',25.0,'N\đ',6,29);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(84,'Ecocardiograma','2020-04-16 09:30',25.0,'N\đ',6,29);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(85,'Eletrocardiograma','2020-04-17 10:30',10.0,'N\đ',6,35);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(86,'Raio_X','2020-04-17 11:30',25.0,'N\đ',6,35);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(87,'Ecocardiograma','2020-04-17 09:30',25.0,'N\đ',6,35);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(88,'Óptica','2020-04-18 09:30',15.0,'N\đ',9,35);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(89,'Analise_sanguinea','2020-04-16 16:30',5.0,'N\đ',8,29);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(90,'Prova_Esforço','2020-04-16 17:30',5.0,'N\đ',10,29);
-- LISBOA
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(91,'Eletrocardiograma','2020-04-16 10:30',10.0,'N\đ',1,15);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(92,'Raio_X','2020-04-16 11:30',25.0,'N\đ',1,15);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(93,'Ecocardiograma','2020-04-16 09:30',25.0,'N\đ',1,15);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(94,'Eletrocardiograma','2020-04-17 10:30',10.0,'N\đ',1,3);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(95,'Raio_X','2020-04-17 11:30',25.0,'N\đ',1,3);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(96,'Ecocardiograma','2020-04-17 09:30',25.0,'N\đ',1,3);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(97,'Óptica','2020-04-18 09:30',15.0,'N\đ',4,3);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(98,'Analise_sanguinea','2020-04-16 16:30',5.0,'N\đ',2,27);
INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES
(99,'Prova_Esforço','2020-04-16 17:30',5.0,'N\đ',5,27);
-- POVOAR PROVAS
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(1,'60 metros', 'Velocidade', 'Menores_18');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(2,'100 metros', 'Velocidade', 'Menores_18');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(3,'200 metros', 'Velocidade', 'Menores_18');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(4,'400 metros', 'Velocidade', 'Menores_18');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(5,'100 metros', 'Barreiras', 'Menores_18');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(6,'400 metros', 'Barreiras', 'Menores_18');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(7,'60 metros', 'Velocidade', 'Entre_18_24');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(8,'100 metros', 'Velocidade', 'Entre_18_24');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(9,'200 metros', 'Velocidade', 'Entre_18_24');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(10,'400 metros', 'Velocidade', 'Entre_18_24');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(11,'100 metros', 'Barreiras', 'Entre_18_24');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(12,'400 metros', 'Barreiras', 'Entre_18_24');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(13,'60 metros', 'Velocidade', 'Maiores_24');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(14,'100 metros', 'Velocidade', 'Maiores_24');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(15,'200 metros', 'Velocidade', 'Maiores_24');
INSERT INTO ClinicaAltaPerformance.Prova(idProva,designacao,modalidade,categoria) VALUES(16,'400 metros', 'Velocidade', 'Maiores_24');

```



```
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (11,'Prova_Esforco');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (11,'Analise_sanguinea');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (12,'Prova_Esforco');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (12,'Analise_sanguinea');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (13,'Prova_Esforco');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (13,'Analise_sanguinea');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (14,'Prova_Esforco');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (14,'Analise_sanguinea');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (15,'Prova_Esforco');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (15,'Analise_sanguinea');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (16,'Prova_Esforco');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (16,'Analise_sanguinea');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (17,'Prova_Esforco');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (17,'Analise_sanguinea');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (18,'Prova_Esforco');
INSERT INTO ClinicaAltaPerformance.Prova_TipoTeste (idProva, designacao) VALUES (18,'Analise_sanguinea');
```

III. Anexo 3 – Queries em SQL

-- Requisito de exploração 1: Obter o número de pacientes de todas as clínicas

```
SELECT count(*) FROM Atleta;
```

-- Requisito de exploração 2: Consultar os testes realizados antes de determinada prova para um dado atleta ordenados do mais recente para o mais antigo

```
SELECT t.designacao as Designação, t.dataHora as "Data e Hora" FROM TesteClinico t
INNER JOIN Atleta_Prova ap ON ap.data > t.dataHora
WHERE ap.id_Atleta = t.idAtleta AND id_Atleta = 1
ORDER BY t.dataHora;
```

-- Requisito de exploração 3: Consultar os nomes dos atletas de determinada modalidade que acusaram positivo nos testes de doping

```
SELECT DISTINCT a.nome
FROM(SELECT ap.id_Atleta
      FROM Atleta_Prova ap
      INNER JOIN Prova p ON p.idProva = ap.id_Prova WHERE p.modalidade = "velocidade") as partial_table
INNER JOIN TesteClinico t ON t.idAtleta = partial_table.id_Atleta
INNER JOIN Atleta a ON a.idAtleta = partial_table.id_Atleta
WHERE designacao = "dopping" AND resultado = "positivo";
```

-- Requisito de exploração 4: Consultar os nomes dos pacientes/atletas que foram submetidos a determinado teste

```
SELECT a.nome as Nome FROM Atleta a
INNER JOIN TesteClinico t ON t.idAtleta = a.idAtleta
WHERE t.designacao = "Analise_sanguinea";
```

-- Requisito de exploração 5: Consultar o número de testes clínicos realizados por um dado médico num determinado ano e o montante faturado

```
SELECT count(*), sum(preco) as Faturado FROM TesteClinico t
WHERE t.idMedico = (SELECT m.idMedico FROM Medico m where m.nome = "Tiago Ferreira")
AND YEAR(t.dataHora) = 2019;
```

-- Requisito de exploração 6: Consultar agendamentos para um determinado tipo de exame numa determinada clínica

```
SELECT t.designacao as Designação, t.dataHora as "Data e Hora", a.nome as Nome FROM TesteClinico t
INNER JOIN Medico m ON m.idMedico = t.idMedico
INNER JOIN Atleta a ON a.idAtleta = t.idAtleta
WHERE t.designacao = "Eletrocardiograma" AND t.dataHora > CURRENT_TIME() AND m.idClinica = 1;
```

-- Requisito de exploração 7: Obter o número de médicos de uma determinada especialidade numa determinada clínica

```
SELECT count(*) FROM Medico WHERE especialidade = "Cardiologia" AND idClinica = 3;
```

-- Requisito de exploração 8: Obter o top 3 atletas que gastaram mais dinheiro em exames

```
SELECT SUM(t.preco) AS "Total Gasto", a.nome AS Atleta FROM TesteClinico t
INNER JOIN Atleta a ON t.idAtleta = a.idAtleta
GROUP BY a.idAtleta
ORDER BY SUM(t.preco) DESC
LIMIT 3;
```

-- Requisito de exploração 9: Obter o top 5 atletas que realizaram mais exames

```
SELECT count(*) AS "Teste Realizados", a.nome AS Atleta FROM TesteClinico t
```

```

INNER JOIN Atleta a ON t.idAtleta = a.idAtleta
GROUP BY a.idAtleta
ORDER BY count(*) DESC
LIMIT 5;

-- Requisito de exploração 10: Obter o top 3 médicos de determinada especialidade que mais faturaram
SELECT m.nome AS "Nome", sum(t.preco) as "Faturado" FROM Medico m
INNER JOIN TesteClinico t ON m.idMedico = t.idMedico
WHERE m.especialidade = "Cardiologia"
GROUP BY m.idMedico
ORDER BY (sum(t.preco)) DESC
LIMIT 3;

-- Requisito de exploração 11: Obter a designação e resultado dos últimos 5 testes a que um atleta foi submetido
SELECT a.nome AS "Nome", t.designacao as "Designação", t.resultado as "Resultado" FROM Atleta a
INNER JOIN TesteClinico t ON a.idAtleta = t.idAtleta
WHERE t.dataHora < CURRENT_TIME() AND a.idAtleta = 1
ORDER BY t.dataHora DESC
LIMIT 5;

-- Requisito de exploração 12: Consultar a lista de exames obrigatórios para poder concorrer em determinada prova
SELECT ptt.designacao AS "Testes" FROM Prova_TipoTeste ptt
INNER JOIN Prova p ON p.idProva = ptt.idProva
WHERE ptt.idProva = 1;

-- Requisito de exploração 13: Consultar a designação das provas realizadas por determinado atleta
SELECT p.designacao AS "Tipo de provas realizadas" FROM Prova p
INNER JOIN Atleta_Prova ap ON ap.id_Prova = p.idProva
INNER JOIN Atleta a ON ap.id_Atleta = a.idAtleta
WHERE a.idAtleta = 1;

-- Requisito de exploração 14: Obter os atletas de uma dada localidade
SELECT a.idAtleta, a.nome AS "Nome" FROM Atleta a
INNER JOIN Morada m ON m.idMorada = a.morada
WHERE localidade = 'Braga';

-- Requisito de exploração 15: Obter o número de testes clínicos realizados numa determinada clínica no ano 2019
SELECT count(*) as "Testes Realizados" FROM TesteClinico t
INNER JOIN Medico m ON t.idMedico = m.idMedico
WHERE year(t.dataHora) = 2019 AND m.idClinica = 1;

```