



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

Unidade Curricular de Bases de Dados

Ano Letivo de 2019/2020

Clínicas *MIEICare*

**Rui Oliveira (a83610), Guilherme Araújo
(a84527), Pedro Henriques (a84794) e
Gonçalo Esteves (a85731)**

Janeiro de 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Clínicas *MIEICare*

**Rui Oliveira (a83610), Guilherme Araújo
(a84527), Pedro Henriques (a84794) e
Gonçalo Esteves (a85731)**

Janeiro de 2020

Resumo

Serve o presente documento, elaborado no âmbito da Unidade Curricular de Base de Dados, para retratar toda a elaboração e execução de uma arquitetura de base de dados de uma companhia de clínicas. Esta será usada para o agendamento e realização de diversos testes clínicos em atletas de diferentes modalidades de atletismo.

Numa fase inicial, retrataremos as fases primordiais do projeto, como por exemplo a contextualização, motivação, objetivos e análise de viabilidade do processo. Será ainda efetuado um levantamento dos requisitos onde estarão identificadas as entidades consideradas essenciais para este sistema.

Ao nível do modelo conceptual serão descritas as entidades e os relacionamentos, sem esquecer os respetivos atributos e a caracterização da associação dos mesmos.

Passando para o modelo lógico, aqui é aprofundado o estudo dos elementos referidos anteriormente, sendo descritos detalhadamente os vários tipos de relacionamentos, as várias associações e os atributos correspondentes. É feita, também, a validação deste modelo através de vários métodos.

Por fim, será feita a conversão para o modelo físico onde será usado o sistema de gestão de bases de dados *MySQL*.

Este trabalho tem como objetivo a construção de um sistema de base de dados que seja capaz de gerir de uma forma correta a informação acerca de um sistema de uma companhia de clínicas de testes clínicos, permitindo o bom funcionamento do mesmo.

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

Palavras-Chave: Clínicas *MIECare*, *SQL*, informação, testes clínicos, atleta, médico, clínica, Bases de Dados Relacionais, Bases de Dados não Relacionais, Análise de Requisitos, Gestão de Índices, Sistemas Operacionais, *Business Intelligence*, Entidades, Atributos, Relacionamentos, Metodologia, Modelo Conceptual, Modelo Lógico, Modelo Físico, SGBD, transações, gatilhos, procedimentos.

Índice

Resumo	i
Índice	ii
Índice de Figuras	v
Índice de Tabelas	vii
1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	2
1.3. Motivação e Objetivos	2
1.4. Estrutura do Relatório	3
1.5. Análise de Viabilidade do Processo	3
2. Levantamento e Análise de Requisitos	5
2.1. Método adotado	5
2.2. Requisitos Levantados	5
2.2.1 Requisitos de Descrição	5
2.2.2 Requisitos de Exploração	7
2.2.3 Requisitos de Controlo	8
2.3. Análise geral dos requisitos	9
3. Modelo de Dados Conceptual	10
3.1. Abordagem de modelação realizada	10
3.2. Identificação e caracterização das entidades	10
3.2.1 Dicionário de dados das entidades do modelo	12
3.3. Identificação e caracterização dos relacionamentos	12
3.3.1 Dicionário de relacionamentos do modelo	15
3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos	15
3.5. Detalhe ou generalização de entidades	18
3.6. Apresentação e explicação do diagrama ER	18
3.7. Revisão e validação do modelo com o utilizador	19
4. Modelo de Dados Lógico	20
4.1. Construção e validação do modelo de dados lógico	20
4.1.1 Entidades Fortes	20
4.1.2 Entidades Fracas	21

4.1.3 Relacionamentos binários um para muitos (1-N)	21
4.1.4 Relacionamentos binários um para um (1-1)	22
4.1.5 Relacionamentos recursivos de um para um (1-1)	22
4.1.6 Relacionamentos superclasse/subclasse	22
4.1.7 Relacionamentos binários muitos para muitos (N-M)	22
4.1.8 Relacionamentos complexos	23
4.2. Desenho do modelo lógico	24
4.3. Validação do modelo através da normalização	24
4.3.1 1FN – 1 ^a Forma Normal	24
4.3.2 2FN – 2 ^a Forma Normal	25
4.3.3 3FN – 3 ^a Forma Normal	25
4.4. Validação do modelo com as interrogações do utilizador	25
4.5. Validação do modelo com as transações estabelecidas	26
4.6. Reavaliação do modelo lógico	27
4.7. Revisão do modelo lógico com o utilizador	27
5. Modelo de Dados Físico	28
5.1. Seleção do sistema de gestão de bases de dados	28
5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL	28
5.2.1 Descrição das relações base	28
5.2.2 Desenho das restrições gerais	32
5.3. Tradução das interrogações do utilizador para SQL	36
5.4. Tradução das transações estabelecidas para SQL	38
5.5. Escolha, definição e caracterização de índices em SQL	39
5.6. Estimativa do espaço em disco da base de dados e taxa de crescimento anual	40
5.6.1 Tamanho de cada uma das tabelas das Clínicas MIEICare, após o povoamento inicial	40
5.6.2 Estimativa do crescimento anual	43
5.7. Definição e caracterização das vistas de utilização em SQL	46
5.8. Definição e caracterização dos mecanismos de segurança em SQL	48
5.9. Revisão do sistema implementado com o utilizador	50
6. Migração para Base de Dados não Relacional	52
6.1. Introdução	52
6.2. Bases de Dados NoSQL	52
6.2.1 Principais Características	53
6.2.2 SQL vs NoSQL	54
6.2.3 Vantagens	55
6.2.4 Desvantagens	55
6.3. Migração de Dados	56

6.3.1 Exportação da informação contida para ficheiros csv	56
6.3.2 Importação da informação contida em ficheiros csv	56
6.3.3 Criação dos nodos	56
6.3.4 Criação dos relacionamentos	57
6.3.5 Remoção de dados redundantes	57
6.4. Grafo Resultante	58
6.4.1 Imagem do grafo resultante	58
6.4.2 Imagem do grafo resultante sem os nodos não relacionados	59
6.5. Tradução das interrogações do utilizador para Cypher	60
6.6. Revisão do sistema implementado com o utilizador	61
6.7. Análise Crítica	61
7. Conclusões e Trabalho Futuro	63
Referências	64
Lista de Siglas e Acrónimos	65
Anexos	66
I. Script de povoamento	67
II. Script de criação de tabelas	71
III. Script funções	74
IV. Script conversão de dados em ficheiros csv	75
V. Script criação dos nodos (e importação de dados)	76
VI. Script criação dos relacionamentos	78
VII. Script criação dos índices	81
VIII. Script remoção dos dados redundantes	82

Índice de Figuras

Figura 1 - Esquema Conceptual	10
Figura 2 - Relacionamentos entre Atleta e TCAgendado/TCRealizado	13
Figura 3 - Relacionamentos entre Clinica e TCAgendado/TCRealizado	14
Figura 4 - Relacionamentos entre TesteClinico e TCAgendado/TCRealizado	14
Figura 5 - Modelo Lógico	24
Figura 6 - Criação da tabela Clinica	32
Figura 7 - Criação da tabela Atleta	33
Figura 8 - Criação da tabela Medico	34
Figura 9 - Criação da tabela TesteClinico	34
Figura 10 - Criação da tabela TCAgendado	35
Figura 11 - Criação da tabela TCRealizado	35
Figura 12 - 1 ^a <i>querie</i> em SQL	36
Figura 13 - 2 ^a <i>querie</i> em SQL	36
Figura 14 - 3 ^a <i>querie</i> (atleta mais velho) em SQL	36
Figura 15 - 3 ^a <i>querie</i> (atleta mais novo) em SQL	36
Figura 16 - 4 ^a <i>querie</i> em SQL	37
Figura 17 - 5 ^a <i>querie</i> em SQL	37
Figura 18 - 6 ^a <i>querie</i> em SQL	37
Figura 19 - Transação de agendamento de um teste	38
Figura 20 - Transação de deleção de um teste agendado	38
Figura 21 - Transação de realização de um teste	39
Figura 22 - Criação de Índices	39
Figura 23 - Código SQL que mostra o conteúdo de TesteClinico	46
Figura 24 - Código SQL que mostra o conteúdo de Atleta	46
Figura 25 - Código SQL que mostra o conteúdo de Medico	46
Figura 26 - Código SQL que mostra o conteúdo de TCAgendado	47
Figura 27 - Código SQL que mostra o conteúdo de TCRealizado	47
Figura 28 - Código SQL que mostra quais os médicos que atenderam cada atleta	47
Figura 29 - Código SQL com as permissões do administrador	48
Figura 30 - Código SQL com as permissões do atleta	48

Figura 31 - Código <i>SQL</i> com as permissões do medico	49
Figura 32 - Código <i>SQL</i> com as permissões do funcionário	50
Figura 33 - Grafo que representa a base de dados com todos os seus elementos	58
Figura 34 - Grafo que representa a base de dados apenas com os elementos relacionados	59
Figura 35 - 1 ^a <i>querie</i> em <i>Cypher</i>	60
Figura 36 - 2 ^a <i>querie</i> em <i>Cypher</i>	60
Figura 37 - 3 ^a <i>querie</i> em <i>Cypher</i> (atleta mais velho)	60
Figura 38 - 3 ^a <i>querie</i> em <i>Cypher</i> (atleta mais novo)	60
Figura 39 - 4 ^a <i>querie</i> em <i>Cypher</i>	60
Figura 40 - 5 ^a <i>querie</i> em <i>Cypher</i>	61
Figura 41 - 6 ^a <i>querie</i> em <i>Cypher</i>	61

Índice de Tabelas

Tabela 1 - Dicionário de dados das entidades do modelo	12
Tabela 2 - Dicionário de relacionamentos do modelo	15
Tabela 3 - Atributos de cada entidade	17
Tabela 4 – CodigoPostal	40
Tabela 5 – Modalidade	40
Tabela 6 – TesteClinico	40
Tabela 7 – Equipamento	40
Tabela 8 – Clinica	41
Tabela 9 – Atleta	41
Tabela 10 – Medico	41
Tabela 11 – TCAgendado	42
Tabela 12 – TCRealizado	42
Tabela 13 – Contacto	42
Tabela 14 – EquipamentoDisponivel	42
Tabela 15 – EquipamentoNecessario	42
Tabela 16 – EquipamentoUtilizado	43
Tabela 17 - Tamanho inicial da base de dados	43
Tabela 18 - Tamanho da base de dados ao fim de um ano	45

1. Introdução

1.1. Contextualização

Devido ao aumento significativo do número de praticantes de desporto e a uma rigidez cada vez maior ao nível do controlo do estado clínico de um atleta (maior quantidade de testes clínicos necessários e maior controlo de *doping*, por exemplo) as clínicas que efetuam os testes necessários para um atleta estar apto a competir têm tido cada vez mais procura, o que leva a um aumento das oportunidades de negócio nesta área: não só são necessários mais profissionais com capacidade de realizar estes testes, como também são necessárias mais infraestruturas que suportem um crescente número de consultas.

Companhias que pretendam despoletar nesta área, começando do zero ou então tentando alargar a sua zona de incidência, deverão sempre tentar ser o mais inovadoras possível, por forma a atrair um maior número de clientes. Isto é observável não só no tipo de equipamentos médicos utilizados, como também na forma como atendem os utentes.

Por forma a providenciar um melhor atendimento aos seus utentes, maioritariamente atletas praticantes de Atletismo, de todas as modalidades e escalões, os proprietários da Clínica MIEICare pretendem utilizar um *software* próprio, que terá acesso a toda a informação da clínica. Este será utilizado por todos serviços de atendimento ao utente, aquando das marcações presenciais dos testes clínicos. Para além disto, e mais importante, os proprietários pretendem ser donos de uma aplicação, que torne o processo de marcação de testes mais fácil por parte dos atletas, uma vez que estes o poderão fazer em qualquer lugar, a partir da aplicação.

Devido ao enorme sucesso que a clínica de Braga está a ter, os proprietários pretendem abrir duas novas clínicas: uma no Porto e outra em Lisboa, não descartando, no entanto, a hipótese de, mais tarde, abrirem clínicas noutras zonas do país e, quem sabe, até mesmo no estrangeiro! Como tal, a base de dados deve guardar a informação de todas as clínicas, nomeadamente todos os atletas que já realizaram algum teste em alguma clínica, por forma a que qualquer atleta previamente atendido, possa voltar a ser noutra clínica, sem ser necessário voltar a fornecer todas as suas informações.

1.2. Apresentação do Caso de Estudo

Os proprietários da clínica *MIECare* pretendem agora alargar horizontes, criando uma companhia, as Clínicas *MIECare*. Por forma a proporcionar o melhor atendimento possível, é intenção deles criar um *software* próprio para a companhia, de gestão de testes clínicos. Este *software* será capaz de guardar a informação de todos os atletas que fizeram testes em qualquer uma das clínicas da companhia. Além disto, será capaz de guardar as informações relativas ao pessoal médico de cada clínica, a todo o equipamento existente numa dada clínica e a todos os testes clínicos, agendados ou realizados, associados a qualquer clínica.

Por forma a usufruir dos serviços prestados por estas clínicas de uma forma ainda mais inovadora, os atletas terão acesso a uma aplicação que permitirá a marcação dos testes clínicos sem ser necessária a deslocação à clínica. Para tal, os utilizadores deverão registar-se na aplicação, preenchendo uma ficha clínica, que ficará guardada na base de dados geral, estando esta disponível para todas as clínicas, o que permite que um atleta possa ser atendido em todas elas. Posto isto, e à medida que for marcando e realizando testes, estes ficarão associados a ele tornando-se fácil saber quais testes o atleta já realizou, e aqueles que já estão marcados para serem realizados.

Tal como já foi referido, são também guardadas as informações de todos os médicos que fazem parte do quadro das clínicas. Como tal, cada médico deverá registar-se também na aplicação, preenchendo uma ficha com algumas informações básicas, de modo a registá-lo na base de dados. Tal como os atletas, estes também terão associados todos os testes que já fizeram, e todas as marcações que possuem.

É guardada, também, uma tabela com todos os tipos de testes clínicos que as clínicas podem realizar, associando o nome do teste, todo o material necessário para a sua realização e o seu custo.

Por fim, podemos afirmar que há dois tipos de testes: os testes agendados e os testes realizados. Sendo estes o motivo pelo qual os atletas se dirigem à clínica, estes devem estar também bem estruturados, por forma a providenciar o melhor atendimento possível.

Posto isto, será com facilidade que, com o acesso à aplicação, qualquer atleta agende todos os testes clínicos que pretender, tendo de ter apenas em conta a eventual disponibilidade do médico com que pretende efetuar o teste, e do material necessário.

1.3. Motivação e Objetivos

Os principais motivos para o desenvolvimento deste projeto têm por base a expansão e inovação do negócio, devido à abertura de mais clínicas, tal como já fora referido a cima. Para além disto, pretende-se não só obter uma melhor organização na marcação das consultas, evitando conflitos de horários em consultas com o mesmo médico, ou então a espera prolongada pela disponibilização do material que está a ser utilizado, como também uma maior facilidade por parte dos atletas na marcação de testes.

Com este sistema operacional, a companhia tem toda a sua informação centralizada e organizada, para posteriormente serem aplicadas técnicas de *Business Intelligence*, de forma a analisar a sua evolução e serem retiradas as devidas conclusões com os dados das marcações que vão sendo recolhidos.

1.4. Estrutura do Relatório

Após termos apresentado o caso de estudo e a motivação e os objetivos por detrás deste, nos capítulos seguintes deste relatório será exposta a construção do modelo conceptual da base de dados, a transição deste para esquema lógico e a consequente implementação do modelo físico.

Primeiramente, na análise de requisitos, são apresentadas todas as entidades e todos os potenciais relacionamentos entre elas, bem como os atributos que as caracterizam. A forma como um utilizador poderá interagir com a base de dados é descrita sob a forma de possíveis operações de manipulação e consulta através de exemplos práticos.

De seguida, expõem-se a análise do modelo conceptual da base de dados, onde são apresentadas em tabelas as entidades e relacionamentos, bem como os respetivos atributos. Para cada entidade, é apresentada toda a informação sobre a sua descrição e ocorrência, enquanto que, para os relacionamentos é caracterizada a multiplicidade entre as entidades envolvidas. Quanto aos atributos, são caracterizados os tipos de dados e o seu tamanho, assim como se podem ser nulos ou não, derivados ou multivvalorados. Posto isto, é também determinado o domínio de valores possíveis. São também apresentadas, para cada entidade, as chaves candidatas, sendo identificada a chave primária e as chaves alternativas.

1.5. Análise de Viabilidade do Processo

Este projeto consiste na implementação de um sistema de base de dados relacional (SBDR). Uma base de dados relacional é composta por um conjunto de tabelas e associações entre estas. A associação entre os dados é o ponto forte dos sistemas relacionais. Estas tabelas são formadas por linhas e colunas onde se encontram os dados, que numa base de dados relacional são representados como valores nas colunas das tabelas.

A elaboração de SBDR, garante inúmeras vantagens que tornam este projeto bastante viável e favorável. Ao contrário de um ficheiro, as tabelas são muito vantajosas na medida em que podem ter mais do que um propósito e os seus dados podem ser classificados com diferentes formas e formatos.

Em relação à implementação da base de dados podemos destacar as cinco vantagens mais relevantes:

- **Resposta rápida aos pedidos de informação**

Um dos ganhos a nível operacional é a resposta rápida aos pedidos de informação. Como os dados estão integrados numa única estrutura, as respostas a questões complexas processam-se mais rapidamente.

- **Acesso múltiplo**

O software de gestão de base de dados permite que os dados sejam acedidos de várias formas, nomeadamente, podem ser vistos através de pesquisas sobre qualquer um dos campos da tabela.

- **Flexibilidade**

A independência entre os dados e programas faz com que qualquer modificação num desses elementos não implique alterações radicais no outro.

- **Integridade da informação**

Dada a obrigação de não permitir a redundância, as alterações de dados são feitas num só sítio, evitando-se assim possíveis conflitos entre as diferentes versões da mesma informação.

- **Melhor gestão da informação**

A localização central dos dados permite saber sempre como e onde se encontra a informação. Deste modo, podemos concluir que a implementação da base de dados tem a finalidade de simplificar e favorecer a forma como toda a informação é processada e armazenada, contribuindo para uma melhor gestão da mesma.

2. Levantamento e Análise de Requisitos

2.1. Método adotado

Por forma a ter uma boa implementação de uma base de dados, torna-se fulcral conhecer bem o funcionamento de uma clínica, nomeadamente uma com as especificidades retratadas, isto é, todas as funcionalidades e características, e aquilo com que interagem.

Deste modo, consideramos que analisar não só as exigências dos proprietários das clínicas, mas também o funcionamento de outras, seria uma boa prática, tendo em vista uma maior compreensão destes espaços.

2.2. Requisitos Levantados

2.2.1 Requisitos de Descrição

Após uma extensa análise de toda a informação obtida, reuniu-se toda aquela que se mostrava relevante, de modo a procedermos posteriormente ao desenho da base de dados. A abordagem desses dados será feita tendo em conta os quatro principais elementos deste sistema: o atleta, o médico, o teste clínico e a clínica.

- **Atleta**

As informações básicas de um atleta são absolutamente necessárias de modo a que se possa associar-lhe um teste. Deste modo, decidimos que, aquando da inscrição na aplicação ou presencialmente, na clínica, um atleta deverá preencher uma ficha clínica, cujas informações ficarão registadas na base de dados. O preenchimento da ficha clínica requer: **email**, **password**, **nome**, **nº de cartão de cidadão**, **nº de identificação fiscal**, **data de nascimento**, **género**, **morada** (rua e código postal), **equipa** a que pertence (podendo não pertencer a nenhuma), **contacto** e a **modalidade** e escalão em que se insere.

Para agendar um novo teste, o atleta poderá apenas entrar na aplicação, escolher o teste que pretende realizar, escolher um médico com quem pretende realizar o exame e uma data para o realizar.

Para além disto, um atleta terá sempre associado um registo com todos os exames previamente realizados, e todos aqueles que tem agendados.

- **Médico**

Também as informações pessoais do corpo médico da clínica deverão ser guardadas. Como tal, cada médico deverá registar-se na aplicação, indicando na sua ficha o seu **email**, **password**, **nome**, **nº de CC**, **nº da cédula profissional**, **data de nascimento**, **género**, **morada**, **contacto** e a **clínica** em que trabalha.

Tal como acontece com os atletas, todos os médicos terão acesso a todos os exames que têm agendados, bem como todos os que já realizaram.

- **Teste Clínico**

Os testes clínicos são o motivo pelo qual as clínicas têm se tornado um verdadeiro sucesso! Devido à grande quantidade de testes disponíveis, estes estão também organizados e guardados na base de dados, sendo caracterizados pela sua **designação** técnica e tendo associado o seu **preço**, bem como todo o material necessário para os realizar. No entanto, podemos afirmar que os testes clínicos podem ser de dois tipos: **agendados** ou **realizados**. Os testes agendados possuem um identificador do teste que irá ser realizado, data de realização, o atleta que se vai submeter ao teste, o médico que irá efetuar o teste e a clínica onde o teste será efetuado. Os testes realizados possuem as mesmas características que os testes agendados, acrescentando-se, no entanto, os identificadores dos equipamentos que foram efetivamente utilizados nesse teste.

- **Clínica**

A clínica é o local onde tudo acontece, possuindo esta um **nome** próprio e também uma **morada** onde se localiza (rua e código postal). No entanto, a clínica representa não só o espaço físico onde os testes são realizados, mas também toda a gestão por detrás da organização dos mesmos. Deste modo, os funcionários da clínica que trabalham diretamente com o **software** deverão ter acesso a todos os tipos de testes disponíveis, bem como a todos os testes clínicos, agendados e realizados. Assim, poderão informar os atletas de tudo aquilo que necessitam, como por exemplo, a disponibilidade para a marcação de um teste com um dado médico, ou então quando foi o último teste que realizaram.

2.2.2 Requisitos de Exploração

Um dos propósitos das bases de dados é tornar acessível toda a informação que é pedida, e apenas aquela que é pedida. Por exemplo, se um funcionário da clínica pedir uma lista com todos os atletas registados na base de dados, tal lista deverá ser facultada, sem adicionar informação extra. Neste exemplo, seriam apenas selecionados os nomes dos atletas, sendo posteriormente apresentada a lista com tais nomes (o funcionário não deverá ter acesso a *emails* e *passwords*, por exemplo). Por outro lado, a tarefa não seria tão simples caso fossem introduzidas mais condições no pedido, por exemplo todos os atletas com mais de 20 anos, que já realizaram mais de 5 testes clínicos. Tal exigência poderia ser um requisito necessário, por forma a que os proprietários, por exemplo, pudessem averiguar quais os exames mais requisitados por cada faixa etária. Assim, a gestão dos dados apresentados deve ser feita pelo sistema de base de dados, pois controla toda a informação existente, garantindo assim uma maior eficiência.

Como tal, são apresentados em seguida alguns requisitos necessários à manipulação da base de dados (os mais básicos, igualmente importantes, não iremos explicitar).

- Consulta de todos os atletas, por parte de um administrador (ou seja, um proprietário);
- Consulta de todos os médicos, por parte de um administrador;
- Consulta de todos os testes clínicos agendados ou realizados, por parte de um administrador;
- Consulta de todos os testes clínicos agendados ou realizados numa dada clínica, por parte de um funcionário da clínica;
- Consulta dos próprios testes clínicos agendados e realizados, por parte de um atleta ou médico;
- Consulta de todos os testes clínicos disponíveis, por parte de qualquer utilizador;
- Consulta da disponibilidade do médico com o id nº15, as 16h30m do dia 25 de julho de 2020, por parte de um atleta;
- Consulta dos três exames mais realizados no mês de outubro de 2016, por parte de um administrador;
- Consulta do atleta mais velho ou mais novo, por parte de um administrador;
- Consulta de todos os atletas que realizaram Eletrocardiogramas, por parte de um administrador;
- Consulta do número de testes realizado por cada médico, por parte de um administrador;
- Consulta de todos os médicos com idades compreendidas entre os 45 e os 60 anos que realizaram exames à urina, por parte de um administrador.

2.2.3 Requisitos de Controlo

Uma base de dados é, por definição, uma ferramenta que permite guardar e gerir da forma mais eficiente toda a informação que um dado sistema (informático) possui. É importante que todos os utilizadores da aplicação possam aceder aos dados que lhes dizem respeito, não obstante, é ainda mais importante garantir que esses mesmo utilizadores não acedam a informações alheias e desnecessárias ao uso da aplicação, algo que está bem patente nas mais recentes leis de proteção de dados. Por esse motivo, é imprescindível definir várias formas de monitorizar e restringir o acesso a vários dados por parte dos utilizadores. Posto isto, na aplicação foram registados quatro tipos de utilizadores: o administrador, o funcionário, o atleta e o médico.

- **Administrador (Proprietários das Clínicas MIECare)**

Na aplicação que estamos a desenvolver, é necessário a existência de alguém que tenha todas as permissões necessárias para resolver qualquer problema que surja com alguma das informações suportadas pela base de dados. Deste modo, fará sentido que sejam os donos das clínicas a ter o poder para alterar, remover, inserir e consultar qualquer informação suportada pela base de dados. A única operação que será impossibilitada será a de remoção total da informação guardada na base de dados.

- **Funcionário**

Apenas deverão ser disponibilizados a um funcionário os dados relativos aos testes clínicos agendados e realizados na clínica onde este trabalha, por forma a que este possa informar atletas e médicos sobre o seu histórico de testes e aqueles que têm agendados.

- **Atleta**

Este utilizador poderá alterar qualquer um dos seus dados pessoais, caso tal seja pertinente, não podendo, no entanto, remover nada ou até mesmo apagar a sua conta.

Para além disto, ele poderá consultar uma lista com todos os testes disponibilizados pelas clínicas, todos os testes que tem agendados, e todos os testes que já realizou.

- **Médico**

Em tudo semelhante ao atleta.

2.3. Análise geral dos requisitos

O levantamento e a análise de requisitos, tal como referido acima, é de extrema importância, por forma a existir uma consciência sobre a totalidade da informação que a base de dados deverá suportar. Estes deverão estar de acordo com o que é pretendido pelo cliente, por forma a criar um projeto que cumpra em tudo com as expectativas.

Tendo estes findado, procederemos à construção de um modelo conceptual que será na sua integra dependente da informação recolhida, explicada anteriormente.

3. Modelo de Dados Conceptual

3.1. Abordagem de modelação realizada

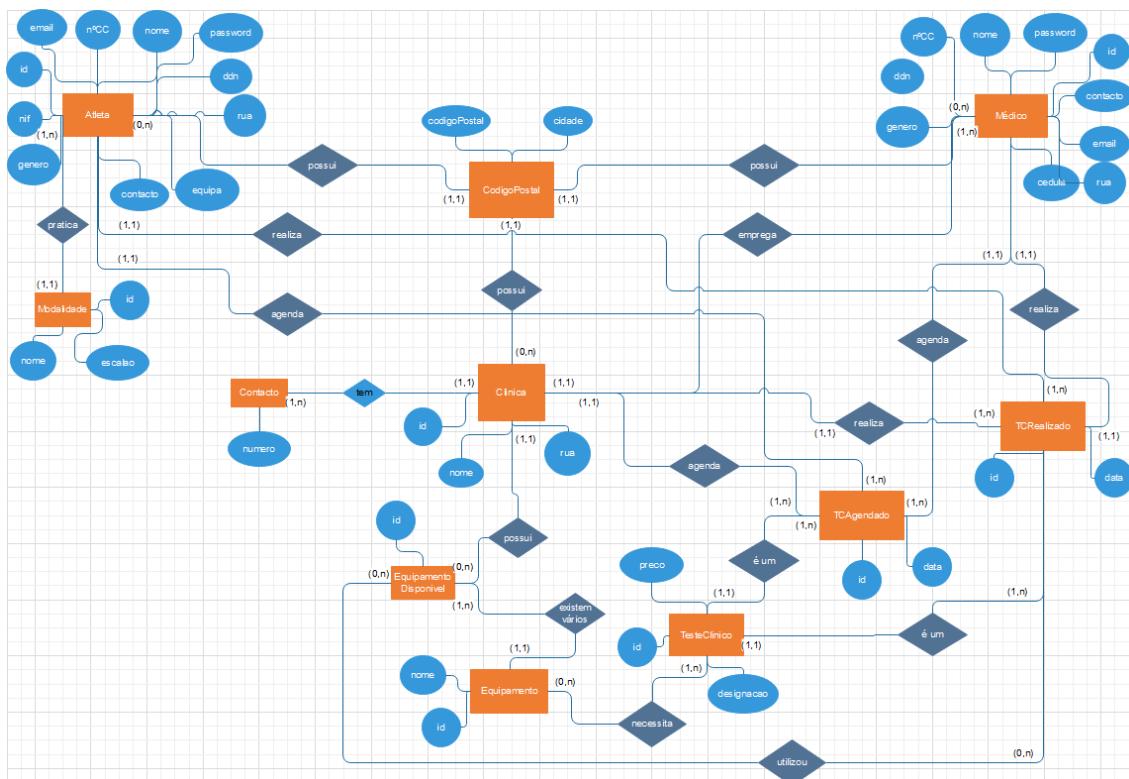


Figura 1 - Esquema Conceptual

3.2. Identificação e caracterização das entidades

Por forma a elaborar um modelo conceptual que retratasse corretamente a situação em causa, começamos por identificar as entidades do problema. Uma análise cuidada dos requisitos é essencial por forma a não haver uma escolha incorreta de entidades, uma vez que podem haver objetos importantes para o modelo, que não sejam necessariamente uma entidade.

Assim, iremos em seguida apresentar as entidades selecionadas, o seu significado e o porquê de terem sido selecionadas.

- **Atleta**

Termo geral que descreve os atletas registados na base de dados. Cada atleta possui atributos próprios, tem uma existência autónoma e pode ser identificado univocamente, sendo assim uma entidade.

- **Médico**

Entidade que representa todos os médicos empregues em alguma clínica da companhia. Estes estão registados na base de dados e são responsáveis por realizar os diversos testes clínicos.

- **Clinica**

Entidade que representa cada clínica pertencente à companhia, estando todas as informações relativas a esta registadas na base de dados.

- **TesteClínico**

Entidade que representa os vários tipos de testes possíveis de efetuar numa clínica (por exemplo: eletrocardiograma, análise à urina, ...).

- **TCAgendado**

Entidade que representa cada teste marcado.

- **TCRealizado**

Entidade que representa cada teste efetuado.

De realçar a existência de outras “entidades secundárias”, tais como Contacto, CódigoPostal, Modalidade e Equipamento, que se relacionam com as entidades acima declaradas. No entanto, por não desempenharem um papel fundamental no modelo criado (só são usadas como auxiliares para suportar alguma informação extra – por exemplo, a entidade CódigoPostal, que é usada para representar os diferentes códigos postais existentes) não serão consideradas como entidades principais do modelo, não sendo, por tanto, estudadas tão profundamente.

3.2.1 Dicionário de dados das entidades do modelo

Entidade	Descrição	Ocorrência
Atleta	Entidade representativa da pessoa que efetua um ou mais exames nas clínicas.	Entidade que faz uso dos serviços prestados pelas clínicas.
Médico	Entidade responsável pela realização dos testes clínicos.	Os médicos são a base da realização dos testes nas clínicas.
Clinica	Entidade representativa dos locais onde se realizam os testes.	Entidade extremamente importante pois é esta que guarda toda a informação relativa a uma dada clínica (testes realizados, pessoal médico, ...)
TesteClinico	Entidade representativa dos tipos de teste existentes para escolha.	Os testes são as entidades nucleares das clínicas.
TCAgendado	Entidade responsável pela marcação de um dado teste.	Entidade base para a realização dos testes.
TCRealizado	Entidade responsável pela sinalização de um teste já realizado.	Entidade base para o registo das informações relativas a testes previamente realizados.

Tabela 1 - Dicionário de dados das entidades do modelo

3.3. Identificação e caracterização dos relacionamentos

- **Atleta e TCAgendado/TCRealizado**

Tanto o relacionamento entre as entidades Atleta e TCAgendado como o relacionamento entre Atleta e TCRealizado são caracterizados por uma cardinalidade de 1 para N, já que um atleta tanto pode agendar vários testes como pode já ter realizado diversos testes.

- **Medico e TCAgendado/TCRealizado**

Tal como nos relacionamentos apresentados anteriormente, para os relacionamentos entre as entidades Medico e TCAgendado e Medico e TCRealizado a cardinalidade é de 1 para N, uma vez que um médico pode ter várias consultas para realizar, bem como muitas consultas já realizadas.

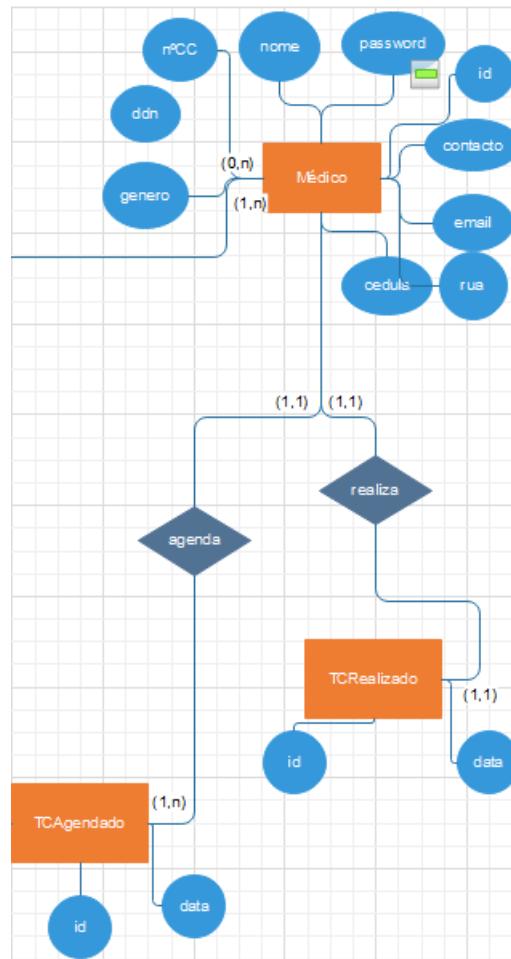


Figura 2 - Relacionamentos entre Atleta e TCAgendado/TCRealizado

- **Clinica e TCAgendado/TCRealizado**

Mais uma vez, os relacionamentos entre Clinica e TCAgendado e Clinica e TCRealizado são de 1 para N, pois uma clínica pode possuir muitos testes marcados, bem como diversos testes previamente realizados.

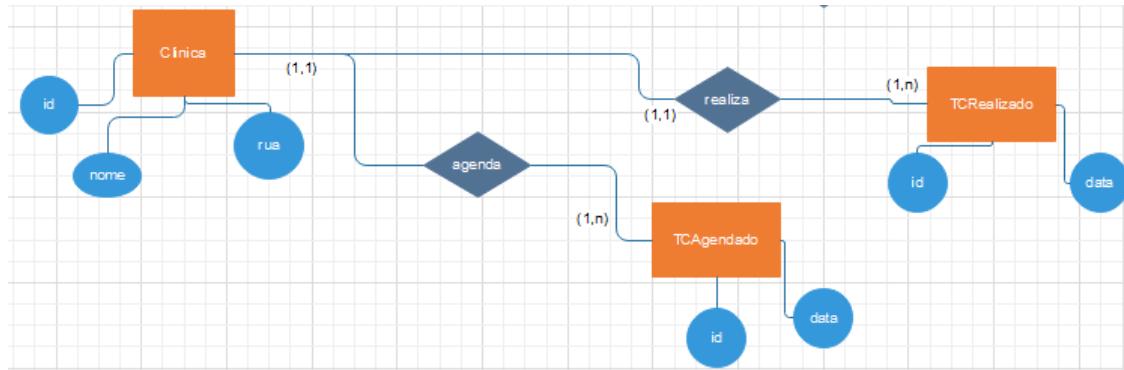


Figura 3 - Relacionamentos entre Clinica e TCAgendado/TCRealizado

- **Clinica e Medico**

Para o relacionamento entre as entidades Clinica e Medico, podemos afirmar que a cardinalidade é de 1 para N, uma vez que cada clínica possui diversos médicos que lá trabalham.

- **TesteClinico e TCAgendado/TCRealizado**

Por fim, constata-se que os relacionamentos entre as entidades TesteClinico e TCAgendado e TesteClinico e TCRealizado possuem uma cardinalidade de 1 para N, uma vez que cada teste, quer esteja agendado ou já tenha sido realizado, pode ser de apenas um tipo, no entanto, podem haver vários testes do mesmo tipo.

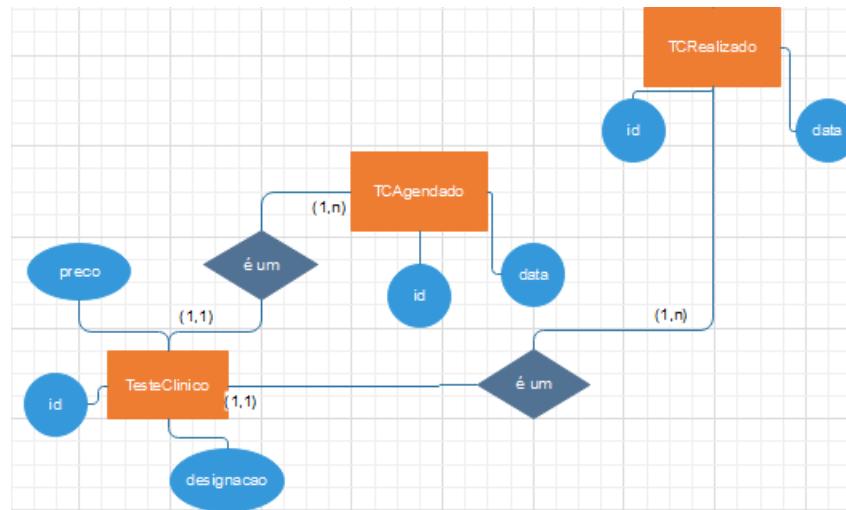


Figura 4 - Relacionamentos entre TesteClinico e TCAgendado/TCRealizado

3.3.1 Dicionário de relacionamentos do modelo

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Atleta	1..1	agenda	1..N	TCAgendado
Atleta	1..1	realiza	1..N	TCRealizado
Medico	1..1	agenda	1..N	TCAgendado
Medico	1..1	realiza	1..N	TCRealizado
Clinica	1..1	agenda	1..N	TCAgendado
Clinica	1..1	realiza	1..N	TCRealizado
Clinica	1..1	emprega	1..N	Medico
TesteClinico	1..1	é um	1..N	TCAgendado
TesteClinico	1..1	é um	1..N	TCRealizado

Tabela 2 - Dicionário de relacionamentos do modelo

3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos

Agora, após a identificação de todas as entidades e, consequentemente, todos os relacionamentos existentes, partiremos para a identificação dos atributos que consideramos relevantes para cada entidade.

Entidade	Atributo	Descrição	Tipo de Dados e Tamanho	NULL	Tipo de atributo
Atleta	idAtleta	Número que identifica um atleta	INT	Não	Chave Primária
	email	Email do atleta usado para iniciar sessão	VARCHAR(45)	Não	Simples
	password	Password do atleta utilizada para iniciar sessão	VARCHAR(45)	Não	Simples
	nome	Nome usado para se registrar na aplicação	VARCHAR(45)	Não	Simples
	nºCC	Nº do cartão de cidadão do atleta	VARCHAR(15)	Não	Simples
	nif	Nº de identificação fiscal	VARCHAR(15)	Não	Simples

	ddn	Data de nascimento	DATETIME	Não	Simples
	genero	Género do atleta	CHAR	Não	Simples
	rua	Rua de residência	VARCHAR(45)	Não	Simples
	CodigoPostal_codigoPostal	Código postal do atleta	VARCHAR(8)	Não	Chave Estrangeira
	equipa	Equipa que o atleta representa	VARCHAR(45)	Sim	Simples
	contacto	Contacto do atleta	VARCHAR(13)	Não	Simples
	Modalidade_idModalidade	Identificador da modalidade praticada pelo atleta	INT	Não	Chave Estrangeira
Medico	idMedico	Número que identifica um médico	INT	Não	Chave Primária
	email	Email do médico utilizado para iniciar sessão	VARCHAR(45)	Não	Simples
	password	Password do médico utilizada para iniciar sessão	VARCHAR(45)	Não	Simples
	nome	Nome usado para se registar na aplicação	VARCHAR(45)	Não	Simples
	nºCC	Nº do cartão de cidadão	VARCHAR(15)	Não	Simples
	cedula	Nº da cédula profissional	VARCHAR(15)	Não	Simples
	ddn	Data de nascimento	DATETIME	Não	Simples
	genero	Género do médico	CHAR	Não	Simples
	rua	Rua de residência	VARCHAR(45)	Não	Simples
	CodigoPostal_codigoPostal	Código postal do médico	VARCHAR(8)	Não	Chave Estrangeira
	contacto	Contacto do médico	VARCHAR(13)	Não	Simples
	Clinica_idClinica	Identificador da clínica onde trabalha	INT	Não	Chave Estrangeira
Clinica	idClinica	Número identificador da clínica	INT	Não	Chave Primária
	nome	Nome da clínica	VARCHAR(45)	Não	Simples
	rua	Rua onde se localiza	VARCHAR(45)	Não	Simples
	CodigoPostal_codigoPostal	Código postal da clínica	VARCHAR(8)	Não	Chave Estrangeira

TesteClinico	idTesteClinico	Número identificador do teste	INT	Não	Chave Primária
	designação	Nome técnico do teste	VARCHAR(45)	Não	Simples
	preco	Custo da realização do teste	FLOAT	Não	Simples
TCAgendado	idTCAgendado	Número identificador de um teste agendado	INT	Não	Chave Primária
	TesteClinico_idTesteClinico	Identificador do teste a realizar	INT	Não	Chave Estrangeira
	data	Data de realização do teste	DATETIME	Não	Simples
	Atleta_idAtleta	Identificador do atleta submetido a teste	INT	Não	Chave Estrangeira
	Clinica_idClinica	Identificador da clínica onde se irá realizar o teste	INT	Não	Chave Estrangeira
	Medico_idMedico	Identificador do médico que irá realizar o teste	INT	Não	Chave Estrangeira
TCRealizado	idTCRealizado	Número identificador de um teste realizado	INT	Não	Chave Primária
	TesteClinico_idTesteClinico	Identificador do teste realizado	INT	Não	Chave Estrangeira
	data	Data de realização do teste	DATETIME	Não	Simples
	Atleta_idAtleta	Identificador do atleta submetido a teste	INT	Não	Chave Estrangeira
	Clinica_idClinica	Identificador da clínica onde se realizou o teste	INT	Não	Chave Estrangeira
	Medico_idMedico	Identificador do médico que realizou o teste	INT	Não	Chave Estrangeira

Tabela 3 - Atributos de cada entidade

3.5. Detalhe ou generalização de entidades

Apesar de inicialmente apenas possuirmos 4 entidades (o atleta, o médico, a clínica e o teste clínico), à medida que fomos desenvolvendo o projeto apercebemo-nos de que a entidade teste clínico teria de ser tratada com mais detalhe, por forma a distinguir os diferentes tipos de testes que existem e os diferentes estados em que podem estar (agendados ou realizados). Desta modo, esta entidade “separou-se” em três: a entidade “principal” TesteClinico, que caracteriza o tipo de teste a realizar; e as entidades TCAgendado e TCRealizado, que nos indicam se um dado teste já foi ou não realizado, os seus intervenientes (médico e atleta envolvidos) e qual a clínica onde se realizou.

Deste modo, aquando da marcação de um teste clínico, um atleta poderá mais facilmente escolher qual o exame que pretende realizar, uma vez que terá acesso a uma lista com todos os exames possíveis de realizar nas clínicas. Para além disto, o atleta poderá indicar a data em que pretende realizar, a clínica onde o pretende efetuar e, caso pretenda realizar o exame com um médico específico, poderá fazê-lo, verificando se este se encontra disponível para o atender.

3.6. Apresentação e explicação do diagrama ER

Ao longo da conceção do modelo conceptual, deparamo-nos com diversas alternativas, parecendo todas elas plausíveis de ser utilizadas. No entanto, como forma de garantir a total funcionalidade do sistema, tendo em conta aquilo que nos era requisitado, e uma melhor acessibilidade e organização dos dados, chegamos a este modelo, que nos aparentou ser o mais assertivo e adequado à base de dados em questão.

Relativamente às entidades Atleta e Médico, podemos afirmar que ambas se registam na aplicação através do seu *email* e *password*, preenchendo também todos os outros campos necessários, específicos de cada um. Posto isto, enquanto que o primeiro se torna assim capaz de agendar exame, o segundo passa a poder visualizar todos os exames que tem agendados, permitindo-lhe saber sempre quais os próximos a realizar. Em ambos os casos, é possível ver uma lista com todos os testes que já foram realizados pelo utilizador em questão.

Quanto à entidade TesteClinico, esta associa sempre um teste clínico ao seu custo, enquanto que as entidades TCAgendado e TCRealizado indicam o tipo de teste que está agendado/foi realizado, aqueles que estão ligados a este, a data em que foi realizado e a clínica onde decorreu.

Por fim, a entidade Clinica, que nos permite quase como que “agregar” toda esta informação, pois possui um registo completo de todos os testes lá realizados, todos os testes que ainda se vão realizar, e uma lista com todo o pessoal médico que lá trabalha.

Observando a cardinalidade dos relacionamentos, podemos verificar que um atleta pode agendar tantos testes quanto desejar, sendo que cada um será realizado por um médico.

Cada teste é de um tipo específico, e decorrerá numa clínica concreta (aquela onde o médico trabalha).

3.7. Revisão e validação do modelo com o utilizador

Para darmos por finalizada esta fase do projeto, é essencial a verificação e consequente validação do modelo por parte dos proprietários das clínicas, ou seja, estes devem assegurar-se de que o modelo apresentado é uma representação adequada daquilo que é pretendido.

Deste modo, após uma análise cuidada de todos os pontos acima referidos, os clientes deram o seu aval, aprovando o modelo construído, permitindo que possamos avançar para a próxima fase.

4. Modelo de Dados Lógico

4.1. Construção e validação do modelo de dados lógico

Nesta fase de transição de uma modelação conceptual para uma modelação lógica é necessário a derivação de todas as relações no modelo lógico que representem as entidades, relacionamentos e atributos que foram criados e usados anteriormente no modelo conceptual.

Para que tal objetivo seja alcançado, foquemos a nossa atenção nos elementos do modelo conceptual, tendo por base a *Database Definition Language*: entidades fortes, entidades fracas, relacionamentos binários um para muitos (1-N), relacionamentos binários um para um (1-1), relacionamentos recursivos um para um (1-1), relacionamentos superclasse/subclasse, relacionamentos binários muitos para muitos (N-M), relacionamentos complexos e atributos multivvalorados.

4.1.1 Entidades Fortes

Uma entidade é considerada forte quando não depende da existência de outra entidade e possui atributos suficientes para formar uma chave primária. Como tal, podemos considerar as seguintes entidades fortes, na criação do nosso modelo:

- **Atleta** (idAtleta, email, password, nome, nºCC, nif, ddn, genero, rua, CodigoPostal_codigoPostal, equipa, contacto, Modalidade_idModalidade)
Chave primária: idAtleta
- **TesteClinico** (idTesteClinico, designação, preco)
Chave primária: idTesteClinico
- **Clinica** (idClinica, nome, rua, CodigoPostal_codigoPostal)
Chave primária: idClinica

Apesar de as entidades Atleta e Clinica dependerem da existência de outros identificadores (código postal e, no caso do atleta, modalidade também), consideramo-las

entidades fortes, uma vez que não dependem de nenhuma das entidades principais do programa.

4.1.2 Entidades Fracas

Considera-se uma entidade como sendo fraca quando a sua existência depende de outras entidades, sendo que a sua existência só não faz completo sentido. Sendo assim, consideramos as seguintes entidades do modelo como fracas:

- **Medico** (idMedico, email, password, nome, nºCC, cedula, ddn, genero, rua, CodigoPostal_codigoPostal, contacto, Clinica_idClinica)
Chave primária: idMédico
- **TCAgendado** (idTCAgendado, TesteClinico_idTesteClinico, data, Atleta_idAtleta, Clinica_idClinica, Medico_idMedico)
Chave primária: idTCAgendado
- **TCRealizado** (idTCRealizado, TesteClinico_idTesteClinico, data, Atleta_idAtleta, Clinica_idClinica, Medico_idMedico)
Chave primária: idTCRealizado

4.1.3 Relacionamentos binários um para muitos (1-N)

Neste tipo particular de relacionamento, a entidade com multiplicidade N adquire um novo atributo, também designado por chave estrangeira, que, na prática, é a chave primária da entidade com multiplicidade 1. O nosso modelo apresenta diversos relacionamentos deste tipo (Clinica-Medico, TesteClinico-TCAgendado, TesteClinico-TCRealizado, Atleta-TCAgendado, Atleta-TCRealizado, Clinica-TCAgendado, Clinica-TCRealizado, Medico-TCAgendado, Medico-TCRealizado).

- **Medico** (idMedico, email, password, nome, nºCC, cedula, ddn, genero, rua, CodigoPostal_codigoPostal, contacto, *Clinica_idClinica*)
Chave primária: idMédico
Chave estrangeira: Clinica_idClinica (proveniente da relação com Clinica)
- **TCAgendado** (idTCAgendado, TesteClinico_idTesteClinico, data, Atleta_idAtleta, Clinica_idClinica, Medico_idMedico)
Chave primária: idTCAgendado

Chaves estrangeiras: TesteClinico_idTesteClinico (proveniente da relação com TesteClinico), Atleta_idAtleta (proveniente da relação com Atleta), Clinica_idClinica (proveniente da relação com Clinica), Medico_idMedico (proveniente da relação com Medico)

- **TCRealizado** (idTCRealizado, *TesteClinico_idTesteClinico*, *data*, *Atleta_idAtleta*, *Clinica_idClinica*, *Medico_idMedico*)

Chave primária: idTCRealizado

Chaves estrangeiras: TesteClinico_idTesteClinico (proveniente da relação com TesteClinico), Atleta_idAtleta (proveniente da relação com Atleta), Clinica_idClinica (proveniente da relação com Clinica), Medico_idMedico (proveniente da relação com Medico)

4.1.4 Relacionamentos binários um para um (1-1)

No caso particular do nosso modelo, não consideramos nenhum relacionamento deste tipo.

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

No caso particular do nosso modelo, não consideramos nenhum relacionamento deste tipo.

4.1.6 Relacionamentos superclasse/subclasse

No caso particular do nosso modelo, não consideramos nenhum relacionamento deste tipo.

4.1.7 Relacionamentos binários muitos para muitos (N-M)

Apesar de não existir nenhum relacionamento binário de muitos para muitos, entre as entidades ditas “principais” do nosso modelo, podemos observar a existência de dois relacionamentos deste tipo, quando consideradas todas as entidades presentes: relacionamento TesteClinico-Equipamento e relacionamento TCRealizado-EquipamentoDisponivel.

Para começar, podemos ilustrar quais as diferenças entre Equipamento e EquipamentoDisponivel: enquanto que a primeira entidade é relativa a todos os tipos de

equipamentos que as clínicas possuem, a segunda indica-nos quantos equipamentos de um dado tipo existem numa dada clínica, identificando esses equipamentos com um id próprio.

Deste modo, podemos descrever o relacionamento TesteClinico-Equipamento como aquele que indica quais os tipos de equipamentos necessários para realizar um dado teste clínico. Assim, podemos afirmar que se trata de um relacionamento N-M, uma vez que um teste pode necessitar de diversos equipamentos, tal como um equipamento pode ser utilizado em diversos testes. Devido a tal, é criada a tabela EquipamentoNecessario.

Por outro lado, o relacionamento TCRealizado-EquipamentoDisponivel identifica quais os equipamentos que foram efetivamente utilizados para realizar um teste. Ou seja, do mesmo modo que vários equipamentos podem ter sido utilizados para realizar um teste, vários testes podem ser realizados usando o mesmo equipamento. Posto isto, é criada a tabela EquipamentoUtilizado.

- **EquipamentoNecessario** (TesteClinico_idTesteClinico, Equipamento_idEquipamento)
Chave primária composta: TesteClinico_idTesteClinico, Equipamento_idEquipamento.
- **EquipamentoUtilizado** (EquipamentoDisponivel_idEquipamentoDisponivel, TCRealizado_idTCRealizado)
Chave primária composta: EquipamentoDisponivel_idEquipamentoDisponivel, TCRealizado_idTCRealizado.

4.1.8 Relacionamentos complexos

No caso particular do nosso modelo, não consideramos nenhum relacionamento deste tipo.

4.2. Desenho do modelo lógico

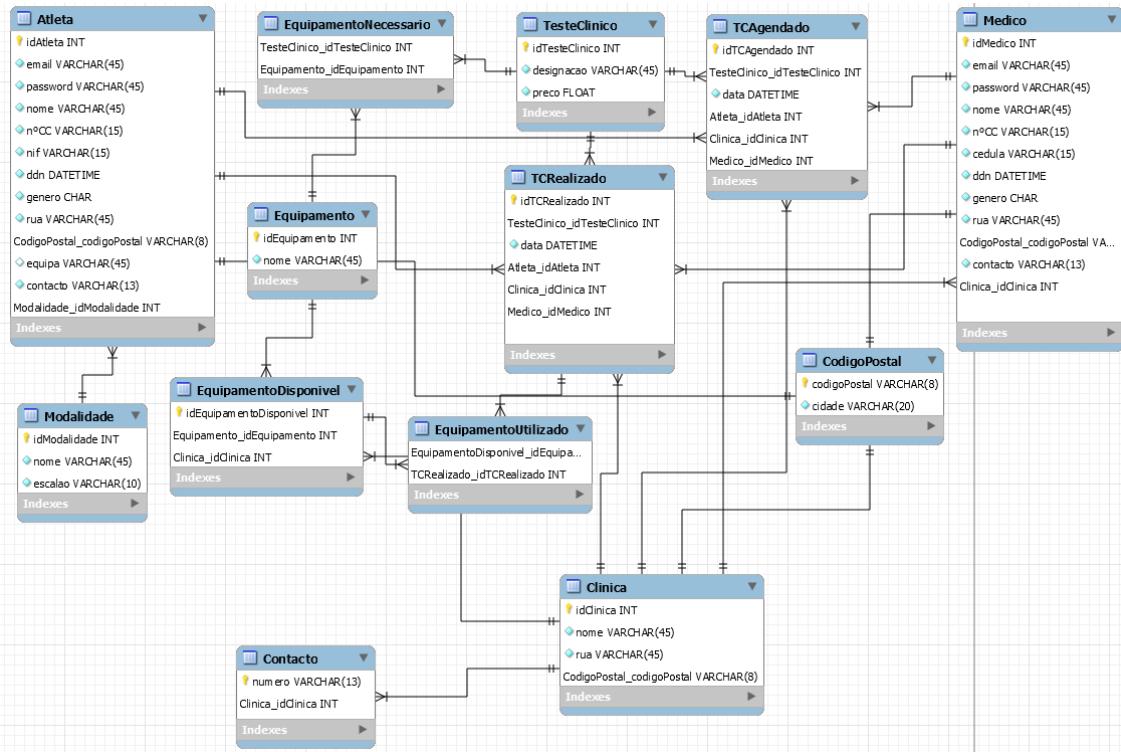


Figura 5 - Modelo Lógico

4.3. Validação do modelo através da normalização

O principal objetivo da normalização de um modelo relacional passa pela resolução de problemas de atualização de bases de dados, minimizando redundâncias. Este pode ser visto como o processo de eliminação de esquemas de relações (tabelas) não satisfatórios, separando os seus atributos em esquemas de relações menos complexas, com o intuito de satisfazerem propriedades desejadas.

Por norma, normalizar até à 3ª Forma Normal (3FN) é suficiente para garantir a inexistência de redundância informacional. Como tal, iremos descrever cada forma normal, indicando, sem seguida, o porquê de o nosso modelo as respeitar.

4.3.1 1FN – 1ª Forma Normal

“Toda relação deve ter uma chave primária e deve-se garantir que todo atributo seja atômico.”

Como podemos verificar, toda a relação do nosso modelo apresenta uma chave primária e todos os atributos são atómicos. Assim, podemos concluir que respeita a 1ª forma normal.

4.3.2 2FN – 2^a Forma Normal

“Uma relação está na 2^a Forma Normal se está na primeira e se todos os atributos não-chave dependerem da totalidade da chave-primária (e não apenas de parte dela - Dependências funcionais parciais).”

As únicas chaves primárias compostas presentes no nosso modelo pertencem às relações EquipamentoNecessario e EquipamentoUtilizado e os atributos dependem da totalidade da chave primária, logo o modelo respeita a 2^a forma normal.

4.3.3 3FN – 3^a Forma Normal

“Uma relação está na 3^a Forma Normal se está na 2^a Formal Normal e nenhum atributo não-chave é transitivamente dependente da chave primária.”

Assim podemos concluir que o nosso modelo respeita a 3^a forma normal.

4.4. Validação do modelo com as interrogações do utilizador

Um requisito básico para obter a validade do modelo, é que este responda a todas as perguntas dos seus utilizadores.

Tendo isto em conta, selecionamos aquelas que foram tomadas como pertinentes, e verificamos se estas podem ser respondidas, tendo em conta o modelo criado:

- Consulta da disponibilidade do médico com o id nº15, as 16h30m do dia 25 de julho de 2020**

Para ter acesso a esta informação é necessária a tabela TCAgendado.

Começamos por filtrar a informação guardada na tabela, mantendo apenas as consultas agendadas para o dado médico. Posto isto, verificamos quais as consultas cujas datas estão inseridas no intervalo compreendido entre meia hora antes e meia hora depois da data (de lembrar que o tipo DATETIME em SQL guarda não só uma data como também uma hora; além disto, consideramos que cada teste demora aproximadamente meia hora). Caso o total destas consultas seja 0, então o médico encontra-se disponível para atender o atleta.

- Consulta dos três exames mais realizados no mês de outubro de 2016**

De modo a obter esta informação, necessitamos das tabelas TCRealizado e TesteClinico. Depois de reunida toda a informação, filtramos os testes cuja data está entre o primeiro e o último dia do mês de outubro de 2016. Agrupando esta informação por testes, faz-se uma soma das

quantidades, que são posteriormente ordenadas por ordem decrescente, sendo, por fim, retiradas da tabela resultante as três primeiras linhas.

- **Consulta do atleta mais velho ou mais novo**

Com o intuito de responder a esta pergunta, necessitamos de aceder à tabela Atleta. Posto isto, ordena-se a tabela por ordem crescente ou decrescente de data de nascimento, consoante desejamos saber qual o atleta mais velho ou mais novo. Por fim, retiramos a primeira linha da coluna.

- **Consulta de todos os atletas que realizaram Eletrocardiogramas**

Por forma a obter a informação necessária a responder a esta questão, necessitamos da informação das tabelas TCRealizado, Atleta e TesteClinico. Em seguida, iremos filtrar a informação destas, por forma a selecionar apenas as entradas da tabela TCRealizado correspondentes a Eletrocardiogramas. Por fim, retiramos o nome dos atletas que realizaram estes testes, ordenados por ordem crescente.

- **Consulta do número de testes realizado por cada médico**

Com a finalidade de responder a esta pergunta, necessitamos da informação das tabelas TCRealizado e Medico. Começamos por agrupar a tabela TCRealizado tendo em conta o id do médico que realizou o teste. Em seguida, contamos o número de entradas que um dado médico tem nessa tabela, ou seja, o número de testes que realizou. Por fim, ordenamos a tabela por ordem decrescente do número de testes realizados.

- **Consulta de todos os médicos com idades compreendidas entre os 45 e os 60 anos que realizaram exames à urina.**

Tendo em vista o acesso a esta informação, necessitamos das tabelas Medico e TCRealizado. Inicialmente, verificamos quais os médicos com as idades compreendidas entre 45 e 60 anos. Seguidamente, verificamos, de entre todos os testes que realizaram, quais aqueles que foram à urina. Por fim, ordenamos a lista resultante, tendo em conta o nome dos médicos.

4.5. Validação do modelo com as transações estabelecidas

Ao nível do modelo lógico, validamos as transações através de mapas de transações. Desta forma, é possível verificar visualmente as interações entre as diferentes Entidades, Atributos e Relacionamentos de forma a obter o resultado pretendido.

- **Agendar um Teste Clínico**

Por forma a registar um teste, é necessário verificar se o teste clínico em questão existe. Como tal, acede-se à tabela TesteClinico, por forma a validar a sua existência. Posto isto, é necessário verificar se o médico com quem o atleta pretende realizar o exame está disponível na data pretendida. Para tal, procede-se como descrito na secção anterior, no primeiro ponto. Posto isto, caso o médico se encontre disponível, podemos prosseguir para o registo de agendamento do teste na base de dados, adicionando o id e toda a respetiva informação na tabela TCAgendado.

- **Remover o agendamento de um Teste Clínico**

De modo a remover um teste agendado, acede-se à tabela TCAgendado e elimina-se o id do teste na tabela.

- **Mudar o estado de um teste de “agendado” para “realizado”**

De maneira a alterar o estado de um teste, “migra-se” a informação da tabela TCAgendado para a tabela TCRealizado, alterando, no entanto, o id do teste, por forma a adequar-se à nova tabela. No fim disto, acede-se à tabela TCAgendado e elimina-se o id do teste da tabela.

4.6. Reavaliação do modelo lógico

Não foi necessário reavaliar o modelo lógico porque a nosso ver, a normalização já se encontra feita.

4.7. Revisão do modelo lógico com o utilizador

Tendo em vista a conclusão desta fase, torna-se novamente necessário que os proprietários das clínicas verifiquem se o modelo lógico construído até agora corresponde as expectativas e necessidades.

Deste modo, tivemos de analisar toda a documentação do modelo lógico, bem como as entidades e os seus relacionamentos e atributos. Após esta revisão, tornou-se possível identificar todas as tabelas importantes no modelo e os seus diversos relacionamentos. Ao revermos a nossa tabela de relacionamentos, confirmamos que os relacionamentos utilizados são aqueles que melhor se identificam com a realidade do problema. Por ultimo, foram validados os atributos, verificando que o utilizador consegue averiguar toda a informação de uma entidade e ter uma melhor noção dos domínios de cada atributo. Posto isto, o nosso modelo foi aprovado pelos proprietários, pois estes veem-no como uma solução capaz de responder às necessidades pretendidas.

5. Modelo de Dados Físico

5.1. Seleção do sistema de gestão de bases de dados

Para construirmos a Base de Dados proposta utilizamos como sistema de gestão de base de dados o *MySQL*. Esta decisão deve-se ao facto deste sistema ter sido o usado nesta Unidade Curricular, o que de certa forma veio facilitar a nossa implementação da Base de Dados.

5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL

A modelação do esquema físico de uma base de dados necessita de uma tradução das relações base, estando estas definidas previamente no modelo lógico, por forma a que estas e as suas restrições sejam suportadas pelo SGBD.

Deste modo, por forma a criar um esquema físico que represente corretamente aquilo que pretendemos, necessitamos de concluir alguns passos, como por exemplo a descrição das relações base e o desenho das restrições gerais.

5.2.1 Descrição das relações base

- Relação Clinica

Domínio idClinica (inteiro)

Domínio nome (String de tamanho variável, de tamanho 45)

Domínio rua (String de tamanho variável, de tamanho 45)

Domínio CodigoPostal_codigoPostal (String de tamanho variável, de tamanho 8)

Clinica(

idClinica (NOT NULL),

nome (NOT NULL),

```

rua (NOT NULL)
CodigoPostal_codigoPostal (NOT NULL),
PRIMARY KEY (idClinica);
FOREIGN      KEY      (CodigoPostal_codigoPostal)      REFERENCES
CodigoPostal(codigoPostal)

```

- **Relação Atleta**

Domínio idAtleta (inteiro)
 Domínio email (String de tamanho variável, de tamanho 45)
 Domínio password (String de tamanho variável, de tamanho 45)
 Domínio nome (String de tamanho variável, de tamanho 45)
 Domínio nºCC (String de tamanho variável, de tamanho 15)
 Domínio nif (String de tamanho variável, de tamanho 15)
 Domínio ddn (*datetime*)
 Domínio genero (char)
 Domínio rua (String de tamanho variável, de tamanho 45)
 Domínio CodigoPostal_codigoPostal (String de tamanho variável, de tamanho 8)
 Domínio equipa (String de tamanho variável, de tamanho 45)
 Domínio contacto (String de tamanho variável, de tamanho 13)
 Domínio Modalidade_idModalidade (inteiro)

Atleta(
 idAtleta (NOT NULL),
 email (NOT NULL),
 password (NOT NULL),
 nome (NOT NULL),
 nºCC (NOT NULL),
 nif (NOT NULL),
 ddn (NOT NULL),
 genero (NOT NULL),
 rua (NOT NULL),
 CodigoPostal_codigoPostal (NOT NULL),
 equipa (NULL),
 contacto (NOT NULL),
 Modalidade_idModalidade (NOT NULL),
 PRIMARY KEY (idAtleta);

FOREIGN	KEY	(CodigoPostal_codigoPostal)	REFERENCES
CodigoPostal(codigoPostal);			
FOREIGN	KEY	(Modalidade_idModalidade)	REFERENCES
Modalidade(idModalidade))			

- **Relação Medico**

Domínio idMedico (inteiro)
 Domínio email (String de tamanho variável, de tamanho 45)
 Domínio password (String de tamanho variável, de tamanho 45)
 Domínio nome (String de tamanho variável, de tamanho 45)
 Domínio nºCC (String de tamanho variável, de tamanho 15)
 Domínio cedula (String de tamanho variável, de tamanho 15)
 Domínio ddn (*datetime*)
 Domínio genero (char)
 Domínio rua (String de tamanho variável, de tamanho 45)
 Domínio CodigoPostal_codigoPostal (String de tamanho variável, de tamanho 8)
 Domínio contacto (String de tamanho variável, de tamanho 13)
 Domínio Clinica_idClinica (inteiro)

```

Medico(
  idMedico (NOT NULL),
  email (NOT NULL),
  password (NOT NULL),
  nome (NOT NULL),
  nºCC (NOT NULL),
  cedula (NOT NULL),
  ddn (NOT NULL),
  genero (NOT NULL),
  rua (NOT NULL),
  CodigoPostal_codigoPostal (NOT NULL),
  contacto (NOT NULL),
  Clinica_idClinica (NOT NULL),
  PRIMARY KEY (idAtleta);
  FOREIGN      KEY      (CodigoPostal_codigoPostal)      REFERENCES
  CodigoPostal(codigoPostal);
  FOREIGN KEY (Clinica_idClinica) REFERENCES Clinica(idClinica))
  
```

- **Relação TesteClinico**

Domínio idTesteClinico (inteiro)

Domínio designacao (String de tamanho variável, de tamanho 45)

Domínio preco (*float*)

TesteClinico(

idTesteClinico (NOT NULL),

designacao (NOT NULL),

preco (NOT NULL),

PRIMARY KEY (idTesteClinico))

- **Relação TCAgendado**

Domínio idTCAgendado (inteiro)

Domínio TesteClinico_idTesteClinico (inteiro)

Domínio data (*datetime*)

Domínio Atleta_idAtleta (inteiro)

Domínio Clinica_idClinica (inteiro)

Domínio Medico_idMedico (inteiro)

TCAgendado(

idTCAgendado (NOT NULL),

TesteClinico_idTesteClinico (NOT NULL),

data (NOT NULL),

Atleta_idAtleta (NOT NULL),

Clinica_idClinica (NOT NULL),

Medico_idMedico (NOT NULL),

PRIMARY KEY (idTCAgendado);

FOREIGN KEY (TesteClinico_idTesteClinico) REFERENCES TesteClinico(idTesteClinico);

FOREIGN KEY (Atleta_idAtleta) REFERENCES Atleta(idAtleta);

FOREIGN KEY (Clinica_idClinica) REFERENCES Clinica(idClinica);

FOREIGN KEY (Medico_idMedico) REFERENCES Medico(idMedico))

- **Relação TCRealizado**

Domínio idTCRealizado (inteiro)

Domínio TesteClinico_idTesteClinico (inteiro)

Domínio data (*datetime*)

Domínio Atleta_idAtleta (inteiro)
 Domínio Clinica_idClinica (inteiro)
 Domínio Medico_idMedico (inteiro)

```

TCRealizado(
  idTCRealizado (NOT NULL),
  TesteClinico_idTesteClinico (NOT NULL),
  data (NOT NULL),
  Atleta_idAtleta (NOT NULL),
  Clinica_idClinica (NOT NULL),
  Medico_idMedico (NOT NULL),
  PRIMARY KEY (idTCRealizado);
  FOREIGN      KEY      (TesteClinico_idTesteClinico)      REFERENCES
  TesteClinico(idTesteClinico);
  FOREIGN KEY (Atleta_idAtleta) REFERENCES Atleta(idAtleta);
  FOREIGN KEY (Clinica_idClinica) REFERENCES Clinica(idClinica);
  FOREIGN KEY (Medico_idMedico) REFERENCES Medico(idMedico))

```

5.2.2 Desenho das restrições gerais

Para este ponto, é necessário apresentar as condições restritivas gerais que dizem respeito ao problema. Assim, vão ser apresentadas todas as restrições agrupadas por relação. As restrições que vão ser apresentadas têm a sua definição no script de criação das tabelas.

- **Clinica**

```

CREATE TABLE IF NOT EXISTS `mieicare`.`Clinica` (
  `idClinica` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `rua` VARCHAR(45) NOT NULL,
  `CodigoPostal_codigoPostal` VARCHAR(8) NOT NULL,
  PRIMARY KEY (`idClinica`, `CodigoPostal_codigoPostal`),
  INDEX `fk_Clinica_CodigoPostal1_idx` (`CodigoPostal_codigoPostal` ASC) VISIBLE,
  CONSTRAINT `fk_Clinica_CodigoPostal1`
    FOREIGN KEY (`CodigoPostal_codigoPostal`)
    REFERENCES `mieicare`.`CodigoPostal`(`codigoPostal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 6 - Criação da tabela Clinica

- **Atleta**

```
CREATE TABLE IF NOT EXISTS `mieicare`.`Atleta` (
  `idAtleta` INT NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `nºCC` VARCHAR(15) NOT NULL,
  `nif` VARCHAR(15) NOT NULL,
  `ddn` DATETIME NOT NULL,
  `genero` CHAR BINARY NOT NULL,
  `rua` VARCHAR(45) NOT NULL,
  `CodigoPostal_codigoPostal` VARCHAR(8) NOT NULL,
  `equipa` VARCHAR(45) NULL,
  `contacto` VARCHAR(13) NOT NULL,
  `Modalidade_idModalidade` INT NOT NULL,
PRIMARY KEY (`idAtleta`, `CodigoPostal_codigoPostal`, `Modalidade_idModalidade`),
INDEX `fk_Atleta_CodigoPostal_idx` (`CodigoPostal_codigoPostal` ASC) VISIBLE,
INDEX `fk_Atleta_Modalidade1_idx` (`Modalidade_idModalidade` ASC) VISIBLE,
CONSTRAINT `fk_Atleta_CodigoPostal`
  FOREIGN KEY (`CodigoPostal_codigoPostal`)
  REFERENCES `mieicare`.`CodigoPostal`(`codigoPostal`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `fk_Atleta_Modalidade1`
  FOREIGN KEY (`Modalidade_idModalidade`)
  REFERENCES `mieicare`.`Modalidade`(`idModalidade`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 7 - Criação da tabela Atleta

- **Medico**

```

CREATE TABLE IF NOT EXISTS `mieicare`.`Medico` (
  `idMedico` INT NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `nºCC` VARCHAR(15) NOT NULL,
  `cedula` VARCHAR(15) NOT NULL,
  `ddn` DATETIME NOT NULL,
  `genero` CHAR BINARY NOT NULL,
  `rua` VARCHAR(45) NOT NULL,
  `CodigoPostal_codigoPostal` VARCHAR(8) NOT NULL,
  `contacto` VARCHAR(13) NOT NULL,
  `Clinica_idClinica` INT NOT NULL,
  PRIMARY KEY (`idMedico`, `CodigoPostal_codigoPostal`, `Clinica_idClinica`),
  INDEX `fk_Medico_CodigoPostal1_idx`(`CodigoPostal_codigoPostal` ASC) VISIBLE,
  INDEX `fk_Medico_Clinica1_idx`(`Clinica_idClinica` ASC) VISIBLE,
  CONSTRAINT `fk_Medico_CodigoPostal1`
    FOREIGN KEY (`CodigoPostal_codigoPostal`)
    REFERENCES `mieicare`.`CodigoPostal` (`codigoPostal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Medico_Clinica1`
    FOREIGN KEY (`Clinica_idClinica`)
    REFERENCES `mieicare`.`Clinica` (`idClinica`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 8 - Criação da tabela Medico

- **TesteClinico**

```

CREATE TABLE IF NOT EXISTS `mieicare`.`TesteClinico` (
  `idTesteClinico` INT NOT NULL AUTO_INCREMENT,
  `designacao` VARCHAR(45) NOT NULL,
  `preco` FLOAT NOT NULL,
  PRIMARY KEY (`idTesteClinico`))
ENGINE = InnoDB;

```

Figura 9 - Criação da tabela TesteClinico

- **TCAgendado**

```

CREATE TABLE IF NOT EXISTS `mieicare`.'TCAgendado` (
    `idTCAgendado` INT NOT NULL AUTO_INCREMENT,
    `TesteClinico_idTesteClinico` INT NOT NULL,
    `data` DATETIME NOT NULL,
    `Atleta_idAtleta` INT NOT NULL,
    `Clinica_idClinica` INT NOT NULL,
    `Medico_idMedico` INT NOT NULL,
    PRIMARY KEY (`idTCAgendado`, `TesteClinico_idTesteClinico`, `Atleta_idAtleta`, `Clinica_idClinica`, `Medico_idMedico`),
    INDEX `fk_TCAgendado_Clinical_idx` (`Clinica_idClinica` ASC) VISIBLE,
    INDEX `fk_TCAgendado_Medico1_idx` (`Medico_idMedico` ASC) VISIBLE,
    INDEX `fk_TCAgendado_TestClinico1_idx` (`TesteClinico_idTesteClinico` ASC) VISIBLE,
    CONSTRAINT `fk_TCAgendado_Atleta1`
        FOREIGN KEY (`Atleta_idAtleta`)
            REFERENCES `mieicare`.'Atleta` ('idAtleta')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_TCAgendado_Clinical1`
        FOREIGN KEY (`Clinica_idClinica`)
            REFERENCES `mieicare`.'Clinica` ('idClinica')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_TCAgendado_Medico1`
        FOREIGN KEY (`Medico_idMedico`)
            REFERENCES `mieicare`.'Medico` ('idMedico')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_TCAgendado_TestClinico1`
        FOREIGN KEY (`TesteClinico_idTesteClinico`)
            REFERENCES `mieicare`.'TesteClinico` ('idTesteClinico')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 10 - Criação da tabela TCAgendado

- **TCRealizado**

```

CREATE TABLE IF NOT EXISTS `mieicare`.'TCRealizado` (
    `idTCRealizado` INT NOT NULL AUTO_INCREMENT,
    `TesteClinico_idTesteClinico` INT NOT NULL,
    `data` DATETIME NOT NULL,
    `Atleta_idAtleta` INT NOT NULL,
    `Clinica_idClinica` INT NOT NULL,
    `Medico_idMedico` INT NOT NULL,
    PRIMARY KEY (`idTCRealizado`, `TesteClinico_idTesteClinico`, `Atleta_idAtleta`, `Clinica_idClinica`, `Medico_idMedico`),
    INDEX `fk_TCRealizado_Clinical1_idx` (`Clinica_idClinica` ASC) VISIBLE,
    INDEX `fk_TCRealizado_Medico1_idx` (`Medico_idMedico` ASC) VISIBLE,
    INDEX `fk_TCRealizado_TestClinico1_idx` (`TesteClinico_idTesteClinico` ASC) VISIBLE,
    CONSTRAINT `fk_TCRealizado_Atleta1`
        FOREIGN KEY (`Atleta_idAtleta`)
            REFERENCES `mieicare`.'Atleta` ('idAtleta')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_TCRealizado_Clinical1`
        FOREIGN KEY (`Clinica_idClinica`)
            REFERENCES `mieicare`.'Clinica` ('idClinica')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_TCRealizado_Medico1`
        FOREIGN KEY (`Medico_idMedico`)
            REFERENCES `mieicare`.'Medico` ('idMedico')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_TCRealizado_TestClinico1`
        FOREIGN KEY (`TesteClinico_idTesteClinico`)
            REFERENCES `mieicare`.'TesteClinico` ('idTesteClinico')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 11 - Criação da tabela TCRealizado

5.3. Tradução das interrogações do utilizador para SQL

- **1^a Querie**

Consulta da disponibilidade do médico com o id nº15, as 16h30m do dia 25 de julho de 2020.

```
SELECT COUNT(idTCAgendado) AS ConsultasAgendadas
FROM TCAgendado AS TCA
    INNER JOIN Medico AS M
    ON Medico_idMedico = '15'
    WHERE TCA.data BETWEEN '2020-07-25 16:00:00' AND '2020-07-25 17:00:00'
```

Figura 12 - 1^a querie em SQL

- **2^a Querie**

Consulta dos três exames mais realizados no mês de outubro de 2016.

```
SELECT T.designacao, COUNT(idTCRealizado) AS NumTestes
FROM TCRealizado AS TCR
    INNER JOIN TesteClinico AS T
    ON TCR.TesteClinico_idTesteClinico = T.idTesteClinico
    WHERE TCR.data BETWEEN '2016-10-01 00:00:00' AND '2016-10-31 23:59:59'
    GROUP BY TesteClinico_idTesteClinico
    ORDER BY NumTestes DESC
    LIMIT 3
```

Figura 13 - 2^a querie em SQL

- **3^a Querie**

Consulta do atleta mais velho ou mais novo.

```
SELECT nome, ddn FROM Atleta
ORDER BY ddn ASC
LIMIT 1
```

Figura 14 - 3^a querie (atleta mais velho) em SQL

```
SELECT nome, ddn FROM Atleta
ORDER BY ddn DESC
LIMIT 1
```

Figura 15 - 3^a querie (atleta mais novo) em SQL

- **4^a Querie**

Consulta de todos os atletas que realizaram Eletrocardiogramas.

```
SELECT A.nome, COUNT(*) AS Num_Testes from TCRealizado AS TCR
    INNER JOIN Atleta AS A
        ON TCR.Atleta_idAtleta = A.idAtleta
    INNER JOIN TesteClinico AS T
        ON TCR.TesteClinico_idTesteClinico = T.idTesteClinico
    WHERE T.designacao = 'Eletrocardiograma'
    GROUP BY TCR.Atleta_idAtleta
    ORDER BY A.nome ASC
```

Figura 16 - 4^a querie em SQL

- **5^a Querie**

Consulta do número de testes realizado por cada médico.

```
SELECT M.nome, COUNT(*) AS TestesRealizados
FROM TCRealizado AS TCR
    INNER JOIN Medico AS M
        ON TCR.Medico_idMedico = M.idMedico
    GROUP BY Medico_idMedico
    ORDER BY TestesRealizados DESC
```

Figura 17 - 5^a querie em SQL

- **6^a Querie**

Consulta de todos os médicos com idades compreendidas entre os 45 e os 60 anos que realizaram exames à urina.

```
SELECT M.nome, COUNT(*) AS TestesRealizados FROM TCRealizado AS TCR
    INNER JOIN Medico as M
        ON TCR.Medico_idMedico = M.idMedico
    INNER JOIN TesteClinico AS T
        ON TCR.TesteClinico_idTesteClinico = T.idTesteClinico
    WHERE T.designacao = 'Urina' AND (M.ddn BETWEEN '1960-01-01 00:00:00' AND '1975-12-31 23:59:59')
    GROUP BY M.nome
    ORDER BY M.nome
```

Figura 18 - 6^a querie em SQL

5.4. Tradução das transações estabelecidas para SQL

Por forma a aumentar a performance do SGBD que desenvolvemos, procedemos a uma análise de algumas transações que ocorrem entre as relações do nosso modelo desenvolvido.

Posto isto, chegamos à conclusão de que existem duas operações bastante frequentes no nosso modelo. Ambas implicam um *INSERT*, uma vez que se estas são o registo do agendamento de um teste clínico e, posteriormente, o registo desse teste como realizado. Sendo estas operações passíveis de ser transações (não implicam apenas operações de leitura), consideram-se as seguintes transações em *SQL*:

```
-- Transação que regista o agendamento de um teste
DELIMITER $$
CREATE PROCEDURE agenda_teste (IN idTCA INT, IN idT INT, IN d DATETIME, IN idA INT, IN idC INT, IN idM INT)

BEGIN
    DECLARE Erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;
    START TRANSACTION;

    -- inserção do teste
    INSERT INTO TCAgendado (idTCAgendado, TesteClinico_idTesteClinico, data, Atleta_idAtleta, Clinica_idClinica, Medico_idMedico)
    VALUES (idTCA, idT, d, idA, idC, idM);

    IF Erro
    THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END $$
```

Figura 19 - Transação de agendamento de um teste

```
-- Transação que regista o agendamento de um teste
DELIMITER $$
CREATE PROCEDURE agenda_teste (IN idTCA INT, IN idT INT, IN d DATETIME, IN idA INT, IN idC INT, IN idM INT)

BEGIN
    DECLARE Erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;
    START TRANSACTION;

    -- inserção do teste
    INSERT INTO TCAgendado (idTCAgendado, TesteClinico_idTesteClinico, data, Atleta_idAtleta, Clinica_idClinica, Medico_idMedico)
    VALUES (idTCA, idT, d, idA, idC, idM);

    IF Erro
    THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END $$
```

Figura 20 - Transação de deleção de um teste agendado

```

-- Transação que regista um teste como realizado
DELIMITER $$ 
CREATE PROCEDURE teste_realizado (IN idTCA INT, IN idTCR INT, IN idT INT, IN d DATETIME, IN idA INT, IN idC INT, IN idM INT, IN eu INT)

BEGIN
    DECLARE Erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;
    START TRANSACTION;

    -- inserção do teste como realizado
    INSERT INTO TCRealizado (idTCRealizado, TesteClinico_idTesteClinico, data, Atleta_idAtleta, Clinica_idClinica, Medico_idMedico)
    VALUES (idTCR, idT, d, idA, idC, idM);

    -- apagar o teste como agendado
    DELETE FROM TCAgendado WHERE idTCAgendado = idTCA;

    -- adicionar o equipamento utilizado, caso algum tenha sido usado
    IF (eu != NULL)
    THEN INSERT INTO EquipamentoUtilizado (EquipamentoDisponivel_idEquipamentoDisponivel, TCRealizado_idTCRealizado) VALUES (eu, idTCR);
    END IF;

    IF Erro
    THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END $$
```

Figura 21 - Transação de realização de um teste

5.5. Escolha, definição e caracterização de índices em SQL

Sendo índices estruturas de dados que permitem ao SGBD localizar registos num ficheiro/arquivo, com o intuito de minimizar o tempo de resposta de *queries* feitas pelo utilizador, achamos por bem implementá-los no nosso projeto, uma vez que poderiam vir a ser úteis em *queries* realizadas futuramente.

Toda a tabela *InnoDB* tem um índice especial, apelidado de *clustered index*. É através deste índice que todos os dados das linhas de uma tabela são armazenados. Por outras palavras, podemos afirmar que este índice corresponde à chave primária da tabela.

Por norma, o *InnoDB* utiliza a chave primária para otimizar as operações de *DML* mais comuns (*SELECT*, *INSERT*, *DELETE* e *UPDATE*). No entanto, é possível dizer-lhe para utilizar outros índices (predefinidos por nós) por forma a otimizar uma operação.

Tendo tudo isto em conta, criamos os seguintes índices:

```

CREATE INDEX IdClinica ON Clinica(idClinica);
CREATE INDEX NomeAtleta ON Atleta(nome);
CREATE INDEX NomeMedico ON Medico(nome);
CREATE INDEX DesignacaoTesteClinico ON TesteClinico(designacao);
CREATE INDEX DataTCAgendado ON TCAgendado(data);
CREATE INDEX DataTCRealizado ON TCRealizado(data);
```

Figura 22 - Criação de Índices

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

5.6.1 Tamanho de cada uma das tabelas das Clínicas MIEICare, após o povoamento inicial

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
CodigoPostal	codigoPostal	VARCHAR(8)	8 bytes
	cidade	VARCHAR(20)	20 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 15*28 = 420		

Tabela 4 – CodigoPostal

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
Modalidade	idModalidade	INT	4 bytes
	nome	VARCHAR(45)	45 bytes
	escalao	VARCHAR(10)	10 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 36*59 = 2124		

Tabela 5 – Modalidade

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
TesteClinico	idTesteClinico	INT	4 bytes
	designacao	VARCHAR(45)	45 bytes
	preco	FLOAT	6 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 11*55 = 605		

Tabela 6 – TesteClinico

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
Equipamento	idEquipamento	INT	4 bytes
	nome	VARCHAR(45)	45 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 2*49 = 98		

Tabela 7 – Equipamento

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
Clinica	idClinica	INT	4 bytes
	nome	VARCHAR(45)	45 bytes
	rua	VARCHAR(45)	45 bytes
	CodigoPostal_codigoPostal	VARCHAR(8)	8 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 3*102 = 306		

Tabela 8 – Clinica

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
Atleta	idAtleta	INT	4 bytes
	email	VARCHAR(45)	45 bytes
	password	VARCHAR(45)	45 bytes
	nome	VARCHAR(45)	45 bytes
	nºCC	VARCHAR(15)	15 bytes
	nif	VARCHAR(15)	15 bytes
	ddn	DATETIME	7 bytes
	genero	CHAR	1 bytes
	rua	VARCHAR(45)	45 bytes
	CodigoPostal_codigoPostal	VARCHAR(8)	8 bytes
	equipa	VARCHAR(45)	45 bytes
	contacto	VARCHAR(13)	13 bytes
	Modalidade_idModalidade	INT	4 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 50*292 = 14600		

Tabela 9 – Atleta

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
Medico	idMedico	INT	4 bytes
	email	VARCHAR(45)	45 bytes
	password	VARCHAR(45)	45 bytes
	nome	VARCHAR(45)	45 bytes
	nºCC	VARCHAR(15)	15 bytes
	cedula	VARCHAR(15)	15 bytes
	ddn	DATETIME	7 bytes
	genero	CHAR	1 bytes
	rua	VARCHAR(45)	45 bytes
	CodigoPostal_codigoPostal	VARCHAR(8)	8 bytes
	contacto	VARCHAR(13)	13 bytes
	Clinica_idClinica	INT	4 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 30*247 = 7410		

Tabela 10 – Medico

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
TCAgendado	idTCAgendado	INT	4 bytes
	TesteClinico_idTesteClinico	INT	4 bytes
	Data	DATETIME	7 bytes
	Atleta_idAtleta	INT	4 bytes
	Clinica_idClinica	INT	4 bytes
	Medico_idMedico	INT	4 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 100*27 = 2700		

Tabela 11 – TCAgendado

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
TCRealizado	idTCRealizado	INT	4 bytes
	TesteClinico_idTesteClinico	INT	4 bytes
	Data	DATETIME	7 bytes
	Atleta_idAtleta	INT	4 bytes
	Clinica_idClinica	INT	4 bytes
	Medico_idMedico	INT	4 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 100*27 = 2700		

Tabela 12 – TCRealizado

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
Contacto	numero	VARCHAR(13)	13 bytes
	Clinica_idClinica	INT	4 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 9*17 = 153		

Tabela 13 – Contacto

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
EquipamentoDisponivel	idEquipamentoDisponivel	INT	4 bytes
	Equipamento_idEquipamento	INT	4 bytes
	Clinica_idClinica	INT	4 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 9*12 = 108		

Tabela 14 – EquipamentoDisponivel

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
EquipamentoNecessario	TesteClinico_idTesteClinico	INT	4 bytes
	Equipamento_idEquipamento	INT	4 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 2*8 = 16		

Tabela 15 – EquipamentoNecessario

Tabela	Atributo	Tipo de Dados	Espaço Ocupado
EquipamentoUtilizado	EquipamentoDisponivel_idEquipmento Disponivel	INT	4 bytes
	TCRealizado_idTCRealizado	INT	4 bytes
TOTAL	Nº de entradas * tamanho de uma entrada = 18*8 = 144		

Tabela 16 – EquipamentoUtilizado

Tabela	Tamanho com os dados inseridos
CodigoPostal	420 bytes
Modalidade	2124 bytes
TesteClinico	605 bytes
Equipmento	98 bytes
Clinica	306 bytes
Atleta	14600 bytes
Medico	7410 bytes
TCAgendado	2700 bytes
TCRealizado	2700 bytes
Contacto	153 bytes
EquipmentoDisponivel	108 bytes
EquipmentoNecessario	16 bytes
EquipmentoUtilizado	144 bytes
TOTAL	31384 bytes

Tabela 17 - Tamanho inicial da base de dados

5.6.2 Estimativa do crescimento anual

Idealizando um cenário hipotético, retiramos algumas conclusões sobre como poderá ser o crescimento na totalidade das clínicas ao fim de um ano, por forma a prever também o quanto a base de dados poderia vir a crescer. Como tal, consideramos que:

- Médicos e atletas de um total de 20 localidades novas registaram-se no sistema, levando a um aumento de 15 para 35 códigos postais registados:

$$35 * 28 \text{ bytes} = 980 \text{ bytes}$$

- Devido à já enorme diversidade de atletas de diversas modalidades atendidos nas clínicas, apenas apareceram atletas de 14 modalidades novas, perfazendo agora um total de 50 modalidades diferentes:

$$50 * 59 \text{ bytes} = 2950 \text{ bytes}$$

- Uma vez que a procura pelas clínicas aumentou, os proprietários decidiram avançar com a criação de novos testes, dispondo agora as clínicas de 15 testes distintos, passíveis de ser realizados:

$$15 * 55 \text{ bytes} = 825 \text{ bytes}$$

- Um dos novos testes inseridos necessitava de um equipamento específico para ser realizado, o que acaba por gerar uma nova entrada na tabela dos equipamentos, aumentando de 2 para 3:

$$3 * 49 \text{ bytes} = 147 \text{ bytes}$$

- As clínicas foram um verdadeiro sucesso nos 3 primeiros trimestres do ano, levando a que os proprietários criassem uma nova, desta feita em Coimbra! A clínica foi construída durante o último trimestre do ano, e estará operacional a partir do primeiro dia do próximo ano, havendo agora um total de 4 clínicas disponíveis:

$$4 * 102 \text{ bytes} = 408 \text{ bytes}$$

- A criação das clínicas do Porto e de Lisboa revelou-se como um investimento muito bom, aumentando o número de atletas clientes de 50 para 200:

$$200 * 292 \text{ bytes} = 58400 \text{ bytes}$$

- Devido ao aumento de clientes, e à criação de uma nova clínica, os proprietários acharam por bem雇用 25 novos médicos (5 para cada uma das clínicas existentes, e 10 para a nova clínica):

$$55 * 247 \text{ bytes} = 13585 \text{ bytes}$$

- A procura pelas clínicas é grande, levando a que haja um enorme agendamento de testes. As estimativas apontam para cerca de 700 testes clínicos agendados, havendo, inclusive, agendamentos de testes que só se irão realizar daqui a 1/2 anos!

$$700 * 27 \text{ bytes} = 18900 \text{ bytes}$$

- Devido à enorme procura que as clínicas tiveram, o número de testes realizados guardados na base de dados aumentou significativamente. Para além dos 100 já registados, foram acrescentados 400 novos testes, totalizando, assim, 500 testes já realizados:

$$500 * 27 \text{ bytes} = 13500 \text{ bytes}$$

- Uma vez que vai ser criada uma nova clínica, 3 novos contactos terão de ser adicionados à base de dados:

$$12 * 17 \text{ bytes} = 204 \text{ bytes}$$

- Por forma a continuar a prestar os melhores serviços, e uma vez que houve a introdução de testes novos, os proprietários decidiram aumentar o número de equipamentos disponíveis em cada uma das clínicas, passando assim de 9 para 15 (não esquecer que também houve a criação de uma nova clínica):

$$15 * 12 \text{ bytes} = 180 \text{ bytes}$$

- Tal como já foi referido, houve a inserção de um novo exame, que necessita de um equipamento específico. Assim sendo, o número total de equipamentos necessários subiu de 2 para 3:

$$3 * 8 \text{ bytes} = 24 \text{ bytes}$$

- Devido ao aumento do número de exames realizados, houve também um aumento do número de vezes que os equipamentos são utilizados. Como tal, observa-se agora um total de 75 exames realizados com o auxílio de um equipamento específico:

$$75 * 8 \text{ bytes} = 600 \text{ bytes}$$

Deste modo, podemos estimar o valor total do tamanho da base de dados ao fim de um ano, usando os valores acima:

Tabela	Tamanho com os dados inseridos
CodigoPostal	980 bytes
Modalidade	2950 bytes
TesteClinico	825 bytes
Equipamento	147 bytes
Clinica	408 bytes
Atleta	58400 bytes
Medico	13585 bytes
TCAgendado	18900 bytes
TCRealizado	13500 bytes
Contacto	204 bytes
EquipamentoDisponivel	180 bytes
EquipamentoNecessario	24 bytes
EquipamentoUtilizado	600 bytes
TOTAL	110703 bytes

Tabela 18 - Tamanho da base de dados ao fim de um ano

5.7. Definição e caracterização das vistas de utilização em SQL

Ao desenvolver um SGBD é importante ter em conta os acessos e as ações que cada utilizador pode ter, ou seja, todas as suas vistas de utilização.

Estas podem ser definidas como uma tabela proveniente da consulta de outras tabelas. As vistas de utilização são por vezes designadas também como tabelas virtuais, permitindo a simplificação de visualização e evitar a escrita constante de consultas frequentes, podendo também esconder certos detalhes que não devem ser mostrados ao utilizador.

Como tal, consideramos a definição de algumas vistas do nosso modelo físico:

- **Tabela que mostra todos os testes clínicos disponíveis**

```
CREATE VIEW Testes_Disponiveis AS
SELECT idTesteClinico, designacao, preco FROM TesteClinico;
```

Figura 23 - Código SQL que mostra o conteúdo de TesteClinico

- **Tabela que mostra as informações sobre todos os atletas**

```
CREATE VIEW Atleta_detalhado AS
SELECT A.email AS Email, A.nome AS Nome, A.nºCC AS CC, A.nif AS Nif, A.ddn AS DataNascimento,
       A.genero AS Género, A.rua AS Rua, A.CodigoPostal_codigoPostal AS CódigoPostal,
       A.equipa AS Equipa, A.contacto AS Contacto, M.nome AS Modalidade
FROM Atleta AS A
INNER JOIN Modalidade AS M
ON A.Modalidade_idModalidade = M.idModalidade
ORDER BY A.nome
```

Figura 24 - Código SQL que mostra o conteúdo de Atleta

- **Tabela que mostra as informações sobre todos os médicos**

```
CREATE VIEW Medico_detalhado AS
SELECT M.email AS Email, M.nome AS Nome, M.nºCC AS CC, M.cedula AS Cédula,
       M.ddn AS DataNascimento, M.genero AS Género, M.rua AS Rua,
       M.CodigoPostal_codigoPostal AS CódigoPostal, M.contacto AS Contacto, C.nome AS Clínica
FROM Medico AS M
INNER JOIN Clinica AS C
ON M.Clinica_idClinica = C.idClinica
ORDER BY M.nome
```

Figura 25 - Código SQL que mostra o conteúdo de Medico

- Tabela que mostra as informações sobre todos os testes clínicos agendados

```
CREATE VIEW TCA_Detalhado AS
    SELECT T.designacao, TCA.data, A.nome AS nomeAtleta, C.nome AS nomeClinica, M.nome AS nomeMedico
    FROM TesteClinico AS T
        INNER JOIN TCAgendado AS TCA
        ON T.idTesteClinico = TCA.TesteClinico_idTesteClinico
            INNER JOIN Atleta AS A
            ON TCA.Atleta_idAtleta = A.idAtleta
            INNER JOIN Clinica AS C
            ON TCA.Clinica_idClinica = C.idClinica
            INNER JOIN Medico AS M
            ON TCA.Medico_idMedico = M.idMedico
    ORDER BY TCA.data
```

Figura 26 - Código SQL que mostra o conteúdo de TCAgendado

- Tabela que mostra as informações sobre todos os testes clínicos realizados

```
CREATE VIEW TCR_Detalhado AS
    SELECT T.designacao, TCR.data, A.nome AS nomeAtleta, C.nome AS nomeClinica, M.nome AS nomeMedico
    FROM TesteClinico AS T
        INNER JOIN TCRealizado AS TCR
        ON T.idTesteClinico = TCR.TesteClinico_idTesteClinico
            INNER JOIN Atleta AS A
            ON TCR.Atleta_idAtleta = A.idAtleta
            INNER JOIN Clinica AS C
            ON TCR.Clinica_idClinica = C.idClinica
            INNER JOIN Medico AS M
            ON TCR.Medico_idMedico = M.idMedico
    ORDER BY TCR.data DESC
```

Figura 27 - Código SQL que mostra o conteúdo de TCRealizado

- Tabela que mostra quais os médicos que atenderam cada atleta

```
CREATE VIEW Medico_Atleta AS
    SELECT A.idAtleta, A.nome AS nome_Atleta, M.nome AS nome_Medico
    FROM Atleta AS A
        INNER JOIN TCRealizado AS T
        ON A.idAtleta = T.Atleta_idAtleta
            INNER JOIN Medico AS M
            ON T.Medico_idMedico = M.idMedico
    ORDER BY A.idAtleta ASC;
```

Figura 28 - Código SQL que mostra quais os médicos que atenderam cada atleta

5.8. Definição e caracterização dos mecanismos de segurança em SQL

Serve o presente capítulo para especificar os mecanismos de segurança/permisões fornecidos aos 4 tipos diferentes de utilizadores identificados no projeto: atletas, médicos, administrador e funcionário.

- O **administrador** deve ter permissão para realizar todas as ações, exceto a de apagar toda a base de dados.

```
CREATE USER 'Administrador'@'localhost'  
    identified by 'adminpassword';  
  
GRANT SELECT, INSERT, DELETE, UPDATE ON *.* TO 'Administrador'@'localhost';  
  
REVOKE DROP, CREATE ON *.* FROM 'Administrador'@'localhost';
```

Figura 29 - Código SQL com as permissões do administrador

- O **atleta** deverá poder alterar qualquer um dos seus dados pessoais, não podendo, no entanto, remover dados específicos. Apenas poderá consultar a tabela dos testes clínicos disponíveis, e dos testes por ele agendados/realizados (permissão não implementada).

```
CREATE USER 'atleta'@'localhost'  
    identified BY 'atletapassword';  
  
GRANT SELECT ON mieicare.TesteClinico to 'atleta'@'localhost';  
GRANT SELECT, DELETE, UPDATE  
ON mieicare.Atleta[idAtleta_atleta, email_atleta, password_atleta, nome_atleta, n°CC_atleta,  
| | | | | nif_atleta, ddn_atleta, genero_atleta, rua_atleta, CódigoPostal_idCódigoPostal_atleta,  
equipa_atleta, contacto_atleta, Modalidade_idModalidade_atleta]  
TO 'atleta'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON mieicare.Atleta to 'atleta'@'localhost';  
  
REVOKE DROP, CREATE, DELETE, UPDATE, INSERT  
ON mieicare.TestesClinicos  
FROM 'atleta'@'localhost';  
  
REVOKE DROP, CREATE  
ON mieicare.Atleta  
FROM 'atleta'@'localhost';
```

Figura 30 - Código SQL com as permissões do atleta

- O **médico** deverá poder alterar qualquer um dos seus dados pessoais, não podendo, no entanto, remover dados específicos. Apenas poderá consultar a tabela dos testes clínicos disponíveis, e dos testes agendados/realizados a ele associados (permissão não implementada).

```

CREATE USER 'medico'@'localhost'
  identified BY 'medicopassword';

GRANT SELECT ON mieicare.TesteClinico to 'medico'@'localhost';
GRANT SELECT, DELETE, UPDATE
ON mieicare.Medico[idMedico_medico, email_medico, password_medico, nome_medico, nºCC_medico,
                   cedula_medico, ddn_medico, genero_medico, rua_medico,
                   CodigoPostal_idCodigoPostal_medico, contacto_medico, Clinica_idClinica_medico]
TO 'medico'@'localhost';
GRANT SELECT, INSERT, UPDATE ON mieicare.Medico to 'medico'@'localhost';

REVOKE DROP, CREATE, DELETE, UPDATE, INSERT
ON mieicare.TestesClinicos
FROM 'medico'@'localhost';

REVOKE DROP, CREATE
ON mieicare.Medico
FROM 'medico'@'localhost';

```

Figura 31 - Código SQL com as permissões do medico

- O funcionário terá acesso à base de dados através da conta da clínica. Portanto, esta deverá ter acesso a todos os dados passíveis de consulta (lista de testes clínicos existentes, lista de atletas clientes, lista de pessoal médico e lista de todos os testes agendados/realizados). No entanto, não terá acesso a informações mais sensíveis, como passwords, por exemplo.

```

CREATE USER 'clinica'@'localhost'
    identified BY 'clinicapassword';

GRANT SELECT ON mieicare.TesteClinico to 'clinica'@'localhost';
GRANT SELECT ON mieicare.Atleta to 'clinica'@'localhost';
GRANT SELECT ON mieicare.Medico to 'clinica'@'localhost';
GRANT SELECT ON mieicare.TCAgendado to 'clinica'@'localhost';
GRANT SELECT ON mieicare.TCRealizado to 'clinica'@'localhost';

REVOKE DROP, CREATE, DELETE, UPDATE, INSERT
ON mieicare.TestesClinicos
FROM 'clinica'@'localhost';

REVOKE DROP, CREATE, DELETE, UPDATE, INSERT
ON mieicare.Atleta
FROM 'clinica'@'localhost';

REVOKE DROP, CREATE, DELETE, UPDATE, INSERT
ON mieicare.Medico
FROM 'clinica'@'localhost';

REVOKE DROP, CREATE, DELETE, UPDATE, INSERT
ON mieicare.TCAgendado
FROM 'clinica'@'localhost';

REVOKE DROP, CREATE, DELETE, UPDATE, INSERT
ON mieicare.TCRealizado
FROM 'clinica'@'localhost';

```

Figura 32 - Código SQL com as permissões do funcionário

5.9. Revisão do sistema implementado com o utilizador

Por forma a dar com terminada esta fase do projeto, o modelo deverá novamente ser observado na perspetiva do utilizador. Este processo é fundamental, uma vez que é o utilizador que deve reconhecer no modelo físico apresentado a retratação da situação real das clínicas que estão a ser modeladas. Como tal, foi necessário avaliar os dados, por forma a traduzir as relações base definidas previamente no modelo lógico, de maneira a que estas e as suas restrições sejam suportadas pelo SGBD. Deste modo, foi necessária a descrição das relações base e o desenho das restrições gerais.

À *posteriori* foram também criados índices, com o intuito de tornar mais eficiente e rápido o acesso aos dados.

Outro aspecto também relevante foi a análise e criação de vistas de utilização, ou tabelas virtuais. Como podemos observar nos exemplos acima dados, estas são na verdade

nada mais nada menos do que consultas pré-definidas, elaboradas com o propósito de facilitar a análise dos dados.

Por último, foram analisadas as permissões de cada utilizador, por forma a proporcionar uma melhor segurança a todos os intervenientes nas clínicas.

Deste modo, podemos afirmar que o nosso modelo é aprovado pelos clientes, uma vez que é tido em conta como uma solução que responde a todas as necessidades pretendidas.

6. Migração para Base de Dados não Relacional

6.1. Introdução

No mercado atual, torna-se cada vez mais comum a criação de bases de dados não relacionais, maioritariamente devido ao facto de que estas resolvem o problema da escalabilidade das bases de dados ditas tradicionais. Outro motivo, é a grande quantidade de dados associados a diversas áreas de negócio, e todos os problemas que daí advêm. Estes novos modelos de modelação de bases de dados permitem, para além de uma fácil manipulação dos dados, o armazenamento dos mesmos de forma sustentável.

No projeto por nós desenvolvido, a ferramenta utilizada para implementar a nossa base de dados não relacional foi o *Neo4j*. Esta ferramenta baseia-se no uso de grafos, por forma a representar todo o esquema do problema associado. Assim sendo, podem ser observadas três componentes básicas:

- **Nodos**, representados pelos vértices do grafo;
- **Relacionamentos** entre nodos, que são as arestas do grafo;
- **Propriedades** associadas a cada nodo ou aos relacionamentos.

6.2. Bases de Dados NoSQL

Inicialmente, o modelo de base de dados maioritariamente escolhido para desenvolver aplicações era o modelo de bases de dados relacionais, utilizando ferramentas como *MySQL*, *DB2*, *SQL Server* e *Oracle*. Mais recentemente, outros modelos começaram a ser adotados, tendo cada vez mais um uso significativo. Por forma a diferenciar esses novos modelos, adotou-se o termo *NoSQL*.

NOSQL (originalmente referindo-se a "no SQL": "não SQL" ou "não relacional", posteriormente estendido para *Not Only SQL* - Não Apenas SQL) é um termo genérico que representa bases de dados não relacionais, sendo estas *open source* e completamente distintas do modelo relacional tradicional, isto porque as bases de dados *NoSQL* foram justamente projetadas a partir de necessidades que as bases de dados tradicionais relacionais não satisfaziam, como alta performance e capacidade de expansão.

Deste modo, podemos dividir os modelos de bases de dados *NoSQL* em quatro tipos:

- **Colunas:** todos os dados são armazenados em linhas particulares de uma tabela no disco.
Ex: *Cassandra, Hibase*;
- **Chave-Valor:** os dados são guardados num padrão chave-valor, algo similar a tabelas de hash.
Ex: *MemcacheD, Riak, REDIS*;
- **Documentos:** armazena os seus dados como documentos, sendo cada um deles uma coleção de pares chave-valor, permitindo a realização de consultas mais elaboradas, através do uso de filtros por atributos e a possibilidade de uso de índices.
Ex: *MongoDB, CouchDB*;
- **Grafos:** modelos baseados na teoria dos grafos, estes armazenam os seus dados em forma de grafo, tirando partido da distribuição dos vértices e arestas.
Ex: *Neo4j, Sesame*.

6.2.1 Principais Características

- **Alta performance e escalabilidade horizontal**

Opondo-se aos servidores de base de dados tradicionais, em que é necessário um número maior de recursos, como disco e memória, por forma a armazenar uma maior quantidade de dados, as bases de dados *NoSQL* redistribuem-se horizontalmente, funcionando como um sistema fracamente acoplado, sendo portanto mais económicas e escaláveis. Devido a isto, a sua performance não advém de um servidor central com elevado poder computacional, mas sim de vários servidores conectados e trabalhando em conjunto.

- **Replicação**

Estratégia que encaixa perfeitamente na arquitetura do *NoSQL*, uma vez que esta segue também o conceito de sistema fracamente acoplado. Essa ferramenta capacita o suporte de dados e backups, independentemente do local físico onde estes se encontram armazenados.

- **Raízes Open Source**

Tal como já fora referido, diversos modelos de *NoSQL* têm raízes na comunidade *open source*. Como tal, podemos afirmar que o seu crescimento acelerado provém dessas origens, acabando por influenciar também à criação

de companhias que oferecem versões comerciais das bases de dados em *NoSQL*, capacitando-as de uma forte estrutura de suporte e serviços.

- **Baixo custo operacional**

Uma vez que estes modelos são originários maioritariamente do *open source*, o custo para iniciar o uso dos mesmos é baixíssimo. Para além disto, uma vez que o *NoSQL* foi desenvolvido para trabalhar em ambientes distribuídos, diminui-se também as despesas na aquisição de computadores topo de gama, necessários para correr bases de dados relacionais.

- **Ausência de esquema ou esquema flexível**

Uma das principais características de uma base de dados *NoSQL* é a ausência parcial ou completa de um esquema que defina a estrutura de dados. Devido a isto, a aplicação da escalabilidade torna-se mais fácil, permitindo também um aumento da disponibilidade. A API também se torna simples, por forma a facilitar o acesso aos dados.

6.2.2 SQL vs NoSQL

- **Linguagem**

As bases de dados *SQL* são estruturadas em torno de um linguagem de consulta, o *SQL*, usando para a definição e manipulação dos dados. Se por um lado, isso torna os modelos relacionais mais capazes, uma vez que o *SQL* é uma escolha segura e especialmente ótima para consultas mais complexas; por outro, o *SQL* pode ser restritivo, uma vez que exige o uso de esquemas lógicos pré-definidos por forma a determinar a estrutura dos dados antes de trabalhar com eles.

- **Escalabilidade**

Em grande parte das situações, as bases de dados relacionais são verticalmente escaláveis, podendo aumentar a capacidade de carregamento do servidor recorrendo ao melhoramento de coisas com *RAM*, *SSD* e *CPU*.

No entanto, sendo as bases de dados não relacionais horizontalmente escaláveis, podem suportar muito mais tráfego por *sharding* (particionamento de dados), ou seja, através da adição de mais servidores na sua base de dados.

Por norma, o segundo caso é mais poderoso, tornando as bases de dados *NoSQL* a escolha de preferência para conjuntos de dados maiores ou em mudança constante.

- **Estrutura**

Enquanto as bases de dados SQL se baseiam em tabelas, as bases de dados *NoSQL* podem ser baseadas em colunas, pares de chave-valor, documentos ou grafos. Como tal, as últimas definem-se como melhores opções para aplicações que requerem transações que retornam várias colunas, como por exemplo, um sistema de contabilidade.

- **Propriedades ACID**

A maior parte das bases de dados relacionais são compatíveis com o modelo ACID (acrônimo de Atomicidade, Consistência, Isolamento e Durabilidade), opondo-se a algumas bases de dados não relacionais, uma vez que, neste caso, a sua compatibilidade ou não varia com as tecnologias adotadas (muitos modelos *NoSQL* abdicam da compatibilidade ACID em detrimento do desempenho e da escalabilidade).

6.2.3 Vantagens

- Facilidade na introdução de novos dados;
- Uma vez que não necessita da ocorrência de carregamentos, os dados encontram-se sempre disponíveis;
- Ótimo método para lidar com o problema causado pelo excesso de dados;
- Custo reduzido, comparativamente a uma base de dados relacional;
- Facilidade em introduzir novos dados;
- Performance superior na consulta tanto de grandes como de pequenas quantidades de dados.

6.2.4 Desvantagens

- Consistência dos dados presentes não garantida, havendo dependência do programador e do modo como são executadas as inserções/alterações dos dados;
- Problemas de escalabilidade de um *website* não são resolvidos;
- Sendo mais recentes que as bases de dados relacionais, não existem tantas alternativas;
- Em problemas onde o modo de estruturação dos dados é importante, uma base de dados relacional é muito mais adequada;
- Neo4J apenas possui *hash indexes*, faltando um índice de alcance (*range index*);

6.3. Migração de Dados

A migração dos dados de um modelo relacional para um modelo não relacional pode ser um ponto crítico, especialmente em casos em que a base de dados possui uma quantidade de informação superior àquela que por nós foi tratada. Estas migrações acontecem muito por causa da falta de flexibilidade dos modelos relacionais, sendo algo em que as empresas começam a apostar bastante.

Apesar de parecer ser um processo simples, a migração de dados entre sistemas de bases de dados possui algumas nuances, uma vez que esta implica a transferência de informação de uma base de dados para outra. Como tal, torna-se imperativo a delinear de uma estratégia por forma a que não se percam quaisquer informações.

Posto isto, definimos que o primeiro ponto a ter em consideração seria a exportação da informação, através de um script em *SQL*, desde o sistema de base de dados onde estavam guardadas as informações até ficheiros do tipo *csv*. Depois, a informação teria de ser importada no *Neo4J*, aquando da criação dos nodos, fazendo nos poupar bastante escrita de código. Nestes passaria a estar guardada toda a informação, por forma a criar automaticamente todos os relacionamentos que devem existir. Por último, removeríamos dados de determinados nodos, aqueles que estão associados a modelos relacionais, que não fariam sentido existir neste caso.

6.3.1 Exportação da informação contida para ficheiros csv

Tal como fora referido acima, recorremos a ficheiros *csv* por forma a guardar a informação exportada. Deste modo, foram gerados 13 ficheiros, suportando cada um a informação de uma determinada tabela. A informação das tabelas fora registada na forma de *strings*, estando os diferentes atributos de uma entrada da tabela separados por vírgulas. Em cada linha do ficheiro é possível encontrar a informação correspondente a uma entrada de uma tabela.

6.3.2 Importação da informação contida em ficheiros csv

Por forma a termos a informação registada na base de dados não relacional, temos de carregá-la dos ficheiros *csv* para os nodos criados na base de dados. Desta feita, esta fase será executada aquando da criação dos nodos. De realçar que para alguns tipos de dados (por exemplo, *datetime*) não foi possível fazer uma conversão correta para o tipo de dados correspondente.

6.3.3 Criação dos nodos

Sabendo à partida que, na nova base de dados, a informação tem que ficar alojada em algum local, é necessário proceder à criação de nodos. Como tal, foram criados nodos de tantos tipos diferentes quanto o número de tabelas que possuímos, existindo assim nodos dos

tipos “Atleta”, “Medico”, “Clinica”, “TesteClinico”, “TCAgendado”, “TCRealizado”, “Equipamento”, “Modalidade”, “Contacto” e “CodigoPostal”.

Posto isto, foram criados tantos nodos de um determinado tipo, quantos os registo existentes na tabela correspondente, associando a cada nodo os valores dos atributos de cada registo, identificados agora como propriedades. Generalizando, podemos afirmar que todos os registo existentes na base de dados relacional passaram agora a ser nodos na nossa base de dados não relacional. Assim sendo, por forma a analisar os dados associados a uma dada tabela, basta selecionar os nodos do tipo afeto a essa tabela.

6.3.4 Criação dos relacionamentos

Os relacionamentos tornam muito mais natural a interpretação e consequente interação do utilizador com a base de dados. Enquanto que numa base de dados relacional teríamos de analisar as chaves contidas noutras tabelas, neste tipo de bases de dados essa representação pode ser efetuada visualmente, através de uma ligação, acompanhada de um nome e das suas propriedades.

Os relacionamentos que no modelo relacional eram 1 para n ou 1 para 1, obrigavam a que existisse uma chave estrangeira numa das tabelas. No caso dos primeiros três relacionamentos referidos, usamos a chave estrangeira que existia em determinados nodos para fazer a correta associação. Por exemplo, na anterior base de dados, um “Medico” tinha a chave estrangeira de uma “Clinica” e na atual sabemos que o valor dessa chave estrangeira ainda existe em todos os nodos do tipo “Medico”. É esse valor que vamos usar para associar uma “Clinica” aos seus “Medicos”, tal como nos outros casos.

Os relacionamentos que no modelo relacional eram de n para m, obrigavam à existência de uma tabela. Para criar estes relacionamentos, ligamos os nodos de acordo com as chaves existentes nos nodos que correspondem a essas tabelas. Como por exemplo, no relacionamento entre “TesteClinico” e “Equipamento”, na antiga base de dados existia uma tabela “EquipamentoNecessario”. Neste momento, existe um tipo de nodos com esse nome, que tem as respetivas propriedades. Ora, o que deve ser feito, é a criação de um relacionamento entre os nodos “TesteClinico” e “Equipamento” por cada nodo “EquipamentoNecessario”, desde que este último contenha em simultâneo os identificadores de um “Equipamento” e de um “TesteClinico”. A cada um destes últimos relacionamentos é também associada uma propriedade que existia no modelo relacional.

6.3.5 Remoção de dados redundantes

Após todo este processo, torna-se fácil observar que existe diversa informação redundante. Esta era necessária no esquema de base de dados relacional de onde provém, no entanto, agora torna-se obsoleta, uma vez que para referirmos a informação associada a vários nodos podemos simplesmente referir os nodos e o correspondente relacionamento.

Deste modo, deixa de ser preciso a existência de qualquer tipo de chave estrangeira, tornando as propriedades dos nodos correspondentes a estas, informação a mais. Para além disto, os nodos do tipo “EquipamentoUtilizado”, “EquipamentoDisponivel” e “EquipamentoNecessario” são, na verdade, inúteis à base de dados.

Assim sendo, optámos por eliminar todas as propriedades que correspondiam a chaves estrangeiras e todos os nodos que correspondiam a tabelas resultantes de relacionamentos n para n.

6.4. Grafo Resultante

6.4.1 Imagem do grafo resultante

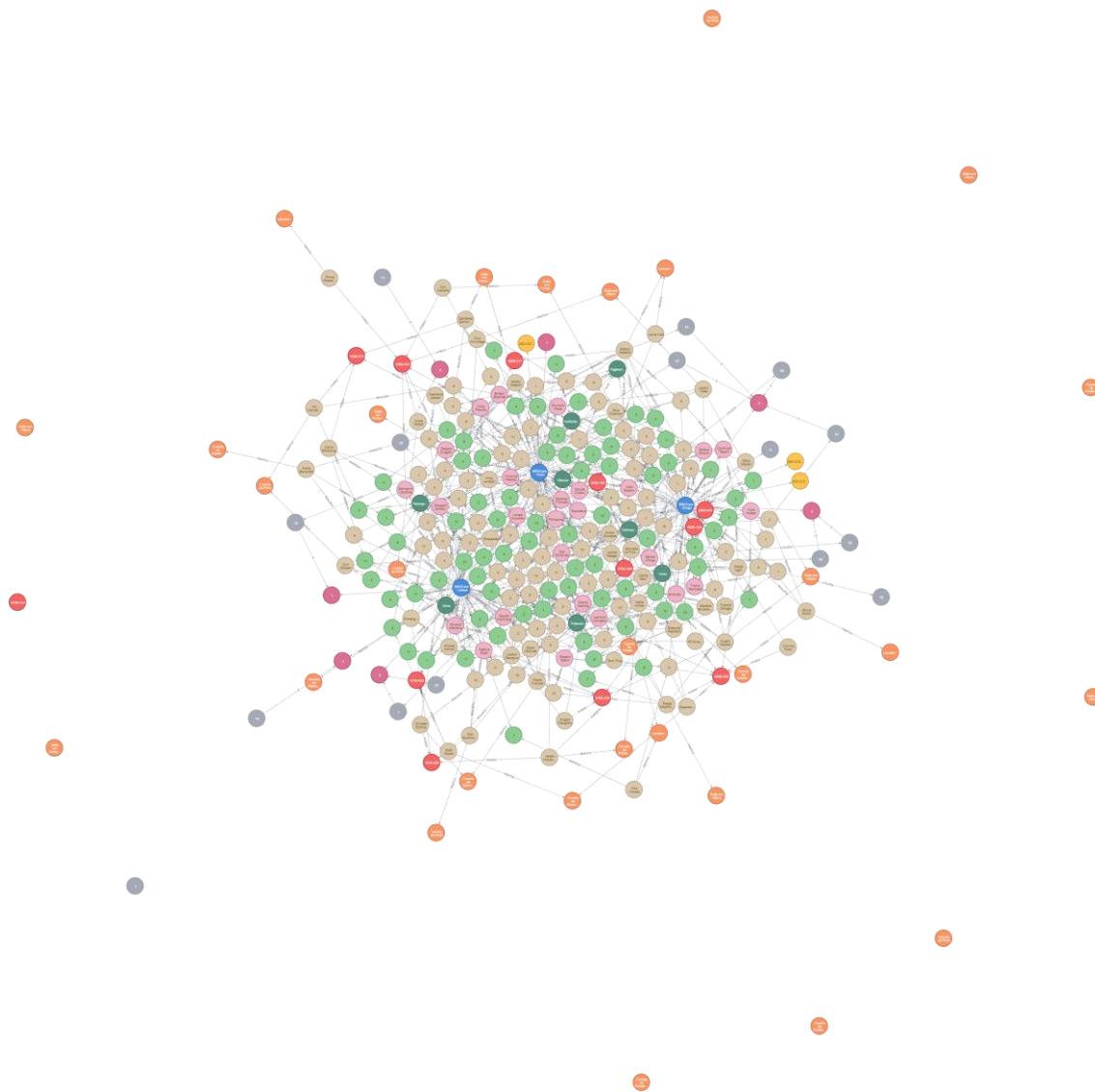


Figura 33 - Grafo que representa a base de dados com todos os seus elementos

6.4.2 Imagem do grafo resultante sem os nodos não relacionados

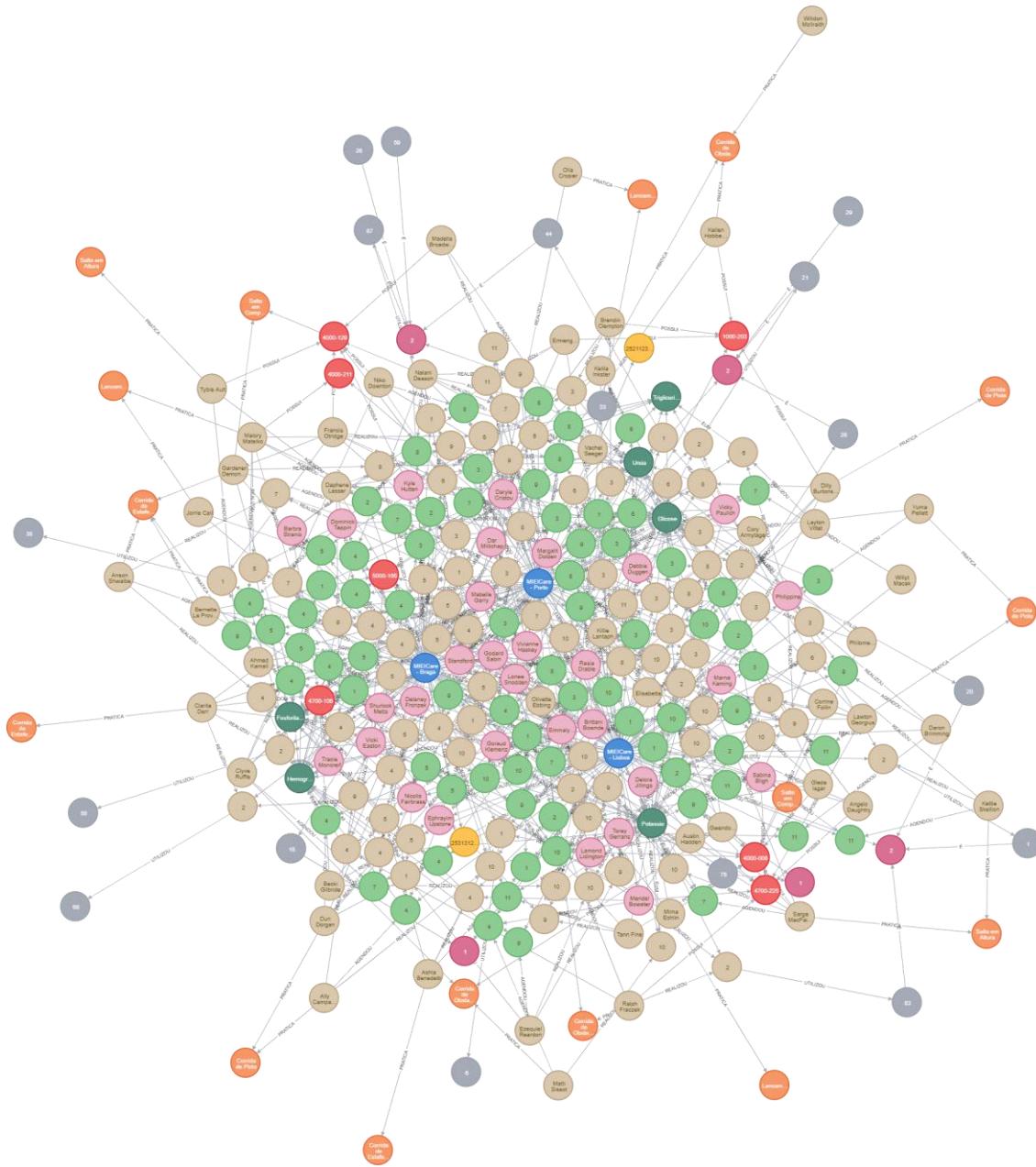


Figura 34 - Grafo que representa a base de dados apenas com os elementos relacionados

6.5. Tradução das interrogações do utilizador para Cypher

- **1^a Querie**

Consulta da disponibilidade do médico com o id nº15, as 16h30m do dia 25 de julho de 2020.

```
1 MATCH (m:Medico)-[:AGENDOU]->(tca:TCAgendado)
2 WITH apoc.date.parse(tca.data, 's', 'yyyy-MM-dd HH:mm:ss') AS date, apoc.date.parse('2020-07-25 16:00:00', 's', 'yyyy-MM-dd HH:mm:ss') AS dateI,
apoc.date.parse('2020-07-25 17:00:00', 's', 'yyyy-MM-dd HH:mm:ss') AS dateF, tca.idTCAgendado AS disp
3 WHERE date>dateI AND date<dateF
4 RETURN COUNT(disp) AS Disponibilidade
```

Figura 35 - 1^a querie em Cypher

- **2^a Querie**

Consulta dos três exames mais realizados no mês de outubro de 2016.

```
1 MATCH (tcr:TCRealizado)-[:ÉUM]->(tc:TesteClinico)
2 WITH apoc.date.parse(tcr.data, 's', 'yyyy-MM-dd HH:mm:ss') AS date, apoc.date.parse('2016-10-01 00:00:00', 's', 'yyyy-MM-dd HH:mm:ss') AS dateI,
apoc.date.parse('2016-10-31 23:59:59', 's', 'yyyy-MM-dd HH:mm:ss') AS dateF, tc.designacao AS Teste
3 WHERE date>dateI AND date<dateF
4 RETURN Teste, COUNT(Teste) AS NúmeroTestes
5 ORDER BY COUNT(Teste) DESC
```

Figura 36 - 2^a querie em Cypher

- **3^a Querie**

Consulta do atleta mais velho ou mais novo.

```
1 MATCH (a:Atleta)
2 WITH apoc.date.parse(a.ddn, 's', 'yyyy-MM-dd HH:mm:ss') AS Date, a.nome AS Nome, a.ddn AS DataNascimento
3 RETURN Nome, DataNascimento
4 ORDER BY Date
5 limit 1
```

Figura 37 - 3^a querie em Cypher (atleta mais velho)

```
1 MATCH (a:Atleta)
2 WITH apoc.date.parse(a.ddn, 's', 'yyyy-MM-dd HH:mm:ss') AS Date, a.nome AS Nome, a.ddn AS DataNascimento
3 RETURN Nome, DataNascimento
4 ORDER BY Date DESC
5 limit 1
```

Figura 38 - 3^a querie em Cypher (atleta mais novo)

- **4^a Querie**

Consulta de todos os atletas que realizaram Eletrocardiogramas.

```
1 MATCH (a:Atleta)-[:REALIZOU]->(tcr:TCRealizado)-[:ÉUM]->(tc:TesteClinico)
2 WITH a.nome AS Nome
3 WHERE tc.designacao = 'Eletrocardiograma'
4 RETURN Nome, COUNT(Nome) AS Testes
5 ORDER BY Nome
```

Figura 39 - 4^a querie em Cypher

- **5^a Querie**

Consulta do número de testes realizado por cada médico.

```
1 MATCH (m:Medico)-[:REALIZOU]->(:TCRealizado)
2 WITH m.nome AS Nome
3 RETURN Nome, COUNT(Nome) AS Testes
4 ORDER BY Testes DESC
```

Figura 40 - 5^a querie em Cypher

- **6^a Querie**

Consulta de todos os médicos com idades compreendidas entre os 45 e os 60 anos que realizaram exames à urina.

```
1 MATCH (m:Medico)-[:REALIZOU]->(tcr:TCRealizado)-[:ÉUM]->(tc:TesteClinico)
2 WITH m.nome AS Nome, apoc.date.parse(m.ddn,'s', 'yyyy-MM-dd HH:mm:ss') AS date, apoc.date.parse('1960-01-01 00:00:00','s', 'yyyy-MM-dd HH:mm:ss') AS dateI,
apoc.date.parse('1975-12-31 23:59:59','s', 'yyyy-MM-dd HH:mm:ss') AS dateF
3 WHERE tc.designacao = 'Urina' AND date>dateI AND date<dateF
4 RETURN Nome, COUNT(Nome) AS Testes
5 ORDER BY Nome
```

Figura 41 - 6^a querie em Cypher

6.6. Revisão do sistema implementado com o utilizador

Tal como efetuado com a base de dados relacional, foi necessário demonstrar o resultado da implementação do projeto aos proprietários das clínicas.

Foi facilmente observável que esta era de muito mais fácil interpretação por parte dos utilizadores, sendo o mesmo referido pelos proprietários, o que motivou diversos elogios a este tipo de implementação. Foi também observável que, em caso de necessidade de implementação de novos campos informativos, tal seria muito mais facilmente executado nesta base de dados não relacional, do que na base de dados relacional previamente construída.

6.7. Análise Crítica

É possível afirmar que, com a realização desta parte do projeto, foi mais fácil compreender e analisar as diferenças entre uma base de dados relacional e uma não relacional, facilitando também a apresentação destas aos nossos clientes.

Ao contrário do que acontecera previamente, não foi necessário construir outros modelos, sendo apenas necessários adequar os já existentes. Assim, atravessamos diferentes etapas, pesquisando diversa informação que facilitou o processo de migração entre bases de dados e posterior implementação.

Desta vez, foi utilizada a linguagem *Cypher*, que permite a criação de nodos e relacionamentos, sendo, no entanto, na mesma possível resolver as mesmas queries usadas previamente no modelo relacional.

Comparando os dois modelos de bases de dados distintos sobre os quais nos debruçamos, pudemos compreender como ambas funcionam e a sua respetiva importância. Além disto, podemos afirmar que se por um lado os modelos relacionais são os mais usados, os modelos não relacionais estão a ganhar cada vez mais força. Deste modo, podemos também afirmar que o grupo de trabalho tornou-se mais capaz de interpretar problemas, conseguindo agora decidir melhor sobre qual o tipo mais indicado de base de dados a escolher para um dado projeto.

7. Conclusões e Trabalho Futuro

Concluído o projeto, podemos afirmar que a implementação e consequente elaboração de um SGBD é uma tarefa árdua, que requer diferentes cuidados ao longo da sua execução. Assim, torna-se necessário seguir diversos pontos, por forma a facilitar no seu desenvolvimento, sem nunca comprometer a integridade e correção do resultado obtido.

A análise dos requisitos com o cliente é, provavelmente, uma das etapas mais importantes de todo um projeto, uma vez que é aí que são definidas as entidades e relacionamentos do projeto, tendo em vista a otimização da gerência e o armazenamento dos dados, de modo a melhorar toda a performance do sistema. Caracterizamos a base de dados obtida como simples, mas não simplista, uma vez que pode facilmente adquirir outras áreas de funcionamento das clínicas.

Seguindo a metodologia sugerida na folha informativa disponibilizada pelo professor da unidade curricular, foi inicialmente elaborado o modelo conceptual, sendo tal realizado com o conhecimento do cliente. Posto isto, partiu-se para a elaboração do modelo lógico e respetivo esquema físico.

Por fim, chegamos à conclusão de que os objetivos foram atingidos, já que conseguimos elaborar duas bases de dados funcionais, que representam as necessidades das clínicas *MIE/Care*, tendo sido tal cumprido devido à manutenção constante da base de dados, por forma a manter ou otimizar o nível de performance da mesma.

Referências

Connolly, T., Begg, C., 2004. *Database Systems, A Practical Approach to Design, Implementation and Management*. 4^a Edição. Addison-Wesley.

Lista de Siglas e Acrónimos

BD	Base de Dados
DDN	Data de Nascimento
<i>DML</i>	<i>Data Manipulation Language</i>
<i>DW</i>	<i>Data Warehouse</i>
<i>OLTP</i>	<i>On-Line Analytical Processing</i>
SGBD	Sistema de Gestão de Bases de Dados
TC	Testes Clínicos

Anexos

I. Script de povoamento

```

1  insert into CodigoPostal (codigoPostal, cidade) values
2   ('4700-023', 'Braga'),
3   ('3000-041', 'Coimbra'),
4   ('4700-106', 'Braga'),
5   ('1000-025', 'Lisboa'),
6   ('4000-129', 'Porto'),
7   ('5000-165', 'Vila Real'),
8   ('1000-152', 'Lisboa'),
9   ('4000-088', 'Porto'),
10  ('1000-203', 'Lisboa'),
11  ('4000-052', 'Porto'),
12  ('4700-225', 'Braga'),
13  ('4000-080', 'Porto'),
14  ('1000-066', 'Evora'),
15  ('1000-074', 'Lisboa'),
16  ('4700-314', 'Braga');

17  insert into TesteClinico (idTesteClinico, designacao, preco) values
18   (1, 'Eletrocardiograma', 14.99),
19   (2, 'Eletroencefalograma', 75),
20   (3, 'Glicose', 3.95),
21   (4, 'Hemograma', 5.71),
22   (5, 'Tosse e Fissas', 26.34),
23   (6, 'Triglicerides', 2.50),
24   (7, 'Colesterol', 4.39),
25   (8, 'Ureia', 0.88),
26   (9, 'Sodio', 1.25),
27   (10, 'Potassio', 1.25),
28   (11, 'Urina', 0.97);

29  insert into Modalidade (idModalidade, nome, escalao) values
30   (1, 'Corrida de Pista', 'Infantis'),
31   (2, 'Corrida de Pista', 'Iniciados'),
32   (3, 'Corrida de Pista', 'Juvenis'),
33   (4, 'Corrida de Pista', 'Juniores'),
34   (5, 'Corrida de Pista', 'Seniores'),
35   (6, 'Corrida de Pista', 'Veteranos'),
36   (7, 'Corrida de Obstaculos', 'Infantis'),
37   (8, 'Corrida de Obstaculos', 'Juvenis'),
38   (9, 'Corrida de Obstaculos', 'Juvenis'),
39   (10, 'Corrida de Obstaculos', 'Juniores'),
40   (11, 'Corrida de Obstaculos', 'Seniores'),
41   (12, 'Corrida de Obstaculos', 'Veteranos'),
42   (13, 'Corrida de Estafetas', 'Infantis'),
43   (14, 'Corrida de Estafetas', 'Iniciados'),
44   (15, 'Corrida de Estafetas', 'Juvenis'),
45   (16, 'Corrida de Estafetas', 'Juniores'),
46   (17, 'Corrida de Estafetas', 'Seniores'),
47   (18, 'Corrida de Estafetas', 'Veteranos'),
48   (19, 'Salto em Comprimento', 'Infantis'),
49   (20, 'Salto em Comprimento', 'Iniciados'),
50   (21, 'Salto em Comprimento', 'Juvenis'),
51   (22, 'Salto em Comprimento', 'Juniores'),
52   (23, 'Salto em Comprimento', 'Seniores'),
53   (24, 'Salto em Comprimento', 'Veteranos'),
54   (25, 'Salto em Altura', 'Infantis'),
55   (26, 'Salto em Altura', 'Iniciados'),
56   (27, 'Salto em Altura', 'Juvenis'),
57   (28, 'Salto em Altura', 'Juniores'),
58   (29, 'Salto em Altura', 'Seniores'),
59   (30, 'Salto em Altura', 'Veteranos'),
60   (31, 'Lancamento', 'Infantis'),
61   (32, 'Lancamento', 'Iniciados'),
62   (33, 'Lancamento', 'Juvenis'),
63   (34, 'Lancamento', 'Juniores'),
64   (35, 'Lancamento', 'Seniores'),
65   (36, 'Lancamento', 'Veteranos');

66  insert into Equipamento (idEquipamento, nome) values
67   (1, 'Eletrocardiografo'),
68   (2, 'Eletroencefalografo');

69  insert into Clinica (idClinica, nome, rua, CodigoPostal.codigoPostal) values
70   (1, 'MIEICare - Braga', 'Antonio Jose Lisboa', '4700-225'),
71   (2, 'MIEICare - Porto', 'Manjericos', '4000-088'),
72   (3, 'MIEICare - Lisboa', 'Entrecampos', '1000-152');

73  insert into Atleta (idAtleta, email, password, nome, noCC, nif, ddn, genero, rua, CodigoPostal.codigoPostal, equipa, contacto, Modalidade.idModalidade) values
74   (1, 'phnaw96@gmail.com', 'Philomena Nollet', '00-145-2441', '19-169-5068', '1973-03-15 23:57:47', 'F', 'Anniversary', '4000-088', '', '6614714553', 22),
75   (2, 'vseeger@google.pl', 'Krzysztof Paweł Wachiel Seeger', '05-123-0713', '20-057-1650', '1974-01-15 09:55:07', 'M', 'Prinied', '5000-165', 'Benfica', '00370496156', 31),
76   (3, 'deleroy1981@outlook.pt', 'Eliza Lurdes', '01-123-0118', '64-496-1118', '00-954-1411', '1977-02-24 18:34:27', 'F', 'Toban', '1000-012', 'Benfica', '00370496156', 31),
77   (4, 'duttonshaw@house.gov', '27k07F', 'Dilly Burtonshaw', '23-151-8124', '09-208-3330', '1958-01-16 19:19:25', 'M', 'Bashford', '1000-203', '5885690166', 6),
78   (5, 'truffles44disqus.com', 'WdKtRAkk9', 'Clive Ruffle', '69-312-7776', '52-976-9090', '1967-02-10 09:31:15', 'M', 'Service', '4700-106', 'Vidigalense', '1762569927', 12),
79   (6, 'ccaslake@tinyurl.com', 'ouAWk9', 'Con Caslake', '46-436-0461', '10-655-5283', '2008-11-17 10:09:49', 'M', 'Helena', '1000-012', 'Conforlimpa', '5902400465', 21),
80   (7, 'lgeorgius@imdb.com', 'dValXlp', 'Lawton Georgius', '00-932-9806', '00-761-6599', '1951-04-14 04:58:16', 'M', 'Dakota', '4700-225', 'Conforlimpa', '1422487618', 3),
81   (8, 'oebbing@ambler.ru', '6637x6uFM', 'Olivette Ebbing', '03-902-9697', '76-705-2893', '1952-03-15 22:34:39', 'F', 'Anderson', '1000-203', 'Conforlimpa', '7189494159', 10),
82   (9, 'crossier@tipod.com', '6lJpHote', 'Olivia Crossier', '37-665-5243', '36-150-3737', '2007-03-15 09:28:06', 'F', 'Kropf', '1000-152', 'Porto', '2376601172', 31),
83   (10, 'mechlin@outlook.es', '171001x1000', 'Ivana Echlin', '38-665-0567', '00-954-1411', '1977-02-24 18:34:27', 'F', 'Eastlawn', '4000-088', 'Porto', '1817602962', 35),
84   (11, 'mellissa@outlook.com', '8Z7yD0NRE', 'Rafaela Melissa', '24-310-2113', '28-770-0211', '1982-10-19 09:17:51', 'M', 'Loring', '1000-012', 'Conforlimpa', '5902400465', 10),
85   (12, 'knightaphararchive.org', 'AN4EXL', 'Killie Lantaph', '53-193-1208', '99-649-5797', '2003-02-12 04:18:49', 'M', 'Mockingbird', '5000-165', 'Conforlimpa', '9215120099', 19),
86   (13, 'ndownton@mud.gov', '31vx3lCthn', 'Niko Downton', '45-454-0416', '53-264-4017', '1953-01-05 05:59:06', 'M', 'Dovetail', '4000-129', 'Porto', '7205965349', 21),
87   (14, 'adaughtryd@google.com.hk', 'EnWBBSuW164', 'Angela Daughtry', '11-998-9941', '27-650-2898', '1978-03-16 12:37:47', 'M', 'Southridge', '4700-225', 'Conforlimpa', '4584751978', 23),
88   (15, 'ereardon@alexa.com', '5YVsPlkY5n', 'Ezequiel Reardon', '01-873-9948', '61-655-0202', '1955-12-19 09:54:35', 'M', 'Burrows', '3000-041', 'Conforlimpa', '6574460419', 32),
89   (16, 'taulfat@instagmail.com', 'gmmYLtg!', 'Tybira Ault', '37-535-1923', '09-483-3733', '1952-04-24 15:45:25', 'F', 'Lindbergh', '4000-129', 'Porto', '6546256513', 25),
90   (17, 'kskelling@myspace.com', 'k9Fkwd', 'Kettie Skellion', '28-194-8211', '46-161-1574', '2009-03-03 11:21:28', 'F', 'Debra', '1000-074', 'Conforlimpa', '7205963683', 26),
91   (18, 'tfinne@shared.com', '0489Gsd', 'Tanni Fine', '16-923-4373', '58-150-3737', '1948-01-15 03:45:13', 'M', 'Holy Cross', '4000-088', 'Sporting', '1823652291', 22),
92   (19, 'yannickstutter@outlook.com', 'w4EBsOPQgfr', 'Yannick Stutter', '24-310-0000', '28-770-0211', '1982-10-19 09:17:51', 'M', 'Loring', '1000-012', 'Conforlimpa', '5902400465', 22),
93   (20, 'yannickstutter@outlook.com', 'w4EBsOPQgfr', 'Yuma Pellett', '81-847-5143', '14-430-0001', '1974-03-05 03:22:53', 'M', 'Rockefeller', '1000-012', 'Benfica', '0034148113', 3),
94   (21, 'ble4amazon.co.uk', 'KD1gman001aT', 'Bennette le Provost', '00-743-1068', '68-545-0675', '1959-08-22 10:00:56', 'F', 'Clyde Gallagher', '5000-165', 'Conforlimpa', '1773863094', 15),
95   (22, 'khobema@gov.uk', 'Aq5SeT111Z', 'Kellen Hobema', '82-330-3104', '33-46-8339', '1991-06-21 15:08:46', 'M', 'Namekagon', '1000-203', 'Sporting', '5015432436', 7),
96   (23, 'mbroadwood@diigo.com', 'p6wz5FC', 'Madella Broadwood', '73-639-4858', '08-468-4449', '1985-06-05 17:29:15', 'F', 'Westend', '4000-129', 'Benfica', '5808977926', 32),
97   (24, 'mcilmraith@smmonitor.com', 'xQnC9ytB0', 'Willdon McIlraith', '56-929-7383', '45-663-6595', '1984-04-16 16:00:09', 'M', 'Sycamore', '1000-074', 'Braga', '2569837235', 7),
98   (25, 'abenedettio@infoseek.co.jp', 'Gc2V4TwNin', 'Ashla Benedetti', '05-495-4274', '27-537-0795', '2007-05-21 14:23:01', 'F', 'Waubesa', '4000-052', 'Conforlimpa', '5397174760', 14),
99   (26, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
100  (27, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
101  (28, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
102  (29, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
103  (30, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
104  (31, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
105  (32, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
106  (33, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
107  (34, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
108  (35, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
109  (36, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
110  (37, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
111  (38, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
112  (39, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
113  (40, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
114  (41, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
115  (42, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
116  (43, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
117  (44, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
118  (45, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
119  (46, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
120  (47, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
121  (48, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
122  (49, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
123  (50, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
124  (51, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
125  (52, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
126  (53, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
127  (54, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
128  (55, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
129  (56, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
130  (57, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
131  (58, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
132  (59, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
133  (60, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
134  (61, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
135  (62, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
136  (63, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
137  (64, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
138  (65, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
139  (66, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
140  (67, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
141  (68, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
142  (69, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
143  (70, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
144  (71, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
145  (72, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
146  (73, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
147  (74, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
148  (75, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
149  (76, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
150  (77, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
151  (78, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
152  (79, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
153  (80, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17:36:34', 'M', 'Holand', '1000-012', 'Vidigalense', '3883408789', 34),
154  (81, 'rglassborow@zimbio.com', '40gv1Qe', 'Rorke Glassborow', '77-019-1622', '58-544-2639', '1981-08-29 17
```

```

105 ('D7', 'gdemonego@cisco.com', 'uqajlcbSwTn', 'Gardener Demonegot', '60-996-9252', '65-984-2539', '1971-03-18 16:07:58', 'M', 'Hintze', '1000-074', 'Porto', '836440124', ⑩),
106 ('28', 'mateikon@youku.com', 'Ebx73CwQ0', 'Malory Mateik', '23-241-5679', '93-955-1939', '1942-08-16 17:52:34', 'F', 'Jay', '4000-211', 'Vidigalense', '5938602015', ⑯),
107 ('29', 'fotrides@logtalkradio.com', '7yC6nHEmU', 'Francis Otridge', '79-669-0489', '86-500-4721', '2005-02-04 11:19:21', 'M', 'Upham', '4000-129', 'Conforlimpa', '9332822117', ⑯),
108 ('30', 'ashwabert@lundi.de', 'xJ6cmU', 'Anson Shalbe', '85-116-4996', '47-005-8794', '2007-07-04 05:43:16', 'M', 'Johnson', '4000-052', 'Porto', '2446379963', ⑯),
109 ('31', 'amacaclawether.com', 'vBhKndy8t', 'Willyt Macak', '87-073-2811', '06-894-8841', '1961-02-16 21:45:01', 'F', 'Vidon', '3000-041', 'Sporting', '989879392', ⑯),
110 ('32', 'econvertv@discoverycorp.com', 'QmQdR00n', 'Lorraine Gengrande Cottier', '17-149-2281', '05-952-0296', '1996-11-26 10:12:20', 'F', 'Porto', '3466308002', '2027021469', ⑧),
111 ('33', 'tspaper@outlook.com', 'FFaIYuZu0n', 'Brenda Clemens', '55-120-2258', '72-493-4259', '1954-04-25 16:51:54', 'M', 'Straubel', '1000-363', 'Porto', '3466308002', '32),
112 ('34', 'brimming@arcopcollective.com', 'sqy3dWVlgh', 'Daron Brimming', '71-880-1734', '61-846-2122', '1978-03-30 23:13:56', 'M', 'Goodland', '1000-074', 'Conforlimpa', '8999907538', ㉑),
113 ('35', 'smacquaisey@o2.net', 'ubxvAFAHxtu', 'Sarge MacPake', '23-297-5532', '21-157-3170', '1983-09-30 00:34:24', 'M', 'Meadow Valley', '4000-088', 'Benfica', '1324827854', ㉙),
114 ('36', 'bgliblride@imesonline.co.uk', 'YKGSwpl7ym2', 'Becky Liblride', '23-832-7624', '23-390-6377', '1953-09-30 15:06:56', 'F', 'Grover', '4700-106', 'Sporting', '4828306666', ㉕),
115 ('37', 'skamall1@army.mil', 'BUBXvOHC', 'Ahmed Kamall', '53-759-4954', '10-028-0249', '1992-06-05 17:07:03', 'M', 'Anhalt', '4000-052', '8388380254', ⑯),
116 ('38', 'ahadden1@oacit.gov.au', 'adzbif1f', 'Austin Hadden', '44-247-3044', '71-196-2450', '2000-01-04 01:30:25', 'M', 'Anhalt', '4700-225', 'Porto', '19933726283', ㉙),
117 ('39', 'jcall12@baidu.com', 'uA0Q1Ykgut', 'Jorrie Call', '04-046-7945', '56-436-8579', '1996-06-06 09:32:28', 'M', 'Westridge', '5000-165', 'Weingart', '9728265732', ㉓),
118 ('40', 'g1g1g14@4igc.com', 'PF2zGZ0mo', 'Glen Gart', '99-261-8971', '08-264-6915', '1987-08-25 18:51:45', 'F', 'Ilene', '4000-088', 'Briggs', '5526421427', ⑯),
119 ('41', 'g1g1g14@4igc.com', 'PF2zGZ0mo', 'Glen Gart', '99-261-8971', '08-264-6915', '1987-08-25 18:51:45', 'F', 'Ilene', '4000-088', 'Briggs', '5526421427', ⑯),
120 ('42', 'kfallini@sakurajp.jp', 'JZn4RH1', 'Corinne Fallini', '11-840-3961', '93-386-0828', '1946-04-14 21:32:46', 'F', 'Dovetail', '3000-041', 'Benfica', '5871987393', ⑦),
121 ('43', 'kfallini@sakurajp.jp', 'JZn4RH1', 'Corinne Fallini', '11-840-3961', '93-386-0828', '1946-04-14 21:32:46', 'F', 'Sunnydays', '4000-088', 'Benfica', '8854710776', ⑪),
122 ('44', 'acampagne@furnl.net', '6uUdz1S6', 'Ally Campanage', '49-413-0846', '80-069-9046', '1972-09-25 16:13:24', 'F', 'Mesta', '1000-074', 'Sporting', '5276696738', ⑯),
123 ('45', 'ndesason1@storiify.com', 'bdYdcL3lpM0', 'Nalani Deason', '05-963-2386', '58-276-8679', '1986-11-26 04:43:50', 'F', 'Holberg', '4000-052', 'Sporting', '3701210906', ㉖),
124 ('46', 'mississ19@gmpg.org', 'uDLFCMPf6psv1', 'Maiti Sissot', '00-894-6003', '61-440-7033', '1966-02-24 11:21:37', 'F', 'Braga', '1954114502', ⑫),
125 ('47', 'greedick1@slidshare.com', 'TH3n1cKc', 'Taylor Vinter', '75-394-8929', '66-280-7332', '1958-08-04 06:45:20', 'F', 'Waywood', '3000-041', 'Benfica', '5985944271', ⑯),
126 ('48', 'udorgan1@gmail.com', 'W3Begp0Hng', 'Dun Dorgan', '67-060-4767', '1990-05-20 03:23:20', 'M', 'Amoth', '1000-152', 'Benfica', '4627448317', ⑯),
127 ('49', 'camaytagel@moonfruit.com', 'H4zlvfurov', 'Cory Armytagel', '18-157-8869', '1979-05-01 15:45:22', 'M', 'Mendota', '1000-012', 'Vidigalense', '9822748229', ⑯),
128 ('50', 'shadenn1@oacit.gov.au', 'adzbif1f', 'Austin Hadden', '44-247-3044', '71-196-2450', '2000-01-04 01:30:25', 'M', 'Anhalt', '4700-225', 'Porto', '19933726283', ㉙),
129 ('51', 'dtappin@ucoz.com', 'oahnMh0j0r', 'Dominick Tappin', '41-444-7331', '00-032-9629', '1947-12-13 22:21:00', 'M', 'Scott', '3000-041', '9531185300', ⑯),
130 ('52', 'pbawood@homestead.com', '61bU7Bn5Zf', 'Nicolis Fairbrass', '10-394-8376', '99-001-5959', '1992-05-11 11:32:16', 'F', 'Sauthoff', '4000-052', 'Benfica', '169144396', ⑯),
131 ('53', 'pbawood@homestead.com', '61bU7Bn5Zf', 'Nicolis Fairbrass', '10-394-8376', '99-001-5959', '1992-05-11 11:32:16', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
132 ('54', 'hasketyl@over-blog.com', 'u0KmKh27M', 'Vivianne Hasketyl', '93-326-7939', '99-001-5960', '1992-05-11 11:32:16', 'F', 'Freudenthal', '4708-104', 'Benfica', '5276698972', ㉖),
133 ('55', 'mckameys@sciencedirect.com', '0kF2zGZ0mo', 'Markus Kamey', '00-077-8877', '58-118-7419', '1973-02-27 11:11:34', 'F', 'Lund', '4000-165', 'Benfica', '5701100017154', ⑯),
134 ('56', 'mkameys@sciencedirect.com', '0kF2zGZ0mo', 'Markus Kamey', '00-077-8877', '58-118-7419', '1973-02-27 11:11:34', 'F', 'Lund', '4000-165', 'Benfica', '5701100017154', ⑯),
135 ('57', 'etichurst@comcast.net', 'i8ADWc', 'Marna Kaming', '86-500-9651', '34-286-6300', '1975-06-23 07:47:29', 'F', 'Charing Cross', '4700-225', '2502997907', ⑯),
136 ('58', 'smetts@abode.com', 'JdXkyqJxBIn', 'Shanelle Metts', '81-429-3316', '72-975-2972', '1949-01-26 10:59:08', 'M', 'Hagan', '1000-152', '4896445319', ⑯),
137 ('59', 'smetts@abode.com', 'JdXkyqJxBIn', 'Shanelle Metts', '81-429-3316', '72-975-2972', '1949-01-26 10:59:08', 'M', 'Hagan', '1000-152', '4896445319', ⑯),
138 ('60', 'etichurst@comcast.net', 'i8ADWc', 'Delaney Fronzek', '58-154-4211', '33-694-7891', '1972-10-06 07:07:29', 'F', 'Kinsman', '4708-106', 'Benfica', '1277793238', ⑯),
139 ('61', 'djillings@ucoz.com', 'ZjdabU0m', 'Delora Jillings', '42-343-2354', '54-075-2880', '1964-12-23 11:04:24', 'F', 'Kinsman', '3000-041', '646913338', ⑯),
140 ('62', 'khunter@tbt.com', 'u0KmKh27M', 'Kyle Hunter', '51-840-8000', '81-599-7000', '1992-05-11 11:20:18', 'F', 'Gate', '4000-125', 'Benfica', '501000000064', ⑯),
141 ('63', 'khunter@tbt.com', 'u0KmKh27M', 'Kyle Hunter', '51-840-8000', '81-599-7000', '1992-05-11 11:20:18', 'F', 'Gate', '4000-125', 'Benfica', '501000000064', ⑯),
142 ('64', 'gashield@willi.org', 'StuPrt', 'Godard Sabine', '99-432-0699', '36-183-6900', '1980-11-26 11:54:01', 'M', 'Hazelwood', '4700-106', 'Benfica', '5893428249', ⑯),
143 ('65', 'mbowater@google.com', 'StuPrt', 'Godard Sabine', '99-432-0699', '36-183-6900', '1980-11-26 11:54:01', 'M', 'Hazelwood', '4700-106', 'Benfica', '5893428249', ⑯),
144 ('66', 'bstranio@weibo.com', 'NSydzSC', 'Britannia Bound', '08-777-8671', '93-856-3488', '1943-02-22 07:51:55', 'F', 'Superior', '7000-066', '8459652970', ⑯),
145 ('67', 'bstranio@weibo.com', 'NSydzSC', 'Britannia Bound', '08-777-8671', '93-856-3488', '1943-02-22 07:51:55', 'F', 'Superior', '7000-066', '8459652970', ⑯),
146 ('68', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
147 ('69', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
148 ('70', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
149 ('71', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
150 ('72', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
151 ('73', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
152 ('74', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
153 ('75', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
154 ('76', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
155 ('77', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
156 ('78', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
157 ('79', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
158 ('80', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
159 ('81', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
160 ('82', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
161 ('83', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
162 ('84', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
163 ('85', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
164 ('86', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
165 ('87', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
166 ('88', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
167 ('89', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
168 ('90', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
169 ('91', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
170 ('92', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
171 ('93', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
172 ('94', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
173 ('95', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
174 ('96', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
175 ('97', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
176 ('98', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
177 ('99', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
178 ('100', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
179 ('101', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
180 ('102', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
181 ('103', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
182 ('104', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
183 ('105', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
184 ('106', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
185 ('107', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
186 ('108', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
187 ('109', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
188 ('110', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
189 ('111', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
190 ('112', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
191 ('113', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
192 ('114', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
193 ('115', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
194 ('116', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
195 ('117', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
196 ('118', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
197 ('119', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
198 ('120', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
199 ('121', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
200 ('122', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
201 ('123', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
202 ('124', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
203 ('125', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
204 ('126', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
205 ('127', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
206 ('128', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
207 ('129', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
208 ('130', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
209 ('131', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077-7571', '1974-01-17 05:10:04', 'F', 'Blaine', '3000-041', '9571984714', ⑯),
210 ('132', 'etichurst@comcast.net', 'i8ADWc', 'Emmallye Tiechurst', '18-077
```

```

207 (19, 2, '2021-09-28 21:35:01', 50, 1, 8),
208 (20, 8, '2020-05-01 09:22:53', 23, 1, 3),
209 (21, 4, '2021-03-23 08:54:04', 3, 2, 14),
210 (22, 8, '2020-02-02 19:56:30', 45, 1, 1),
211 (23, 9, '2020-02-27 17:27:33', 35, 2, 25),
212 (24, 2, '2020-03-03 10:44:43', 30, 2, 25),
213 (25, 2, '2021-08-21 01:25:33', 28, 3, 8),
214 (26, 7, '2021-03-24 08:49:26', 35, 1, 12),
215 (27, 2, '2022-10-07 16:28:35', 28, 2, 16),
216 (28, 10, '2022-05-17 14:52:01', 4, 2, 4),
217 (29, 4, '2020-10-04 20:37:59', 45, 1, 18),
218 (30, 7, '2021-09-07 20:00:14', 13, 3, 19),
219 (31, 2, '2020-03-24 10:44:23', 7, 3, 5),
220 (32, 11, '2022-02-20 23:42:30', 30, 2, 24),
221 (33, 1, '2022-06-04 05:51:48', 3, 2),
222 (34, 2, '2022-12-27 14:51:26', 47, 3, 28),
223 (35, 7, '2022-03-02 18:27:15', 31, 2, 13),
224 (36, 7, '2021-05-23 06:35:34', 10, 1, 28),
225 (37, 9, '2022-05-07 07:40:34', 43, 2, 18),
226 (38, 1, '2021-01-18 17:23:38', 48, 1, 23),
227 (39, 6, '2021-01-11 18:18:02', 50, 3, 1),
228 (40, 13, '2020-04-10 09:13:08', 11, 2, 17),
229 (41, 3, '2020-04-04 10:09:20', 26, 3, 18),
230 (42, 3, '2022-04-03 18:52:04', 26, 3, 18),
231 (43, 5, '2020-04-22 00:10:29', 10, 1, 8),
232 (44, 8, '2021-12-09 16:24:49', 1, 3, 26),
233 (45, 11, '2022-12-21 12:57:19', 18, 1, 25),
234 (46, 8, '2021-09-17 01:23:56', 49, 3, 27),
235 (47, 3, '2021-12-02 00:17:85', 20, 3, 3),
236 (48, 9, '2020-03-01 06:57:23', 34, 3, 19),
237 (49, 1, '2022-03-03 12:39:43', 1, 3, 1),
238 (50, 9, '2021-02-21 07:46:33', 12, 3, 27),
239 (51, 4, '2022-04-25 11:46:09', 33, 1, 38),
240 (52, 11, '2022-07-06 15:19:24', 39, 1, 28),
241 (53, 5, '2022-09-12 05:44:21', 30, 1, 27),
242 (54, 3, '2020-04-13 12:24:02', 47, 1, 30),
243 (55, 9, '2022-02-07 12:59:48', 42, 2, 13),
244 (56, 18, '2022-03-16 08:44:29', 18, 2, 19),
245 (57, 11, '2022-01-16 22:32:24', 49, 3, 8),
246 (58, 7, '2022-12-29 11:32:51', 43, 3, 1),
247 (59, 4, '2021-03-25 19:06:45', 46, 3, 15),
248 (60, 1, '2021-09-27 07:05:41', 25, 3, 4),
249 (61, 5, '2020-07-03 18:17:58', 5, 2, 17),
250 (62, 3, '2020-09-21 10:32:05', 16, 1, 21),
251 (63, 5, '2022-03-08 06:10:11', 5, 2, 10),
252 (64, 8, '2020-04-04 00:21:11', 29, 2, 9),
253 (65, 1, '2020-03-06 15:11:42', 1, 3, 1),
254 (66, 4, '2021-03-27 04:57:54', 49, 3, 17),
255 (67, 10, '2020-08-26 21:56:34', 38, 1, 25),
256 (68, 8, '2020-06-07 10:08:40', 45, 2, 26),
257 (69, 7, '2020-11-29 10:42:44', 37, 2, 26),
258 (70, 2, '2020-04-25 17:31:54', 41, 1, 6),
259 (71, 9, '2020-04-12 00:44:44', 16, 1, 2),
260 (72, 1, '2021-12-13 04:01:28', 1, 2, 29),
261 (73, 11, '2022-06-24 15:11:26', 49, 3, 8),
262 (74, 8, '2022-05-17 15:38:38', 12, 2, 29),
263 (75, 3, '2021-05-14 17:33:07', 28, 1, 5),
264 (76, 2, '2022-12-28 11:06:00', 28, 1, 4),
265 (77, 9, '2021-05-21 21:00:53', 45, 3, 7),
266 (78, 9, '2020-08-31 10:49:42', 41, 2, 7),
267 (79, 4, '2020-09-14 01:44:44', 46, 3, 22),
268 (80, 1, '2021-08-15 00:24:50', 49, 3, 22),
269 (81, 2, '2022-12-16 12:02:47', 41, 1, 1),
270 (82, 3, '2020-09-14 03:02:19', 43, 1, 12),
271 (83, 9, '2021-03-11 22:18:41', 36, 2, 21),
272 (84, 11, '2021-06-11 01:05:06', 13, 3, 22),
273 (85, 4, '2022-10-02 22:44:43', 27, 1, 29),
274 (86, 1, '2022-06-18 16:26:20', 7, 1, 30),
275 (87, 1, '2020-03-29 19:27:20', 14, 2, 25),
276 (88, 1, '2021-03-29 00:48:29', 14, 2, 3),
277 (89, 3, '2020-12-17 06:03:01', 25, 3, 27),
278 (90, 5, '2022-07-11 22:27:25', 36, 2, 15),
279 (91, 4, '2022-12-26 19:48:49', 19, 2, 5),
280 (92, 10, '2020-02-11 15:13:08', 12, 1, 10),
281 (93, 8, '2021-05-17 07:44:47', 13, 2, 16),
282 (94, 7, '2021-06-02 08:45:25', 27, 2, 17),
283 (95, 2, '2020-02-11 15:13:36', 12, 2, 26),
284 (96, 3, '2022-09-01 15:31:36', 10, 1, 17),
285 (97, 1, '2020-01-01 13:41:49', 14, 2, 28),
286 (98, 3, '2021-05-25 01:44:57', 37, 3, 29),
287 (99, 9, '2022-09-13 13:44:31', 36, 3, 11),
288 (100, 1, '2020-01-31 06:42:57', 5, 1, 18);
289
290
291 insert into TCRelizado (idTCRelizado, TesteClinico_idTesteClinico, data, Atleta_idAtleta, Clinica_idClinica, Medico_idMedico) values
292 (1, 1, '2010-04-23 22:17:07', 17, 3, 5),
293 (2, 4, '2019-08-20 21:54:45', 30, 3, 30),
294 (3, 8, '2016-12-08 08:02:58', 40, 3, 13),
295 (4, 3, '2016-08-11 08:53:52', 13, 1, 25),
296 (5, 3, '2016-05-23 23:23:05', 33, 2, 8),
297 (6, 1, '2016-10-10 14:15:39', 40, 1, 25),
298 (7, 8, '2019-08-22 05:04:32', 47, 2, 8),
299 (8, 5, '2016-04-12 12:59:51', 38, 3, 29),
300 (9, 10, '2017-03-31 10:53:07', 29, 3, 13),
301 (10, 1, '2017-03-30 07:07:01', 1, 1, 1),
302 (11, 10, '2018-05-30 15:12:34', 35, 3, 14),
303 (12, 6, '2016-01-10 16:43:16', 1, 2, 27),
304 (13, 9, '2018-06-20 21:06:51', 21, 2, 14),
305 (14, 9, '2017-04-02 01:39:36', 19, 2, 24),
306 (15, 6, '2015-05-04 09:11:21', 17, 3, 28),
307 (16, 5, '2019-01-15 21:53:41', 7, 1, 28),
308 (17, 18, '2019-07-26 22:24:25', 23, 2, 2),
309

```

```

309 (18, 1, "2018-01-15 18:23:46", 45, 3, 8),
310 (19, 4, "2015-05-30 04:04:24", 45, 3, 4),
311 (20, 2, "2018-05-25 07:27:44", 2, 3, 18),
312 (21, 2, "2019-06-12 06:29:34", 12, 2, 19),
313 (22, 3, "2018-10-31 14:31:16", 41, 2, 18),
314 (23, 1, "2016-01-01 00:00:00", 46, 3, 11),
315 (24, 19, "2017-01-22 02:42:17", 19, 1, 20),
316 (25, 10, "2016-03-13 16:33:11", 44, 1, 7),
317 (26, 1, "2018-05-22 12:17:40", 11, 2, 21),
318 (27, 4, "2017-11-15 16:08:34", 25, 3, 15),
319 (28, 2, "2017-05-04 01:28:28", 7, 2, 2),
320 (29, 2, "2017-12-23 15:27:36", 29, 2, 2),
321 (30, 7, "2019-01-01 00:00:00", 38, 1, 1),
322 (31, 7, "2019-08-10 11:09:48", 38, 1, 1),
323 (32, 14, "2019-07-19 07:48:00", 26, 2, 20),
324 (33, 2, "2019-12-14 14:38:34", 19, 1, 20),
325 (34, 3, "2017-02-15 03:20:49", 32, 3, 29),
326 (35, 4, "2018-03-14 08:04:30", 27, 2, 7),
327 (36, 6, "2019-02-20 01:01:37", 29, 1, 18),
328 (37, 5, "2017-01-23 03:01:48", 6, 2, 13),
329 (38, 1, "2019-05-13 06:38:20", 29, 1, 23),
330 (39, 8, "2018-01-10 00:00:00", 29, 1, 1),
331 (40, 19, "2019-09-30 20:31:49", 41, 1, 29),
332 (41, 6, "2015-01-13 03:32:17", 19, 3, 18),
333 (42, 10, "2018-04-24 01:46:00", 4, 3, 29),
334 (43, 9, "2018-10-23 02:05:29", 44, 2, 9),
335 (44, 1, "2015-09-03 06:41:44", 12, 2, 19),
336 (45, 9, "2018-05-15 06:18:40", 18, 3, 21),
337 (46, 6, "2017-05-12 09:49:53", 45, 1, 19),
338 (47, 9, "2018-05-18 00:33:34", 3, 3, 1),
339 (48, 8, "2017-06-17 08:41:43", 45, 2, 24),
340 (49, 5, "2017-10-04 08:00:22", 28, 1, 1),
341 (50, 5, "2019-02-14 20:56:00", 25, 3, 18),
342 (51, 6, "2016-02-22 21:33:38", 28, 2, 4),
343 (52, 18, "2016-02-20 20:56:52", 36, 3, 7),
344 (53, 9, "2019-07-04 14:48:45", 40, 3, 3),
345 (54, 13, "2018-08-10 00:00:00", 7, 1, 24),
346 (55, 2, "2018-08-16 04:49:17", 3, 3, 1),
347 (56, 5, "2017-09-09 10:32:17", 3, 3, 24),
348 (57, 9, "2018-04-08 07:57:01", 38, 1, 6),
349 (58, 11, "2017-05-14 05:26:36", 23, 2, 18),
350 (59, 2, "2019-10-31 07:09:43", 28, 1, 6),
351 (60, 7, "2018-03-02 07:44:51", 45, 3, 8),
352 (61, 3, "2015-08-17 08:03:34", 38, 2, 30),
353 (62, 18, "2016-02-20 20:56:50", 46, 1, 29),
354 (63, 8, "2019-07-12 09:35:33", 37, 1, 19),
355 (64, 1, "2016-10-31 08:06:10", 50, 3, 23),
356 (65, 5, "2015-07-22 16:33:56", 32, 3, 8),
357 (66, 4, "2015-11-20 15:56:42", 18, 2, 26),
358 (67, 1, "2017-08-20 01:43:26", 32, 2, 18),
359 (68, 2, "2019-05-04 01:26:38", 3, 1, 2),
360 (69, 6, "2018-04-07 09:27:51", 48, 2, 3),
361 (70, 8, "2018-04-10 00:14:14", 3, 3, 15),
362 (71, 1, "2016-07-26 03:21:05", 4, 1, 17),
363 (72, 4, "2019-09-30 02:02:59", 49, 1, 20),
364 (73, 11, "2017-05-06 02:57:25", 32, 2, 24),
365 (74, 3, "2017-12-31 13:21:12", 33, 2, 12),
366 (75, 5, "2019-02-03 04:10:21", 40, 3, 27),
367 (76, 7, "2015-08-27 01:06:07", 13, 1, 22),
368 (77, 11, "2019-10-21 21:32:00", 8, 3, 23),
369 (78, 1, "2016-05-10 00:20:17", 3, 3, 20),
370 (79, 1, "2015-11-19 15:08:05", 37, 2, 5),
371 (80, 4, "2019-11-30 13:35:16", 19, 1, 25),
372 (81, 10, "2015-08-21 19:07:34", 4, 1, 24),
373 (82, 3, "2017-07-24 15:14:15", 28, 1, 30),
374 (83, 2, "2015-06-22 07:03:54", 11, 3, 15),
375 (84, 5, "2015-10-03 19:43:13", 39, 3, 26),
376 (85, 11, "2016-09-24 06:04:15", 18, 1, 20),
377 (86, 7, "2017-03-15 11:12:50", 4, 2, 2),
378 (88, 8, "2016-03-30 03:42:06", 18, 1, 17),
380 (89, 3, "2018-05-24 23:57:47", 35, 2, 5),
381 (90, 7, "2019-01-08 09:55:21", 41, 2, 5),
382 (91, 9, "2019-12-23 19:38:17", 15, 3, 4),
383 (92, 7, "2017-09-04 01:51:17", 30, 2, 1),
384 (93, 1, "2017-09-24 01:59:49", 15, 1, 9),
385 (94, 4, "2018-01-10 00:00:00", 1, 1, 1),
386 (95, 5, "2016-05-11 03:10:18", 2, 1, 1),
387 (96, 10, "2017-04-12 09:41:22", 31, 2, 20),
388 (97, 10, "2018-07-16 10:42:35", 34, 2, 11),
389 (98, 4, "2015-05-03 06:16:59", 49, 2, 38),
390 (99, 3, "2016-11-10 16:01:20", 34, 3, 19),
391 (100, 7, "2019-08-26 07:04:39", 33, 3, 28);
392
393 insert into EquipamentoUtilizado (EquipamentoDisponivel_idEquipamentoDisponivel, TCRelizado_idTCRelizado) values
394
395 (0, 20),
396 (6, 21),
397 (6, 28),
398 (6, 29),
399 (3, 33),
400 (4, 55),
401 (3, 59),
402 (6, 68),
403 (6, 73),
404 (1, 6),
405 (7, 18),
406 (3, 26),
407 (2, 38),
408 (3, 44),
409 (3, 67),
410 (8, 78),
411 (1, 93);

```

II. Script de criação de tabelas

```
1 -- MySQL Workbench Forward Engineering
2
3 SET @OLD_UNIQUE_CHECKS =@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
4 SET @OLD_FOREIGN_KEY_CHECKS =@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
5 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';
6
7
8 -- Schema mieicare
9 -----
10
11 -- Schema mieicare
12
13
14 CREATE SCHEMA IF NOT EXISTS `mieicare` DEFAULT CHARACTER SET utf8 ;
15 USE `mieicare` ;
16
17
18 -- Table `mieicare`.`CodigoPostal`
19
20 CREATE TABLE IF NOT EXISTS `mieicare`.`CodigoPostal` (
21   `codigoPostal` VARCHAR(8) NOT NULL,
22   `cidade` VARCHAR(20) NOT NULL,
23   PRIMARY KEY (`codigoPostal`)
24 ) ENGINE = InnoDB;
25
26
27
28 -- Table `mieicare`.`Modalidade`
29
30 CREATE TABLE IF NOT EXISTS `mieicare`.`Modalidade` (
31   `idModalidade` INT NOT NULL AUTO_INCREMENT,
32   `nome` VARCHAR(45) NOT NULL,
33   `escalao` VARCHAR(10) NOT NULL,
34   PRIMARY KEY (`idModalidade`)
35 ) ENGINE = InnoDB;
36
37
38
39 -- Table `mieicare`.`Atleta`
40
41 CREATE TABLE IF NOT EXISTS `mieicare`.`Atleta` (
42   `idAtleta` INT NOT NULL AUTO_INCREMENT,
43   `email` VARCHAR(45) NOT NULL,
44   `password` VARCHAR(45) NOT NULL,
45   `nome` VARCHAR(45) NOT NULL,
46   `nºCC` VARCHAR(15) NOT NULL,
47   `nif` VARCHAR(15) NOT NULL,
48   `ddn` DATETIME NOT NULL,
49   `genero` CHAR BINARY NOT NULL,
50   `rua` VARCHAR(45) NOT NULL,
51   `CodigoPostal_codigoPostal` VARCHAR(8) NOT NULL,
52   `equipa` VARCHAR(45) NULL,
53   `contacto` VARCHAR(13) NOT NULL,
54   `Modalidade_idModalidade` INT NOT NULL,
55   PRIMARY KEY (`idAtleta`, `CodigoPostal_codigoPostal`, `Modalidade_idModalidade`),
56   INDEX `fk_Atleta_CodigoPostal_idx`(`CodigoPostal_codigoPostal` ASC) VISIBLE,
57   INDEX `fk_Atleta_Modalidade_idx`(`Modalidade_idModalidade` ASC) VISIBLE,
58   CONSTRAINT `fk_Atleta_CodigoPostal`
59     FOREIGN KEY (`CodigoPostal_codigoPostal`)
60       REFERENCES `mieicare`.`CodigoPostal`(`codigoPostal`)
61     ON DELETE NO ACTION
62     ON UPDATE NO ACTION,
63   CONSTRAINT `fk_Atleta_Modalidade`
64     FOREIGN KEY (`Modalidade_idModalidade`)
65       REFERENCES `mieicare`.`Modalidade`(`idModalidade`)
66     ON DELETE NO ACTION
67     ON UPDATE NO ACTION)
68 ENGINE = InnoDB;
69
70
71
72 -- Table `mieicare`.`Clinica`
73
74 CREATE TABLE IF NOT EXISTS `mieicare`.`Clinica` (
75   `idClinica` INT NOT NULL AUTO_INCREMENT,
76   `name` VARCHAR(45) NOT NULL,
77   `rua` VARCHAR(45) NOT NULL,
78   `CodigoPostal_codigoPostal` VARCHAR(8) NOT NULL,
79   PRIMARY KEY (`idClinica`, `CodigoPostal_codigoPostal`),
80   INDEX `fk_Clinica_CodigoPostal_idx`(`CodigoPostal_codigoPostal` ASC) VISIBLE,
81   CONSTRAINT `fk_Clinica_CodigoPostal`
82     FOREIGN KEY (`CodigoPostal_codigoPostal`)
83       REFERENCES `mieicare`.`CodigoPostal`(`codigoPostal`)
84     ON DELETE NO ACTION
85     ON UPDATE NO ACTION)
86 ENGINE = InnoDB;
87
88
89
90 -- Table `mieicare`.`Medico`
91
92 CREATE TABLE IF NOT EXISTS `mieicare`.`Medico` (
93   `idMedico` INT NOT NULL AUTO_INCREMENT,
94   `email` VARCHAR(45) NOT NULL,
95   `password` VARCHAR(45) NOT NULL,
96   `name` VARCHAR(45) NOT NULL,
```

```

97 `nRCC` VARCHAR(15) NOT NULL,
98 `cedula` VARCHAR(15) NOT NULL,
99 `ddn` DATETIME NOT NULL,
100 `genero` CHAR BINARY NOT NULL,
101 `rua` VARCHAR(45) NOT NULL,
102 `CodigoPostal.codigoPostal` VARCHAR(8) NOT NULL,
103 `contacto` VARCHAR(15) NOT NULL,
104 `Clinica_idClinica` INT NOT NULL,
105 PRIMARY KEY (`idMedico`, `CodigoPostal.codigoPostal`, `Clinica_idClinica`),
106 INDEX `fk_Medico_CodigoPostal_idx` (`CodigoPostal.codigoPostal` ASC) VISIBLE,
107 INDEX `fk_Medico_Clinical_idx` (`Clinica_idClinica` ASC) VISIBLE,
108 CONSTRAINT `fk_Medico_CodigoPostal1`
109 FOREIGN KEY (`CodigoPostal.codigoPostal`)
110 REFERENCES `mieicare`.`CodigoPostal` (`codigoPostal`)
111 ON DELETE NO ACTION
112 ON UPDATE NO ACTION,
113 CONSTRAINT `fk_Medico_clinical1`
114 FOREIGN KEY (`Clinica_idClinica`)
115 REFERENCES `mieicare`.`Clinica` (`idClinica`)
116 ON DELETE NO ACTION
117 ON UPDATE NO ACTION)
118 ENGINE = InnoDB;
119
120
121 -- Table: `mieicare`.`TesteClinico`
122
123 CREATE TABLE IF NOT EXISTS `mieicare`.`TesteClinico` (
124 `idtesteClinico` INT NOT NULL AUTO_INCREMENT,
125 `designacao` VARCHAR(45) NOT NULL,
126 `preco` FLOAT NOT NULL,
127 PRIMARY KEY (`idtesteClinico`)
128 ) ENGINE = InnoDB;
129
130
131
132 -- Table: `mieicare`.`TCAgendado`
133
134 CREATE TABLE IF NOT EXISTS `mieicare`.`TCAgendado` (
135 `idTCAgendado` INT NOT NULL AUTO_INCREMENT,
136 `TesteClinico_idTesteClinico` INT NOT NULL,
137 `data` DATETIME NOT NULL,
138 `Atleta_idAtleta` INT NOT NULL,
139 `Clinica_idClinica` INT NOT NULL,
140 `Medico_idMedico` INT NOT NULL,
141 PRIMARY KEY (`idTCAgendado`, `testeClinico.idTesteClinico`, `Atleta_idAtleta`, `Clinica_idClinica`, `Medico_idMedico`),
142 INDEX `fk_TCagendado_Clinica_idx` (`Clinica_idClinica` ASC) VISIBLE,
143 INDEX `fk_TCagendado_Medico_idx` (`Medico_idMedico` ASC) VISIBLE,
144 INDEX `fk_TCagendado_TesteClinico_idx` (`TesteClinico_idTesteClinico` ASC) VISIBLE,
145 CONSTRAINT `fk_TCagendado_Atleta1`
146 FOREIGN KEY (`Atleta_idAtleta`)
147 REFERENCES `mieicare`.`Atleta` (`idAtleta`)
148 ON DELETE NO ACTION
149 ON UPDATE NO ACTION,
150 CONSTRAINT `fk_TCagendado_Clinical1`
151 FOREIGN KEY (`Clinica_idClinica`)
152 REFERENCES `mieicare`.`Clinica` (`idClinica`)
153 ON DELETE NO ACTION
154 ON UPDATE NO ACTION,
155 CONSTRAINT `fk_TCagendado_Medical1`
156 FOREIGN KEY (`Medico_idMedico`)
157 REFERENCES `mieicare`.`Medico` (`idMedico`)
158 ON DELETE NO ACTION
159 ON UPDATE NO ACTION,
160 CONSTRAINT `fk_TCagendado_TesteClinico1`
161 FOREIGN KEY (`TesteClinico_idTesteClinico`)
162 REFERENCES `mieicare`.`TesteClinico` (`idTesteClinico`)
163 ON DELETE NO ACTION
164 ON UPDATE NO ACTION)
165 ENGINE = InnoDB;
166
167
168
169 -- Table: `mieicare`.`TCTRealizado`
170
171 CREATE TABLE IF NOT EXISTS `mieicare`.`TCTRealizado` (
172 `idTCTRealizado` INT NOT NULL AUTO_INCREMENT,
173 `TesteClinico_idTesteClinico` INT NOT NULL,
174 `data` DATETIME NOT NULL,
175 `Atleta_idAtleta` INT NOT NULL,
176 `Clinica_idClinica` INT NOT NULL,
177 `Medico_idMedico` INT NOT NULL,
178 PRIMARY KEY (`idTCTRealizado`, `TesteClinico_idTesteClinico`, `Atleta_idAtleta`, `Clinica_idClinica`, `Medico_idMedico`),
179 INDEX `fk_TCTRealizado_Clinica_idx` (`Clinica_idClinica` ASC) VISIBLE,
180 INDEX `fk_TCTRealizado_Medico_idx` (`Medico_idMedico` ASC) VISIBLE,
181 INDEX `fk_TCTRealizado_TesteClinico1_idx` (`TesteClinico_idTesteClinico` ASC) VISIBLE,
182 CONSTRAINT `fk_TCTRealizado_Atleta1`
183 FOREIGN KEY (`Atleta_idAtleta`)
184 REFERENCES `mieicare`.`Atleta` (`idAtleta`)
185 ON DELETE NO ACTION
186 ON UPDATE NO ACTION,
187 CONSTRAINT `fk_TCTRealizado_Clinical1`
188 FOREIGN KEY (`Clinica_idClinica`)
189 REFERENCES `mieicare`.`Clinica` (`idClinica`)
190 ON DELETE NO ACTION
191 ON UPDATE NO ACTION,
```

```

193  CONSTRAINT `fk_TCRealizado_Medico1`  

194    FOREIGN KEY (`Medico_idMedico`)  

195      REFERENCES `mieicare`.`Medico` (`idMedico`)  

196    ON DELETE NO ACTION  

197    ON UPDATE NO ACTION,  

198  CONSTRAINT `fk_TCRealizado_TesteClinico1`  

199    FOREIGN KEY (`TesteClinico_idTesteClinico`)  

200      REFERENCES `mieicare`.`TesteClinico` (`idTesteClinico`)  

201    ON DELETE NO ACTION  

202    ON UPDATE NO ACTION  

203 ENGINE = InnoDB;  

204  

205  

206  -- Table `mieicare`.`Contacto`  

207  

208  CREATE TABLE IF NOT EXISTS `mieicare`.`Contacto` (  

209    `numero` VARCHAR(13) NOT NULL,  

210    `Clinica_idClinica` INT NOT NULL,  

211    PRIMARY KEY (`numero`, `Clinica_idClinica`),  

212    INDEX `fk_Contacto_Clinical_idx`(`Clinica_idClinica` ASC) VISIBLE,  

213    CONSTRAINT `fk_Contacto_Clinical`  

214      FOREIGN KEY (`Clinica_idClinica`)  

215          REFERENCES `mieicare`.`Clinica` (`idClinica`)  

216        ON DELETE NO ACTION  

217        ON UPDATE NO ACTION  

218  ENGINE = InnoDB;  

219  

220  

221  

222  -- Table `mieicare`.`Equipamento`  

223  

224  CREATE TABLE IF NOT EXISTS `mieicare`.`Equipamento` (  

225    `idEquipamento` INT NOT NULL AUTO_INCREMENT,  

226    `name` VARCHAR(45) NOT NULL,  

227    PRIMARY KEY (`idEquipamento`)  

228  ENGINE = InnoDB;  

229  

230  

231  

232  -- Table `mieicare`.`EquipamentoDisponivel`  

233  

234  CREATE TABLE IF NOT EXISTS `mieicare`.`EquipamentoDisponivel` (  

235    `idEquipamentoDisponivel` INT NOT NULL AUTO_INCREMENT,  

236    `Equipamento_idEquipamento` INT NOT NULL,  

237    `Clinica_idClinica` INT NOT NULL,  

238    PRIMARY KEY (`idEquipamentoDisponivel`, `Equipamento_idEquipamento`, `Clinica_idClinica`),  

239    INDEX `fk_EquipamentoDisponivel_Equipamento1_idx`(`Equipamento_idEquipamento` ASC) VISIBLE,  

240    INDEX `fk_EquipamentoDisponivel_Clinical_idx`(`Clinica_idClinica` ASC) VISIBLE,  

241    CONSTRAINT `fk_EquipamentoDisponivel_Equipamento1`  

242      FOREIGN KEY (`Equipamento_idEquipamento`)  

243          REFERENCES `mieicare`.`Equipamento` (`idEquipamento`)  

244        ON DELETE NO ACTION  

245        ON UPDATE NO ACTION,  

246        CONSTRAINT `fk_EquipamentoDisponivel_Clinical`  

247          FOREIGN KEY (`Clinica_idClinica`)  

248              REFERENCES `mieicare`.`Clinica` (`idClinica`)  

249            ON DELETE NO ACTION  

250            ON UPDATE NO ACTION  

251  ENGINE = InnoDB;  

252  

253  

254  

255  

256  -- Table `mieicare`.`EquipamentoUtilizado`  

257  

258  CREATE TABLE IF NOT EXISTS `mieicare`.`EquipamentoUtilizado` (  

259    `EquipamentoDisponivel_idEquipamentoDisponivel` INT NOT NULL,  

260    `TCRealizado_idTCRealizado` INT NOT NULL,  

261    PRIMARY KEY (`EquipamentoDisponivel_idEquipamentoDisponivel`, `TCRealizado_idTCRealizado`),  

262    INDEX `fk_Equipamento_has_TCRealizado_idx`(`TCRealizado_idTCRealizado` ASC) VISIBLE,  

263    INDEX `fk_Equipamento_has_TCRealizado_Equipamento1_idx`(`EquipamentoDisponivel_idEquipamentoDisponivel` ASC) VISIBLE,  

264    CONSTRAINT `fk_Equipamento_has_TCRealizado_Equipamento1`  

265      FOREIGN KEY (`EquipamentoDisponivel_idEquipamentoDisponivel`)  

266          REFERENCES `mieicare`.`EquipamentoDisponivel` (`idEquipamentoDisponivel`)  

267        ON DELETE NO ACTION  

268        ON UPDATE NO ACTION,  

269        CONSTRAINT `fk_Equipamento_has_TCRealizado_TCRealizado1`  

270          FOREIGN KEY (`TCRealizado_idTCRealizado`)  

271              REFERENCES `mieicare`.`TCRealizado` (`idTCRealizado`)  

272            ON DELETE NO ACTION  

273            ON UPDATE NO ACTION  

274  ENGINE = InnoDB;  

275  

276  

277  -- Table `mieicare`.`EquipamentoOpcional`  

278  

279  CREATE TABLE IF NOT EXISTS `mieicare`.`EquipamentoOpcional` (  

280    `TesteClinico_idTesteClinico` INT NOT NULL,  

281    `Equipamento_idEquipamento` INT NOT NULL,  

282    PRIMARY KEY (`TesteClinico_idTesteClinico`, `Equipamento_idEquipamento`),  

283    INDEX `fk_Testeclinico_has_Equipamento_Equipamento1_idx`(`Equipamento_idEquipamento` ASC) VISIBLE,  

284    INDEX `fk_Testeclinico_has_Equipamento_Equipamento1_idx`(`TesteClinico_idTesteClinico` ASC) VISIBLE,  

285    CONSTRAINT `fk_Testeclinico_has_Equipamento_Equipamento1`  

286      FOREIGN KEY (`TesteClinico_idTesteClinico`)  

287          REFERENCES `mieicare`.`TesteClinico` (`idTesteClinico`)  

288        ON DELETE NO ACTION  

289        ON UPDATE NO ACTION,  

290        CONSTRAINT `fk_Testeclinico_has_Equipamento_Equipamento1`  

291          FOREIGN KEY (`Equipamento_idEquipamento`)  

292              REFERENCES `mieicare`.`Equipamento` (`idEquipamento`)  

293            ON DELETE NO ACTION  

294            ON UPDATE NO ACTION  

295  ENGINE = InnoDB;  

296  

297  

298  SET SQL_MODE=@OLD_SQL_MODE;  

299  SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  

300  SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

III. Script funções

```
1  USE mieicare;
2
3  SET SQL_SAFE_UPDATES = 0;
4  SET GLOBAL log_bin_trust_function_creators = 1;
5
6  -- Determina o id do teste com uma dada designacao
7  DELIMITER $$
8  ▼ CREATE FUNCTION mieicare.escolheTeste(des VARCHAR(45)) RETURNS INT
9  ▼   BEGIN
10     DECLARE resultado INT;
11
12     SELECT idTesteClinico INTO resultado FROM TesteClinico
13     WHERE designacao = des;
14
15     return resultado;
16 END $$
17
18 -- Determina o preco do teste com uma dada designacao
19 DELIMITER $$
20 ▼ CREATE FUNCTION mieicare.precioTeste(des VARCHAR(45)) RETURNS FLOAT
21 ▼   BEGIN
22     DECLARE resultado FLOAT;
23
24     SELECT preco INTO resultado FROM TesteClinico
25     WHERE designacao = des;
26
27     return resultado;
28 END $$
29
30 -- Calcula o valor total faturado por uma clinica
31 DELIMITER $$
32 ▼ CREATE FUNCTION mieicare.faturacaoClinica(idC INT) RETURNS FLOAT
33 ▼   BEGIN
34     DECLARE resultado FLOAT;
35
36     SELECT SUM(TC.preco) INTO resultado FROM TCRealizado AS TCR
37         INNER JOIN TesteClinico AS TC
38         ON TCR.TesteClinico_idTesteClinico = TC.idTesteClinico
39         WHERE TCR.Clinica_idClinica = idC;
40
41     RETURN resultado;
42 END $$
43
44 USE mieicare;
45
46 SET SQL_SAFE_UPDATES = 0;
47 SET GLOBAL log_bin_trust_function_creators = 1;
48
49 -- Calcula o valor total faturado por todas as clinicas
50 DELIMITER $$
51 ▼ CREATE FUNCTION mieicare.faturacaoTotal() RETURNS FLOAT
52 ▼   BEGIN
53     DECLARE resultado FLOAT;
54
55     SELECT SUM(TC.preco) INTO resultado FROM TCRealizado AS TCR
56         INNER JOIN TesteClinico AS TC
57         ON TCR.TesteClinico_idTesteClinico = TC.idTesteClinico;
58
59     RETURN resultado;
60 END $$
```

IV. Script conversão de dados em ficheiros csv

```
1 USE mieicare;
2
3 SELECT * FROM Atleta AS A
4 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/atleta.csv'
5 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
6 LINES TERMINATED BY '\r\n';
7
8 SELECT * FROM Clinica AS C
9 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/clinica.csv'
10 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
11 LINES TERMINATED BY '\r\n';
12
13 SELECT * FROMCodigoPostal AS CP
14 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/codigopostal.csv'
15 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
16 LINES TERMINATED BY '\r\n';
17
18 SELECT * FROM Contacato AS C
19 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/contacto.csv'
20 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
21 LINES TERMINATED BY '\r\n';
22
23 SELECT * FROM Equipamento AS E
24 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/equipamento.csv'
25 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
26 LINES TERMINATED BY '\r\n';
27
28 SELECT * FROM EquipamentoDisponivel AS ED
29 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/equipamentodisponivel.csv'
30 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
31 LINES TERMINATED BY '\r\n';
32
33 SELECT * FROM EquipamentoNecessario AS EN
34 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/equipamentonecessario.csv'
35 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
36 LINES TERMINATED BY '\r\n';
37
38 SELECT * FROM EquipamentoUtilizado AS EU
39 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/equipamentoutilizado.csv'
40 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
41 LINES TERMINATED BY '\r\n';
42
43 SELECT * FROM Medico AS M
44 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/medico.csv'
45 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
46 LINES TERMINATED BY '\r\n';
47
48 SELECT * FROM Modalidade AS Mo
49 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/modalidade.csv'
50 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
51 LINES TERMINATED BY '\r\n';
52
53 SELECT * FROM TCAgendado AS TCA
54 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/tcagendado.csv'
55 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
56 LINES TERMINATED BY '\r\n';
57
58 SELECT * FROM TCREalizado AS TCR
59 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/tcrealizado.csv'
60 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
61 LINES TERMINATED BY '\r\n';
62
63 SELECT * FROM TesteClinico AS TC
64 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/testeclinico.csv'
65 FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY ''
66 LINES TERMINATED BY '\r\n';
```

V. Script criação dos nodos (e importação de dados)

```
1 //modalidade nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://modalidade.csv' AS line
4 CREATE(c:Modalidade {idModalidade: toInteger(line[0]), nome: line[1], escalao: line[2]})

1 //atleta nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://atleta.csv' AS line
4 CREATE(c:Atleta {idAtleta: toInteger(line[0]), email: line[1], password: line[2], nome: line[3],
    n°CC: line[4], nif: line[5], ddn: line[6], genero: line[7], rua: line[8],
    CódigoPostal_codigoPostal: line[9], equipa: line[10], contacto: line[11],
    Modalidade_idModalidade: toInteger(line[12])})

1 //TCAgendado nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://tcagendado.csv' AS line
4 CREATE(c:TCAgendado {idTCAgendado: toInteger(line[0]), TesteClinico_idTesteClinico:
    toInteger(line[1]), data: line[2], Atleta_idAtleta: toInteger(line[3]), Clinica_idClinica:
    toInteger(line[4]), Medico_idMedico: toInteger(line[5])})

1 //clinica nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://clinica.csv' AS line
4 CREATE(c:Clinica {idClinica: toInteger(line[0]), nome: line[1], rua: line[2],
    CódigoPostal_codigoPostal: line[3]})

1 //equipamentoNecessario nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://equipamentonecessario.csv' AS line
4 CREATE(c:EquipamentoNecessario {TesteClinico_idTesteClinico: toInteger(line[0]),
    Equipamento_idEquipamento: toInteger(line[1])})

1 //codigoPostal nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://codigopostal.csv' AS line
4 CREATE(c:CodigoPostal {codigoPostal: line[0], cidade: line[1]})

1 //testeClinico nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://testeclinico.csv' AS line
4 CREATE(c:TesteClinico {idTesteClinico: toInteger(line[0]), designacao: line[1], preco:
    toFloat(line[2])})
```

```

1 //equipamento nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://equipamento.csv' AS line
4 CREATE(c:Equipamento {idEquipmento: toInteger(line[0]), nome: line[1]})

1 //medico nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://medico.csv' AS line
4 CREATE(c:Medico {idMedico: toInteger(line[0]), email: line[1], password: line[2], nome: line[3],
nºCC: line[4], cedula: line[5], ddn: line[6], genero: line[7], rua: line[8],
CodigoPostal_codigoPostal: line[9], contacto: line[10], Clinica_idClinica: toInteger(line[11])})

1 //TCRealizado nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://tcrealizado.csv' AS line
4 CREATE(c:TCRealizado {idTCRealizado: toInteger(line[0]), TesteClinico_idTesteClinico:
toInteger(line[1]), data: line[2], Atleta_idAtleta: toInteger(line[3]), Clinica_idClinica:
toInteger(line[4]), Medico_idMedico: toInteger(line[5])})

1 //contacto nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://contacto.csv' AS line
4 CREATE(c:Contacto {numero: line[0], Clinica_idClinica: toInteger(line[1])})

1 //equipamentoUtilizado nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://equipamentoutilizado.csv' AS line
4 CREATE(c:EquipamentoUtilizado {EquipmentoDisponivel_idEquipmentoDisponivel: toInteger(line[0]),
TCRealizado_idTCRealizado: toInteger(line[1])})

1 //equipmentoDisponivel nodes
2 USING PERIODIC COMMIT
3 LOAD CSV FROM 'file://equipamentodisponivel.csv' AS line
4 CREATE(c:EquipmentoDisponivel {idEquipmentoDisponivel: toInteger(line[0]),
Equipmento_idEquipmento: toInteger(line[1]), Clinica_idClinica: toInteger(line[2])})

```

VI. Script criação dos relacionamentos

```
1 MATCH (a:Atleta), (b:CodigoPostal)
2 WHERE a.CodigoPostal_codigoPostal = b.codigoPostal
3 CREATE (a)-[:POSSUI]->(b)
```

```
1 MATCH (a:Atleta), (b:Modalidade)
2 WHERE a.Modalidade_idModalidade = b.idModalidade
3 CREATE (a)-[:PRATICA]->(b)
```

```
1 MATCH (a:Atleta), (b:TCAgendado)
2 WHERE a.idAtleta = b.Atleta_idAtleta
3 CREATE (a)-[:AGENDOU]->(b)
```

```
1 MATCH (a:Atleta), (b:TCRealizado)
2 WHERE a.idAtleta = b.Atleta_idAtleta
3 CREATE (a)-[:REALIZOU]->(b)
```

```
1 MATCH (a:Clinica), (b:CodigoPostal)
2 WHERE a.CodigoPostal_codigoPostal = b.codigoPostal
3 CREATE (a)-[:POSSUI]->(b)
```

```
1 MATCH (a:Clinica), (b:Contacto)
2 WHERE a.idClinica = b.Clinica_idClinica
3 CREATE (a)-[:TEM]->(b)
```

```
1 MATCH (a:Clinica), (b:EquipamentoDisponivel)
2 WHERE a.idClinica = b.Clinica_idClinica
3 CREATE (a)-[:POSSUI]->(b)
```

```
1 MATCH (a:Clinica), (b:Medico)
2 WHERE a.idClinica = b.Clinica_idClinica
3 CREATE (a)-[:EMPREGA]->(b)
```

```
1 MATCH (a:Clinica), (b:TCAgendado)
2 WHERE a.idClinica = b.Clinica_idClinica
3 CREATE (a)-[:AGENDOU]->(b)
```

```
1 MATCH (a:Clinica), (b:TCRealizado)
2 WHERE a.idClinica = b.Clinica_idClinica
3 CREATE (a)-[:REALIZOU]->(b)
```

```
1 MATCH (a:EquipamentoDisponivel), (b:Equipamento)
2 WHERE a.Equipamento_idEquipamento = b.idEquipamento
3 CREATE (a)-[:É]->(b)
```

```
1 MATCH (a:Medico), (b:CodigoPostal)
2 WHERE a.CodigoPostal_codigoPostal = b.codigoPostal
3 CREATE (a)-[:POSSUI]->(b)
```

```
1 MATCH (a:Medico), (b:TCAgendado)
2 WHERE a.idMedico = b.Medico_idMedico
3 CREATE (a)-[:AGENDOU]->(b)
```

```
1 MATCH (a:Medico), (b:TCRealizado)
2 WHERE a.idMedico = b.Medico_idMedico
3 CREATE (a)-[:REALIZOU]->(b)
```

```
1 MATCH (tcr:TCRealizado), (en:EquipamentoUtilizado), (e:EquipamentoDisponivel)
2 WHERE tcr.idTCRealizado = en.TCRealizado_idTCRealizado AND
en.EquipamentoDisponivel_idEquipamentoDisponivel = e.idEquipamentoDisponivel
3 CREATE (tcr)-[:UTILIZOU]->(e)
```

```
1 MATCH (a:TCRealizado), (b:TesteClinico)
2 WHERE a.TesteClinico_idTesteClinico = b.idTesteClinico
3 CREATE (a)-[:ÉUM]->(b)
```

```
1 MATCH (tc:TesteClinico), (en:EquipamentoNecessario), (e:Equipamento)
2 WHERE tc.idTesteClinico = en.TesteClinico_idTesteClinico AND en.Equipamento_idEquipamento =
e.idEquipamento
3 CREATE (tc)-[:NECESSITA]->(e)■
```

VII. Script criação dos índices

```
1 //clinica index
2 CREATE INDEX ON :Clinica(idClinica);
3
4 //atleta index
5 CREATE INDEX ON :Atleta(nome);
6
7 //medico index
8 CREATE INDEX ON :Medico(nome);
9
10 //testeclinico index
11 CREATE INDEX ON :TesteClinico(designacao);
12
13 //tcagendado index
14 CREATE INDEX ON :TCAgendado(idTCAgendado);
15
16 //tcrealizado index
17 CREATE INDEX ON :TCRealizacao(idTCRealizado);
18
19 //equipamento index
20 CREATE INDEX ON :Equipamento(nome);
```

VIII. Script remoção dos dados redundantes

```
1 MATCH (eu: EquipamentoUtilizado)
2 DELETE eu;
3
4 MATCH (en: EquipamentoNecessario)
5 DELETE en;
6
7 MATCH (c: Clinica)
8 DELETE c.CodigoPostal_codigoPostal
9 RETURN c;
10
11 MATCH (a: atleta)
12 DELETE a.CodigoPostal_codigoPostal, a.Modalidade_idModalidade
13 RETURN a;
14
15 MATCH (m: Medico)
16 DELETE m.CodigoPostal_codigoPostal, m.Clinica_idClinica
17 RETURN m;
18
19 MATCH (tca: TCAgendado)
20 DELETE tca.TesteClinico_idTesteClinico, tca.Atleta_idAtleta, tca.Clinica_idClinica, tca.Medico_idMedico
21 RETURN tca;
22
23 MATCH (tcr: TCRealizado)
24 DELETE tcr.TesteClinico_idTesteClinico, tcr.Atleta_idAtleta, tcr.Clinica_idClinica, tcr.Medico_idMedico
25 RETURN tcr;
26
27 MATCH (ed: EquipamentoDisponivel)
28 DELETE ed.Equipamento_idEquipamento, ed.Clinica_idClinica
29 RETURN ed;
30
31 MATCH (c: ContacTo)
32 DELETE c.Clinica_idClinica
33 RETURN c;
```