

**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**

Janeiro, 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**

Janeiro, 2020

# Resumo

Este relatório foi realizado no âmbito do desenvolvimento de uma Base de Dados para a *Clínica 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](#_Toc28894977)

[Índice ii](#_Toc28894978)

[Índice de Figuras v](#_Toc28894979)

[Índice de Tabelas vii](#_Toc28894980)

[1. Introdução 1](#_Toc28894981)

[1.1. Contextualização 1](#_Toc28894982)

[1.2. Apresentação do Caso de Estudo 1](#_Toc28894983)

[1.3. Fundamentação da Implementação da Base de Dados 2](#_Toc28894984)

[1.4. Análise da viabilidade do projeto 2](#_Toc28894985)

[1.5. Estrutura do Relatório 2](#_Toc28894986)

[2. Levantamento e Análise de requisitos 4](#_Toc28894987)

[2.1. Método de Levantamento e de Análise de Requisitos Adotados 4](#_Toc28894988)

[2.2. Requisitos Levantados 4](#_Toc28894989)

[2.2.1 Requisitos de Descrição 4](#_Toc28894990)

[2.2.2 Requisitos de Exploração 5](#_Toc28894991)

[2.2.3 Requisitos de Controlo 5](#_Toc28894992)

[2.3. Análise de Requisitos 6](#_Toc28894993)

[2.4. Caraterização dos perfis de utilização 6](#_Toc28894994)

[3. Modelação Conceptual 7](#_Toc28894995)

[3.1. Apresentação da Abordagem de Modelação Realizada 7](#_Toc28894996)

[3.2. Identificação e Caracterização das Entidades 7](#_Toc28894997)

[3.3. Identificação e Caracterização dos Relacionamentos 8](#_Toc28894998)

[3.3.1 Dicionário de relacionamentos 11](#_Toc28894999)

[3.4. Identificação e Caracterização da Associação dos Atributos com as Entidades e Relacionamentos 12](#_Toc28895000)

[3.5. Chaves Candidatas, Primárias e Alternativas 13](#_Toc28895001)

[3.6. Detalhe ou Generalização de Entidades 13](#_Toc28895002)

[3.7. Apresentação e Explicação do Diagrama ER 14](#_Toc28895003)

[3.8. Validação do Modelo de Dados com o Utilizador 14](#_Toc28895004)

[4. Modelação Lógica 16](#_Toc28895005)

[4.1. Construção e Validação do Modelo de Dados Lógico 16](#_Toc28895006)

[4.1.1 Entidades fortes 16](#_Toc28895007)

[4.1.2 Relacionamentos 1:N 17](#_Toc28895008)

[4.1.3 Relacionamentos N:M 18](#_Toc28895009)

[4.1.4 Atributos Multivalorados 19](#_Toc28895010)

[4.2. Desenho do Modelo Lógico 20](#_Toc28895011)

[4.3. Validação do Modelo através da Normalização 20](#_Toc28895012)

[4.3.1 Primeira Forma Normal (1FN) 20](#_Toc28895013)

[4.3.2 Segunda Forma Normal (2FN) 21](#_Toc28895014)

[4.3.3 Terceira Forma Normal (3FN) 21](#_Toc28895015)

[4.4. Validação do Modelo com as Interrogações do Utilizador 21](#_Toc28895016)

[4.5. Validação do Modelo com as Transações Estabelecidas 24](#_Toc28895017)

[4.6. Verificação das Restrições de Integralidade 26](#_Toc28895018)

[4.7. Revisão do Modelo Lógico com o Utilizador 26](#_Toc28895019)

[5. Implementação Física 27](#_Toc28895020)

[5.1. Seleção do Sistema de Gestão de Base de Dados 27](#_Toc28895021)

[5.2. Tradução do Esquema Lógico para o Sistema de Gestão de Bases de Dados escolhido em SQL 27](#_Toc28895022)

[5.2.1 Desenho das relações base 27](#_Toc28895023)

[5.2.2 Desenho das restrições 31](#_Toc28895024)

[5.3. Tradução das Interrogações do Utilizador para SQL 34](#_Toc28895025)

[5.4. Tradução das transações estabelecidas para SQL 35](#_Toc28895026)

[5.5. Triggers SQL 38](#_Toc28895027)

[5.6. Escolha, Definição e Caracterização de Índices em SQL 38](#_Toc28895028)

[5.7. Estimativa do Espaço em Disco da Base de Dados e Taxa de Crescimento Anual 38](#_Toc28895029)

[5.8. Definição e Caracterização das vistas de utilização em SQL 42](#_Toc28895030)

[5.9. Definição e Caracterização dos Mecanismos de Segurança em SQL 43](#_Toc28895031)

[5.10. Revisão do Sistema Implementado com o Utilizador 44](#_Toc28895032)

[6. Sistema *NoSQL*: *Neo4j* 45](#_Toc28895033)

[6.1. Justificação da utilização de um sistema *NoSQL* 45](#_Toc28895034)

[6.2. Identificação dos objetivos da base de dados 45](#_Toc28895035)

[6.3. Identificação e explicação do tipo de questões (necessidades) que serão realizadas sobre o sistema de dados NoSQL 45](#_Toc28895036)

[6.4. Definição da estrutura base para o sistema de dados NoSQL que satisfaça os requisitos e as questões apresentadas anteriormente 46](#_Toc28895037)

[6.5. Identificação dos objetos de dados no sistema SQL que serão utilizados para alimentar o novo sistema 48](#_Toc28895038)

[6.6. Mapeamento do processo de migração de dados, descrevendo o processo de conversão dos vários objetos de dados 49](#_Toc28895039)

[6.7. Explicação do processo de migração de dados, explicando de forma detalhada as suas principais etapas - extração, transformação e carregamento 50](#_Toc28895040)

[6.8. Apresentação e descrição da implementação do processo de migração de dados 50](#_Toc28895041)

[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 53](#_Toc28895042)

[8. Conclusões e Trabalho Futuro 55](#_Toc28895043)

[Referências 57](#_Toc28895044)

[Lista de Siglas e Acrónimos 58](#_Toc28895045)

[Anexos 59](#_Toc28895046)

[I. Anexo 1 – Script Completo de Criação 60](#_Toc28895047)

[II. Anexo 2 – Script Parcial de Povoamento 64](#_Toc28895048)

[III. Anexo 3 – Queries em SQL 66](#_Toc28895049)

# Índice de Figuras

[Figura 1 - Relacionamento Atleta - Teste Clínico 8](#_Toc28895946)

[Figura 2 - Relacionamento Teste Clínico – Médico 9](#_Toc28895947)

[Figura 3 - Relacionamento Médico - Clínica 10](#_Toc28895948)

[Figura 4 - Relacionamento Atleta - Prova 11](#_Toc28895949)

[Figura 5 - Modelo Conceptual 14](file:////Users/mariapires/Desktop/BD/relatorio.docx#_Toc28895950)

[Figura 6 - Modelo Lógico 20](#_Toc28895951)

[Figura 7 - Árvore representativa da Interrogação nº1 22](#_Toc28895952)

[Figura 8 - Árvore representativa da Interrogação nº2 22](#_Toc28895953)

[Figura 9 - Árvore representativa da Interrogação nº3 23](#_Toc28895954)

[Figura 10 - Árvore representativa da Interrogação nº4 23](#_Toc28895955)

[Figura 11 - Árvore representativa da Interrogação nº5 24](#_Toc28895956)

[Figura 12 - Mapa de Transações para registar um atleta 25](#_Toc28895957)

[Figura 13 - Tabela TesteClinico 25](#_Toc28895958)

[Figura 14 - Mapa de Transações para adicionar uma prova 25](#_Toc28895959)

[Figura 15 - Criação da tabela Atleta 31](#_Toc28895960)

[Figura 16 - Criação da tabela TesteClinico 31](#_Toc28895961)

[Figura 17 - Criação da Tabela Médico 32](#_Toc28895962)

[Figura 18 - Criação da tabela Clínica 32](#_Toc28895963)

[Figura 19 - Criação da tabela Contacto 32](#_Toc28895964)

[Figura 20 - Criação da Tabela Prova\_TipoTeste 33](#_Toc28895965)

[Figura 21 - Criação da tabela Prova 33](#_Toc28895966)

[Figura 22 - Criação da tabela Atleta\_Prova 33](#_Toc28895967)

[Figura 23 - Criação da tabela Morada 33](#_Toc28895968)

[Figura 24 - Resolução da interrogação nº1 34](#_Toc28895969)

[Figura 25 - Resolução da interrogação nº2 34](#_Toc28895970)

[Figura 26 - Resolução da interrogação nº3 35](#_Toc28895971)

[Figura 27 - Resolução da interrogação nº4 35](#_Toc28895972)

[Figura 28 - Resolução da interrogação nº5 35](#_Toc28895973)

[Figura 29 - Inserir um novo atleta na base de dados 36](#_Toc28895974)

[Figura 30 - Inserir provas do atleta 36](#_Toc28895975)

[Figura 31 - Inserir um agendamento na base de dados 37](file:////Users/mariapires/Desktop/BD/relatorio.docx#_Toc28895976)

[Figura 32 - Inserir o registo de um novo tipo de prova 37](#_Toc28895977)

[Figura 33 - Imagem do código obtido para o trigger 38](#_Toc28895978)

[Figura 34 - View de agendamentos 42](#_Toc28895979)

[Figura 35 – View dos exames realizados 42](#_Toc28895980)

[Figura 36 – View dos medicos e as suas especialidades 43](#_Toc28895981)

[Figura 37 - View dos dados dos pacientes 43](#_Toc28895982)

[Figura 38 - Permissões dos diferentes utilizadores 44](#_Toc28895983)

[Figura 39 - Estrutura Neo4j 46](#_Toc28895984)

[Figura 40 - Ilustração da base de dados NoSQL criada em Neo4j 47](#_Toc28895985)

[Figura 41 - Ilustração da base de dados NoSQL criada em Neo4j 47](#_Toc28895986)

[Figura 42 - Ilustração da base de dados NoSQL criada em Neo4j 48](#_Toc28895987)

# Índice de Tabelas

[Tabela 1 – Dicionário de dados das entidades 8](#_Toc28894957)

[Tabela 2 - Dicionário de relacionamentos 11](#_Toc28894958)

[Tabela 3 - Caraterização de todos os atributos existentes 12](#_Toc28894959)

[Tabela 4 - Representação da entidade Atleta 16](#_Toc28894960)

[Tabela 5 - Representação da entidade TesteClinico 17](#_Toc28894961)

[Tabela 6 - Representação da entidade Medico 17](#_Toc28894962)

[Tabela 7- Representação da entidade Clínica 17](#_Toc28894963)

[Tabela 8 - Representação da entidade Prova 17](#_Toc28894964)

[Tabela 9 - Espaço ocupado em disco por cada tipo de dados 39](#_Toc28894965)

[Tabela 10 - Espaço ocupado pela tabela Atleta 39](#_Toc28894966)

[Tabela 11 - Espaço ocupado pela tabela Clínica 39](#_Toc28894967)

[Tabela 12 - Espaço ocupado pela tabela Médico 39](#_Toc28894968)

[Tabela 13 - Espaço ocupado pela tabela Teste Clínico 40](#_Toc28894969)

[Tabela 14 - Espaço ocupado pela tabela Prova 40](#_Toc28894970)

[Tabela 15 - Espaço ocupado pela tabela Morada 40](#_Toc28894971)

[Tabela 16 - Espaço ocupado pela tabela Contacto 40](#_Toc28894972)

[Tabela 17 - Espaço ocupado pela tabela Atleta\_Prova 41](#_Toc28894973)

[Tabela 18 - Espaço ocupado pela tabela Prova\_TipoTeste 41](#_Toc28894974)

[Tabela 19 - Espaço ocupado em disco pela população atual 41](#_Toc28894975)

[Tabela 20 - Espaço ocupado em disco pela população no futuro 42](#_Toc28894976)

1. Introdução
   1. Contextualização

Antes de iniciar a prática de qualquer atividade física é importante realizar uma avaliação médica detalhada, especialmente quando um individuo 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. 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. 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. 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. 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.

1. Levantamento e Análise de requisitos
   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.

* 1. Requisitos Levantados
     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;
    * 1. Requisitos de Exploração
13. Obter o número de pacientes de todas as clínicas;
14. Consultar os testes realizados antes de determinada prova para um dado atleta ordenados do mais recente para o mais antigo;
15. Consultar os nomes dos atletas de determinada modalidade que acusaram positivo nos testes de *doping*;
16. Consultar os nomes dos pacientes/atletas que foram submetidos a determinado teste;
17. Consultar o número de testes clínicos realizados por um dado médico num determinado ano e o montante faturado;
18. Consultar agendamentos para um determinado tipo de exame numa determinada clínica;
19. Obter o número de médicos de uma determinada especialidade numa determinada clínica;
20. Obter o top 3 atletas que gastaram mais dinheiro em exames;
21. Obter o top 5 atletas realizaram mais exames;
22. Obter o top 3 médicos de determinada especialidade que mais faturaram;
23. Obter a designação e resultado dos últimos 5 testes a que um atleta foi submetido;
24. Consultar a lista de exames obrigatórios para poder concorrer em determinada prova;
25. Consultar a designação das provas realizadas por determinado atleta;
26. Obter os atletas de uma determinada localidade;
27. Obter o número de testes clínicos realizados numa determinada clínica no ano 2019;
    * 1. Requisitos de Controlo
28. O médico apenas poderá consultar os aspetos do sistema que lhe estão direta ou indiretamente associados;
29. O médico poderá inserir e modificar o resultado dos testes clínicos por ele realizados;
30. 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;
31. A rececionista poderá agendar testes clínicos;
32. A rececionista poderá adicionar, editar e apagar atletas, testes clínicos agendados, contactos, moradas e provas de atletas do sistema;
33. O médico e a rececionista não poderão alterar propriedades do esquema da base de dados.
    1. 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 aspetos 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.

* 1. 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.

1. Modelação Conceptual
   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.

* 1. 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 entidadeMédico representa o funcionário responsável pela realização dos testes clínicos. É caraterizado 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 entidadeClínica representa cada uma das clínicas da empresa. É caraterizada pelo seu **id**, **nome** e **morada**.

**Prova**

A entidade prova refere-se às provas em que os atletas competem. É caraterizada 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

* 1. Identificação e Caracterização dos Relacionamentos

**Relacionamento: Atleta – Teste Clínico**

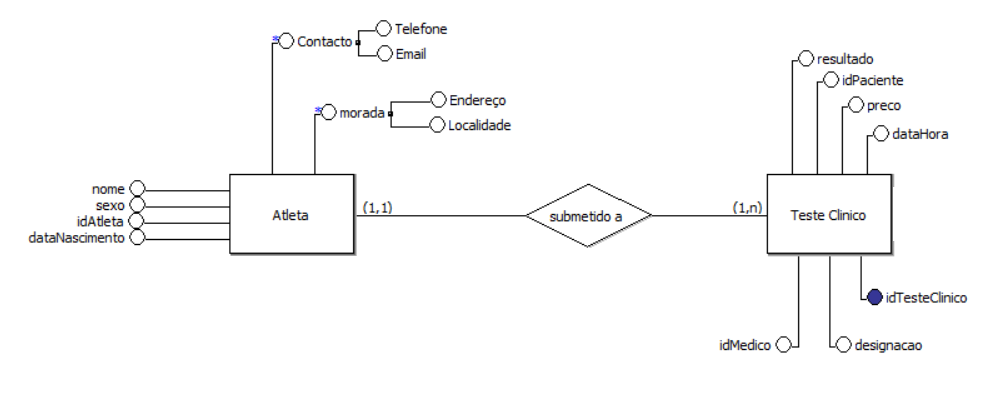


Figura 1 - Relacionamento Atleta - Teste Clínico

**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**

Uma imagem com mapa, texto

Descrição gerada automaticamente

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**

Uma imagem com texto

Descrição gerada automaticamente

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**

Uma imagem com texto, mapa

Descrição gerada automaticamente

Figura 4 - Relacionamento Atleta - Prova

**Relacionamento:** Atleta compete numa Prova

**Descrição:** Um Atleta pdoe 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.

* + 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

* 1. 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:  Email  Telefone |  | Composto |  |
| Email do atleta | VARCHAR(45) |
| Número de telefone do atleta | INT |
| Morada:  Endereço  Localidade |  | Composto |  |
| Endereço postal do atleta | VARCHAR(100) |
| 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  Endereço  Localidade |  | Composto |  |
| Endereço postal do médico | VARCHAR(100) |
| Localidade do médico | VARCHAR(100) |
| Especialidade | Área de especialização do médico | Simples |  |
| Data de Inicio de Serviço | Data de inicio de serviço do médico na clínica | Simples | DATE |
| IdHospital | Identificador do hospital | Simples | INT |
| **Clínica** | IdClinica | Identificador da clínica | Chave primária | INT |
| Nome | Nome da Clínica | Simples |  |
| Morada  Endereço  Localidade |  | Composto |  |
| Endereço postal da clínica | VARCHAR(100) |
| Localidade da clínica | VARCHAR(100) |
| **Prova** | 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

* 1. 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

* 1. Detalhe ou Generalização de Entidades

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

* 1. 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:

Uma imagem com texto, mapa

Descrição gerada automaticamente

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.

* 1. 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 respostas 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.

1. **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.

1. **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.

1. **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.

1. **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.

1. 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.

* 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.

* + 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 será́ 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

**TesteClínico** (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

* + 1. 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

* + 1. 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

* + 1. 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

* 1. Desenho do Modelo Lógico

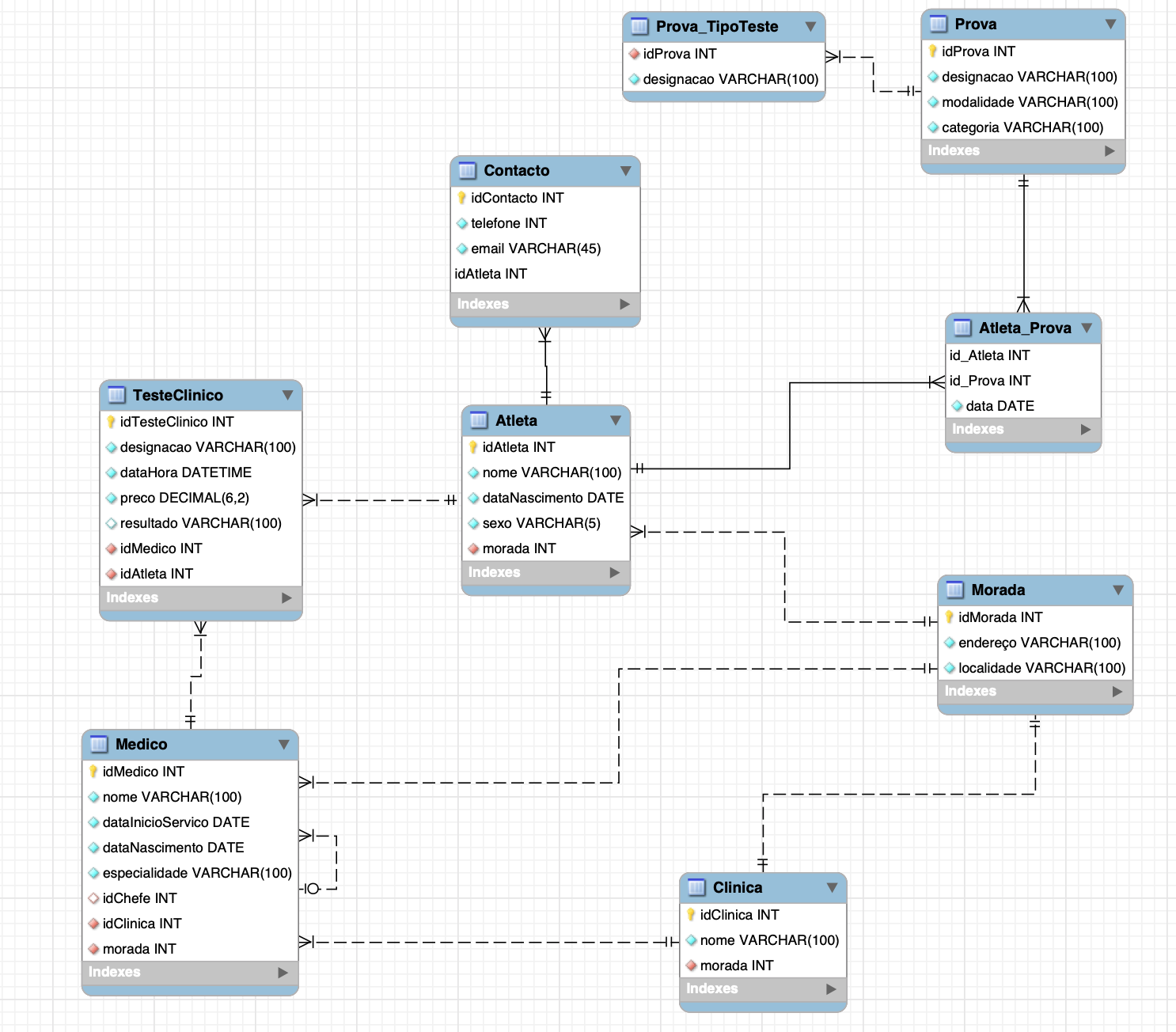


Figura 6 - Modelo Lógico

* 1. 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.

* + 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.

* + 1. 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 bse de dados não existe qualquer dependência funcional total em relação à chave primária. Assim, o nosso modelo está de acordo com a 2FN.

* + 1. 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 unicamente 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.

* 1. 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)**

Uma imagem com texto, mapa

Descrição gerada automaticamente

Figura 7 - Árvore representativa da Interrogação nº1

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

**Uma imagem com texto

Descrição gerada automaticamente**

Figura 8 - Árvore representativa da Interrogação nº2

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

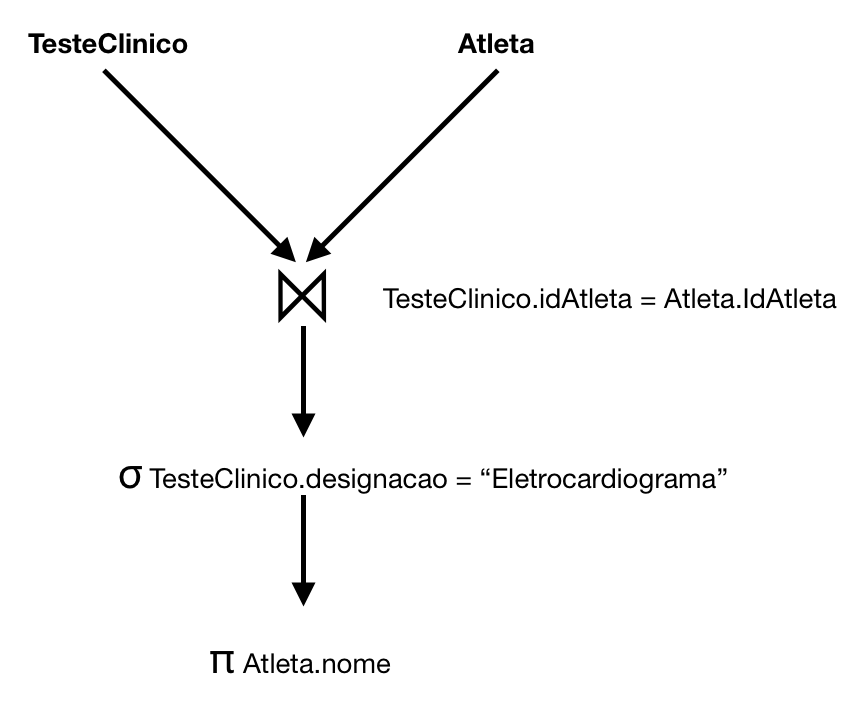


Figura 9 - Árvore representativa da Interrogação nº3

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

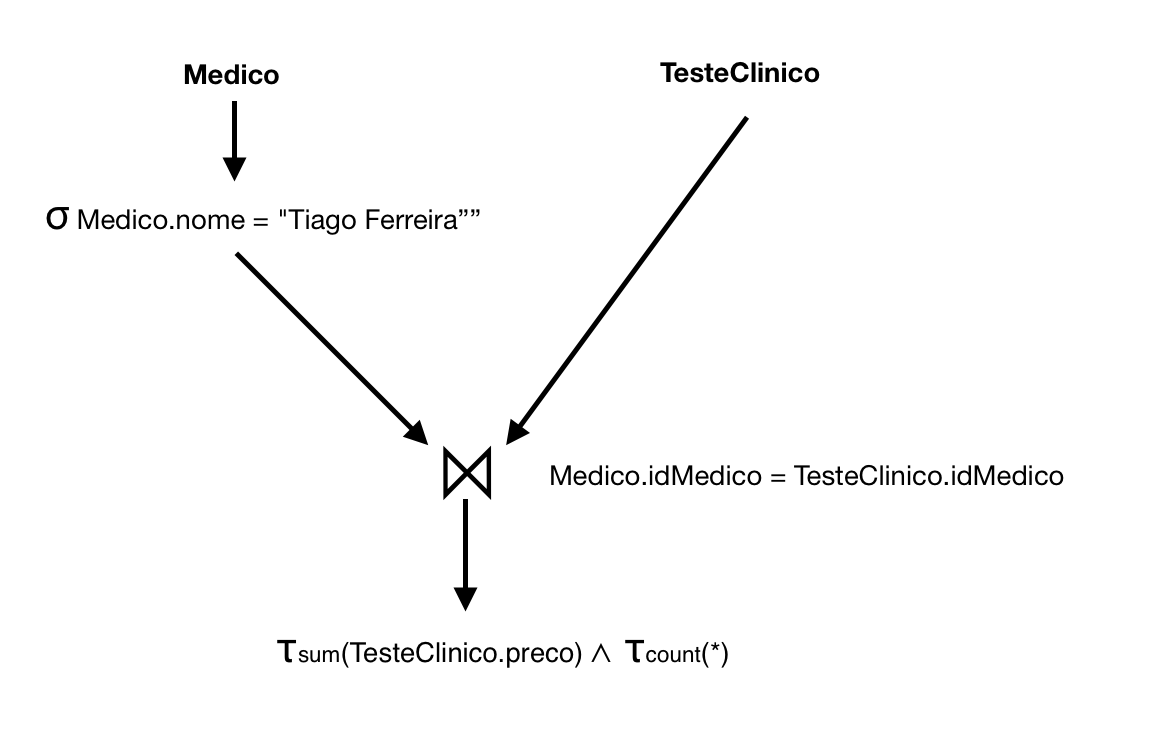
****

Figura 10 - Árvore representativa da Interrogação nº4

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

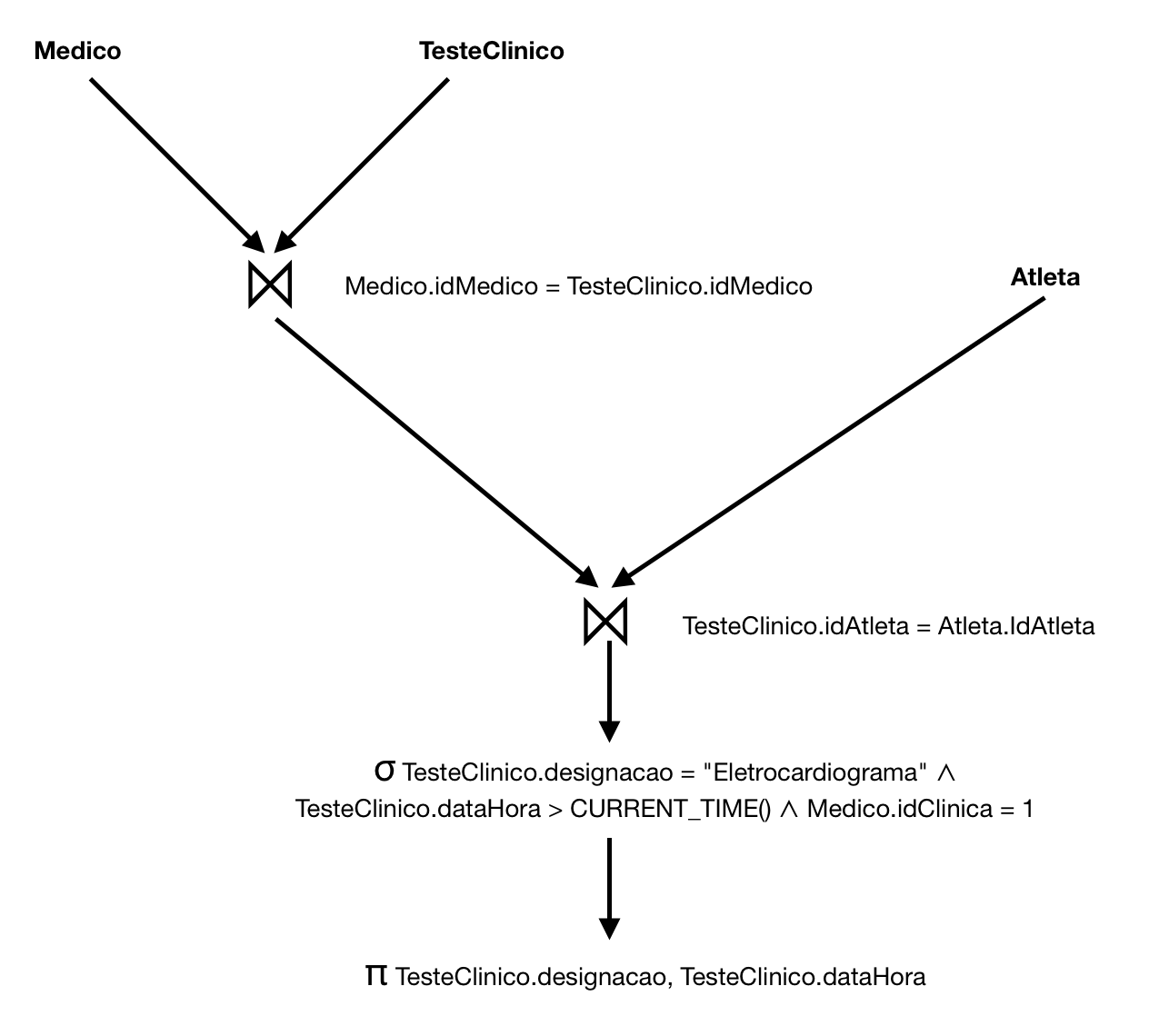


Figura 11 - Árvore representativa da Interrogação nº5

* 1. 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.

* **Registar um atleta:** Para registar 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.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

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.*

**Uma imagem com captura de ecrã

Descrição gerada automaticamente**

Figura 13 - Tabela TesteClinico

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

**Uma imagem com captura de ecrã

Descrição gerada automaticamente**

Figura 14 - Mapa de Transações para adicionar uma prova

* 1. 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 primarias. No SGBD que estamos a implementar, todos os campos são de preenchimento obrigatório.

**Integridade de Entidade:** 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 de 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 de Multiplicidade:** As multiplicidades são definidas na criação do modelo concetual, e como tal devem transitar para o modelo lógico. Contudo, existem casos particulares como os atributos multivalorados que devem ser analisados, como é retratado na seção *4.1*, onde é possível encontrar os relacionamentos estabelecidos.

* 1. 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.

1. Implementação Física
   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 princípiosACID (Atomicidade, Consistência, Isolamento e Durabilidade) também contribuiu positivamente para a nossa escolha.

* 1. 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 logico 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.

* + 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 hota

Domínio preco: Número décimal

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 designação: 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));

* + 1. Desenho das restrições

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

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 15 - Criação da tabela Atleta

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 16 - Criação da tabela TesteClinico

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 17 - Criação da Tabela Médico

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 18 - Criação da tabela Clínica

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 19 - Criação da tabela Contacto

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 20 - Criação da Tabela Prova\_TipoTeste

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 21 - Criação da tabela Prova

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 22 - Criação da tabela Atleta\_Prova

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 23 - Criação da tabela Morada

* 1. 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)**

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 24 - Resolução da interrogação nº1

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

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 25 - Resolução da interrogação nº2

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

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 26 - Resolução da interrogação nº3

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

**Uma imagem com captura de ecrã

Descrição gerada automaticamente**

Figura 27 - Resolução da interrogação nº4

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

**Uma imagem com captura de ecrã

Descrição gerada automaticamente**

Figura 28 - Resolução da interrogação nº5

* 1. Tradução das transações estabelecidas para SQL

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

**Inserir Atleta:**

Uma imagem com captura de ecrã

Descrição gerada automaticamente

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*.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 30 - Inserir provas do atleta

**Efetuar Agendamento:**

Uma imagem com captura de ecrã

Descrição gerada automaticamente

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:**

Uma imagem com captura de ecrã

Descrição gerada automaticamente

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

* 1. 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.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 33 - Imagem do código obtido para o trigger

* 1. Escolha, Definição e Caracterização de Índices em SQL

Apesar de a opção de *Foward Engeneering* 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.

* 1. 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

|  |  |  |  |
| --- | --- | --- | --- |
| ATLETA | Atributos | Tipo de Dados | Espaço Ocupado |
| 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

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

Tabela 11 - Espaço ocupado pela tabela Clínica

|  |  |  |  |
| --- | --- | --- | --- |
| Médico | Atributos | Tipo de Dados | Espaço Ocupado |
| 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

|  |  |  |  |
| --- | --- | --- | --- |
| TesteClínico | Atributos | Tipo de Dados | Espaço Ocupado |
| 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

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

Tabela 14 - Espaço ocupado pela tabela Prova

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

Tabela 15 - Espaço ocupado pela tabela Morada

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

Tabela 16 - Espaço ocupado pela tabela Contacto

|  |  |  |  |
| --- | --- | --- | --- |
| Atleta\_Prova | Atributos | Tipo de Dados | Espaço Ocupado |
| idAtleta | INT | 4 |
| idProva | INT | 4 |
| data | DATE | 4 |
| TOTAL | 12 | | |

Tabela 17 - Espaço ocupado pela tabela Atleta\_Prova

|  |  |  |  |
| --- | --- | --- | --- |
| Prova\_TipoTeste | Atributos | Tipo de Dados | Espaço Ocupado |
| 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 quadruplique, 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.

* 1. 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.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 34 - View de agendamentos

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

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 35 – View dos exames realizados

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

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 36 – View dos medicos e as suas especialidades

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

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 37 - View dos dados dos pacientes

* 1. 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 terá́ 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.



Figura 38 - Permissões dos diferentes utilizadores

* 1. 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.

1. Sistema *NoSQL*: *Neo4j*
   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 complementar, 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.

* 1. 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.

* 1. 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 subconjunto 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.

Decidiu-se demonstrar as queries que mais sobrecarregam a base de dados, ou seja, as que recorrem a *inner joins*, visto que estas são as que justificam a utilização do sistema NoSQL:

1. **Consultar os testes realizados antes de determinada prova para um dado atleta ordenados do mais recente para o mais antigo.**
2. **Consultar os nomes dos atletas de determinada modalidade que acusaram positivo nos testes de *doping*.**
3. **Consultar os nomes dos pacientes/atletas que foram submetidos a determinado teste.**
4. **Consultar agendamentos para um determinado tipo de exame numa determinada clínica.**
   1. 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:

Uma imagem com captura de ecrã

Descrição gerada automaticamente

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”.

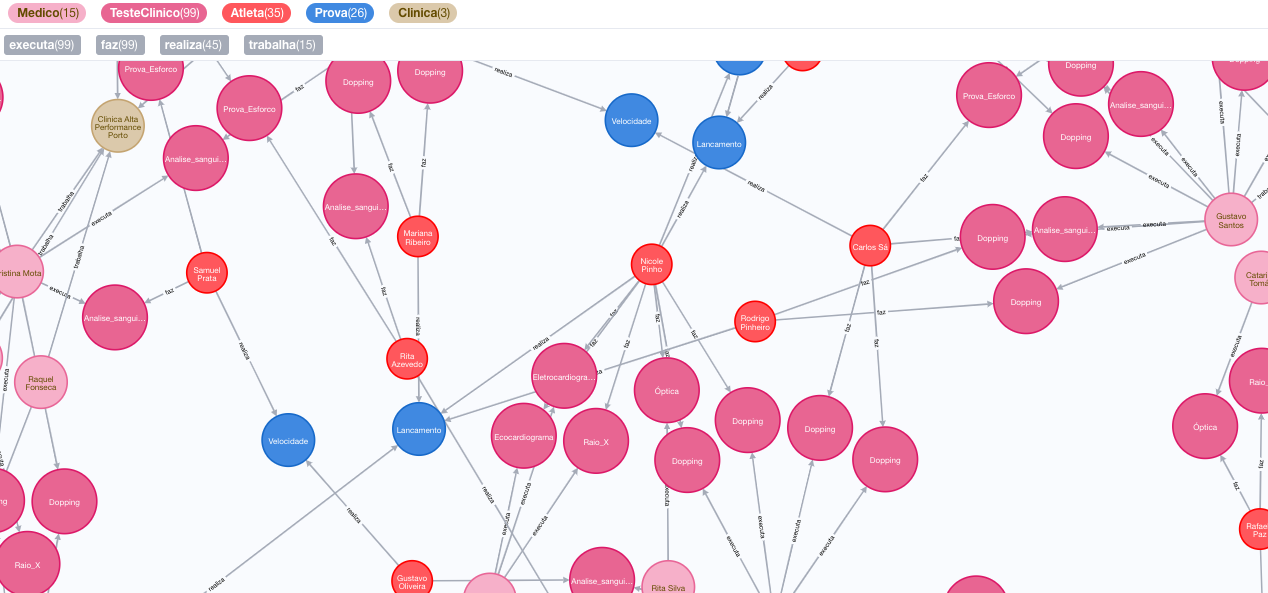


Figura 40 - Ilustração da base de dados NoSQL criada em Neo4j

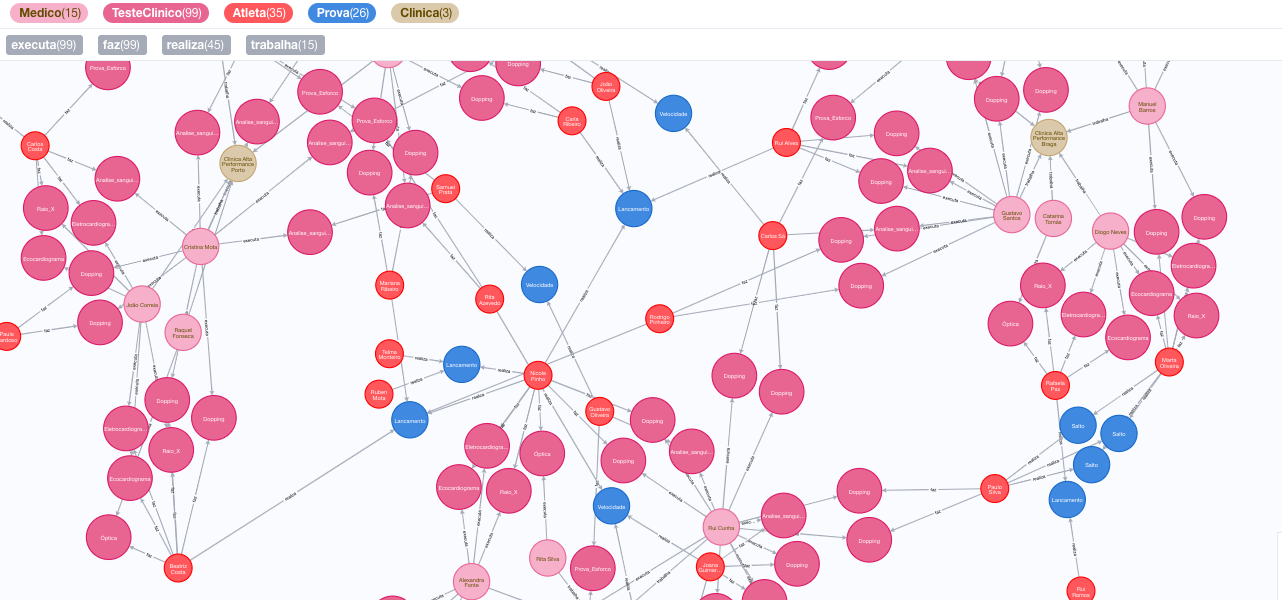


Figura 41 - Ilustração da base de dados NoSQL criada em Neo4j

Uma imagem com texto, mapa

Descrição gerada automaticamente

Figura 42 - Ilustração da base de dados NoSQL criada em Neo4j

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

Para a base de dados NoSQL os objetos utilizados serão provenientes de 6 tabelas da base de dados do sistema relacional: Clinica, Atleta, TesteClinico, Atleta\_Prova, Prova e Medico.

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 e o nodo Medico através da tabela Medico.

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.

* 1. 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;

**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” se 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 diração do Medico para a Clinica;

* 1. 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.

* 1. 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 FROM 'file:///Clinica.csv' AS line

CREATE (:Clinica { idClinica: line[0], nome: line[1]})

**Criação do nodo Atleta:**

USING PERIODIC COMMIT

LOAD CSV FROM 'file:///Atleta.csv' AS line

CREATE (:Atleta { idAtleta: line[0], nome: line[1], dataNascimento: line[2], sexo: line[3]})

**Criação do nodo Prova:**

USING PERIODIC COMMIT

LOAD CSV FROM 'file:///Prova.csv' AS line

CREATE (:Prova { idProva: line[0], designacao: line[1], modalidade: line[2],categoria:line[3]})

**Criação do nodo TesteClinico:**

USING PERIODIC COMMIT

LOAD CSV FROM 'file:///Teste\_Clinico.csv' AS line

CREATE (:TesteClinico { idTesteClinico: line[0], designacao: line[1], dataHora: line[2], preco: line[3], resultado: line[4]})

**Criação do nodo Medico:**

USING PERIODIC COMMIT

LOAD CSV FROM 'file:///Medico.csv' AS line

CREATE (:Medico { idMedico: line[0], nome: line[1], dataInicioServico: line[2],dataNascimento: line[3], especialidade: line[4]})

Foram criados índices para assegurar uma procura mais 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);

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 FROM 'file:///Atleta\_Prova.csv' AS line

MATCH (a:Atleta {idAtleta: line[0]})

MATCH (p:Prova {idProva: line[1]})

MERGE (a)-[r:realiza {data: line[2]}]->(p);

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

USING PERIODIC COMMIT

LOAD CSV FROM 'file:///Teste\_Clinico.csv' AS line

MATCH (a:Atleta {idAtleta: line[6] })

MATCH (t:TesteClinico {idTesteClinico: line[0] })

MERGE (a)-[r:faz]->(t);

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

USING PERIODIC COMMIT

LOAD CSV FROM 'file:///Medico.csv' AS line

MATCH (m:Medico {idMedico: line[0]})

MATCH (c:Clinica {idClinica: line[6] })

MERGE (m)-[r:trabalha]->(c);

**Criação do relacionamento entre o nodo Medico e o nodoTesteClinico:**

USING PERIODIC COMMIT

LOAD CSV FROM 'file:///Teste\_Clinico.csv' AS line

MATCH (m:Medico {idMedico: line[5]})

MATCH (t:TesteClinico {idTesteClinico: line[0] })

MERGE (m)-[r:executa]->(t);

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.

* 1. 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.

Decidiu-se demonstrar as queries que mais sobrecarregam a base de dados, ou seja, as que recorrem a *inner joins*, visto que estas são as que justificam a utilização do sistema *NoSQL*.

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. **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;

1. **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;

1. **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;

1. **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;

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 caraterísticas associadas, uma vez que a base de dados NoSQL foi criada de forma complementar à base de dados relacional.

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** *Structured Query Language*

**ER** *Entity–Relationship*

**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**
4. 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\_ENGINE\_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,

`endereço` 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\_TesteClinico\_Tecnico1\_idx` (`idMedico` ASC) VISIBLE,

INDEX `fk\_TesteClinico\_Atleta\_idx` (`idAtleta` ASC) VISIBLE,

CONSTRAINT `fk\_TesteClinico\_Tecnico1`

FOREIGN KEY (`idMedico`)

REFERENCES `ClinicaAltaPerformance`.`Medico` (`idMedico`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_TesteClinico\_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,

`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;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

1. Anexo 2 – Script Parcial 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');

(...)

-- 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');

(...)

-- 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);

(...)

-- Clinicas

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);

(...)

-- 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);

(...)

-- 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);

(...)

-- 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);

(...)

-- TESTES CLÍNICOS REALIZADOS

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

(36,'Prova\_Esforco','2019-07-15 10:30',5.0,'Valores a controlar',10,18);

(...)

-- TESTES AGENDADOS

INSERT INTO ClinicaAltaPerformance.TesteClinico (idTesteClinico,designacao,dataHora,preco,resultado,idMedico,idAtleta) VALUES

(73,'Eletrocardiograma','2020-04-16 10:30',10.0,'N\d',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\d',11,19);

(...)

-- 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');

(...)

-- POVOAR ATLETA\_PROVA COM AS DATAS DAS PROVAS REALIZADAS

INSERT INTO ClinicaAltaPerformance.Atleta\_Prova (id\_Atleta,id\_Prova,data) VALUES (1,38,'2019-09-15');

INSERT INTO ClinicaAltaPerformance.Atleta\_Prova (id\_Atleta,id\_Prova,data) VALUES (1,39,'2019-09-15');

INSERT INTO ClinicaAltaPerformance.Atleta\_Prova (id\_Atleta,id\_Prova,data) VALUES (1,40,'2019-09-15');

(...)

-- PROVAS QUE REQUEREM TESTE DE DOPPING

INSERT INTO ClinicaAltaPerformance.Prova\_TipoTeste (idProva, designacao) VALUES (9,'Dopping');

INSERT INTO ClinicaAltaPerformance.Prova\_TipoTeste (idProva, designacao) VALUES (11,'Dopping');

INSERT INTO ClinicaAltaPerformance.Prova\_TipoTeste (idProva, designacao) VALUES (13,'Dopping');

(...)

-- TODAS AS PROVAS DE VELOCIDADE E DE BARREIRAS REQUEREM TESTE DE ESFORCO E ANALISE SANGUINEAS

INSERT INTO ClinicaAltaPerformance.Prova\_TipoTeste (idProva, designacao) VALUES (1,'Prova\_Esforco');

INSERT INTO ClinicaAltaPerformance.Prova\_TipoTeste (idProva, designacao) VALUES (1,'Analise\_sanguinea');

INSERT INTO ClinicaAltaPerformance.Prova\_TipoTeste (idProva, designacao) VALUES (2,'Prova\_Esforco');

(...)

1. 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 dopping

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 obrigatorios 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;