



Universidade do Minho

Mestrado Integrado em Engenharia Informática

Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

Testes Clínicos de Atletas de atletismo

Ana Margarida Campos A85166

Ana Catarina Gil A85266

Tânia Rocha A85176

Janeiro, 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Testes Clínicos de Atletas de atletismo

Ana Margarida Campos A85166

Ana Catarina Gil A85266

Tânia Rocha A85176

Janeiro, 2020

Resumo

Este projeto tem como principal objetivo o desenvolvimento de um sistema de gestão de bases de dados relacional e não relacional, que irá permitir administrar, de forma adequada, o agendamento e realização de testes clínicos de atletas de Atletismo de diferentes modalidades e categorias.

De maneira a obter um modelo de base de dados viável, procuramos seguir um conjunto de passos bem definidos. Assim, começamos o nosso trabalho pelo levantamento de requisitos. Esta fase consiste principalmente na escolha e na procura de informação relevante e imprescindível para o desenvolvimento do projeto bem como solucionar alguns conflitos que podem aparecer ao longo do mesmo. Seguiu-se o desenho do modelo conceptual caracterizado por ser um modelo de dados independente da implementação lógica e física. Com base neste, foi realizado o modelo lógico, modelo esse que aperfeiçoa o modelo conceptual. Por fim, realizou-se o modelo físico, onde já é possível a manipulação de dados.

Em síntese, podemos concluir que o trabalho aqui apresentado cumpre as condições necessárias para que seja gerada uma base de dados simples de acordo com o problema inicialmente proposto.

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

Palavras-Chave: Bases de Dados, levantamento de requisitos, modelo conceptual, modelo lógico, modelo físico, manipulação de dados, mysql, neo4j.

Conteúdo

Resumo	i
Índice de Figuras	iv
Índice de Tabelas	vi
1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	1
1.3. Motivação e Objetivos	2
1.4. Estrutura do Relatório	2
2. Requisitos levantados	3
2.1. Requisitos de descrição (RD)	3
2.2. Requisitos de exploração (RE)	3
2.3. Requisitos de controlo	4
3. Desenvolvimento do Modelo Conceptual	5
4. Modelação lógica	8
4.1. Construção do modelo de dados lógico	8
4.2. Derivar relações a partir do modelo concetual	14
4.3. Validar relações usando a normalização	15
4.4. Validação do Modelo com Interrogações do Utilizador	16
4.5. Validação do Modelo com as Transações Estabelecidas	16
4.6. Desenho do Modelo Lógico	17
4.7. Rever modelo lógico com o utilizador	17
5. Implementação Física	18
5.1. Seleção do sistema de gestão de base de dados	18
5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL	18
5.3. Tradução de alguns dos requisitos do utilizador para o SQL	22
5.4. Definições e caracterização dos mecanismos de segurança em SQL	32
5.5. Definição e caracterização das vistas de utilização em SQL	33
5.6. Revisão do sistema implementado com o utilizador	33
6. Base de Dados não relacional (NoSQL)	34
7. Migração da base de dados	35
7.1. Passagem do modelo anterior para o modelo orientado a grafos	35
8. Migração das interrogações para o modelo não relacional	37
9. Análise comparativa entre SQL e NEO4J	45
10. Conclusão	46
Referências	47
Lista de Siglas e Acrónimos	48
Anexos	49

Anexo 1 – Entrevista	49
Anexo 2 – Modelo Físico no <i>MySQL</i>	50

Índice de Figuras

Figura 1: Modelo Conceptual	5
Figura 2: Passagem da entidade Atleta para o modelo lógico	8
Figura 3: Passagem da entidade Tecnico para o modelo lógico	9
Figura 4: Passagem da entidade Modalidade para o modelo lógico	9
Figura 5: Passagem da entidade Categoria para o modelo lógico	10
Figura 6: Passagem da entidade Especialidade para o modelo lógico	10
Figura 7: Passagem da entidade Recurso para o modelo lógico	11
Figura 8: Passagem da entidade Localizacao para o modelo lógico	11
Figura 9: Passagem da entidade TesteClinico para o modelo lógico	12
Figura 10: Passagem da relação de N para N e do atributo Medicamento para o modelo lógico	13
Figura 11: Modelo lógico	17
Figura 12: Tabela Categoria no modelo físico	18
Figura 13: Tabela Modalidade no modelo físico	19
Figura 14: Tabela Atleta no modelo físico	19
Figura 15: Tabela Especialidade no modelo físico	20
Figura 16: Tabela Tecnico no modelo físico	20
Figura 17: Tabela Medicamento no modelo físico	20
Figura 18: Tabela TesteClinico no modelo físico	21
Figura 19: Tabela Recurso no modelo físico	21
Figura 20: Tabela Atleta_has_Medicamento no modelo físico	22
Figura 21: Listagem de atletas por modalidade	22
Figura 22: Resultado obtido de RE1	23
Figura 23: Atletas listados por localidade	23
Figura 24: Resultado obtido de RE2	23
Figura 25: Listagem de dados referentes aos testes clínicos bem como o seu respectivo ano.	24
Figura 26: Resultado obtido RE3	24
Figura 27: Quantia gasta por cada modalidade	24
Figura 28: Resultado obtido RE4	25
Figura 29: Consultas para um determinado atleta	25

Figura 30: Resultado obtido RE5	25
Figura 31: Ranking do número de consultas para cada especialidade	25
Figura 32: Resultado obtido RE6	26
Figura 33: Ranking da faturação por especialidade	26
Figura 34: Resultado obtido RE7	26
Figura 35: Atletas que faltaram a consultas	26
Figura 36: Resultado obtido (RE7)	26
Figura 37: Ranking dos médicos pelo número de consultas	27
Figura 38: Resultado obtido (RE9)	27
Figura 39: Agendamento de um teste clínico	28
Figura 40: Cancelamento de um teste clínico	29
Figura 41: Listagem de medicamento de um dado atleta	29
Figura 42: Resultado obtido RE12	29
Figura 43: Listagem de atletas por categoria	30
Figura 44: Resultado obtido RE13	30
Figura 45: Percentagem de equipamentos usados num certo dia	30
Figura 46: Resultado obtido RE14	31
Figura 47: Adicionar um Atleta	31
Figura 48: Criação de utilizadores no <i>MySQL</i>	32
Figura 49: Vista do Técnico em relação à tabela Teste Clínico	33
Figura 50: Vista do Técnico em relação à tabela Recurso	33
Figura 51: Vista do Técnico em relação à tabela Atleta	33
Figura 52: Exemplo de importação	35
Figura 53: Grafo exemplo	36
Figura 54: Grafo Final	36
Figura 55: Atletas por modalidade	37
Figura 56: Atletas por localidade	38
Figura 57: Dados de consulta por cada ano	38
Figura 58: Quantia gasta por modalidade	39
Figura 59: Próximas consultas para um atleta	39
Figura 60: Ranking de número de consultas por especialidade	40
Figura 61: Ranking de faturação por especialidade	40
Figura 62: Atletas que faltaram a consultas	41
Figura 63: Ranking do número de consultas por técnico	41
Figura 64: Agendar um teste clínico	42
Figura 65: Cancelar teste clínico	42
Figura 66: Listagem de medicamentos de um dado atleta	42
Figura 67: Atletas por categoria	43
Figura 68: Percentagem de equipamentos usados num certo dia	43

Índice de Tabelas

Tabela 1: Identificação e associação dos atributos com as entidades	7
Tabela 2: Identificação das chaves primárias e estrangeiras	15

1. Introdução

No âmbito da unidade curricular de Base de Dados foi-nos proposto a análise, planeamento e implementação de um Sistema de Gestão de Base de Dados relacional e não relacional, tendo como base o agendamento e realização de Testes Clínicos de atletas de Atletismo de diferentes modalidades e categorias. Neste relatório, são apresentados todos os passos dados no sentido a cumprir o objetivo proposto.

1.1. Contextualização

Desde o ano 2000 que é realizada em Portugal uma competição nacional de atletismo, que visa a participação de todos os atletas das diversas modalidades e categorias. Esta competição foi originalmente concebida pelo representante da federação portuguesa de atletismo que pediu auxílio a uma empresa para a elaboração dos testes clínicos de todos os atletas participantes.

Esta empresa tem várias clínicas espalhadas por todo o país (Braga, Lisboa, Barcelos, Setúbal, Faro) e é caracterizada pelos seus bons equipamentos e excelentes técnicos. No entanto, os métodos usados para guardar informações necessárias para o bom funcionamento destas clínicas usam tecnologias ultrapassadas para os dias que correm tornando a gerência da empresa cada vez mais difícil.

1.2. Apresentação do Caso de Estudo

Devido a problemas financeiros o dono da empresa de clínicas teve de a vender passando esta a possuir atualmente uma nova gerência.

A chegada da nova gerência implicou alterações a nível de guardar e alterar informações no agendamento e realização de testes clínicos pois estas encontravam-se ainda a ser manipuladas de uma maneira não informatizada.

De maneira a tornar a gestão mais fácil, não só na atualidade, mas também no futuro, a empresa decidiu ver opções para implementar um sistema mais adequado.

1.3. Motivação e Objetivos

Uma das motivações que levou o Senhor Rui, novo gerente da empresa, a contratar um grupo de engenheiros informáticas foi o facto do antigo dono guardar todas as informações em formato papel (agendas e livros com todas as informações sobre atletas, técnicos e testes clínicos) sendo bastante difícil a organização de todos os documentos. Outro dos motivos apresentado, foi o facto de este querer expandir o número de clínicas por mais locais ao longo do país.

O principal objetivo na elaboração deste projeto é o desenvolvimento de um sistema de gestão de bases de dados relacional e não relacional, que irá permitir administrar, de forma adequada, os recursos de várias clínicas para que desta forma seja possível agendar e realizar testes clínicos de atletas de Atletismo de diferentes modalidades e categorias.

1.4. Estrutura do Relatório

Após a apresentação do contexto do problema, do caso de estudo, das suas motivações e objetivos surge agora os passos para o desenvolvimento da base de dados propriamente dita. Primeiramente foi elaborado o levantamento de requisitos. Seguidamente foi desenvolvido os modelos conceptual, lógico e físico através da utilização do *MySQLWorkbench*. Posteriormente, foi elaborado o modelo não relacional através da utilização do software *Neo4j*. No fim, é ainda apresentada uma secção dedicada à análise crítica do projeto bem como algumas referências utilizadas ao longo do mesmo.

2. Requisitos levantados

2.1. Requisitos de descrição (RD)

RD1: A base de dados deverá ser capaz de guardar informação relativa aos atletas, técnicos, testes clínicos, recursos;

RD2: Todos os atletas devem ser registados no sistema;

RD3: Os atletas devem ser caracterizados por: nome, idade, género, email, cartão de cidadão, localidade, modalidade, categoria, data de nascimento e contacto telefónico;

RD4: Todos os técnicos atualmente empregues devem ser registados no sistema;

RD5: Os técnicos devem ser caracterizados por: nome, número de identificação do técnico (id), morada, contacto telefónico e especialidade;

RD6: O sistema deverá ser capaz de registar os testes clínicos de cada atleta;

RD7: Cada teste clínico deve ser caracterizado por: tipo, duração, estado, data e identificação do teste clínico (id);

2.2. Requisitos de exploração (RE)

RE1: O sistema deverá ser capaz de agrupar os atletas de acordo com a sua modalidade;

RE2: O sistema deverá ser capaz de agrupar os atletas de acordo com a sua localidade;

RE3: O sistema deverá ser capaz de listar dados referentes aos testes clínicos bem como o seu respetivo ano.

RE4: O sistema deverá ser capaz de listar as modalidades e a respetiva quantia gasta por cada uma até ao momento.

RE5: O sistema deverá ser capaz de apresentar a um determinado cliente todas as suas consultas efetuadas até ao momento.

RE6: O sistema deverá ser capaz de apresentar o ranking do número de consultas para cada especialidade.

RE7: O sistema deverá ser capaz de apresentar o ranking da faturação por especialidade.

RE8: O sistema deverá ser capaz de apresentar por ano quantos atletas foram realizar testes clínicos.

RE9: O sistema deverá ser capaz de mostrar o ranking dos médicos pelo número de consultas.

RE10: O sistema deverá ser capaz de agendar testes clínicos.

RE11: O sistema deverá ser capaz de cancelar testes clínicos.

RE12: O sistema deverá ser capaz de listar os medicamentos de cada atleta.

RE13: O sistema deverá ser capaz de listar os atletas por categoria.

RE14: O sistema deverá ser capaz de apresentar a percentagem de equipamentos usados num determinado dia.

RE15: O sistema deverá ser capaz de adicionar um novo atleta.

2.3. Requisitos de controlo

RC1: O dono da clínica de testes clínicos para atletas deverá possuir controlo total à base de dados;

RC2: Os técnicos da clínica não podem aceder a qualquer informação dos outros técnicos;

RC3: Os atletas não podem ter acesso à base de dados.

3. Desenvolvimento do Modelo Conceptual

Após a análise de requisitos, passamos para a concepção do Modelo Concetual. Esta modelação consiste no processo de construção de um modelo que é capaz de relacionar informação independentemente da implementação.

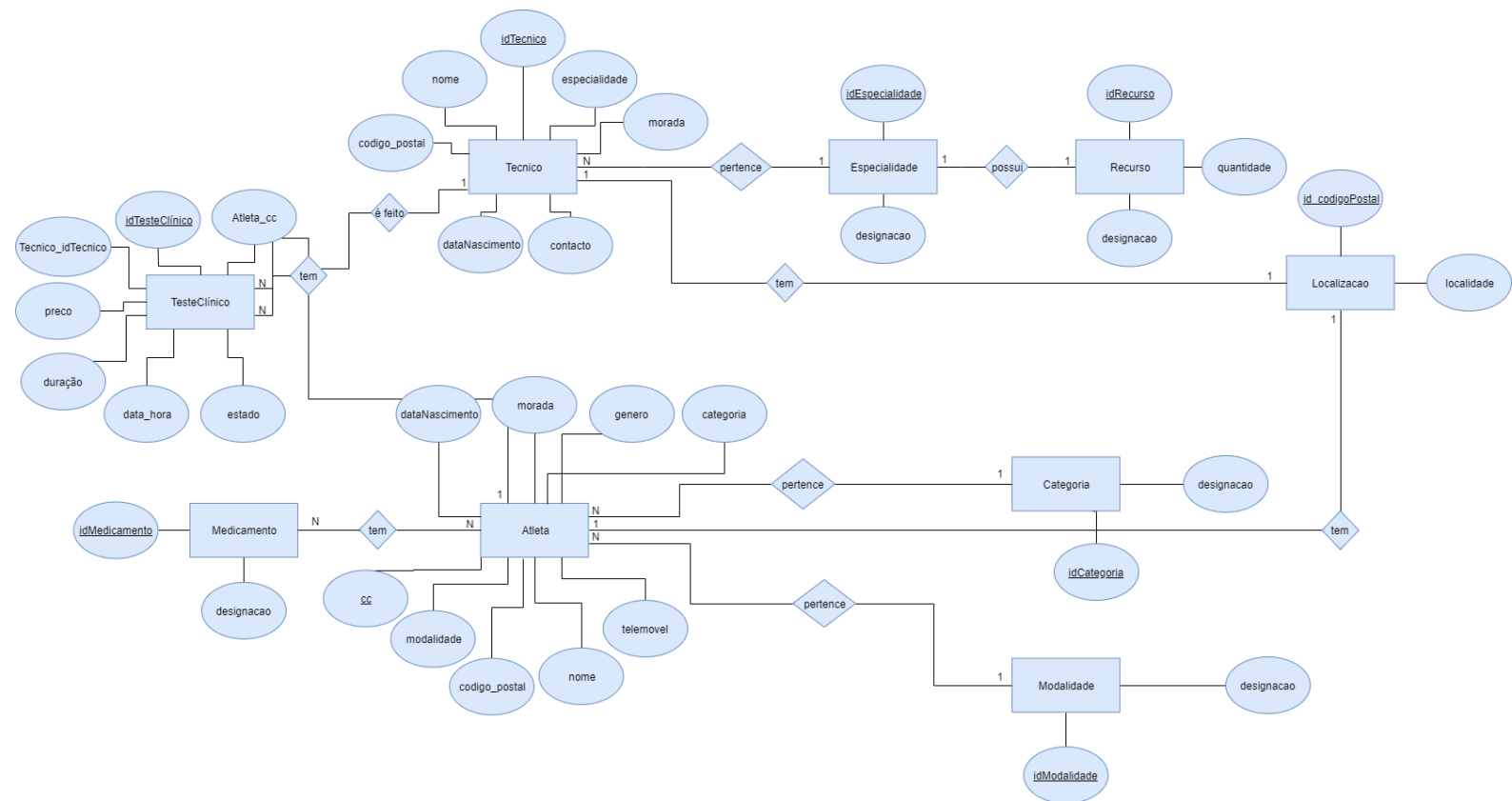


Figura 1: Modelo Conceptual

3.1 Identificação e caracterização das associações dos atributos com as entidades

Depois de analisados os requisitos chegamos a um conjunto de entidades que se apresentam nas seguintes tabelas.

Entidade	Atributos	Descrição	Dados
TesteClinico	idTesteClinico	Identifica unicamente um teste clínico	Inteiro
	data_hora	Data e hora do teste clínico	DATETIME
	preco	Preço do teste clínico	Double(4,2)
	estado	Estado do teste clínico (se já se realizou ou não, ou se foi cancelado)	Variável até 45 caracteres
	idTecnico	Identifica o técnico que realizou o teste	Inteiro
	idAtleta	Identifica o atleta que realizou o teste	Inteiro
	duracao	Indica a duração de um teste clínico	Inteiro
Especialidade	idEspecialidade	Identifica unicamente a especialidade	Inteiro
	designacao	Designação da especialidade	Variável até 45 caracteres
Localização	id_codigoPostal	Identifica unicamente a localização a partir do código postal	Variável até 45 caracteres
	Localidade	Identifica a Localidade	Variável até 45 caracteres
Modalidade	idModalidade	Identifica unicamente a modalidade do atleta	Inteiro entre 1 e MAXINT
	designacao	Designação da modalidade	Variável até 45 caracteres
Categoria	idCategoria	Identifica unicamente a categoria do atleta	Inteiro
	designacao	Designação da categoria	Variável até 45 caracteres
Recurso	idRecurso	Identifica unicamente um recurso (equipamento)	Inteiro
	designacao	Designação do recurso	Variável até 45 caracteres
	quantidade	Quantidade de recursos existentes	Inteiro
	idEspecialidade	Identifica a especialidade que está interligada com o recurso	Inteiro

Entidade	Atributos	Descrição	Dados
Atleta	cc	Cartão de cidadão do atleta	Inteiro
	modalidade	Modalidade em que se encontra o atleta	Inteiro
	codigo_postal	Código postal do atleta	Variável até 45 caracteres
	nome	Nome do atleta	Variável até 45 caracteres
	telemovel	Número de telemóvel do atleta	Inteiro
	dataNascimento	Data de Nascimento do atleta	DATETIME
	morada	Morada do atleta	Variável até 45 caracteres
	genero	Género do atleta	Variável até 45 caracteres
	categoria	Categoria em que se encontra o atleta	Inteiro
Tecnico	idTecnico	Identifica unicamente um técnico	Inteiro
	nome	Nome do técnico	Variável até 45 caracteres
	codigo_postal	Código postal do técnico	Variável até 45 caracteres
	especialidade	Especialidade do técnico	Inteiro
	morada	Morada do técnico	Variável até 45 caracteres
	dataNascimento	Data de nascimento do técnico	DATETIME
	contacto	Contacto do técnico	Inteiro
Medicamento	idMedicamento	Identifica unicamente um medicamento	Inteiro
	idAtleta	Identificador do atleta que consome o medicamento	Inteiro

Tabela 1: Identificação e associação dos atributos com as entidades

4. Modelação lógica

4.1. Construção do modelo de dados lógico

Com vista a construir o modelo lógico do projeto, utilizamos o modelo conceptual para nos orientarmos. Apesar deste ser muito semelhante, existem algumas opções que têm de ser tomadas de forma independente para permitir a consistência e coerência nos dados. De seguida é especificado todo o processo:

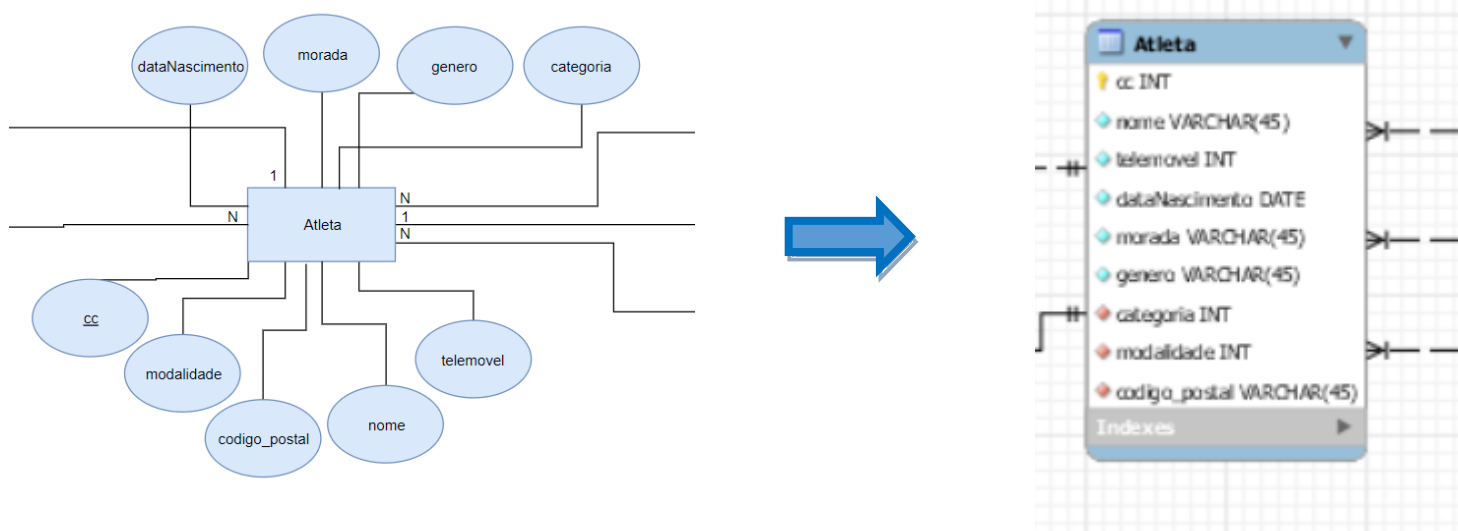


Figura 2: Passagem da entidade Atleta para o modelo lógico

No que diz respeito ao Atleta, o seu cc representa a chave primária e será representado como um inteiro, pois este representa o número do cartão de cidadão de um determinado atleta. Como é uma chave primária nunca poderá ter o valor nulo. O seu nome será representado com um *VARCHAR(45)*, visto que o nome de um atleta poderá ter bastantes caracteres, dá-se o limite de 45. No caso do atributo telemóvel, é também um inteiro pois representa o contacto telefónico do atleta. Relativamente à data de nascimento, esta é representada como uma *DATE* com o formato “YYYY-MM-DD” (ano, mês, dia), e pode variar entre “1000-01-01” até “9999-12-31”. À morada está associado um *VARCHAR(45)* para assim ser possível adicionar diversas informações sobre esta. Por último temos o género de um atleta

que é representado por um `VARCHAR(45)`, permitindo escrever o género com que cada atleta se identifica.

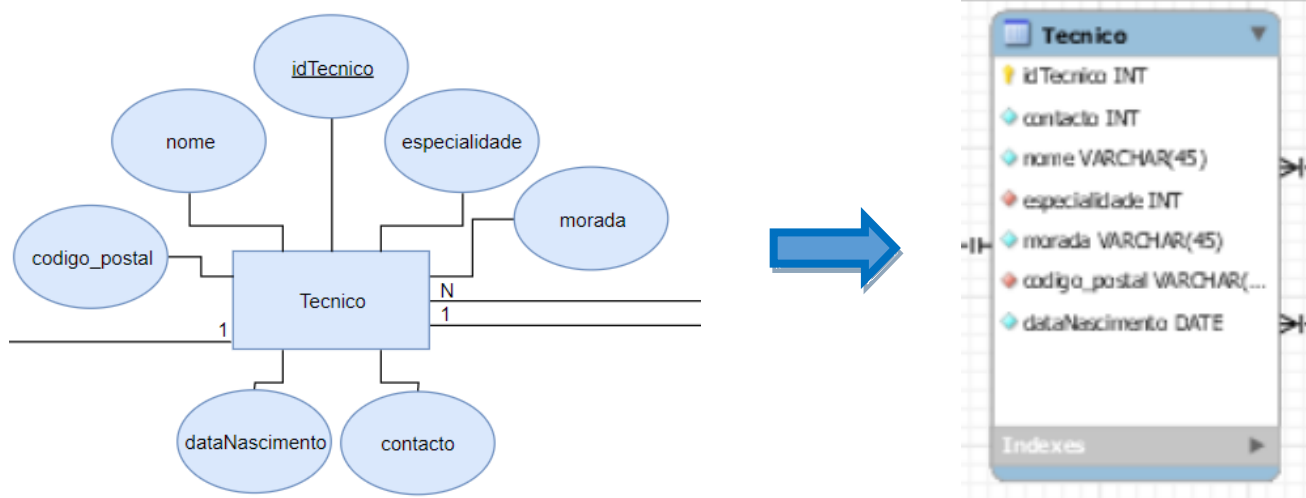


Figura 3: Passagem da entidade Tecnico para o modelo lógico

Já no técnico, temos o idTecnico que é também representado como um inteiro. Este será diferente para todos os técnicos pois é o seu identificador. É por isso a sua chave primária. O seu contacto é um inteiro pois identifica o número de telemóvel de um determinado técnico. No caso do nome e da morada, serão um `VARCHAR(45)` pois, como no caso do atleta, foi o que consideramos mais adequado. Por último, a data de Nascimento é também representada por uma `DATE`.

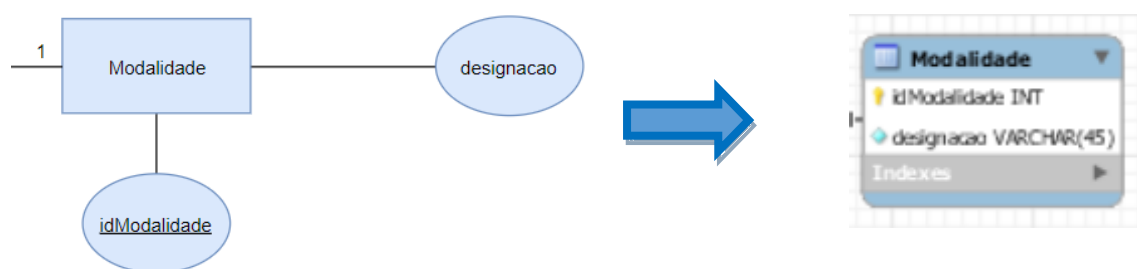


Figura 4: Passagem da entidade Modalidade para o modelo lógico

Para a modalidade, o idModalidade é representado como um inteiro. Este é a chave primária da modalidade e, no caso da nossa base de dados, pode tomar os valores de 1 até 6, pois cada um está associado a uma determinada designação. A designação é então representada por um `VARCHAR(45)` e pode ser: corrida de pista, associado ao id 1; corrida de

obstáculos associada ao id 2; salto em altura associado ao id 3; salto em comprimento associado ao id 4; lançamento associado ao id 5 e por último corrida de estafetas associado ao id 6.

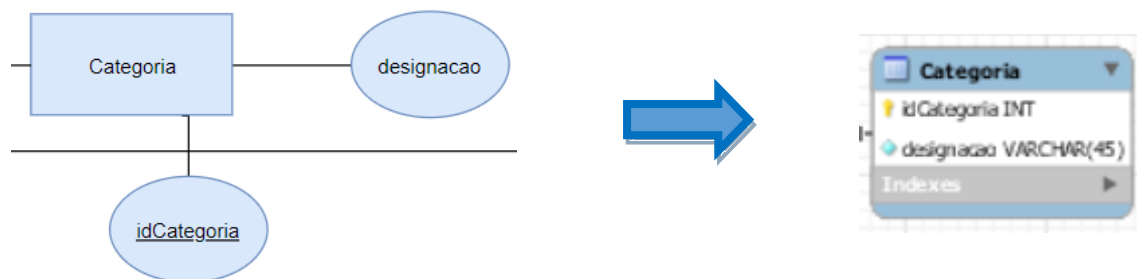


Figura 5: Passagem da entidade Categoria para o modelo lógico

Para a categoria, o idCategoria é representado como um inteiro. Este é a chave primária da categoria e, no caso da nossa base de dados, pode tomar os valores de 0 até 4, pois cada um está associado a uma determinada designação. A designação é então representada por um *VARCHAR(45)* e pode ser: infantis, associado ao id 0; iniciados, associado ao id 1; juniores associado ao id 2; seniores associados ao id 3 e por último veteranos associado ao id 4. Estas representam as categorias existentes para cada atleta.

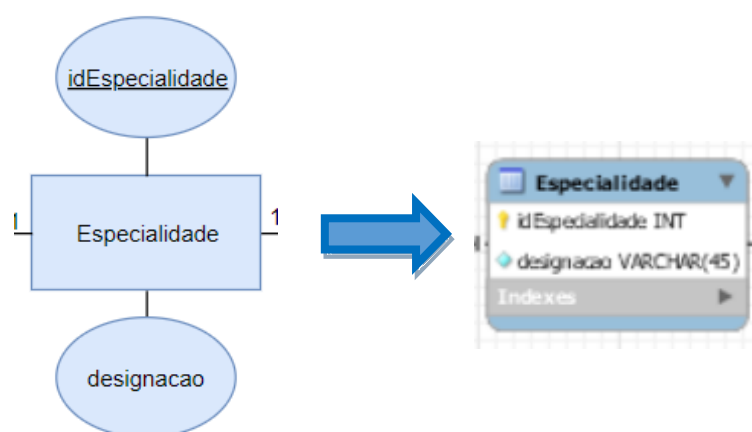


Figura 6: Passagem da entidade Especialidade para o modelo lógico

A especialidade tem como chave primária o idEspecialidade que é representado por um inteiro pois este será incrementado de acordo com o número de especialidades existentes nas clínicas. A designação é caracterizada como um `VARCHAR(45)` e identifica cada especialidade associada ao id.

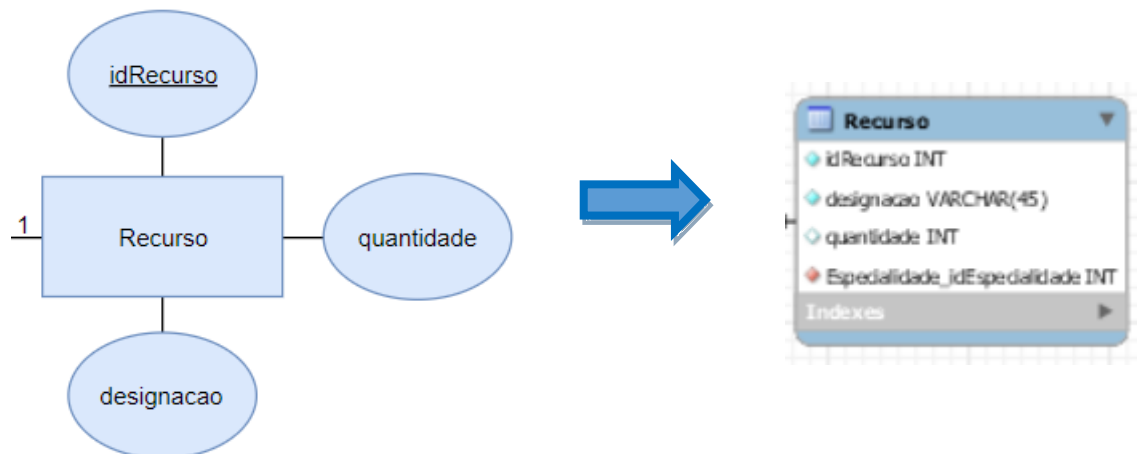


Figura 7: Passagem da entidade Recurso para o modelo lógico

No que diz respeito ao Recurso, o seu idRecurso representa a chave primária e é identificado como um inteiro. Este vai incrementando de acordo com o tipo de equipamentos distintos das diferentes especialidades. A designação é um `VARCHAR(45)` e indica para que especialidade são os equipamentos (por exemplo equipamentos de cardiologia). A quantidade é também um inteiro e indica o número de equipamentos existentes.

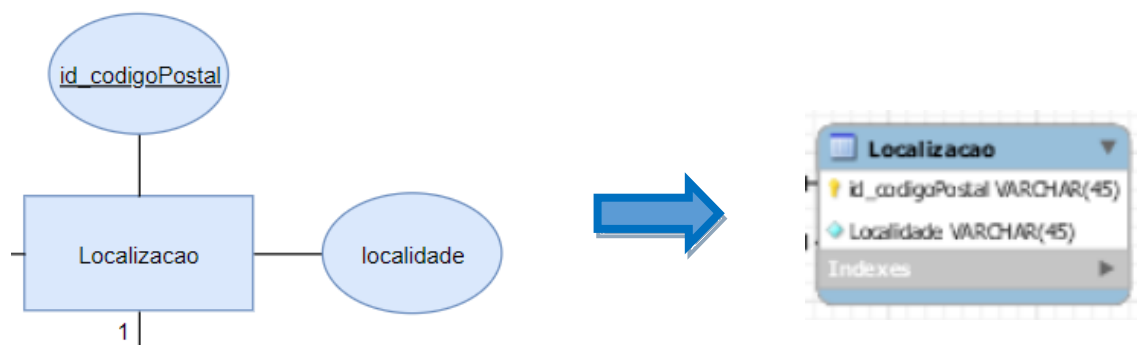


Figura 8: Passagem da entidade Localizacao para o modelo lógico

Para a Localização, o id_codigo_postal é um inteiro e identifica o código postal de cada localidade, é, portanto, a chave primária. A localidade é representada por um `VARCHAR(45)` e indica a localidade associada a cada código postal.

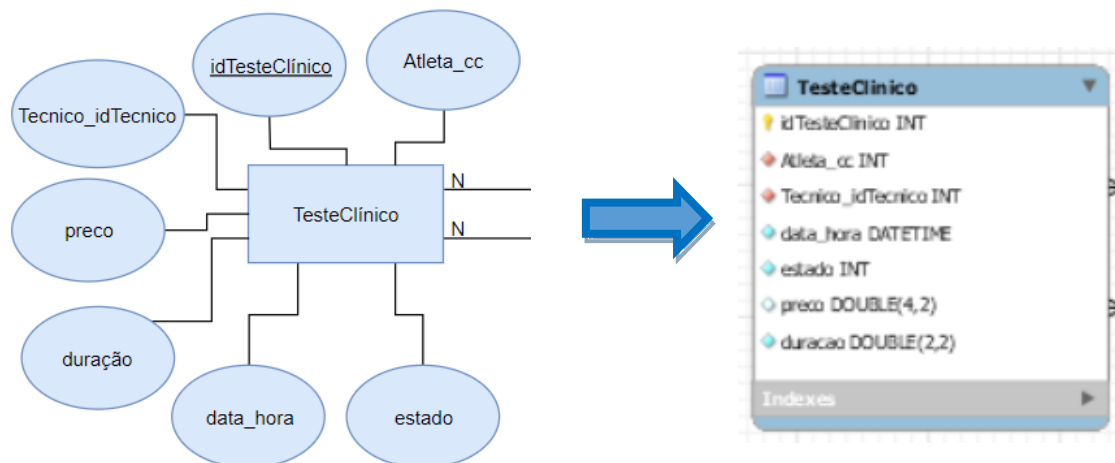


Figura 9: Passagem da entidade TesteClínico para o modelo lógico

O Teste Clínico tem como chave primária o idTesteClínico que é um inteiro que identifica unicamente cada teste. A sua data e hora será representada por um *DATETIME* com o formato “YYYY-MM-DD HH:MM:SS”, e pode variar entre “1000-01-01 00:00:00” até “9999-12-31 23:59:59”. O estado é um inteiro e no caso da nossa base de dados toma os seguintes valores: 1 se o teste clínico já tiver sido realizado, 2 caso ainda não foi realizado e 3 caso o atleta tenha cancelado o teste. O preço e a duração são representados como um *DOUBLE(4,2)* e como um *DOUBLE(2,2)* respetivamente, e ambos necessitam de uma parte decimal com 2 dígitos no máximo.

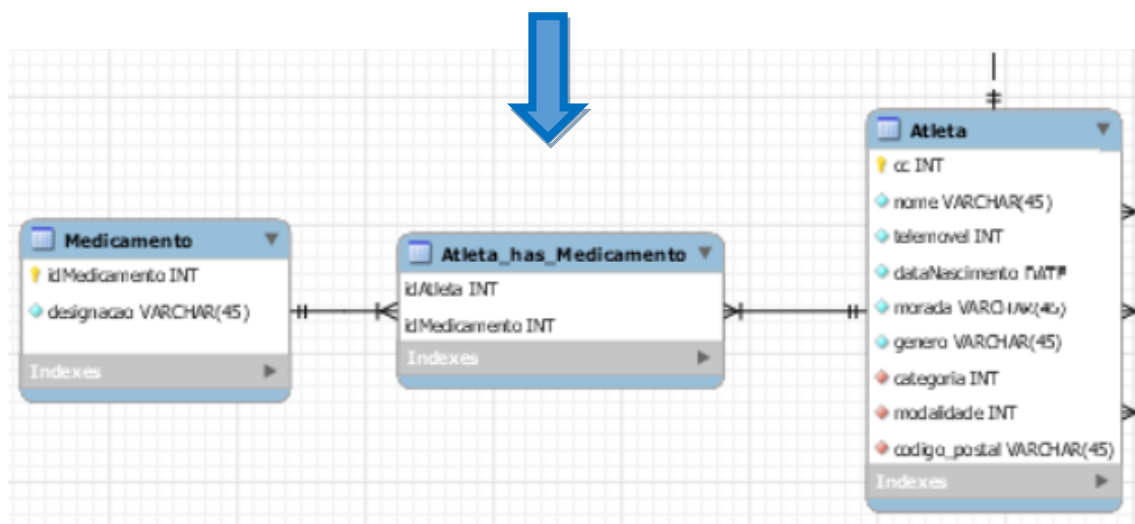
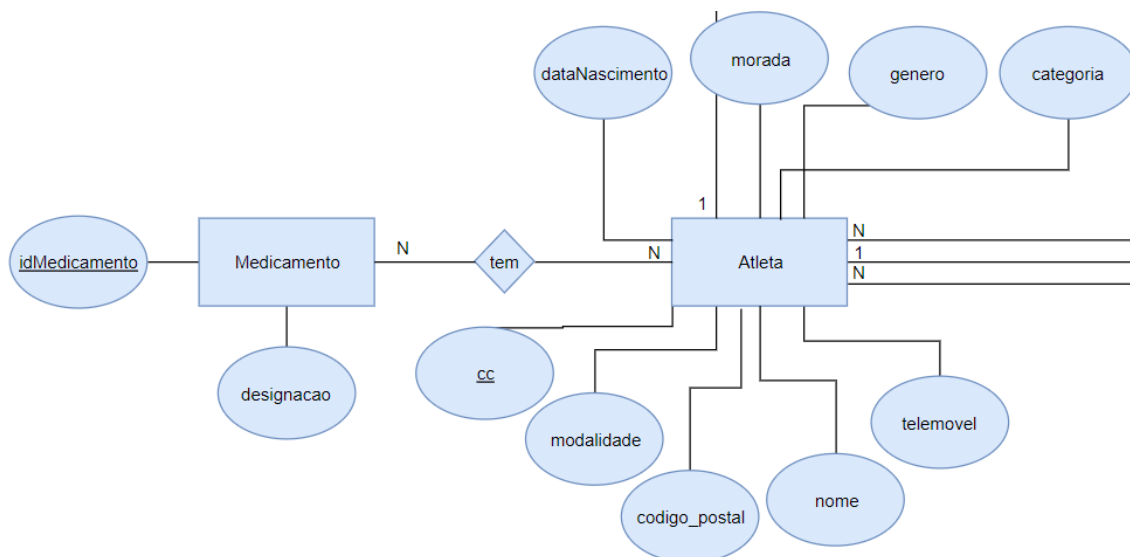


Figura 10: Passagem da relação de N para N e do atributo Medicamento para o modelo lógico

Como do Atleta para o Medicamento existe uma ligação de N para N é necessária a criação de uma nova tabela entre estas duas. É, portanto, criada a tabela Atleta_has_Medicamento. Esta possui tanto o identificador do Atleta como o identificador do Medicamento (chaves primárias das outras tabelas), ambos representados como inteiros. No medicamento, o atributo idMedicamento é um inteiro e a designação um VARCHAR(45) que indica o nome do medicamento registado.

4.2. Derivar relações a partir do modelo concetual

Primeiramente, com a análise do modelo conceptual retratado anteriormente, começa-se por derivar as relações nele presentes de forma a apresentar as entidades, relações e atributos.

Na transição do modelo conceptual para o modelo lógico começa-se por escolher as chaves primárias para cada entidade.

Nesta fase, em relacionamentos do tipo 1:*, a chave primária da entidade “pai” (1), é utilizada como chave estrangeira na entidade “filho” (*). Este é o caso referente à maior parte dos relacionamentos presentes no nosso modelo.

Os relacionamentos *: * originam uma nova entidade que inclui as chaves primárias de ambas as entidades. Como é o caso do relacionamento entre as entidades *Atleta* e *Medicamento*, formando assim a tabela *Atleta_has_Medicamento*.

Para o último tipo de relacionamento 1:1, optamos por escolher aleatoriamente qual a entidade que iria ficar com a chave estrangeira da outra.

De seguida é apresentada uma tabela com as entidades e as suas respetivas chaves primárias e estrangeiras:

Entidade	Atributos	Chave Primária	Chave Estrangeira
Atleta	cc nome telemovel dataNascimento morada genero, categoria modalidade codigo_Postal	cc	modalidade : Modalidade(idModalidade) categoria : Categoria(idCategoria) codigo_postal : Localizacao(id_CodigoPostal)
Tecnico	idTecnico, contacto nome especialidade morada codigo_postal dataNascimento	idTecnico	especialidade : Especialidade (idEspecialidade) codigo_postal : Localizacao(id_CodigoPostal)
TesteClinico	idTesteClinico Atleta_cc Tecnico_idTecnico data_hora estado preco duracao	idTecnico idAtleta idTesteClinico	Atleta_cc : Atleta(cc) Tecnico_idTecnico : Tecnico (idTecnico)

Localizacao	id_codigoPostal localidade	id_codigo_Posta l	-
Modalidade	idModalidade designacao	idModalidade	-
Categoria	idCategoria designacao	idCategoria	-
Especialidade	idEspecialidade designacao	idEspecialidade	-
Recurso		idRecurso	Especialidade_idEspecilaidade: Especialidade(id_Especilidade)
Medicamento	idMedicamento designacao	idMedicamento	-
Atleta_has_Medicamento	idAtleta idMedicamento	idAtleta idMedicamento	-

Tabela 2: Identificação das chaves primárias e estrangeiras

4.3. Validar relações usando a normalização

O processo de normalização, é um passo fundamental para a construção de um sistema de base de dados rigoroso.

Para isso, baseamo-nos nos três passos do processo de normalização.

1. First Normal Form (1NF)

A interseção entre cada linha e cada coluna (i.e. cada célula) deve conter um e um só valor.

2. Second Normal Form (2NF)

Todos os atributos que não sejam chave primária de uma tabela devem ser (totalmente) funcionalmente dependentes da chave primária dessa tabela.

3. Third Normal Form (3NF)

Nenhum atributo que não seja chave primária deve ser transitivamente dependente em relação à chave primária (i.e. todas as colunas de uma tabela devem apenas poder ser determinadas pela coluna da PK e não por qualquer outra coluna da tabela).

Como podemos verificar pela tabela (Tabela 2) o nosso sistema de base de dados vai de encontro ao que é pressuposto anteriormente.

4.4. Validação do Modelo com Interrogações do Utilizador

Na validação do modelo pretendemos assegurar que o mesmo suporta as interrogações por parte do utilizador, como tal usaremos álgebra relacional para demonstrar a metodologia usada. Asseguraremos assim que será possível efetuar as interrogações requisitadas.

1ª Interrogação: Pesquisa de Atletas de uma determinada modalidade/categoria

Atleta \bowtie (modalidade=idModalidade) $(\sigma(\text{designacao}='Corrida de estafetas') \text{ Modalidade})$

Atleta \bowtie (categoria=idCategoria) $(\sigma(\text{designacao}= 'Iniciados') \text{ Categoria})$

2ª Interrogação: Histórico de testes clínicos de um determinado atleta

$(\sigma(\text{data_hora}<\text{data}) \text{ TesteClinico} \bowtie (\text{Atleta_cc}=\text{cc}) (\sigma(\text{cc}= '17125602') \text{ Atleta}))$

3ª Interrogação: Ranking de atletas por faturação

$\sqcap \text{ Nome } (\text{DES } \tau (\text{sum}(\text{preco})) (\forall \text{Atleta } (\text{TesteClinico} \bowtie (\text{Atleta_cc}=\text{cc}) \text{ Atleta})))$

4ª Interrogação: Consultas que ocorrem no ano 2019

$\sigma(\text{year}(\text{data_hora})=2019) \text{ TesteClinico}$

4.5. Validação do Modelo com as Transações Estabelecidas

1ª Transação: Adicionar um Técnico/Atleta

A inserção de um novo Técnico/Atleta só é possível caso o id_Tecnico e o cc respetivos não constem no sistema. Aquando da inserção, estas serão as chaves primárias das respetivas entidades.

Caso os parâmetros referentes às chaves estrangeiras de cada um não estejam presentes nas respetivas tabelas das entidades, estes serão acrescentados nas mesmas.

2ª Transação: Registrar um novo teste clínico

Para registar um teste clínico terá de lhe ser fornecido o cc do Atleta que o pretende realizar assim como a especialidade e a data do mesmo assim e o local da Clinica onde este vai ser

realizado. Posteriormente será verificado se para estes parâmetros se encontram médicos disponíveis para realizar o teste clínico.

4.6. Desenho do Modelo Lógico

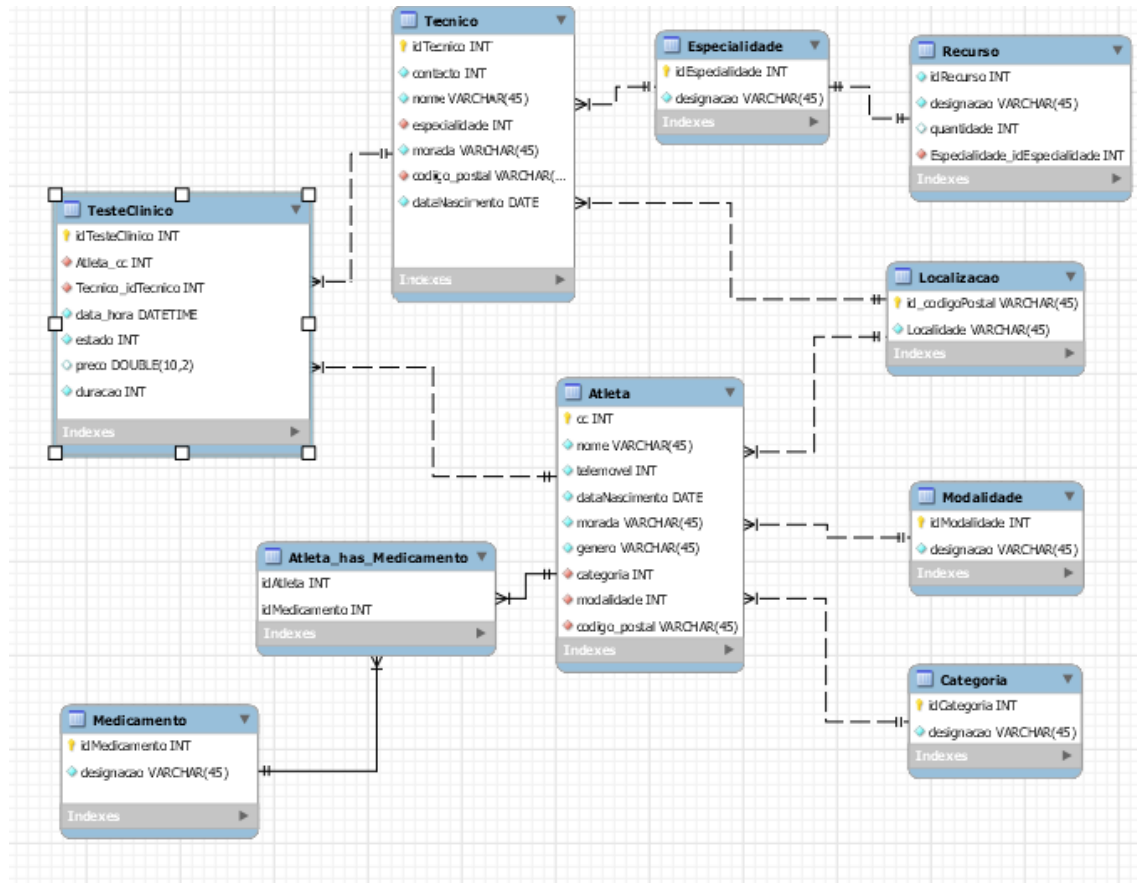


Figura 11: Modelo lógico

4.7. Rever modelo lógico com o utilizador

Foi marcado mais um encontro com o Sr. Rui, onde foi discutido o modelo lógico obtido e o modo como seriam satisfeitos os requisitos pedidos e explicado como se processariam cada uma das interrogações e transações. Este mostrou-se bastante satisfeito dando-nos permissão para continuar com o projeto.

5. Implementação Física

5.1. Seleção do sistema de gestão de base de dados

Para o desenvolvimento deste projeto na parte relacional optamos por usar o MySQL. Isto deve-se ao facto de este ser uma ferramenta bastante acessível e das mais populares a nível de gerenciamento de base de dados. Possibilita a escrita das nossas instruções SQL no software *MySQL WORKBENCH*, o que permite ir visualizando ao mesmo tempo o modelo lógico, tornando assim mais fácil relacionar as interrogações que pretendíamos escrever.

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

Com a vista a desenvolver o modelo físico, foi utilizada a ferramenta de *Forward Engineering* disponibilizada pelo MySQL. Desta forma, o código foi gerado de maneira automática. Em seguida é apresentado o código que foi criado.

- Categoria:

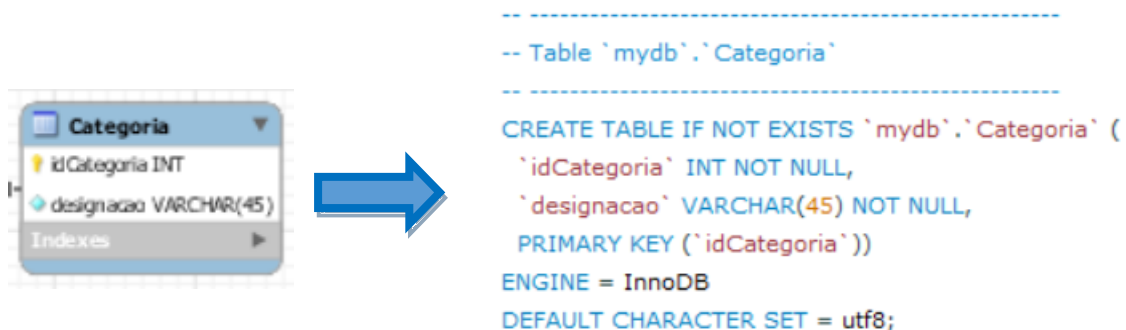
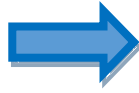
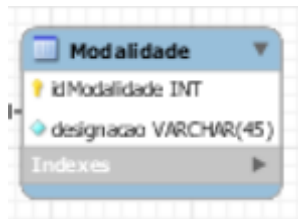


Figura 12: Tabela Categoria no modelo físico

- Modalidade

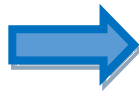
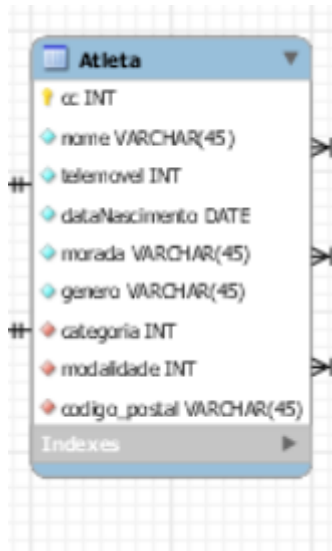


```
-- Table `mydb`.`Modalidade`

CREATE TABLE IF NOT EXISTS `mydb`.`Modalidade` (
  `idModalidade` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idModalidade`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

Figura 13: Tabela Modalidade no modelo físico

- Atleta

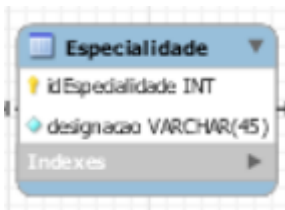


```
-- Table `mydb`.`Atleta`

CREATE TABLE IF NOT EXISTS `mydb`.`Atleta` (
  `cc` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `telemovel` INT NOT NULL,
  `dataNascimento` DATE NOT NULL,
  `morada` VARCHAR(45) NOT NULL,
  `genero` VARCHAR(45) NOT NULL,
  `categoria` INT NOT NULL,
  `modalidade` INT NOT NULL,
  `codigo_postal` VARCHAR(45) NOT NULL,
  INDEX `fk_Atleta_2_idx` (`categoria` ASC) VISIBLE,
  INDEX `fk_Atleta_3_idx` (`modalidade` ASC) VISIBLE,
  INDEX `fk_Atleta_1_idx` (`codigo_postal` ASC) VISIBLE,
  PRIMARY KEY (`cc`),
  CONSTRAINT `fk_Atleta_1`
    FOREIGN KEY (`codigo_postal`)
    REFERENCES `mydb`.`Localizacao` (`id_codigoPostal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Atleta_2`
    FOREIGN KEY (`categoria`)
    REFERENCES `mydb`.`Categoria` (`idCategoria`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Atleta_3`
    FOREIGN KEY (`modalidade`)
    REFERENCES `mydb`.`Modalidade` (`idModalidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

Figura 14: Tabela Atleta no modelo físico

- Especialidade

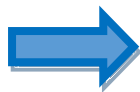


```
-- Table `mydb`.`Especialidade`

CREATE TABLE IF NOT EXISTS `mydb`.`Especialidade` (
  `idEspecialidade` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idEspecialidade`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

Figura 15: Tabela Especialidade no modelo físico

- Técnico:



```
-- Table `mydb`.`Tecnico`

CREATE TABLE IF NOT EXISTS `mydb`.`Tecnico` (
  `idTecnico` INT NOT NULL,
  `contacto` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `especialidade` INT NOT NULL,
  `morada` VARCHAR(45) NOT NULL,
  `codigo_postal` VARCHAR(45) NOT NULL,
  `dataNascimento` DATE NOT NULL,
  PRIMARY KEY (`idTecnico`),
  INDEX `fk_Tecnico_1_idx` (`especialidade` ASC) VISIBLE,
  INDEX `fk_Tecnico_2_idx` (`codigo_postal` ASC) VISIBLE,
  CONSTRAINT `fk_Tecnico_1`
    FOREIGN KEY (`especialidade`)
    REFERENCES `mydb`.`Especialidade` (`idEspecialidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Tecnico_2`
    FOREIGN KEY (`codigo_postal`)
    REFERENCES `mydb`.`Localizacao` (`id_codigoPostal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 16: Tabela Tecnico no modelo físico

- Medicamento



```
-- Table `mydb`.`Medicamento`

CREATE TABLE IF NOT EXISTS `mydb`.`Medicamento` (
  `idMedicamento` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idMedicamento`))
ENGINE = InnoDB;
```

Figura 17: Tabela Medicamento no modelo físico

- Teste Clínico

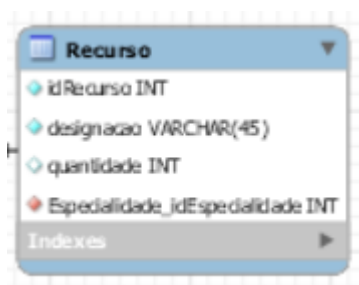


```
-- Table `mydb`.`TesteClinico`

CREATE TABLE IF NOT EXISTS `mydb`.`TesteClinico` (
  `idTesteClinico` INT NOT NULL,
  `Atleta_cc` INT NOT NULL,
  `Tecnico_idTecnico` INT NOT NULL,
  `data_hora` DATETIME NOT NULL,
  `estado` INT NOT NULL,
  `preco` DOUBLE(4,2) NULL,
  `duracao` DOUBLE(2,2) NOT NULL,
  PRIMARY KEY (`idTesteClinico`),
  INDEX `fk_TestesClinico_Tecnico1_idx` (`Tecnico_idTecnico` ASC) VISIBLE,
  INDEX `fk_TestesClinico_Atleta1_idx` (`Atleta_cc` ASC) VISIBLE,
  CONSTRAINT `fk_TestesClinico_Tecnico1`
    FOREIGN KEY (`Tecnico_idTecnico`)
      REFERENCES `mydb`.`Tecnico` (`idTecnico`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TestesClinico_Atleta1`
    FOREIGN KEY (`Atleta_cc`)
      REFERENCES `mydb`.`Atleta` (`cc`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 18: Tabela TesteClinico no modelo físico

- Recurso:



```
-- Table `mydb`.`Recurso`

CREATE TABLE IF NOT EXISTS `mydb`.`Recurso` (
  `idRecurso` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
  `quantidade` INT NULL,
  `Especialidade_idEspecialidade` INT NOT NULL,
  INDEX `fk_Recurso_Especialidade1_idx` (`Especialidade_idEspecialidade` ASC) VISIBLE,
  CONSTRAINT `fk_Recurso_Especialidade1`
    FOREIGN KEY (`Especialidade_idEspecialidade`)
      REFERENCES `mydb`.`Especialidade` (`idEspecialidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

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

Figura 19: Tabela Recurso no modelo físico

- **Atleta_has_Medicamento**

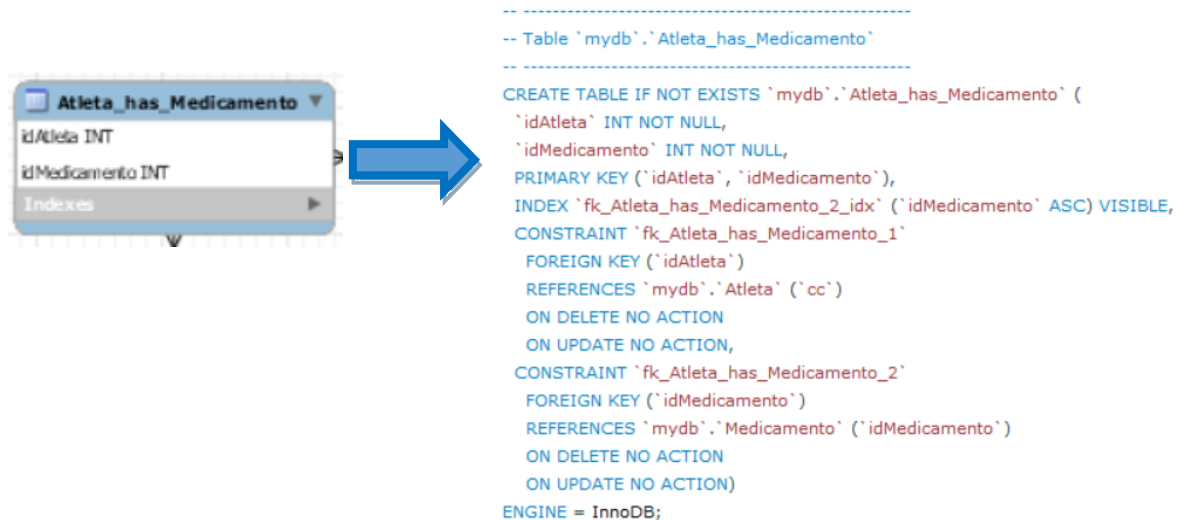


Figura 20: Tabela Atleta_has_Medicamento no modelo físico

5.3. Tradução de alguns dos requisitos do utilizador para o SQL

Após a povoação da base de dados, implementamos os requisitos necessários em *MySQL*. No sentido de apenas apresentarmos alguns requisitos, selecionamos os que consideramos mais importantes.

- **Passagem do requisito de exploração um (RE1) para o *MySQL*:**

```
select a.nome as nomeAtleta, (select m.designacao from Modalidade m
where a.modalidade=m.idModalidade) as modalidade
from Atleta a
order by a.nome;
```

Figura 21: Listagem de atletas por modalidade

#	nomeAtleta	modalidade
1	Abel Quintanilha Brião	Corrida de obstaculos
2	Adonal Morales Bento	Salto em Comprimento
3	Alícia Faleiro Carvalhal	Salto em Altura
4	Breno Condorcet Naves	Corrida de obstaculos
5	Felipe Alvelos Pinto	Lançamento
6	Gabriella Estrada Portela	Corrida de estafetas
7	Ícaro Simões Santarém	Corrida de obstaculos
8	Ivo Brito Doutis	Corrida de Pista
9	Julian Santos Araujo	Lançamento
10	Kauã Azevedo Barbosa	Corrida de Pista
11	Leila Martins Cardoso	Salto em Altura

12	Luciana Cancela	Salto em Altura
13	Luz Milhães Penteado	Corrida de Pista
14	Melissa Martins Cunha	Corrida de Pista
15	Messias Lamenha Bouça	Salto em Comprimento
16	Nancy Curado Rico	Salto em Altura
17	Paulo Negromonte Ma...	Corrida de estafetas
18	Seerat Moura Valverde	Corrida de Pista
19	Valdemar Brás Maciel	Corrida de Pista
20	Yangchen Cortesão F...	Lançamento
21	Yanni Feltosa Amaro	Salto em Comprimento
22	Yara César Rua	Salto em Altura

Figura 22: Resultado obtido de RE1

- Passagem do requisito de exploração dois (RE2) para o *MySQL*:

```
select a.nome as NomeAtleta, (select l.Localidade
from Localizacao l
where a.codigo_postal=l.id_codigoPostal) as Localidade
from Atleta a;
```

Figura 23: Atletas listados por localidade

#	NomeAtleta	Localidade
1	Abel Quintanilha Brião	FARO
2	Valdemar Brás Maciel	FARO
3	Luz Milhães Penteado	FARO
4	Felipe Alvelos Pinto	BRAGA
5	Messias Lamenha Bouça	LISBOA
6	Paulo Negromonte Marcondes	BARCELOS
7	Gabriella Estrada Portela	BARCELOS
8	Adonal Morales Bento	BARCELOS
9	Yara César Rua	LISBOA
10	Ivo Brito Doutis	SETUBAL
11	Ícaro Simões Santarém	BRAGA

12	Julian Santos Araujo	BRAGA
13	Yanni Feltosa Amaro	SETUBAL
14	Seerat Moura Valverde	FARO
15	Breno Condorcet Naves	LISBOA
16	Kauã Azevedo Barbosa	BRAGA
17	Nancy Curado Rico	SETUBAL
18	Yangchen Cortesão Ferraço	LISBOA
19	Leila Martins Cardoso	BRAGA
20	Melissa Martins Cunha	BRAGA
21	Alícia Faleiro Carvalhal	BRAGA
22	Luciana Cancela	SETUBAL

Figura 24: Resultado obtido de RE2

- Passagem do requisito de exploração três (RE3) para o *MySQL*:

```
select a.nome as NomeAtleta ,tec.nome as NomeTecnico, t.estado, year(t.data_hora) as ano
from TesteClinico t, Atleta a, Tecnico tec
where a.cc=t.Atleta_cc
and tec.idTecnico = t.Tecnico_idTecnico
order by year(t.data_hora);
```

Figura 25: Listagem de dados referentes aos testes clínicos bem como o seu respetivo ano.

#	NomeAtleta	NomeTecnico	estado	ano
1	Alicia Faleiro Carvalho	Adam Salgueiro Durão	1	2012
2	Lella Martins Cardoso	Adam Salgueiro Durão	1	2012
3	Ícaro Simões Santarém	Ivan Almada Ferraço	1	2012
4	Yangchen Cortesão Ferraço	Tiara Parracho Lagoa	1	2012
5	Seerat Moura Valverde	Ivan Almada Ferraço	1	2013
6	Valdemar Brás Maciel	Ivan Almada Ferraço	1	2014
7	Alicia Faleiro Carvalho	Adam Salgueiro Durão	1	2015
8	Kauã Azevedo Barbosa	Ivan Almada Ferraço	1	2015
9	Yangchen Cortesão Ferraço	Assunção Cantanhede Matosinhos	1	2016
10	Ivo Brito Doutis	Assunção Cantanhede Matosinhos	1	2016
11	Ícaro Simões Santarém	Ivan Almada Ferraço	3	2017
12	Kauã Azevedo Barbosa	Viviane Correia Botica	1	2017
13	Luz Milhães Penteadó	Raphael Pinho Junqueira	1	2017
14	Julian Santos Araujo	Viviane Correia Botica	3	2017
15	Alicia Faleiro Carvalho	Tiara Parracho Lagoa	1	2017
16	Kauã Azevedo Barbosa	Adam Salgueiro Durão	1	2017
17	Breno Condorcet Naves	Damien Arantes Sabrosa	1	2018
18	Kauã Azevedo Barbosa	Raphael Pinho Junqueira	2	2020
19	Ivo Brito Doutis	Raphael Pinho Junqueira	2	2020
20	Felipe Alvelos Pinto	Adam Salgueiro Durão	2	2020
21	Julian Santos Araujo	Viviane Correia Botica	2	2020
22	Valdemar Brás Maciel	Raphael Pinho Junqueira	2	2020
23	Breno Condorcet Naves	Damien Arantes Sabrosa	2	2020
24	Seerat Moura Valverde	Tiara Parracho Lagoa	2	2020
25	Kauã Azevedo Barbosa	Raphael Pinho Junqueira	2	2020
26	Julian Santos Araujo	Viviane Correia Botica	2	2020
27	Ivo Brito Doutis	Raphael Pinho Junqueira	2	2020
28	Lella Martins Cardoso	Adam Salgueiro Durão	2	2020
29	Yangchen Cortesão Ferraço	Assunção Cantanhede Matosinhos	2	2020
30	Seerat Moura Valverde	Raphael Pinho Junqueira	2	2020
31	Messias Lamenha Bouça	Damien Arantes Sabrosa	2	2020

Figura 26: Resultado obtido RE3

- Passagem do requisito de exploração quatro (RE4) para o *MySQL*:

```
select m.designacao as modalidade,(select sum(t.preco)
from Atleta a, TesteClinico t
where a.modalidade=m.idModalidade
and t.Atleta_cc=a.cc) AS totalFaturado
from Modalidade m
order by m.designacao;
```

Figura 27: Quantia gasta por cada modalidade

#	modalidade	totalFaturado
1	Corrida de estafetas	NULL
2	Corrida de obstaculos	730.00
3	Corrida de Pista	2188.00
4	Lançamento	1166.00
5	Salto em Altura	491.00
6	Salto em Comprimento	212.00

Figura 28: Resultado obtido RE4

- Passagem do requisito de exploração cinco (RE5) para o *MySQL*:

```
DELIMITER //
create procedure `testePorAtleta` (nome_atleta VARCHAR(45))
begin
select a.nome , c.preco,c.data_hora,e.designacao from atleta a,testeclinico c,tecnico t,especialidade e
where nome_atleta=a.nome and a.cc=c.Atleta_cc and t.idTecnico=c.Tecnico_idTecnico and t.especialidade=e.idEspecialidade
order by c.data_hora;
end //
DELIMITER ;

call testePorAtleta ("Alícia Faleiro Carvalho");
```

Figura 29: Consultas para um determinado atleta

	nome	preco	data_hora	designacao
▶	Alícia Faleiro Carvalho	63.00	2012-04-07 18:27:00	Neurologia
	Alícia Faleiro Carvalho	108.00	2015-12-05 08:48:00	Neurologia
	Alícia Faleiro Carvalho	182.00	2017-10-07 11:55:00	Clinica Geral

Figura 30: Resultado obtido RE5

- Passagem do requisito de exploração seis (RE6) para o *MySQL*:

```
select e.designacao as Especialidade, (select count(*) from tecnico t, testeclinico tc
where tc.Tecnico_idTecnico=t.idTecnico and t.especialidade=e.idEspecialidade) as numConsultas
from especialidade e
group by e.designacao
order by numConsultas desc;
```

Figura 31: Ranking do número de consultas para cada especialidade

	Especialidade	numConsultas
►	Oftalmologia	9
	Cardiologia	8
	Clinica Geral	7
	Neurologia	6

Figura 32: Resultado obtido RE6

- **Passagem do requisito de exploração sete (RE7) para o MySQL:**

```
select e.designacao as Especialidade, (select sum(tc.preco) from tecnico t, testeclinico tc
where tc.Tecnico_idTecnico=t.idTecnico and t.especialidade=e.idEspecialidade) as faturacao
from especialidade e
group by e.designacao
order by faturacao desc;
```

Figura 33: Ranking da faturação por especialidade

Especialidade	faturacao
Cardiologia	1402.00
Clinica Geral	1352.00
Oftalmologia	1258.00
Neurologia	532.00

Figura 34: Resultado obtido RE7

- **Passagem do requisito de exploração oito (RE8) para o MySQL:**

```
select distinct a.cc as N°cc,a.nome as Nome from atleta a
where exists (select * from testeclinico ct where ct.Atleta_cc=a.cc and ct.estado=3);
```

Figura 35: Atletas que faltaram a consultas

N°cc	Nome
14925884	Ícaro Simões Santarém
15020247	Julian Santos Araujo

Figura 36: Resultado obtido (RE7)

- Passagem do requisito de exploração nove (RE9) para o *MySQL*:

```
SELECT
  (SELECT
    COUNT(*)
  FROM
    TesteClinico c
  WHERE
    m.idTecnico = c.Tecnico_idTecnico) AS ranking,
  m.nome
FROM
  Tecnico m
ORDER BY ranking;
```

Figura 37: Ranking dos médicos pelo número de consultas

#	ranking	nome
	2	Luana Espargosa Ribeiro
	2	Ariel Esparteiro Ramalho
	3	Charlotte Vasconcelos Tristão
	3	Assunção Cantanhede Matosinhos
	3	Tiara Parracho Lagoa
	4	Viviane Correia Botica
	4	Damien Arantes Sabrosa
	5	Ivan Almada Ferraço
	6	Raphael Pinho Junqueira
	6	Adam Salgueiro Durão

Figura 38: Resultado obtido (RE9)

- Passagem do requisito de exploração dez (RE10) para o MySQL:

```

DELIMITER //
create function `existeMedico`(idEspecial Int(11), datinha Varchar(45)) returns boolean
READS SQL DATA
DETERMINISTIC

begin

declare numero_medicos_livres int(11) default 0;

SELECT
    COUNT(m.idTecnico) into numero_medicos_livres
FROM
    Tecnico m
WHERE m.especialidade =idEspecial and
    m.idTecnico NOT IN (SELECT
        a.idTecnico
    FROM
        Tecnico a,
        TesteClinico u
    WHERE
        a.especialidade = idEspecial
        AND u.Tecnico_idTecnico = a.idTecnico
        AND u.data_hora = datinha);

IF numero_medicos_livres = 0 THEN RETURN FALSE;
ELSE RETURN TRUE;
END IF;

END //
DELIMITER ;

/*procedura*/
DELIMITER //
create procedure `fazInsert`(in idAt Int(11) ,in datinha Varchar(45),in especial Int(11))
READS SQL DATA
DETERMINISTIC

begin
declare resultado varchar(45) default "nada";

IF existeMedico(especial,datinha) then begin
INSERT into mydb.TesteClinico values (
(select (FLOOR(RAND()*(999999-100000+1))+100000) as aa from TesteClinico m where "aa" NOT in (SELECT m.idTesteClinico FROM TesteClinico m) limit 1)
,idAt,
(select m.idTecnico FROM
Tecnico m
WHERE m.especialidade =especial and
m.idTecnico NOT IN (SELECT
a.idTecnico
FROM
Tecnico a,
TesteClinico u
WHERE
a.especialidade = especial
AND u.Tecnico_idTecnico = a.idTecnico
AND u.data_hora = datinha) order by rand() limit 1),
datinha,1,FLOOR(RAND()*(200-50+1))+50,FLOOR(RAND()*(2-1+1))+1);

end;
end if;

END //
DELIMITER ;

/*call procedure*/
CALL fazInsert(12169329,'2017-11-03 11:54',2);

```

Figura 39: Agendamento de um teste clínico

- Passagem do requisito de exploração onze (RE11) para o *MySQL*:

```

3  DELIMITER //
4  • CREATE FUNCTION `temConsulta` (id_at INT(11), dataa varchar(45)) RETURNS BOOLEAN
5  READS SQL DATA
6  DETERMINISTIC
7
8  BEGIN
9
10 DECLARE numerC INT(11) DEFAULT 0;
11
12 SELECT COUNT(*) INTO numerC FROM mydb.TesteClinico con
13 WHERE con.Atleta_cc=id_at and con.data_hora=dataa;
14
15 IF numerC = 0 THEN RETURN False;
16 ELSE RETURN True;
17 END IF;
18
19 END //
20 DELIMITER ;
21 • /* EXECUTAR OPERACAO */
22 DELETE FROM mydb.TesteClinico
23 WHERE
24     TEMCONSULTAA(14756486, '2020-12-31 07:43:00');
25

```

Figura 40: Cancelamento de um teste clínico

- Passagem do requisito de exploração doze (RE12) para o *MySQL*:

```

DELIMITER //
create procedure `medAtleta` (atleta INT(11) )
begin
Select a.nome as Atleta ,m.designacao as Medicamento from atleta_has_Medicamento am, atleta a, Medicamento m
where am.idAtleta=a.cc and am.idMedicamento=m.idMedicamento and atleta=a.cc;
end //
DELIMITER ;

call medAtleta (14188128);

```

Figura 41: Listagem de medicamento de um dado atleta

Atleta	Medicamento
Paulo Negromonte Marcondes	Voltaren

Figura 42: Resultado obtido RE12

- Passagem do requisito de exploração treze (RE13) para o *MySQL*:

```
select a.nome,(select c.designacao
from Categoria c
where a.categoria=c.idCategoria)
as categoria from Atleta a;
```

Figura 43: Listagem de atletas por categoria

#	nome	categoria
1	Abel Quintanilha Brião	VETERANOS
2	Valdemar Brás Maciel	VETERANOS
3	Luz Milhães Penteado	INFANTIS
4	Felipe Alvelos Pinto	VETERANOS
5	Messias Lamenha Bouça	VETERANOS
6	Paulo Negromonte Marcondes	SENIORES
7	Gabriella Estrada Portela	VETERANOS
8	Adonal Morais Bento	JUNIORES
9	Yara César Rua	INICIADOS
10	Ivo Brito Doutis	SENIORES
11	Ícaro Simões Santarém	INFANTIS

12	Julian Santos Araujo	JUNIORES
13	Yanni Feitosa Amaro	SENIORES
14	Seerat Moura Valverde	JUNIORES
15	Breno Condorcet Naves	SENIORES
16	Kauã Azevedo Barbosa	SENIORES
17	Nancy Curado Rico	JUNIORES
18	Yangchen Cortesão Ferraço	VETERANOS
19	Leila Martins Cardoso	INFANTIS
20	Melissa Martins Cunha	VETERANOS
21	Alícia Faleiro Carvalhal	JUNIORES
22	Luciana Cancela	SENIORES

Figura 44: Resultado obtido RE13

- Passagem do requisito de exploração catorze (RE14) para o *MySQL*:

```
DELIMITER //
CREATE PROCEDURE `RecursoPorDia` (IN dia DATE)
BEGIN
select r.designacao, (select (count(*)/r.quantidade)*100 from especialidade e,testeClinico tc,tecnico t
where r.Especialidade_idEspecialidade=e.idEspecialidade
and e.idEspecialidade=t.especialidade
and t.idTecnico=tc.Tecnico_idTecnico
and date_format(tc.data_hora,'%Y-%m-%d')=dia) as perc from recurso r
group by r.designacao;
END //
DELIMITER ;
/* EXECUTAR PROCEDIMENTO */
CALL RecursoPorDia('2017-07-01') ;
```

Figura 45: Percentagem de equipamentos usados num certo dia

designacao	perc
Equip. Oftalmologia	33.3333
Equip. ClinicaGeral	0.0000
Equip. Cardiologia	0.0000
Equip. Neurologia	0.0000

Figura 46: Resultado obtido RE14

- **Passagem do requisito de exploração quinze (RE15) para o MySQL:**

```
delimiter //
create function `InsereCodigoPostal` (codPost VARCHAR(45),localidade VARCHAR(45) ) returns VARCHAR(45)
READS SQL DATA
DETERMINISTIC
begin
declare cpexists INT(4) default 0;
select count(*) into cpexists from localizacao l where l.id_codigoPostal=codPost;
if (cpexists=0) then insert into localizacao (id_codigoPostal, localidade) values (codPost,localidade);
return codPost;
end if;
end //
DELIMITER ;

delimiter //
create procedure `criaAtleta`(id_atleta INT(11), nome VARCHAR(45),tele INT, datN DATE, morada VARCHAR(45),genero
VARCHAR(45),cat VARCHAR(45),moda VARCHAR(45), codPost VARCHAR(45),localidade VARCHAR(45),cat VARCHAR(45),moda VARCHAR(45)

begin
declare cmexists INT(4) default 0;#verifica se a modalidade e a categoria são válidos
declare numero_atletas INT(11) DEFAULT 0; #verifica se o atleta (a partir do cc) já não está registado no sistema.

select count(*) into cmexists from categoria c,modalidade m where cat=c.idCategoria and moda=m.idModalidade;
select count(*) into numero_atletas from atleta a where a.cc=id_atleta;

if (numero_atletas = 0 and cmexists=1) then insert into atleta (cc, nome, telemovel, dataNascimento, morada, genero,
categoria, modalidade, codigo_Postal) values(id_atleta,nome,tele,datN,morada,genero,cat,moda,
InsereCodigoPostal(codPost,localidade) );
end if;
end //
DELIMITER ;

call criaAtleta(17125645,'Catarina',964871609,'1982-12-23','Rua Amorim','Feminino',3,1,'4000-002','Porto');
```

Figura 47: Adicionar um Atleta

5.4. Definições e caracterização dos mecanismos de segurança em SQL

De maneira a respeitar os requisitos de controlo definidos anteriormente, resolvemos criar dois tipos de utilizadores. O primeiro refere-se a um administrador (“admin”) que tem o privilégio de ter acesso completo à base de dados. O segundo, foi desenvolvido para qualquer técnico (“técnico”) e tem acesso a apenas algumas tabelas: Atleta, Recurso e TesteClinico. Apenas pode executar algumas instruções como o select e o update.

```
3 • create USER 'admin'@'localhost'
4     identified by 'admin';
5
6 • grant all privileges on mydb.*
7     to 'admin'@'localhost';
8
9 • create USER 'tecnico'@'localhost'
10    identified by 'tecnico';
11
12 • grant select, update on mydb.viewTecnico_TestesClinico to
13     'tecnico'@'localhost';
14
15 • grant select, update on mydb.viewTecnico_Recurso to
16     'tecnico'@'localhost';
17
18 • grant select, update on mydb.viewTecnico_Atleta to
19     'tecnico'@'localhost';
```

Figura 48: Criação de utilizadores no MySQL

5.5. Definição e caracterização das vistas de utilização em SQL

Devido à criação dos diferentes utilizadores foi necessária também a criação das vistas necessárias para que estes vejam apenas os pontos de que têm acesso.

```
22 • create view viewTecnico_TestesClinico as
23     select idTesteClinico, data_hora, estado, preco, duracao
24     from TesteClinico;
```

Figura 49: Vista do Técnico em relação à tabela Teste Clínico

```
26 • create view viewTecnico_Recurso as
27     select idRecurso, designacao, quantidade
28     from Recurso;
```

Figura 50: Vista do Técnico em relação à tabela Recurso

```
30 • create view viewTecnico_Atleta as
31     select cc, nome, telemovel, dataNascimento, morada, genero
32     from Atleta;
```

Figura 51: Vista do Técnico em relação à tabela Atleta

5.6. Revisão do sistema implementado com o utilizador

Após a apresentação do sistema final relacional ao Sr. Rui, este mostrou-se bastante satisfeito pois implementava tudo o que era necessário para o bom funcionamento da sua empresa de clínicas.

6. Base de Dados não relacional (*NoSQL*)

Nesta fase do projeto é necessária a conversão da base de dados relacional para uma base de dados não relacional. Para tal foi utilizado o software *Neo4J*. Com vista à construção desta, foi analisado novamente o modelo relacional desenvolvido anteriormente e procurada uma maneira para construir os grafos na plataforma. Por último, foram transformadas as interrogações desenvolvidas em *MySQL* e feita uma análise crítica relativamente aos dois tipos de modelos.

De forma a relacionar os nodos, utilizamos o seguinte comando:

```
MATCH (a:atleta), (mod:modalidade)
WHERE a.modalidade=mod.idModalidade
MERGE (a)-[:belongs]-(mod)
```

Dando origem ao seguinte grafo:

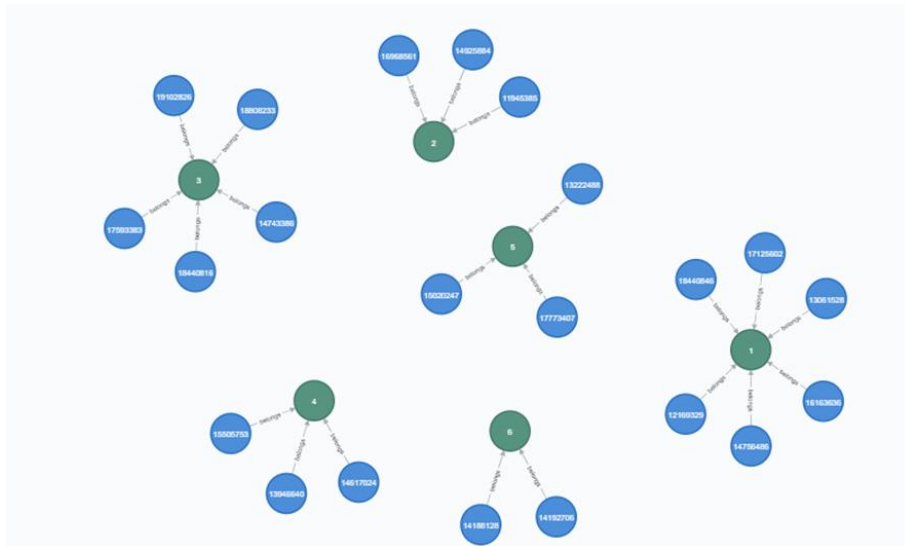


Figura 53: Grafo exemplo

Tal como explicado no exemplo anterior, é feita a mesma coisa para o resto das tabelas e devidas ligações, obtendo o grafo final.

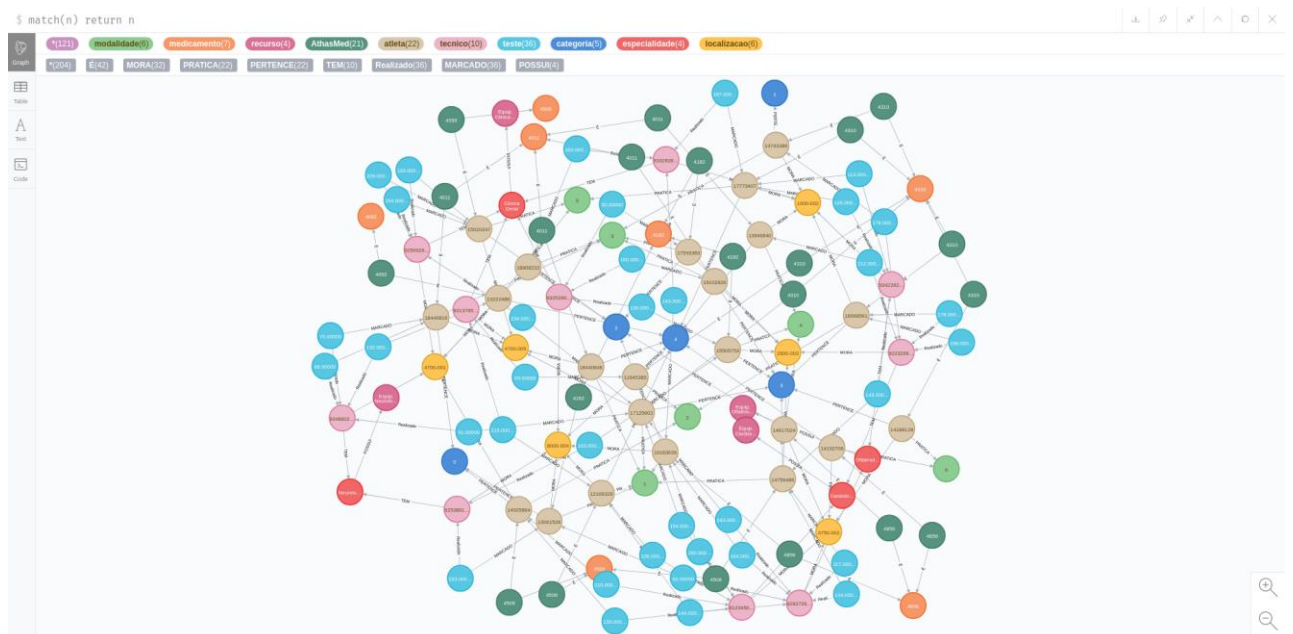


Figura 54: Grafo Final

8. Migração das interrogações para o modelo não relacional

Depois de elaboradas as interrogações em *SQL*, decidimos transformá-las em *Cypher* (linguagem utilizada em Neo4J) de maneira a que estas repondam de maneira semelhante ao modelo relacional. A seguir são apresentadas as queries desenvolvidas.

- **Passagem do requisito de exploração um (RE1) para o Neo4J:**

Comando:

```
match (a:atleta)-[p:pertence]->(m:modalidade) where a.modalidade=m.idModalidade
return a.nome as NomeAtleta ,m.designacao as Modalidade
```

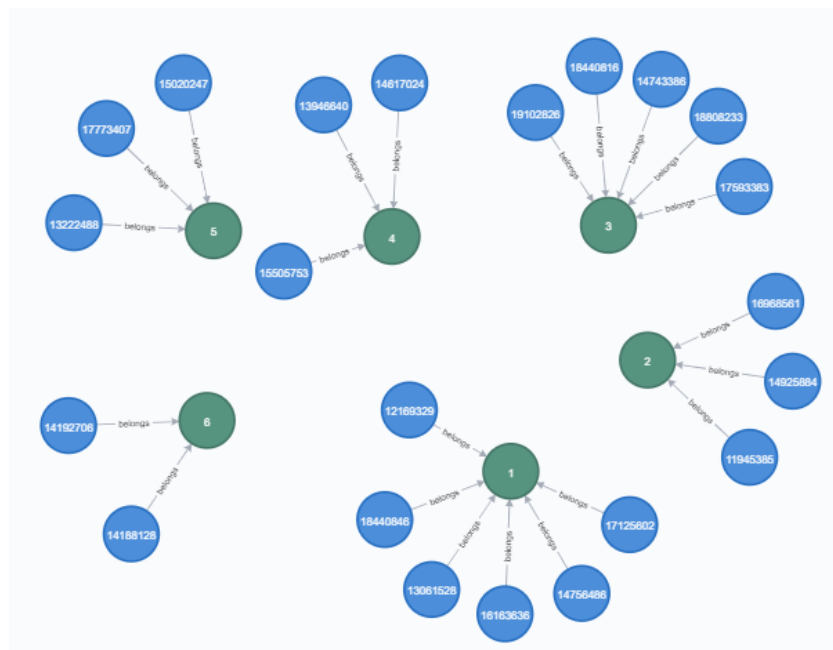


Figura 55: Atletas por modalidade

- **Passagem do requisito de exploração dois (RE2) para o Neo4J:**

Comando:

```
match (a:atleta)-[t:tem]->(l:localizacao) where a.codigo_postal=l.id_codigoPostal return
a.nome as NomeAtleta ,l.Localidade as Localidade
```

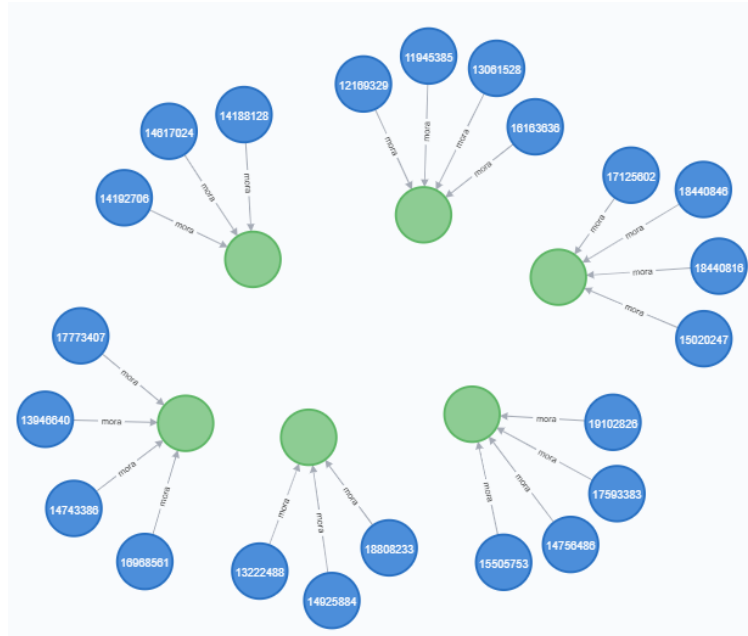


Figura 56: Atletas por localidade

- **Passagem do requisito de exploração três (RE3) para o Neo4J:**

Comando:

```
match(tec:tecnico)-[p:precisa]-(t:testeClinico)-[te:ter]-(a:atleta)where
t.Tecnico_idTecnico=tec.idTecnico and t.Atleta_cc=a.cc return a.nome as NomeAtleta,
tec.nome as NomeTecnico, t.estado as Estado, t.data_hora.year
```

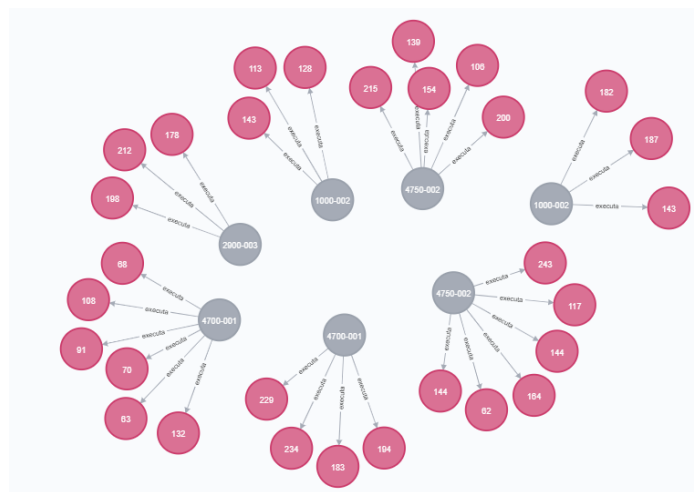


Figura 57: Dados de consulta por cada ano

- **Passagem do requisito de exploração quatro (RE4) para o Neo4J:**

Comando:

```
match (m:modalidade)->[p:pertence]-(a:atleta)->[te:ter]-(t:testeClinico) where
a.modalidade=m.idModalidade and a.cc=t.Atleta_cc set t.preco=toInt(t.preco) return distinct
m.designacao as Modalidade, sum(t.preco)
```

Modalidade	Faturacao
"Corrida de Pista"	1945
"Corrida de obstaculos"	730
"Salto em Altura"	491
"Salto em Comprimento"	212
"Lançamento"	1166

Figura 58: Quantia gasta por modalidade

- **Passagem do requisito de exploração cinco (RE5) para o Neo4J:**

Comando:

```
MATCH (a:atleta)-[r:marcou]-(tc:testeClinico) where date(tc.data_hora)>date() RETURN a, r, tc
```

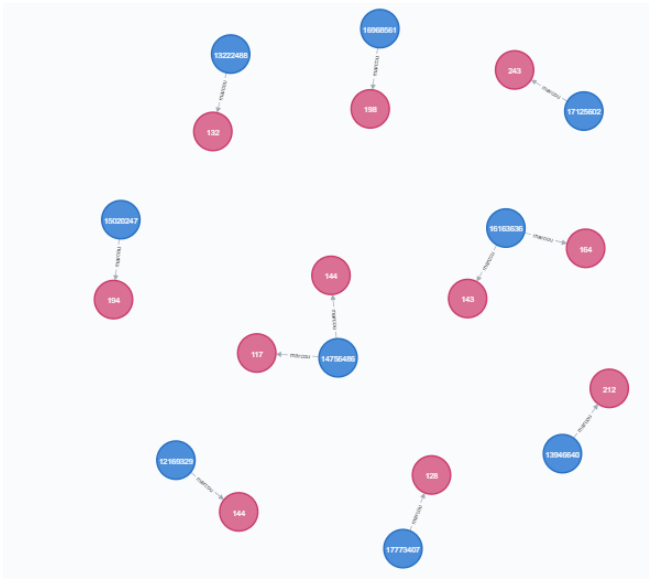


Figura 59: Próximas consultas para um atleta

- **Passagem do requisito de exploração seis (RE6) para o Neo4J:**

Comando:

```
MATCH (e:especialidade)-[r2:especializado]-(t:tecnico)-[r:executa]->(tc:testeClinico)
where e.idEspecialidade=t.especialidade and t.idTecnico=tc.Tecnico_idTecnico
RETURN e.designacao as Especialidade, count(tc) as nTestes
order by count(tc) desc
```

Especialidade	nTestes
"Cardiologia"	1402
"Clinica Geral"	1352
"Oftalmologia"	1258
"Neurologia"	532

Figura 60: Ranking de número de consultas por especialidade

- **Passagem do requisito de exploração sete (RE7) para o Neo4J:**

Comando:

```
MATCH (e:especialidade)-[r2:especializado]-(t:tecnico)-[r:executa]->(tc:testeClinico)
where e.idEspecialidade=t.especialidade and t.idTecnico=tc.Tecnico_idTecnico
set tc.preco=tolnt(tc.preco)
RETURN e.designacao as Especialidade, sum(tc.preco) as nTestes
order by nTestes desc
```

Especialidade	nTestes
"Cardiologia"	1402
"Clinica Geral"	1352
"Oftalmologia"	1258
"Neurologia"	532

Figura 61: Ranking de faturação por especialidade

- **Passagem do requisito de exploração oito (RE8) para o Neo4J:**

Comando:

```
MATCH (a:atleta)-[r:marcou]->(tc:testeClinico)
```

```
where a.cc=tc.Atleta_cc and tc.estado="3"
```

```
RETURN a.cc as cc,a.nome as Nome,count(*)as Nconsultas
```

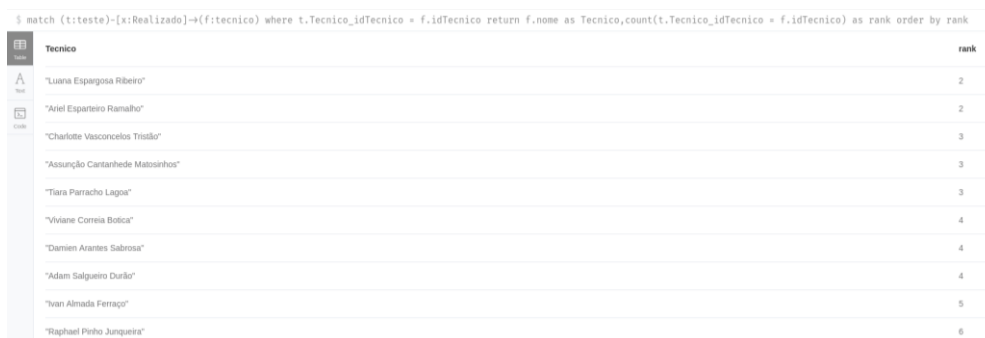
cc	Nome	Nconsultas
"15020247"	"Julian Santos Araujo"	1
"14925884"	"Ícaro Simões Santarém"	1

Figura 62: Atletas que faltaram a consultas

- **Passagem do requisito de exploração nove (RE9) para o Neo4J:**

Comando:

```
match (t:teste)-[x:Realizado]->(f:tecnico) where t.Tecnico_idTecnico = f.idTecnico return f.nome as Tecnico,count(t.Tecnico_idTecnico = f.idTecnico) as rank order by rank
```



The screenshot shows a Neo4J Cypher query editor with the following query: `$ match (t:teste)-[x:Realizado]->(f:tecnico) where t.Tecnico_idTecnico = f.idTecnico return f.nome as Tecnico,count(t.Tecnico_idTecnico = f.idTecnico) as rank order by rank`. Below the query, a table displays the results, listing the name of each technician and their corresponding rank (number of consultations).

Tecnico	rank
"Luana Espargosa Ribeiro"	2
"Ariel Esparteiro Ramalho"	2
"Charlotte Vasconcelos Tristão"	3
"Assunção Cantanhede Matosinhos"	3
"Tiara Parracho Lagoa"	3
"Viviane Correia Bolica"	4
"Damien Arantes Sabrosa"	4
"Adem Salgueiro Durião"	4
"Ivan Almada Ferraco"	5
"Raphael Pinho Junqueira"	6

Figura 63: Ranking do número de consultas por técnico

- **Passagem do requisito de exploração dez (RE10) para o Neo4J:**

Comando:

```
match (t:tecnico)-[:Realiza]->(f:teste) where t.especialidade = '3' and (not (f.data_hora)='2014-04-04 10:33:00') match (l:tecnico)-[:Realiza]->(q:teste) where l.especialidade = '3' and ((q.data_hora)='2014-04-04 10:33:00')) with distinct t.idTecnico AS v, l.idTecnico as s FOREACH ( ignoreMe in CASE WHEN v<>s THEN [1] ELSE [] END | create(t:teste {idTesteClinico:rand()*999999,Atleta_cc : '12371208',data_hora: '2014-04-04 10:33:00', Tecnico_idTecnico :v,preco:rand()*200,duracao:rand()*2}))
```



Figura 64: Agendar um teste clínico

- **Passagem do requisito de exploração onze (RE11) para o Neo4J:**

Comando:

```
match (t:teste) where t.Atleta_cc = '18808233' and t.data_hora = '2012-04-07 18:27:00' detach delete t
```



Figura 65: Cancelar teste clínico

- **Passagem do requisito de exploração doze (RE12) para o Neo4J:**

Comando:

```
match (m:AthasMed)-[c:É]->(h:medicamento) where m.idAtleta='18440816' and m.idMedicamento =h.idMedicamento return h.designacao as Medicamento
```



Figura 66: Listagem de medicamentos de um dado atleta

- **Passagem do requisito de exploração treze (RE13) para o Neo4J:**

Comando:

```
match (a:atleta)-[t:tem]->(c:categoria) where a.categoria=c.idCategoria return a.nome, c.designacao
```

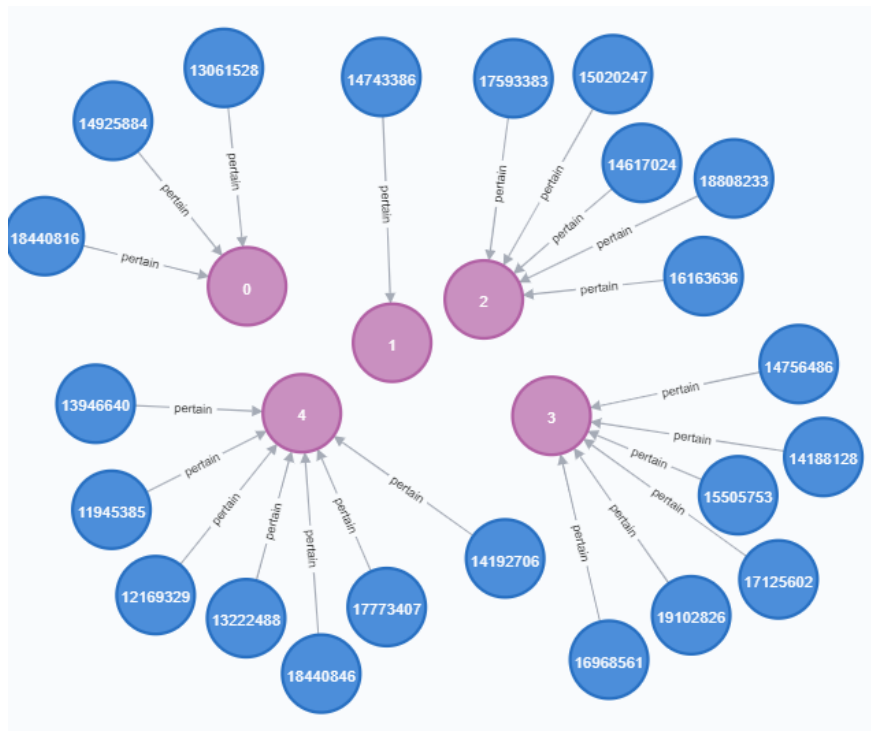


Figura 67: Atletas por categoria

- **Passagem do requisito de exploração catorze (RE14) para o Neo4J:**

Comando:

```
MATCH (rec:recurso)->[n:necessita]-(e:especialidade)->[r2:especializado]-(t:tecnico)->[r:executa]->(tc:testeClinico)
where e.idEspecialidade=t.especialidade and t.idTecnico=tc.Tecnico_idTecnico and
rec.Especialidade_idEspecialidade=e.idEspecialidade
and rec.Especialidade_idEspecialidade=e.idEspecialidade and (tc.data_hora.year=2017 and
tc.data_hora.month=07 and tc.data_hora.day=01)
set rec.quantidade=toInt(rec.quantidade)
return rec.designacao,((count(*)/toFloat(rec.quantidade))*100)
```

rec.designacao	Percentagem
"Equip. Oftalmologia"	33.33333333333333

Figura 68: Percentagem de equipamentos usados num certo dia

- **Passagem do requisito de exploração quinze (RE15) para o Neo4J:**

Cria um atleta:

Comando:

```
Match      (mod:modalidade),(a:atleta),(cat:categoria)      where      a.cc='111112'and
mod.idModalidade='3' and cat.idCategoria='1' with count(*) as c FOREACH(ignoreme in CASE
WHEN c=0 THEN [1] ELSE [] END
| create (a:atleta {cc:'111112', nome:'Catarina', telemovel:'964871609', dataNascimento:'1982-
12-23',      morada:'Rua      AMORIM',      genero:'FEM',      categoria:'3',      modalidade:'1',
codigo_Postal:'4000-002'}));
```

9. Análise comparativa entre *SQL* e *NEO4J*

Com a realização deste trabalho podemos concluir que em termos de rapidez o Neo4J é considerado o melhor uma vez que o tempo de criação do povoamento foi bastante menor. Também podemos comparar o nível de dificuldade do desenvolvimento das queries. Nesta comparação concluímos que a realização destas em *Neo4J* foram bastante mais acessíveis. Para além disso, verificamos que o tamanho de armazenamento em *mySQL* é menor do que em Neo4J, fazendo uma grande diferença entre estes dois modelos de base de dados.

10. Conclusão

No desenvolvimento deste projeto foi-nos permitido a aplicação de diversos conhecimentos adquiridos nas aulas práticas e teóricas desta unidade curricular. Ao longo da sua construção deparamos-nos com algumas dificuldades, as quais vão ser apresentadas nesta secção.

Numa fase inicial, uma das dificuldades foi a fixação dos requisitos uma vez que à medida do desenvolvimento do trabalho surgiram alguns que nos pareciam mais importantes para a execução do mesmo. A nível do modelo conceptual, lógico e físico não sentimos grandes adversidades pelo que consideramos que estes foram cumpridos em relação ao objetivo proposto. Na passagem de modelo relacional para o modelo não relacional, sentimos algumas dificuldades a nível de povoamento da base de dados que mais tarde conseguimos alcançar. Também na execução das queries, no *NEO4J*, que envolviam procedures e functions tivemos alguns problemas uma vez que foi um conceito pouco consolidado.

Em suma, embora este projeto apresente uma grande complexidade, consideramos que o trabalho cumpre todos os objetivos propostos inicialmente.

Referências

- Connolly, T., Begg, C., *Database Systems, A Practical Approach to Design, Implementation, and Management*, Addison - Wesley, 4a Edição, 2004
- <https://neo4j.com/>
- https://elearning.uminho.pt/webapps/blackboard/execute/modulepage/view?course_id=_41084_1&cmp_tab_id=71000_1&mode=view
- <https://www.mysql.com/>

Lista de Siglas e Acrónimos

BD	Base de Dados
SQL	Structured Query Language
RD	Requisitos de Descrição
RE	Requisitos de Exploração
RC	Requisitos de Controlo
ID	Identidade

Anexos

Anexo 1 – Entrevista

1ª Pergunta: Porque é que recorreu aos nossos serviços?

R: Pois necessito de um sistema que seja capaz de guardar todas as informações necessárias relativas aos atletas, técnicos, equipamentos e teste clínicos. Este tem de ser seguro e de permitir o agendamento e realização de testes clínicos.

2ª Pergunta: O que precisa de saber de cada atleta?

R: É necessário ser possível saber o nome do atleta, a idade que este possui, o número do seu cartão de cidadão, o seu contacto telefónico, a sua data de nascimento, o seu género, código postal bem como a sua categoria e modalidade. Também acho que seja necessário guardar dados sobre os medicamentos que os atletas devam tomar caso seja necessário.

Todas estas informações são necessárias para o bom funcionamento das clínicas.

3ª Pergunta: E sobre os técnicos?

R: Como a empresa tem várias clínicas, um dado importante sobre o técnico é a localidade onde este trabalha. É também necessário ter guardado o seu nome, o contacto, a sua especialidade, a morada e a data de nascimento.

6ª Pergunta: Que tipo de informações quer ter acesso sobre os testes clínicos?

R: De maneira a ter um correto agendamento e realização dos testes acho necessário armazenar informaticamente os seguintes dados: o atleta que estará no teste clínico, o

técnico que realizou o teste, a data e a hora a que o teste aconteceu, o estado (se já ocorreu ou não) e o respetivo preço.

5ª Pergunta: Para além de guardar todas as informações que o Sr. Rui já referiu, há algum tipo de funcionalidades extra que pretende que a base de dados faça gestão?

R: Sim! Gostaria também de guardar informações relativas aos equipamentos que são necessários utilizar na realização dos testes.

6ª Pergunta: Quem poderá ter acesso à base de dados?

R: Apenas eu poderei aceder às informações dos técnicos e claro a tudo o que diz respeito à clínica. Os técnicos só podem aceder a aspetos relacionados com os atletas, testes clínicos e equipamentos utilizados. Os atletas não poderão obter qualquer tipo de informação relativa às clínicas.

Anexo 2 – Modelo Físico no *MySQL*

-- MySQL Workbench Forward Engineering

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

```
-----
-- Schema mydb
-----
```

```
-----
-- Schema mydb
-----
```

```
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;
```

```
-- Table `mydb`.`Localizacao`
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Localizacao` (  
  `id_codigoPostal` VARCHAR(45) NOT NULL,  
  `Localidade` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`id_codigoPostal`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

```
-- Table `mydb`.`Categoria`
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Categoria` (  
  `idCategoria` INT NOT NULL,  
  `designacao` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idCategoria`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

```
-- Table `mydb`.`Modalidade`
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Modalidade` (  
  `idModalidade` INT NOT NULL,  
  `designacao` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idModalidade`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

```
-- Table `mydb`.`Atleta`
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Atleta` (  
  `cc` INT NOT NULL,
```

```

`nome` VARCHAR(45) NOT NULL,
`telemovel` INT NOT NULL,
`dataNascimento` DATE NOT NULL,
`morada` VARCHAR(45) NOT NULL,
`genero` VARCHAR(45) NOT NULL,
`categoria` INT NOT NULL,
`modalidade` INT NOT NULL,
`codigo_postal` VARCHAR(45) NOT NULL,
INDEX `fk_Atleta_2_idx` (`categoria` ASC) VISIBLE,
INDEX `fk_Atleta_3_idx` (`modalidade` ASC) VISIBLE,
INDEX `fk_Atleta_1_idx` (`codigo_postal` ASC) VISIBLE,
PRIMARY KEY (`cc`),
CONSTRAINT `fk_Atleta_1`
  FOREIGN KEY (`codigo_postal`)
  REFERENCES `mydb`.`Localizacao` (`id_codigoPostal`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `fk_Atleta_2`
  FOREIGN KEY (`categoria`)
  REFERENCES `mydb`.`Categoria` (`idCategoria`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `fk_Atleta_3`
  FOREIGN KEY (`modalidade`)
  REFERENCES `mydb`.`Modalidade` (`idModalidade`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

```

-----
-- Table `mydb`.`Especialidade`
-----

```

```

CREATE TABLE IF NOT EXISTS `mydb`.`Especialidade` (
  `idEspecialidade` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idEspecialidade`))

```

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- Table `mydb`.`Tecnico`

```
CREATE TABLE IF NOT EXISTS `mydb`.`Tecnico` (  
  `idTecnico` INT NOT NULL,  
  `contacto` INT NOT NULL,  
  `nome` VARCHAR(45) NOT NULL,  
  `especialidade` INT NOT NULL,  
  `morada` VARCHAR(45) NOT NULL,  
  `codigo_postal` VARCHAR(45) NOT NULL,  
  `dataNascimento` DATE NOT NULL,  
  PRIMARY KEY (`idTecnico`),  
  INDEX `fk_Tecnico_1_idx` (`especialidade` ASC) VISIBLE,  
  INDEX `fk_Tecnico_2_idx` (`codigo_postal` ASC) VISIBLE,  
  CONSTRAINT `fk_Tecnico_1`  
    FOREIGN KEY (`especialidade`)  
      REFERENCES `mydb`.`Especialidade` (`idEspecialidade`)  
      ON DELETE NO ACTION  
      ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Tecnico_2`  
    FOREIGN KEY (`codigo_postal`)  
      REFERENCES `mydb`.`Localizacao` (`id_codigoPostal`)  
      ON DELETE NO ACTION  
      ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

-- Table `mydb`.`TesteClinico`

```
CREATE TABLE IF NOT EXISTS `mydb`.`TesteClinico` (  
  `idTesteClinico` INT NOT NULL,  
  `Atleta_cc` INT NOT NULL,  
  `Tecnico_idTecnico` INT NOT NULL,
```

```

`data_hora` DATETIME NOT NULL,
`estado` INT NOT NULL,
`preco` DOUBLE(4,2) NULL,
`duracao` DOUBLE(2,2) NOT NULL,
PRIMARY KEY (`idTesteClinico`),
INDEX `fk_TestesClinico_Tecnico1_idx` (`Tecnico_idTecnico` ASC) VISIBLE,
INDEX `fk_TestesClinico_Atleta1_idx` (`Atleta_cc` ASC) VISIBLE,
CONSTRAINT `fk_TestesClinico_Tecnico1`
  FOREIGN KEY (`Tecnico_idTecnico`)
  REFERENCES `mydb`.`Tecnico` (`idTecnico`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `fk_TestesClinico_Atleta1`
  FOREIGN KEY (`Atleta_cc`)
  REFERENCES `mydb`.`Atleta` (`cc`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `mydb`.`Medicamento`
-----

```

```

CREATE TABLE IF NOT EXISTS `mydb`.`Medicamento` (
  `idMedicamento` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idMedicamento`))
ENGINE = InnoDB;

```

```

-----
-- Table `mydb`.`Atleta_has_Medicamento`
-----

```

```

CREATE TABLE IF NOT EXISTS `mydb`.`Atleta_has_Medicamento` (
  `idAtleta` INT NOT NULL,
  `idMedicamento` INT NOT NULL,
  PRIMARY KEY (`idAtleta`, `idMedicamento`),
  INDEX `fk_Atleta_has_Medicamento_2_idx` (`idMedicamento` ASC) VISIBLE,

```

```

CONSTRAINT `fk_Atleta_has_Medicamento_1`
  FOREIGN KEY (`idAtleta`)
  REFERENCES `mydb`.`Atleta` (`cc`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `fk_Atleta_has_Medicamento_2`
  FOREIGN KEY (`idMedicamento`)
  REFERENCES `mydb`.`Medicamento` (`idMedicamento`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb`.`Recurso`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`Recurso` (
  `idRecurso` INT NOT NULL,
  `designacao` VARCHAR(45) NOT NULL,
  `quantidade` INT NULL,
  `Especialidade_idEspecialidade` INT NOT NULL,
  INDEX `fk_Recurso_Especialidade1_idx` (`Especialidade_idEspecialidade` ASC) VISIBLE,
  CONSTRAINT `fk_Recurso_Especialidade1`
    FOREIGN KEY (`Especialidade_idEspecialidade`)
    REFERENCES `mydb`.`Especialidade` (`idEspecialidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

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

```