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

Unidade Curricular de Bases de Dados

Ano Letivo de 2019/2020

Clínica Pé de Atleta

Filipa Alves dos Santos (A83631)

Hugo André Coelho Cardoso (A85006)

João da Cunha e Costa (A84775)

Válter Ferreira Picas Carvalho (A84664)

Janeiro, 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Clínica Pé de Atleta

Filipa Alves dos Santos (A83631)

Hugo André Coelho Cardoso (A85006)

João da Cunha e Costa (A84775)

Válter Ferreira Picas Carvalho (A84664)

Janeiro, 2020

Resumo

O projeto abordado e explicado neste relatório está dividido em duas partes: a elaboração de um sistema de base de dados relacional e a migração para um sistema não relacional.

Numa primeira fase, é detalhado todo o processo de elaboração do SGBDR a implementar no caso de estudo apresentado, a clínica Pé de Atleta.

Começa-se por contextualizar o problema e apresentar o caso de estudo em concreto, abordando as motivações e objetivos para a implementação do SGBD, procedendo ao respetivo levantamento de requisitos.

De seguida, são expostas em pormenor as etapas práticas da criação da base de dados, que consistem no desenvolvimento de três modelos diferentes – conceptual, lógico e físico -, sendo detalhadas as decisões tomadas em cada um, devidamente justificadas, e a validação final dos mesmos para assegurar que satisfazem todos os requisitos.

Numa segunda fase, é descrita a migração do SGBD implementado anteriormente em MySQL para um sistema não relacional, neste caso o Neo4j. O processo é detalhado ao pormenor, indicando todos os *softwares* usados e exemplificando o código de construção do sistema em *Neo4j.*

Por fim, são criadas interrogações em *Cypher*, a linguagem de programação usada pelo SGBDNR escolhido, para testar a operacionalidade do sistema e testemunhar as vantagens do contexto não relacional em ação.

Área de Aplicação: Modelação e implementação de Sistemas de Bases de Dados
Palavras-Chave: Bases de Dados Relacionais, Requisitos, Modelação Conceptual, Modelação Lógica, MySQL, SQL, Sistema Não Relacional, Neo4j, Migração, Cypher.

Índice

Anexos

Índice de Figuras

Índice de Tabelas

1. Introdução

A clínica Pé de Atleta, responsável por realizar testes médicos a desportistas profissionais de alta competição de Atletismo, vai recorrer à implementação de um sistema de base de dados para arquivar e organizar toda a informação relativa aos clientes e aos serviços prestados. É pretendido que a base de dados relacione toda a informação entre si, de maneira a ser possível consultar o estado do sistema a qualquer momento e ficar com uma perspetiva sobre os serviços mais recorridos, os períodos mais ocupados da clínica e o desempenho geral dos atletas. Logo, a base de dados deve estar equipada com meios que permitam consultar, por exemplo, as provas ganhas por um dado atleta ou os participantes de uma prova específica.

O desenvolvimento do sistema será dividido em várias fases, de maneira a obter um produto final sólido e consistente: investigação do caso de estudo e definição do sistema, análise da viabilidade dos projetos, levantamento de requisitos e caracterização dos perfis de utilizadores, modelação conceptual e lógica, implementação física, povoamento e, por fim, exploração/consulta da informação armazenada, com eventuais ações corretivas ou de melhoria do projeto.

Nesta primeira secção, procede-se à definição do sistema de base de dados, com recurso a uma breve contextualização e análise do caso de estudo. Efetuar-se-á o estudo das motivações da implementação do sistema, bem como os objetivos pretendidos para o mesmo.

1.1. Contextualização

O sr. Arménio Coelho cresceu em Braga nos arredores do estádio 1º de maio e, graças a tal, desde cedo foi exposto à maravilha das competições profissionais de atletismo, que são frequentemente realizadas no mesmo. Como tal, o desporto tornou-se uma constante de sua vida e o sr. Arménio desenvolveu uma profunda paixão pela modalidade de atletismo em específico, eventualmente fazendo a transição de assistir das bancadas para correr nas pistas. Infelizmente, teve de abandoná-las após alguns anos de uma carreira bem-sucedida e recheada de medalhas, devido a uma lesão na rótula direita que o incapacitou de competir a alto nível para o resto da vida. Porém, embora tenha ficado sem a capacidade de frequentar as pistas, nunca perdeu o entusiasmo pelo desporto, pelo que acabou por aproveitar o conhecimento adquirido ao longo de anos sobre a área, tanto vestindo a pele de espectador como de praticante da modalidade, para criar uma clínica de testes médicos e servir atletas

profissionais. Na sua habitual boa disposição e sentido de humor característicos, fez um trocadilho e deu à clínica o nome “Clínica Pé de Atleta”.

O negócio não tardou a descolar, graças não só ao renome do sr. Arménio no mundo do atletismo, como à vasta rede de contactos que desenvolveu ao longo dos anos, da qual faziam parte atletas, treinadores e médicos, bem como outros perfis do desporto. Além disto, o sr. Arménio possuía já uma grande vantagem quando entrou no mundo do comércio: a confiança e amizade de muita gente da área, especialmente a um nível mais local, pelo que não tardou a receber não só a adesão de caras familiares, como de nomes desconhecidos e personalidades com um estatuto mais elevado, devido às frequentes recomendações que os clientes faziam a companheiros de competição e amigos.

Face a isto, o sr. Arménio viu-se obrigado a expandir as instalações num curto período de tempo e a contratar mais empregados, e chegou ainda a uma outra conclusão: o sistema de registo de dados em papel, sobre as consultas e os atletas, que usara até então deixara de ser viável, devido à grande escala que a clínica atingira. Começou, então, a explorar vias alternativas e acabou por deduzir que a opção mais viável para resolver o problema era a implementação de um sistema de base de dados.

1.2. Apresentação do Caso de Estudo

O sr. Arménio promoveu várias reuniões com os profissionais contratados e os empregados com o objetivo de discutir os serviços pretendidos da futura base de dados, bem como a sua estrutura. Aproveitou também esta mudança de panorama para adotar uma ideia mais ambiciosa que andava a ponderar há já bastante tempo: para além de registar as fichas técnicas dos atletas e os exames médicos, guardaria também no sistema informação sobre as provas de atletismo em que os clientes tomavam parte. Desta forma, mais tarde, poderia analisar os dados e perceber de que maneira é que os serviços prestados afetavam a performance dos atletas e como seria possível melhorar ambos.

No final, elaborou um caderno com os requisitos que ele e os empregados acharam mais importantes e que gostariam de ver refletidos na futura base de dados. Os pontos essenciais desse documento são os seguintes:

- A base de dados deve possuir uma lista de todas as modalidades de atletismo praticadas pelos clientes, com a designação oficial de cada uma e o género dos praticantes, dado que não há provas mistas.
- Associada a cada modalidade deve existir também uma lista de todas as suas categorias, entre as quais varia, por norma, a distância da prova, o tipo de objeto arremessado ou o tipo de salto.
- Deve ser possível aceder a todas as provas em que os clientes participam. Cada uma deve ter a modalidade e categoria disputadas, o respetivo nome, a data em que é realizada, os três atletas que chegam ao pódio e alguma maneira de saber se foi cancelada ou não.

- A clínica deve possuir uma ficha técnica de cada atleta. Nesta deve constar o nome do atleta, a idade, a data de nascimento, o género, a nacionalidade, a morada e a modalidade praticada, bem como um registo de todas as provas em que participou.
- A base de dados deve também guardar toda a informação relativa aos testes clínicos dos clientes: o atleta, o nome do médico que o atende, a data e o preço da consulta. Deve ser possível distinguir se o cliente compareceu ou não ao teste, ou se este foi cancelado.
- Caso o cliente falte a um teste, tem de pagar uma multa do mesmo valor e deve haver um indicador que deia a saber se já foi paga ou não.

1.3. Motivação e Objetivos

Como foi mencionado acima, na clínica Pé de Atleta sempre recorreu a folhas em formato físico para o armazenamento e gestão de dados, mas o processo tem-se tornado cada vez mais caótico e confuso com o aumento exponencial da clientela, pelo que o sr. Arménio sente a necessidade de adotar um método melhor. Após muita pesquisa e ponderação, decidiu contratar profissionais para desenvolver uma base de dados, de maneira a registar toda a informação relevante relativa aos atletas que frequentam a clínica, passada e futura.

Esta mudança permitirá a redução das despesas da clínica, devido à redução da quantidade de papel usado e à desnecessidade de um espaço de arrecadações, e tornará o acesso e gestão de dados muito mais rápido, automatizado e eficiente. Onde antes era desperdiçado bastante tempo na procura das fichas e sumários de consultas passadas, que frequentemente acabavam trocados, em arquivos errados, ou mesmo perdidos, futuramente poder-se-ia simplesmente aceder à base de dados e usufruir das funcionalidades da mesma. Isto constitui uma grande vantagem não só para os empregados, aos quais facilita imenso a análise de dados, nomeadamente consultas passadas para ver a evolução da condição dos clientes, como aos próprios atletas, que podem consultar o seu historial de provas muito mais prontamente, por exemplo.

Além disso, a centralização e fácil gestão dos dados proporcionada pela base de dados abrirá novos horizontes à clínica no que toca a análises mais detalhadas e extensivas do desempenho dos atletas, que levarão a serviços mais especializados e focados nas necessidades individuais de cada um.

1.4. Estrutura do Relatório

Uma vez familiarizados com o problema, vamos então prosseguir ao desenvolvimento do SGBD propriamente dito. Nesta secção, será possível observar um pequeno resumo do que será abordado nos demais capítulos deste relatório.

Capítulo 2 – Análise da Viabilidade do Projeto: analisamos em detalhe as vantagens e desvantagens da implementação de um SGBD no contexto do problema e comparamos a outras alternativas para verificar que é a solução ideal.

Capítulo 3 – Levantamento de Requisitos: reunimos os requisitos pretendidos para a base de dados através da análise de dados e observação, tanto a nível estrutural (requisitos de descrição) como funcional (requisitos de exploração e controlo).

Capítulo 4 – Modelação Conceptual: descrição do processo de criação do modelo conceptual, que serve como pilar do projeto, desde a abordagem de modelação utilizada à validação do modelo criado.

Capítulo 5 – Modelação Lógica: abordagem do desenvolvimento de um modelo lógico a partir do modelo conceptual e verificação do mesmo com interrogações do utilizador e transações necessárias.

Capítulo 6 – Implementação Física: detalhamento da fase final da implementação do SGBDR, justificando a escolha do sistema de gestão de base de dados escolhido e mostrando a tradução do modelo lógico, interrogações do utilizador e transações para código, bem como a definição de mecanismos próprios da linguagem.

Capítulo 7 – Abordagem Não Relacional: relato do processo de migração da base de dados relacional para um contexto não relacional e implementação de um conjunto de interrogações equivalente ao usado anteriormente para testar a base de dados.

Capítulo 8 – Conclusões e Trabalho Futuro: nesta secção refletimos sobre o trabalho efetuado, fazendo um balanço do desempenho do grupo e indicando as áreas do projeto que podiam ser polidas, recapitulando ainda os conhecimentos práticos adquiridos ao longo deste trabalho e como nos poderão ser úteis no futuro.

2. Análise da Viabilidade do Projeto

O objetivo deste projeto é a implementação de um sistema de base de dados relacional para o armazenamento e gestão de dados da clínica Pé de Atleta. Antes de contratar os profissionais e discutir com eles as funcionalidades da futura base de dados, o sr. Arménio realizou uma análise da viabilidade do projeto, para averiguar realmente o quão exequível e beneficial seria a implementação deste novo sistema.

Começou por confrontar o problema atual objetivamente e comparar as soluções que via para o mesmo. De momento, os ficheiros da clínica estão uma tremenda confusão devido à quantidade avultada de clientes que tem e de testes que realiza todos os dias. Não só a organização de todos estes papéis é difícil e demorada, acabando alguns papéis nas pastas erradas ou mesmo perdidos, como o espaço para armazenamento de pastas e afins está a atingir a capacidade máxima, sendo já preciso improvisar e ocupar algumas prateleiras fora da sala de arquivos.

O sr. Arménio pensou primeiro em passar os dados para formato digital e guardá-los, por exemplo, em folhas Excel, mas rapidamente percebeu que esta estratégia não era viável. Não só não tinha tempo e mão de obra para registar todos os arquivos da clínica, como os ficheiros Excel são uma maneira bastante pouco eficaz de o fazer, pois a quantidade de ficheiros rapidamente se multiplicaria e voltar-se-ia a estabelecer a desordem. Seria difícil aceder a vários dados do mesmo atleta, por exemplo, sendo necessário procurar e aceder a vários ficheiros. Além disso, as instalações atuais não possuem grandes capacidades tecnológicas.

Dado que o sr. Arménio vai expandir e modernizar as instalações e contratar profissionais de fora para implementar o novo sistema, em qualquer dos casos, compensa adotar uma estratégia mais eficiente do que as folhas Excel.

Considerou então a implementação de um SBD. Quanto a vantagens, é de salientar a rapidez no acesso a qualquer informação do sistema, por exemplo, de atletas ou testes. Caso um empregado pretendesse verificar as provas ganhas por um atleta, bastaria inserir o nome do atleta. Também possibilitaria eventualmente a execução de pesquisas mais específicas, nomeadamente os médicos que realizaram mais testes. Estas pesquisas mais detalhadas seriam comunicadas aos profissionais responsáveis por construir a base de dados para serem implementadas como funcionalidades da mesma. Logo, é inquestionável que a rapidez e eficácia de uma base de dados é muito elevada.

Estas características evidenciam-se também como soluções para certos problemas mencionados no capítulo anterior: as perdas de tempo por parte dos funcionários, ao pesquisar

fichas específicas e pastas no meio de todos os arquivos armazenados na clínica, e a possível mistura, extravio ou perda de ficheiros, dado que o sistema armazena toda a informação no mesmo sítio, com fácil e rápido acesso.

Além disso, como é um SBDR, sabemos que toda a informação está relacionada e é íntegra, pelo que a probabilidade de surgirem conflitos devido a informação solta ou contraditória é basicamente nula.

Portanto, após toda esta pesquisa e reflexão, o sr. Arménio concluiu que a implementação de um sistema de base de dados era, de facto, a solução mais eficaz e rentável para a sua situação, dado que prometia resolver vários problemas da clínica de uma só vez.

3. Levantamento de Requisitos

Como foi mencionado na análise do caso de estudo, no primeiro capítulo, o sr. Arménio entregou um documento bastante detalhado e preciso relativo à estruturação da futura base de dados. Seguidamente, mencionou ainda algumas práticas importantes da clínica, que imaginou que restringiriam até certo ponto o funcionamento do SBD:

- Para um atleta participar numa prova, é necessário realizar um exame médico na semana imediatamente anterior, para verificar que está apto a competir.
- Cada atleta pratica uma e uma só modalidade, no entanto pode participar em provas de qualquer categoria dentro da sua modalidade.
- Um atleta tem de marcar os seus testes com mínima antecedência, isto é, só pode marcar um teste se não tiver nenhuma prova no mesmo dia.
- Caso possua uma multa por pagar, fica impedido de participar em testes ou provas até que a pague.
- Se o atleta perceber que não vai conseguir comparecer a uma consulta, pode cancelar a mesma, desde que seja antes do dia marcado. Não é possível cancelar um teste no próprio dia.

Referiu ainda algumas das funcionalidades que desejava ver implementadas, nomeadamente historiais de multas (totais e por pagar), para facilitar a gestão financeira do negócio, estatísticas dos atletas (provas ganhas e atletas com melhor desempenho), informações das provas (atletas participantes) e dados da própria clínica (total faturado e médicos mais ocupados).

De seguida, foi inquirido quanto aos utilizadores da base de dados e os privilégios que pretendia conceder a cada um deles. Estas foram as conclusões tiradas das suas respostas:

- O sr. Arménio será o administrador do SBD, com liberdade total para aceder e alterar qualquer informação do sistema, de maneira a poder vigiar a evolução do negócio, gerir o lado financeiro e corrigir eventuais erros que descubra. É o único com privilégios totais sobre as multas e as provas.
- Os médicos da clínica podem aceder e alterar os registos dos atletas, bem como manipular fichas de consultas.
- Os atletas, por fim, podem consultar os seus próprios dados, inclusive a informação das provas em que participaram, bem como inscrever-se e desistir de provas e marcar/cancelar testes médicos.

3.1. Requisitos Levantados

3.1.1 Requisitos de Descrição

- **Atleta:** Nome, idade, género, data de nascimento, nacionalidade e morada;
- **Modalidade:** Designação e género dos praticantes;
- **Categoria:** Designação oficial;
- **Prova:** Nome, data, indicador de cancelamento e vencedores (pódio);
- **Teste:** Nome do médico, data, indicador de comparecimento e preço;
- **Multa:** Valor e indicador de pagamento.

3.1.2 Requisitos de Exploração

1. Atletas que ganharam mais provas;
2. Atleta mais novo a ganhar;
3. Atletas numa prova;
4. Provas em que certo atleta chegou ao pódio;
5. Provas ganhas por nacionalidade;
6. Multas a pagar por atleta;
7. Historial de multas de um certo atleta;
8. Médicos que deram mais consultas;
9. Total faturado global (multas + consultas).

3.1.3 Requisitos de Controlo

- Consulta de fichas de atleta e respetivas provas – clientes, médicos e administrador;
- Inscrição/desistência de provas – clientes e administrador;
- Marcação/Cancelamento de testes – clientes e administrador;
- Manipulação de registos de atletas e consultas – médicos e administrador;
- Consulta/gestão de multas e provas – administrador.

4. Modelação Conceptual

Após a definição do sistema e o levantamento de requisitos, avançamos para a modelação conceptual da base de dados, onde se procura averiguar as várias entidades que existirão no modelo, respetivos atributos, e a maneira como as mesmas se relacionam.

4.1. Abordagem de Modelação Realizada

A modelação conceptual foi abordada da seguinte maneira:

- Identificação das identidades que constituem o sistema e relacionamentos entre elas;
- Identificação dos seus atributos e definição do domínio destes;
- Determinação das chaves das entidades;
- Construção do diagrama conceptual;
- Validação do modelo de dados com o utilizador.

4.2. Identificação das Entidades do Sistema

Entidade	Descrição	Informação
Atleta	Clientes da clínica	Nome, idade, género, data de nascimento, nacionalidade e morada
Modalidade	Variante do desporto praticada	Designação e género de quem a pratica
Categoria	Variante da modalidade praticada	Designação
Prova	Competição disputada pelos atletas	Nome, data, *flag* e vencedores (pódio)
Teste	Exame feito a um atleta para testar a condição física	Nome do médico, data, *flag* e preço
Multa	Coima a pagar em caso de falta de comparecência a um teste	Valor e *flag*

Tabela 1 – Entidades do sistema

4.3. Identificação dos Relacionamentos

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Atleta	N	Pratica	1	Modalidade
Atleta	N	Realiza	N	Prova
Atleta	1	Realiza	N	Teste
Atleta	1	Paga	N	Multa
Modalidade	1	Possui	N	Categoria
Modalidade	1	Disputada numa	N	Prova
Categoria	1	Disputada numa	N	Prova
Teste	1	Gera	1	Teste

Tabela 2 – Relacionamentos das entidades

4.4. Identificação dos Atributos e Respetivos Domínios

Entidade: Atleta

Atributo	Tipo de dados	NULL	Composto	Derivado	Multivalor
idAtleta	INT	X	X	X	X
Nome	VARCHAR(45)	X	X	X	X
Idade	INT	X	X	✓	X
Género	CHAR(1) *	X	X	X	X
Data de Nascimento	DATE	X	X	X	X
Nacionalidade	VARCHAR(45)	X	X	X	X
Morada	VARCHAR(100)	X	X	X	X
* M – masculino, F – feminino					

Tabela 3 – Atributos da entidade Atleta

Entidade: Modalidade

Atributo	Tipo de dados	NULL	Composto	Derivado	Multivalor
idModalidade	INT	X	X	X	X
Designação	VARCHAR(45)	X	X	X	X
Género	CHAR(1) *	X	X	X	X
* M – masculino, F – feminino					

Tabela 4 – Atributos da entidade Modalidade

Entidade: Categoria

Atributo	Tipo de dados	NULL	Composto	Derivado	Multivalor
IdCategoria	INT	X	X	X	X
Designação	VARCHAR(45)	X	X	X	X

Tabela 5 – Atributos da entidade Categoria

Entidade: Prova

Atributo	Tipo de dados	NULL	Composto	Derivado	Multivalor
idProva	INT	X	X	X	X
Nome	VARCHAR(45)	X	X	X	X
Data	DATETIME	X	X	X	X
Flag	CHAR(1) *	X	X	X	X
Vencedor	DATE	X	✓	X	X
* S – cancelada, N – não cancelada					

Tabela 6 – Atributos da entidade Prova

Entidade: Teste

Atributo	Tipo de dados	NULL	Composto	Derivado	Multivalor
idTeste	INT	X	X	X	X
Médico	VARCHAR(45)	X	X	X	X
Data	DATETIME	X	X	X	X
Flag	CHAR(1) *	X	X	X	X
Preço	DOUBLE	X	X	X	X
* N – atleta faltou, S – atleta compareceu, C – cancelado					

Tabela 7 – Atributos da entidade Teste

Entidade: Multa

Atributo	Tipo de dados	NULL	Composto	Derivado	Multivalor
idTeste	INT	X	X	X	X
Flag	CHAR(1) *	X	X	X	X
Valor	DOUBLE	X	X	X	X
* S – paga, N – não paga					

Tabela 8 – Atributos da entidade Multa

4.5. Definição das Chaves das Entidades

Entidade	Chave Primária	Chaves Estrangeiras
Atleta	idAtleta	idModalidade
Modalidade	idModalidade	-
Categoria	idCategoria	idModalidade
Prova	idProva	idModalidade, idCategoria
Teste	idTeste	idAtleta
Multa	idMulta	idTeste, idAtleta
*Não colocamos uma coluna para as chaves candidatas porque a única chave candidata de cada entidade é o respetivo id		

Tabela 9 – Chaves das entidades

4.6. Modelo Conceptual

Tendo em conta o supramencionado e os requisitos levantados no capítulo anterior, chegamos ao seguinte diagrama:

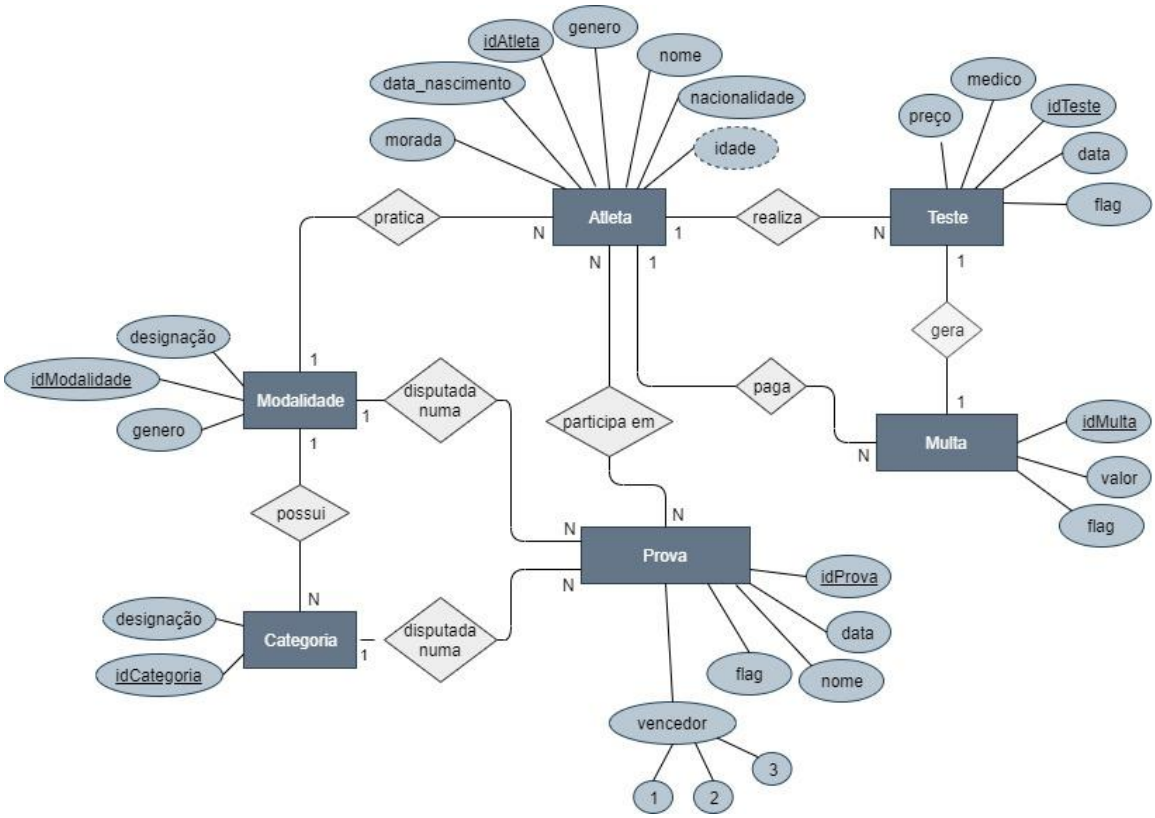

Figura 1 – Modelo conceptual do sistema de base de dados

4.7. Validação do Modelo de Dados com o Utilizador

Após a finalização do modelo que figura acima, foi necessária a aprovação do sr. Arménio para avançar com o projeto. Como tal, realizou-se uma reunião presencial com o mesmo, na qual se lhe foi explicado o significado de cada parte do diagrama, nomeadamente as entidades, atributos, e a maneira como se relacionavam entre si. Um dos pontos que recebeu mais ênfase foi o facto de a relação entre as entidades *Prova* e *Atleta* originar uma outra tabela, com o historial de provas por atleta pretendido para a clínica. Uma vez abordada a estrutura, foi dado a entender como funcionaria internamente cada funcionalidade do programa. A título de exemplo, podemos olhar para a funcionalidade de consultar as multas por pagar de um dado atleta: efetua-se uma travessia por todas as multas do sistema, filtrando as que têm o identificador do atleta em questão e a *flag* indicativa de que o pagamento está pendente.

Uma vez explicado todo o sistema, o sr. Arménio mostrou-se satisfeito com o modelo e deu luz verde à continuação do desenvolvimento do projeto.

5. Modelação Lógica

Uma vez construído e validado o modelo conceptual, temos então uma base sólida para o projeto. O próximo passo é a modelação lógica do SBD, que consiste na representação da estrutura de dados, a um nível de abstração ainda superior ao plano físico.

5.1. Construção e Validação do Modelo Lógico

A lógica utilizada para a construção do modelo lógico a partir do conceptual for a seguinte:

1. Cada entidade deu origem a uma relação (tabela), onde cada atributo foi convertido numa coluna da mesma, do respetivo tipo; o atributo composto Vencedor da entidade *Prova* foi decomposto nos seus vários valores e o derivado *Idade* de *Atleta* não foi introduzido na relação uma vez que, sendo um atributo derivado, é calculado a partir de outro, neste caso a data de nascimento.
2. Foram marcadas em cada relação as chaves primárias (os identificadores) e as variáveis que nunca seriam nulas (NN);
3. Um multivalor (equivalente a um relacionamento de 1 para N) seria convertido numa relação também, com o identificador da outra relação a que estava associado como chave estrangeira da mesma – no caso da clínica, não há multivalores;
4. Foram estabelecidos os relacionamentos de 1 para N, sendo que a relação com multiplicidade N fica com o identificador da outra como chave estrangeira – é possível observar estas chaves estrangeiras nas relações *Atleta*, *Categoria*, *Prova*, *Teste* e *Multa*;
5. De seguida foi o relacionamento de 1 para 1 entre *Teste* e *Multa* – neste caso, qualquer relação poderia ficar com a chave estrangeira da outra, tendo sido atribuída a *Multa* dado que, desta maneira, a base de dados gera menos valores nulos;
6. Por fim, traçou-se o relacionamento de N para N entre *Atleta* e *Prova*, dando origem a uma outra relação *Provas_Atleta*, que possui como colunas as chaves primárias das relações a que está ligada; caso o próprio relacionamento também possuísse atributos, estes também seriam convertidos em colunas da relação.

5.2. Modelo Lógico

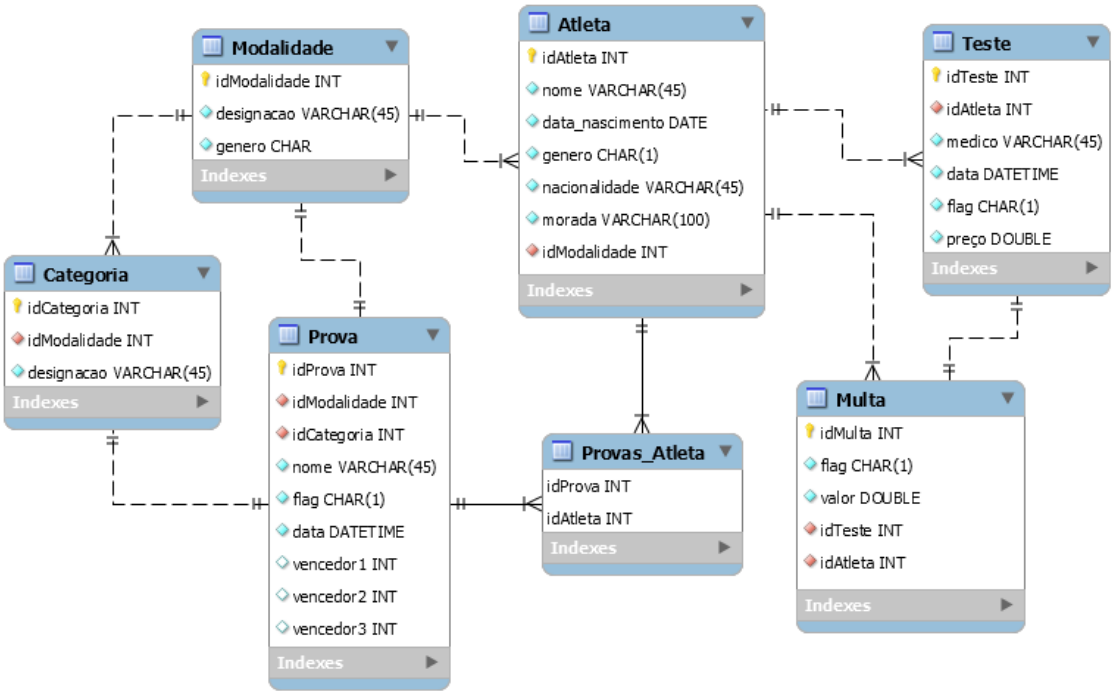

Figura 2 – Modelo lógico do SBD

5.3. Validação do Modelo através da Normalização

A normalização consiste num conjunto de regras que visa organizar o modelo de maneira a reduzir a redundância de dados, incrementando a sua integridade e o desempenho do SBD. Este processo é feito com recurso a três formas diferentes.

5.3.1 Primeira Forma Normal

Condição: todos os valores das colunas das tabelas têm de ser atómicos.

Verificação: nenhuma tabela do modelo possui colunas repetidas nem atributos multivalor, logo estão na 1FN.

5.3.2 Segunda Forma Normal

Condição: todas as relações estão na 1FN e qualquer atributo não-chave é totalmente dependente da respetiva chave primária.

Verificação: as relações estão na 1FN, como verificamos anteriormente, e não existe nenhuma chave primária composta, pelo que não há dependências parciais. Logo, as relações do modelo estão na 2FN.

5.3.3 Terceira Forma Normal

Condição: as relações estão na 2FN e nenhum atributo não-chave possui uma dependência transitiva em relação às chaves candidatas da mesma tabela.

Verificação: esta condição traduz-se grosseiramente em "todas as colunas de uma tabela devem depender única e exclusivamente da respetiva chave primária". Após cuidada análise do modelo, podemos concluir que isto se verifica, pelo que as suas relações estão na 3FN. Isto não se verificaria se, por exemplo, o atributo *Idade* da entidade *Atleta* tivesse sido convertido numa coluna da respetiva relação, dado que a idade depende da data de nascimento.

5.4. Validação do Modelo com Interrogações do Utilizador

Semelhante ao que se fez no processo de validação do modelo conceptual, é de novo necessário verificar que o modelo lógico assegura o funcionamento dos requisitos por exploração, isto é, as interrogações que o utilizador pode fazer ao sistema.

Os processos de resolução dos requisitos descritos abaixo não correspondem necessariamente à solução implementada no sistema físico, dado que esta é apenas uma versão precoce cujo propósito é unicamente verificar se os requisitos são exequíveis com a estrutura atual da base de dados.

Requisito 1 – Atletas que ganharam mais provas

A relação *Prova* possui atributos respetivos aos atletas que ficam no pódio da competição. Para saber os atletas que ganharam mais provas, basta percorrer todas as provas existentes na base de dados, contado o número de ocorrências dos identificadores de cada atleta nas tabelas.

Requisito 2 – Atleta mais novo a ganhar

Semelhante ao que se faz no requisito anterior, realiza-se uma travessia das provas na base de dados, desta vez guardando apenas os identificadores dos atletas que ficaram em primeiro lugar. A relação *Atleta* possui a data de nascimento do desportista e será codificada uma função para calcular a idade a partir da mesma na implementação do modelo físico. Logo, após a referida travessia, basta calcular as idades dos atletas vencedores e averiguar qual deles é o mais novo.

Requisito 3 – Atletas numa prova

As relações *Atleta* e *Prova* estão ligadas através de uma terceira tabela *Provas_Atleta*, que faz a correspondência entre todas as provas e os atletas que nelas participaram. Logo, dada uma prova específica, é possível saber todos os participantes.

Requisito 4 – Provas em que certo atleta ficou no pódio

Este requisito é semelhante ao primeiro, mas em vez de todos, pretende-se verificar as provas ganhas por um único atleta específico. Como tal, o raciocínio é análogo ao do requisito 1, sendo que se compara os identificadores guardados nas provas com o do atleta fornecido e se contam as respetivas ocorrências.

Requisito 5 – Provas ganhas por nacionalidade

A tabela *Atleta* apresenta um atributo respetivo à nacionalidade do indivíduo. Semelhante ao que se faz no requisito 2, após armazenar os primeiros vencedores das provas, proceder-se-ia à verificação das respetivas nacionalidades e contagem das mesmas, organizando-as no fim por ordem decrescente de vitórias.

Requisito 6 – Multas por pagar por atleta

A relação *Multa* possui uma chave estrangeira que identifica o atleta a quem está associada, bem como uma *flag* relativa ao estado de pagamento da multa. Como tal, para obter o historial de multas por pagar de todos os atletas, é necessário percorrer todas as multas na base de dados, filtrando aquelas cuja *flag* esteja a 'N', indicando que ainda não foram pagas, e os respetivos infratores. No fim, são dispostas por ordem alfabética dos atletas.

Requisito 7 – Historial de multas de um certo atleta

Este requisito é semelhante ao anterior. O raciocínio é análogo, mas guarda-se a multa para exibir no fim apenas se a chave estrangeira corresponder ao identificador ao atleta especificado, independentemente do estado de pagamento.

Requisito 8 – Médicos que deram mais consultas

A relação *Teste* armazena o nome do médico que atende o atleta. Logo, para este requisito, apenas é preciso percorrer todos os testes da base de dados, guardando os nomes dos respetivos médicos, e no fim contá-los e apresentá-los por ordem decrescente.

Requisito 9 – Total faturado global

As fontes de rendimento da clínica são as multas e os testes. Para calcular o total faturado pela clínica, é necessário percorrer todos os testes da clínica, filtrando aqueles aos quais os clientes compareceram (o que implica que foram efetivamente pagos) e extraindo o respetivo preço, bem como todas as multas, verificando as que já foram pagas. No fim, soma-se os preços de todos eles.

5.5. Validação do Modelo com as Transações estabelecidas

As alterações na base de dados são realizadas através de transações. Estes procedimentos têm como objetivo garantir a consistência do SBD, permitindo a recuperação correta de falhas e encapsulando os programas que acedem à base de dados, de maneira a evitar que se induzam em erro.

De seguida, iremos abordar as transações implementadas no SBD da clínica:

5.5.1 Adicionar Atleta

A inserção de um novo atleta na base de dados só é possível se o atleta e a modalidade que pratica existirem e se a data de nascimento fornecida for válida (data passada). Caso isto se verifique, é criada uma entrada na tabela *Atleta* com os atributos dados e a chave primária da modalidade como chave estrangeira.

5.5.2 Marcar Prova

A inserção de uma nova prova é possível apenas se a mesma não estiver já registada, a sua modalidade e categoria existirem no sistema e a data for futura. Se for o caso, é criada uma entrada na tabela *Prova* com o nome e a data dados como colunas, uma *flag* indicativa de que a prova ainda não foi cancelada, as chaves primárias da modalidade e categoria como chaves estrangeiras e colunas para os atletas do pódio ainda a nulo, dado que essa informação ainda é desconhecida.

5.5.3 Marcar Teste

A inserção de um teste está dependente de o atleta estar registado no SBD, não ter já um teste marcado nem uma prova no mesmo dia, dado que apenas pode ter um de cada vez, não possuir nenhuma multa por pagar e a data pretendida ser futura. Caso isto se verifique, é criada uma entrada na tabela *Teste* com os dados fornecidos, a chave primária do atleta como chave estrangeira e uma *flag* com valor predefinido a indicar que o atleta não compareceu, que será atualizada caso ele, de facto, compareça, ou se a prova for cancelada.

5.6. Revisão do Modelo Lógico com o Utilizador

Mais uma vez, foi marcada outra reunião com o sr. Arménio com o objetivo de o manter a par do progresso e verificar se estava de acordo com o caminho tomado. Foi-lhe mostrado o modelo lógico e explicado o que foi constatado na validação do mesmo, desde a maneira como o modelo permitia relacionar as diferentes tabelas, de maneira a possibilitar a implementação

dos requisitos de exploração pedidos, à forma como funcionavam as transações e procedimentos de consulta.

O sr. Arménio expressou o seu agrado com o trabalho realizado até então e sugeriu que se passasse à implementação final do SBD.

6. Implementação Física

A fase final do desenvolvimento do SBD consiste na sua implementação física, que culmina numa versão operacional do produto encomendado. Neste capítulo, será abordada a escolha do sistema de gestão de base de dados usado e a tradução do sistema lógico, dos requisitos de exploração e dos de controlo para código SQL, bem como os mecanismos de segurança implementados.

6.1. Seleção do Sistema de Gestão da Base de Dados

Nesta fase do projeto, fomos confrontados com a decisão do SGBD a usar, a qual acabou por recair para o *MySQL* por diversos motivos:

- Permite a delineação de vários perfis de utilizador, o que possibilita o registo dos diferentes tipos de pessoas que vão usufruir da base de dados e a restrição do acesso às respetivas funcionalidades:
- É seguro e mais economicamente viável do que as alternativas existentes no mercado;
- Suporta mecanismos de controlo de concorrência entre utilizadores, o que é fulcral para que as várias operações realizadas simultaneamente sejam sempre coerentes, nomeadamente no caso de um atleta desistir de uma prova ao mesmo tempo que outra pessoa consulta os atletas inscritos da mesma;
- É dotado de um alto desempenho, o que é altamente relevante dado que, hoje em dia, é necessário garantir uma resposta rápida do sistema, sobretudo num ambiente como o da clínica, em que serão feitas interrogações constantes ao sistema.

6.2. Tradução do Esquema Lógico para o SGBD

O *MySQL* possui um ambiente próprio de trabalho conhecido por *MySQL Workbench*, que foi usado no desenvolvimento do projeto. Este possui uma ferramenta de *Forward Engineering* que se encarrega de traduzir o esquema lógico para código SQL, à qual recorremos. O documento de conversão gerado por esta ferramenta está disponível para consulta, tendo sido enviado juntamente com os ficheiros de código com o nome “ForwardEngineer”.

O passo seguinte foi povoar a base de dados para ser possível testar a mesma. O ficheiro de povoamento também foi enviado com os ficheiros de código, com o nome “Povoamento”.

6.3. Tradução das Interrogações do Utilizador para SQL

Nesta secção serão exibidos alguns exemplos do código SQL correspondente aos requisitos de exploração, que constituem as várias operações que os utilizadores podem realizar.

Requisito 1 – Atletas que ganharam mais provas

```
SELECT  a.nome , a.data_nascimento, a.genero, a.nacionalidade, m.designacao,
        Count(p.vencedor1) as TotalProvasGanhas
from Prova p
INNER JOIN Atleta a on a.idAtleta = p.vencedor1
INNER JOIN Modalidade m on m.idModalidade = a.idModalidade
group by p.vencedor1
order by TotalProvasGanhas DESC, a.nome ASC
LIMIT 3;
```

Figura 3 – Código SQL do requisito de exploração 1

Requisito 2 – Atleta mais novo a ganhar uma prova

```
Select a.nome , idade(a.data_nascimento) as idadeMin
from Prova p
INNER JOIN Atleta a on p.vencedor1 = a.idAtleta
group by a.idAtleta
order by idadeMin ASC
LIMIT 1;
```

Figura 4 – Código SQL do requisito de exploração 2

Requisito 3 – Atletas numa prova

```
DELIMITER $$
CREATE PROCEDURE atletasProva (IN idP INT)
BEGIN
    select a.nome, a.data_nascimento , a.genero, a.nacionalidade ,p.nome as NomeProva,
           p.data, m.designacao as Modalidade, c.designacao as Categoria
    from Provas_atleta pa
    INNER JOIN Atleta a on pa.idAtleta = a.idAtleta and pa.idProva = idP
    INNER JOIN prova p on p.idProva = idP
    INNER JOIN modalidade m on m.idModalidade= p.idModalidade
    INNER JOIN categoria c on c.idCategoria= p.idCategoria;

END$$
```

Figura 5 – Código SQL do requisito de exploração 3

Requisito 4 – Provas ganhas por um certo atleta

```
DELIMITER $$
CREATE PROCEDURE provasGanhas (IN idA INT)
BEGIN
    select a.nome, p.data , p.nome, p.vencedor1 , p.vencedor2, p.vencedor3, m.designacao as Modalidade , c.designacao
    from prova p , Atleta a, modalidade m , Categoria c
    where m.idModalidade = p.idModalidade and c.idCategoria = p.idCategoria and a.idAtleta = idA
          and (p.vencedor1 = idA or p.vencedor2 = idA or p.vencedor3 = idA);
END$$
```

Figura 6 – Código SQL do requisito de exploração 4

Requisito 7 – Historial de multas de um certo atleta

```
DELIMITER $$
CREATE PROCEDURE historialMultas (IN idA INT)
BEGIN
    SELECT a.nome as Nome ,m.idMulta, t.medico as Médico, t.data as Data, t.preço as PreçoConsulta,
           m.valor as ValorMulta , m.flag as Flag
    from Atleta a, Multa m, Teste t
    where a.idAtleta = idA and m.idAtleta = idA and t.idTeste = m.idTeste;
END$$
```

Figura 7 – Código SQL do requisito de exploração 7

Requisito 8 – Médicos que deram mais consultas

```
SELECT t.medico as NomeMedico, count(t.idTeste) as TotalTestes
from Teste t
where t.flag = 'S'
group by t.medico
order by TotalTestes DESC, t.medico
LIMIT 3;
```

Figura 8 – Código SQL do requisito de exploração 9

Requisito 9 – Total faturado pela clínica

```
SELECT ((Select Sum(t.preço) from Teste t where t.flag = 'S')
       +(Select Sum(m.valor) from Multa m where m.flag = 'S')) as TotalFaturado;
```

Figura 9 – Código SQL do requisito de exploração 10

6.4. Tradução de Outras Operações do Utilizador para SQL

Como foi descrito anteriormente, no levantamento de requisitos, para além das consultas de carácter puramente de observação, há ainda outras operações exequíveis pelos vários tipos de utilizadores do sistema, que não se traduzem em transações - os atletas da clínica podem inscrever-se/desistir de provas e marcar/cancelar testes através da aplicação, os médicos podem atualizar as fichas de teste e o administrador pode atualizar multas e cancelar provas. Como tal, temos, de seguida, o código SQL que possibilita tais operações:

Médico – Atualizar uma ficha de teste após o mesmo

```
DELIMITER $$
CREATE PROCEDURE atualizarTeste (IN idT INT, IN flag CHAR(1))
BEGIN
    declare test INT;
    set test = (Select count(idTeste) from Teste t where t.idTeste = idT and t.flag = 'N') ;
    if test = 1
    then
    UPDATE Teste set flag = flag where idTeste = idT;
    end if;
END$$
```

Figura 10 – Código SQL para atualizar um teste

Administrador – Atualizar uma multa para mostrar que foi paga

```
DELIMITER $$
CREATE PROCEDURE pagarMulta (IN idM INT)
BEGIN
    declare test INT;
    set test = (Select count(idMulta) from Multa t where t.idMulta = idM and t.flag = 'N') ;
    if test = 1
    then
    UPDATE Multa set flag = 'S' where idMulta = idM;
    end if;
END$$
```

Figura 11 – Código SQL para pagar uma malta

Atleta – Inscrever-se numa prova

```
DELIMITER $$
CREATE PROCEDURE inscreverProva (IN idP INT,IN idA INT)
BEGIN
    declare test1 INT;
    declare test2 INT;
    declare test3 INT;
    declare test4 INT;
    set test1 = (Select count(idAtleta) from Multa m where m.idAtleta = idP and m.flag = 'N');
    set test2 = (Select count(idProva) from Prova t where t.idProva = idP and t.flag = 'N' and t.data > now());
    set test3 = (Select count(idAtleta) from Atleta t where t.idAtleta = idA);
    set test4 = (Select count(idTeste) from Teste t where t.idAtleta = idA and (t.data >=  NOW() - INTERVAL 1 WEEK) and t.flag = 'S');

    if (test1 = 0 and test2 = 1 and test3 = 1 and test4 = 1)
    then
    INSERT INTO Provas_Atleta
    (idProva, idAtleta)
    VALUES  (idP,idA);
    end if;
END$$
```

Figura 12 – Código SQL para inscrever um atleta numa prova

Atleta – Desistir de uma prova

```
DELIMITER $$
CREATE PROCEDURE desistirProva (IN idP INT,IN idA INT)
BEGIN
    declare test1 INT;
    declare test2 INT;
    declare test3 INT;
    set test1 = (Select count(idProva) from Provas_Atleta t where t.idProva = idP and t.idAtleta = idA) ;
    set test2 = (Select count(idProva) from Prova t where t.idProva = idP and t.flag = 'N' and t.data >= now());
    set test3 = (Select count(idAtleta) from Atleta t where t.idAtleta = idA);
    if (test1 = 1 and test2 = 1 and test3 = 1)
    then
    Delete From Provas_Atleta
    where idProva = idP and idAtleta = idA;
    end if;
END$$
```

Figura 13 – Código SQL para desistir de uma prova

Administrador – Atualizar uma prova decorrida com os vencedores

```
DELIMITER $$
CREATE PROCEDURE atualizarVencedoresProva (IN idP INT, IN idA1 INT, IN idA2 INT, IN idA3 INT)
BEGIN
    declare test INT;
    declare test1 INT;
    declare test2 INT;
    declare test3 INT;
    set test = (Select count(idProva) from Prova t where t.idProva = idP and t.flag = 'N') ;
    set test1 = (Select count(idAtleta) from Atleta t where t.idAtleta = idA1);
    set test2 = (Select count(idAtleta) from Atleta t where t.idAtleta = idA2);
    set test3 = (Select count(idAtleta) from Atleta t where t.idAtleta = idA3);

    if (test = 1 and test1 = 1 and test2 = 1 and test3 = 1)
    then
    UPDATE Prova t set t.vencedor1 = idA1 , t.vencedor2 = idA2 , t.vencedor3 = idA3 where t.idProva = idP;
    end if;
END$$
```

Figura 14 – Código SQL para atualizar uma prova com os vencedores

Administrador – Cancelar uma prova

```
DELIMITER $$
CREATE PROCEDURE cancelarProva (IN idP INT)
BEGIN
    declare test INT;
    set test = (Select count(idProva) from Prova t where t.idProva = idP and t.flag = 'N' and t.data >= now());
    if test = 1
    then
    UPDATE Prova set flag = 'S' where idProva = idP;
    end if;
END$$
```

Figura 15 – Código SQL para cancelar uma prova

6.5. Tradução das Transações para SQL

Adicionar um atleta

```
DELIMITER $$
CREATE PROCEDURE addAtleta (IN nomeA VARCHAR(45), IN dat DATETIME, IN gen CHAR(1),IN nac VARCHAR(45), IN mor VARCHAR(45), IN modl VARCHAR(45))
BEGIN
    DECLARE test1 INT;
    DECLARE test2 INT;
    DECLARE test3 INT;
    DECLARE mol int;
    DECLARE sucesso BOOL DEFAULT 0;

    SET autocommit = 0;
    SET test1 = (Select count(idAtleta) from Atleta a where a.nome = nomeA and a.data_nascimento = dat);
    SET test2 = (Select count(idModalidade) from Modalidade c where c.designacao= modl and c.genero = gen);
    IF dat < now() then set test3 = 0;# verifica se é uma data passada
    END IF;


    START TRANSACTION;

        IF(test1=0 AND test2 = 1 AND test3=0) THEN
            SET sucesso = 1;
            set mol = (select m.idModalidade from Modalidade m where m.designacao = modl and m.genero = gen);
            INSERT INTO Atleta
            (nome, data_nascimento, genero, nacionalidade, morada, idModalidade)
            VALUES
            (nomeA,dat,gen,nac, mor,mol);
            END IF;

            IF sucesso THEN
                select('Atleta adicionada!')
                COMMIT;
            ELSE
                select('Adição inválida!')
                ROLLBACK;
            END IF;

    END $$
```

Figura 16 – Código SQL para adicionar um atleta

Marcar um teste

```
DELIMITER $$
CREATE PROCEDURE marcarTeste (IN idA VARCHAR(45),IN med VARCHAR(45), IN dat DATETIME)
BEGIN
    DECLARE test1 INT;
    DECLARE test2 INT;
    DECLARE test3 INT;
    DECLARE test4 INT;
    DECLARE pre DOUBLE;
    DECLARE sucesso BOOL DEFAULT 0;
    SET autocommit = 0;
    SET pre = (SELECT RAND()*(50-10)+10);
    SET test1 = (Select count(idAtleta) from Teste t where t.idAtleta= idA and t.flag='N' and t.data > now());
    SET test2 = (Select count(idAtleta) from Atleta a where a.idAtleta= idA);
    SET test3 = (Select count(idAtleta) from Multa m where m.idAtleta= idA and m.flag='N');
    IF dat > now() then set test4 = 0;# verifica se é uma data futura
    end if;
    START TRANSACTION;

        IF(test1=0 AND test2 = 1 AND test3=0 and test4 = 0) THEN
            SET sucesso = 1;
            INSERT INTO Teste
            (idAtleta, medico, data, flag, preço)
            VALUES (idA,med,dat,'N',pre);
        END IF;

        IF sucesso THEN
            select('Teste marcado!')
            COMMIT;
        ELSE
            select('Marcação inválida!')
            ROLLBACK;
        END IF;
END $$
```

Figura 17 – Código SQL para marcar um teste

Marcar uma prova

```
DELIMITER $$
CREATE PROCEDURE adicionarProva (IN idM INT,IN idC INT,IN nomeP VARCHAR(45), IN dat DATETIME)
BEGIN
    DECLARE test1 INT;
    DECLARE test2 INT;
    DECLARE test3 INT;
    DECLARE sucesso BOOL DEFAULT 0;

    SET autocommit = 0;
    SET test1 = (Select count(idProva) from Prova p where p.nome= nomeP and p.data = dat
                and p.flag='N' and p.idModalidade = idM and p.idCategoria = idC and p.data > now());
    SET test2 = (Select count(idCategoria) from Categoria c where c.idModalidade= idM and c.idCategoria = idC);
    IF dat > now() then set test3 = 0;
    end if;

    START TRANSACTION;

        IF(test1=0 AND test2 = 1 AND test3=0) THEN
            SET sucesso = 1;
            INSERT INTO  Prova
        (idModalidade, idCategoria, nome, flag, data, vencedor1, vencedor2, vencedor3)
        VALUES
        (idM, idC,nomeP,'N', dat, NULL, NULL, NULL);
        END IF;

        IF sucesso THEN
            select('Prova marcado!')
            COMMIT;
        ELSE
            select('Marcação inválida!')
            ROLLBACK;
        END IF;

END $$
```

Figura 18 – Código SQL para marcar uma prova

6.6. Definição de Vistas de Utilização em SQL

As vistas de utilização são utilizadas para gerar tabelas auxiliares às existentes no modelo lógico, em memória, com o propósito de facilitar o acesso a certos dados e evitar a repetição de código, melhorando os tempos de resposta. Tendo isto em conta, foram criadas duas vistas de utilização para o SGBD da clínica:

Requisito 6 - Multas a pagar por atleta

```
create view MultasAPagar as SELECT m.idAtleta, a.nome, m.idMulta , m.valor, m.idTeste, m.flag
FROM Atleta a
INNER JOIN Multa m on a.idAtleta = m.idAtleta
where m.flag = 'N'
order by m.idAtleta ASC;
```

Figura 19 – Código SQL para a vista de utilização das multas por pagar

Requisito 5 - Nacionalidades ordenadas por vitórias

```
create view ProvasGanhasPorNacionalidade as Select a.nacionalidade, count(p.vencedor1) as NtProvasGanhas
from Prova p
inner join Atleta a on a.idAtleta = p.vencedor1
group by a.nacionalidade
order by NtProvasGanhas desc;
```

Figura 20 – Código SQL para a vista de utilização das nacionalidades

6.7. Definição de Mecanismos de Segurança em SQL

Dado que a base de dados será acedida por vários tipos de utilizadores diferentes – clientes, funcionários e administrador -, é necessário criar diferentes perfis no sistema e atribuir permissões diferentes a cada um, de maneira a restringir os seus acessos apenas às respetivas funcionalidades.

```
Use mydb;
#DROP USER 'atleta'@'localhost';

CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin';

GRANT ALL ON mydb.* TO 'admin'@'localhost';

CREATE USER 'medico'@'localhost' IDENTIFIED BY 'medico';

GRANT SELECT ON mydb.Prova TO 'medico'@'localhost';
GRANT SELECT ON mydb.Categoria TO 'medico'@'localhost';
GRANT SELECT ON mydb.Modalidade TO 'medico'@'localhost';
GRANT SELECT, UPDATE ON mydb.Atleta TO 'medico'@'localhost';
GRANT SELECT, INSERT, UPDATE ON mydb.Teste TO 'medico'@'localhost';

CREATE USER 'atleta'@'localhost' IDENTIFIED BY 'atleta';

GRANT SELECT ON mydb.Atleta TO 'atleta'@'localhost';
GRANT SELECT ON mydb.Multa TO 'atleta'@'localhost';
GRANT SELECT ON mydb.Prova TO 'atleta'@'localhost';
GRANT SELECT ON mydb.Categoria TO 'atleta'@'localhost';
GRANT SELECT ON mydb.Modalidade TO 'atleta'@'localhost';
GRANT INSERT, DELETE ON mydb.provas_atleta TO 'atleta'@'localhost';
GRANT INSERT, UPDATE ON mydb.Teste TO 'atleta'@'localhost';
```

Figura 21 – Código SQL para a criação dos diversos utilizadores

6.8. Mecanismos de Preservação da Integridade de Dados

De modo a garantir a coerência dos dados a qualquer momento, foram criados dois *triggers* para atualizar certas tabelas face a possíveis ocorrências no sistema:

Adicionar uma multa

```
DELIMITER $$
create trigger after_Teste_update
after update on Teste
for each row
BEGIN
    if(new.flag = 'N' and new.data < now())
    then insert into Multa (flag, valor, idTeste, idAtleta) values ('N',NEW.preço,NEW.idTeste,NEW.idAtleta);
    end if;
END $$
```

Figura 22 – *Trigger* para adicionar uma multa

Atualizar a vista de utilização das multas por pagar

```
DELIMITER $$
create trigger after_Multa
after Update on Multa
for each row
BEGIN
    Declare nm VARCHAR(45);
    insert into MultasAPagar(idAtleta, idMulta , valor, idTeste, flag) values (NEW.idAtleta,NEW.idMulta,NEW.valor,NEW.idTeste,'N');
END $$
```

Figura 23 – *Trigger* para atualizar a vista das multas por pagar

6.9. Revisão do Modelo Físico com o Utilizador

Por fim, foi marcado um último encontro com o sr. Arménio, de maneira a exibir o produto final e deixar que o cliente o testasse pessoalmente. O sr. Arménio ficou bastante contente com o resultado obtido, uma vez que satisfazia todos os requisitos encomendados e todas as funcionalidades aparentavam operar corretamente quando testadas.

Assim se chegou ao fim deste trabalho para o sr. Arménio, ambos os lados satisfeitos com o produto final, mas não seria muito tempo até voltarem a estabelecer contacto para, desta vez, mudar o sistema para uma base de dados não-relacional.

7. Abordagem Não Relacional

O paradigma *NoSQL* (***N**ot **O**nly **SQL***) engloba os sistemas de base de dados não relacionais, cuja necessidade surgiu das limitações dos sistemas tradicionais. Como tal, os SGBDNR apresentam várias vantagens sobre os sistemas relacionais, nomeadamente uma alta performance e escalabilidade, ideais para o armazenamento e tratamento de *big data*, bem como o uso de estruturas de dados mais eficientes e flexíveis do que as tabelas do modelo relacional. Além do mais, constituem também uma opção mais económica em comparação aos SGBDR.

O sistema de gestão de base de dados não relacional que vamos usar é o *Neo4j*, cuja implementação assenta na representação da informação em grafos, convertendo as tabelas tradicionais em nodos e relacionando-os entre si.

7.1. Representação em Grafo da Base de Dados

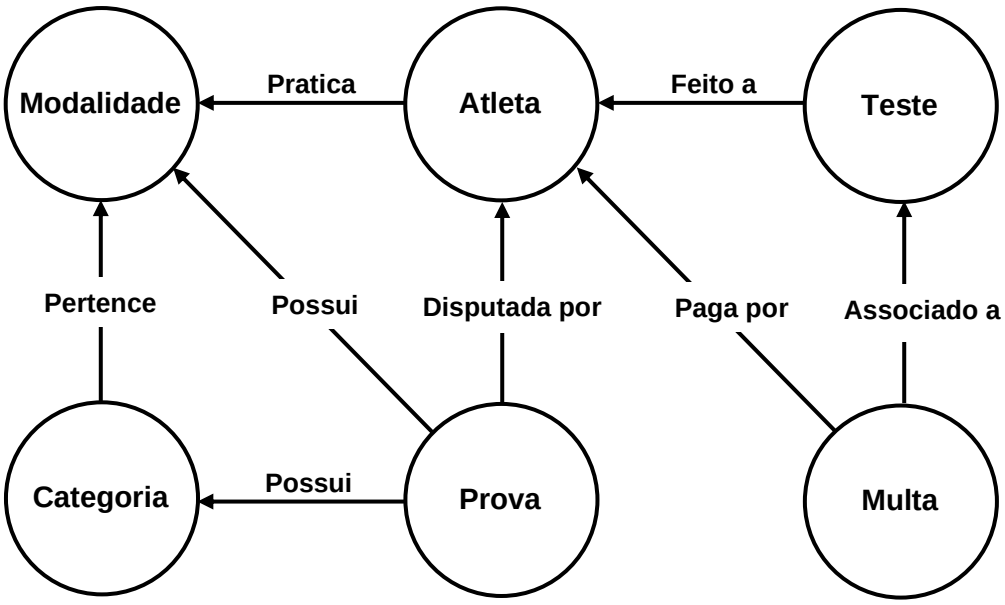

Figura 24 – Representação em grafo da base de dados

7.2. Migração de MySQL para Neo4j

Há vários métodos possíveis para efetuar a migração de dados de uma base de dados relacional para Neo4j. Enquanto que seria possível criar uma aplicação *Java* para importar os dados de *MySQL* e guardá-los em *Neo4j*, concluiu-se que havia uma maneira mais simples de realizar esta operação, recorrendo apenas às ferramentas disponibilizadas pela *MySQL Workbench.*

Durante este processo, foi necessário ter em conta várias diferenças estruturais entre *MySQL* e *Neo4j:*

- As tabelas do modelo relacional são convertidas em nodos;
- Os atributos passam a denominar-se propriedades;
- Um nodo é um tuplo;
- Uma etiqueta corresponde ao nome de uma tabela;
- As tabelas resultantes de relações N para N não originam nodos, ligando-se as duas tabelas das extremidades da relação diretamente;
- Não existem valores nulos;
- Não existem chaves estrangeiras: os relacionamentos são estabelecidos diretamente entre os nodos, com uma descrição mais concreta da relação.

Com estas regras em mente, chegou-se então ao grafo apresentado na página anterior, que representa a estrutura do sistema de base de dados não relacional.

7.2.1 Exportação dos dados de MySQL

Para exportar a informação da base de dados para ficheiros CSV, dos quais será lida mais tarde e convertida em dados não relacionais, foi usada a ferramenta *Data Export Wizard* da *MySQL Workbench.*

Cada tabela foi exportada individualmente para o respetivo ficheiro, sendo o processo bastante automático e intuitivo. A única resguarda consistiu em alterar o separador de campos para uma vírgula, em vez de usar o ponto e vírgula predefinido. No fim, em cada ficheiro ficaram registadas todas as unidades da tabela, com os valores dos respetivos atributos.

Após o processamento de todas as tabelas, criou-se um projeto *Neo4j* e os ficheiros CSV criados foram recambiados para a pasta de importação do mesmo.

7.2.2 Criação dos nodos

Para originar os nodos, foi necessário ler a informação dos ficheiros gerados no passo anterior. Analisou-se cada campo individualmente, dado que estavam separados por vírgulas, como configuramos durante a exportação, convertendo os dados para o respetivo tipo. Dado que todo o conteúdo do ficheiro CSV consiste em caracteres, os atributos do tipo *String* podem

ser lidos diretamente, mas propriedades de tipos como *Integer* e *Double* têm de ser transformadas com recurso a funções de conversão do *Neo4j*.

De seguida, é possível observar a criação dos nodos *Atleta*. Os restantes nodos seguem uma lógica análoga.

```
LOAD CSV WITH HEADERS FROM 'file:///D:/AtletaN.csv' AS line
CREATE (atleta:Atleta {idAtleta: toInteger(line.idAtleta)})
SET atleta.nome = line.nome,
 atleta.data_nascimento = line.data_nascimento,
 atleta.genero = line.genero,
 atleta.nacionalidade = line.nacionalidade,
 atleta.morada = line.morada,
 atleta.idModalidade = toInteger(line.idModalidade)
 return atleta;
```

Figura 25 – Código *Cypher* para originar os nodos *Atleta*

7.2.3 Criação dos relacionamentos

Para criar os relacionamentos apresentados no grafo em 7.1., associamos os nodos através dos respetivos ids, um dos quais era, no modelo relacional, uma chave estrangeira, e criamos o relacionamento com a respetiva descrição:

```
LOAD CSV WITH HEADERS FROM 'file:///Teste.csv' AS line
MATCH (teste:Teste {idTeste: ToInteger(line.idTeste) })
MATCH (atleta:Atleta {idAtleta: ToInteger(line.idAtleta) })

CREATE (teste)-[:Feito_a]→(atleta)
RETURN teste, atleta
```

Figura 26 – Exemplo de código *Cypher* para relacionar nodos

O único relacionamento que divergiu desta norma foi entre *Atleta* e *Prova*, dado que a relação N para N entre eles originara uma tabela nova no modelo não relacional. Portanto, ao criar o relacionamento, abriu-se o ficheiro da tabela *Provas_Atleta* para leitura e ligaram-se os nodos *Atleta* e *Prova* diretamente, através do comando *Match.* Outra particularidade do código deste relacionamento é o facto de não possuir um *Return,* efetuando apenas o *Create* com a respetiva designação da relação.

Com isto, deu-se por concluída a migração de dados de *MySQL* para *Neo4j* e procedeu-se ao teste e validação da base de dados não relacional.

7.3. Validação da Operacionalidade do Sistema Implementado

Tal como foi feito no final da implementação física relacional da base de dados, é necessário elaborar algumas interrogações para testar o sistema, bem como, neste caso, testemunhar as tão faladas vantagens do contexto não relacional, nomeadamente a alta performance.

Algo a ter em conta é o facto de programas *Cypher* não suportarem a passagem de argumentos, o que impossibilita a emulação das *procedures* de SQL nesta linguagem. Para contornar esta inconveniência, nas interrogações correspondentes a *procedures* – 3, 4 e 7 -, colocou-se o argumento diretamente dentro da função para propósitos de teste.

Como tal, foram criadas as seguintes *queries* em *Cypher:*

Requisito 1 - Top 3 atletas que ganharam mais provas

```
MATCH (p:Prova)-[:Disputada]→(a:Atleta)
where p.vencedor1 = a.idAtleta
return a.nome as atl,Count(p.vencedor1) as cnt
order by cnt DESC, a.nome
Limit 3;
```

Figura 27 – Código *Cypher* para o requisito 1

Requisito 2 - Vencedor mais novo de uma prova

```
MATCH (p:Prova)-[:Disputada]→(a:Atleta)
where p.vencedor1 = a.idAtleta
return a.nome as atl, a.data_nascimento
order by a.data_nascimento DESC , a.nome
Limit 1;
```

Figura 28 – Código *Cypher* para o requisito 2

Requisito 3 - Atletas numa prova

```
MATCH (p:Prova)-[r:Disputada]→(a:Atleta)-[:Pratica]→(m:Modalidade)
where p.idProva = 2
return p.idProva, a.nome, a.data_nascimento, a.nacionalidade ,

m.designacao;
```

Figura 29 – Código *Cypher* para o requisito 3

Requisito 4 – Provas em que certo atleta ficou no pódio

```
MATCH (p:Prova)-[r:Disputada]→(a:Atleta)-[:Pratica]→(m:Modalidade)
where a.idAtleta = 10 and (p.vencedor1 = 10 or p.vencedor2 = 10 or
p.vencedor3 = 10)
return a.nome, p , m.designacao;
```

Figura 30 – Código *Cypher* para o requisito 4

Requisito 5 – Provas ganhas por nacionalidade

```
MATCH (p:Prova)-[r:Disputada]→(a:Atleta)
where p.vencedor1 = a.idAtleta
return a.nacionalidade , Count(a.nacionalidade) as cnt
order by cnt DESC;
```

Figura 31 - Código *Cypher* para o requisito 5

Requisito 6 – Multas a pagar por atleta

```
MATCH (m:Multa)-[r:Paga_por]→(a:Atleta)
where m.idAtleta = a.idAtleta and m.flag = 'N'
return a.nome, m.idMulta, m.valor, m.flag
order by a.nome;
```

Figura 32 – Código *Cypher* para o requisito 6

Requisito 7 – Historial de multas de um certo atleta

```
MATCH (m:Multa)-[r:Paga_por]→(a:Atleta)
where m.idAtleta = 4
return a.nome, m.idMulta, m.valor, m.flag
order by a.nome;
```

Figura 33 – Código *Cypher* para o requisito 7

Requisito 8 – Médicos que deram mais consultas

```
MATCH (p:Teste)
where p.flag = "S"
return p.medico as Médico , Count(p.medico) as NrDeConsultasDadas
order by NrDeConsultasDadas DESC , p.medico
Limit 3;
```

Figura 34 – Código *Cypher* para o requisito 8

8. Conclusões e Trabalho Futuro

Este trabalho prático de Base de Dados acabou por se provar um projeto desafiante e mais complexo do que se esperava inicialmente, dado que foram necessárias várias correções e reformulações ao longo do seu desenvolvimento, pois só então nos dávamos conta de certos pormenores que nos obrigavam a voltar atrás e fazer alterações, bem como à nossa maneira de pensar. A aplicação dos conhecimentos obtidos nas aulas de Base de Dados foi essencial para o sucesso do grupo.

A fase inicial de contextualização do problema e apresentação do caso de estudo, objetivos e motivações provou-se das mais complicadas, dado que foi a parte onde nos foi dada mais liberdade criativa. Fomos encarregues de criar uma simulação o mais realista possível e decidir qual seria a estrutura do sistema a estudar e como os elementos do mesmo se relacionariam. Inventar um contexto tão expansivo de raiz que permitisse o desenvolvimento de uma base de dados de relativa complexidade sem, todavia, complicar demasiado provou-se um grande desafio e foi a causa de um início lento.

Isto também levou a que negligenciássemos de certa forma esta parte, com a avidez de fazer progresso mais substancial, avançando para as próximas antes de ter as arestas da mesma bem limadas, o que originou a necessidade de retroceder, eventualmente, e reformular certas partes do caso de estudo. Isto constitui uma valiosa lição de que o faseamento da criação de uma base de dados é fulcral e deve ser respeitado. Como se costuma dizer, “a pressa é inimiga da perfeição”.

Contudo, uma vez conquistada esta etapa, o resto do projeto foi bem mais fluido e linear, auxiliado da prática obtida da resolução de exercícios semelhantes nas aulas práticas. A modelação conceptual permitiu dissecar os requisitos levantados e extrair unicamente a informação relevante da simulação. A modelação lógica possibilitou o aprimoramento da estrutura do SGBDR, através do recurso a matérias como a normalização (que impulsionou uma das tais reestruturações do sistema). A modelação física contribuiu para o aprofundamento dos conhecimentos da linguagem SQL, bem como do respetivo SGBD usado *MySQL* e ambiente de trabalho *MySQL Workbench*.

O grupo acredita que o resultado final da primeira fase do trabalho – desenvolvimento de uma base de dados relacional – é bastante satisfatório e coeso, contudo não deixa de possuir potencial para ser ainda mais desenvolvido e melhorado. Possíveis aperfeiçoamentos incluem a abordagem dos funcionários da clínica como uma entidade independente e a consequente recompensa díspar conforme o trabalho observado (número de consultas), a distinção de

vários tipos de testes clínicos e listagem de preços mais precisa e o desenvolvimento de mais interrogações do utilizador úteis e realistas.

No que toca à segunda fase do projeto, a implementação de um SGBDNR acabou por se provar mais simples do que inicialmente esperado. A migração de dados de um contexto relacional para um ambiente não relacional acabou por ser trivial, com recurso a um tutorial oficial do próprio *Neo4j*, que está listado nas Referências deste relatório.

Já a programação de *queries* em *Cypher* relevou-se desafiante no início, dada a falta de hábito com a linguagem. Constatamos que tínhamos migrado mal a informação na primeira tentativa, pois tínhamos guardado todos os nodos com um identificador principal “id”, tornando-os impossíveis de distinguir, e este erro empatou-nos durante uma quantidade significativa de tempo, dado que mesmo a consola não imprimia nada ao testar *queries*.

Contudo, uma vez resolvido este impasse, o resto do projeto foi feito com relativa rapidez e desembaraço, dado que a solução não relacional baseada em grafos facilita bastante a representação e navegação pelos dados, o que torna também a implementação de interrogações em *Cypher* mais simples do que em *MySQL*.

Referências

Tutorial de migração do *Neo4j*:

- https://www.youtube.com/watch?v=IRTgsxL9V8g&t=629s

Tutorial de exportação de dados da *MySQL Workbench:*

- https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-table.html

Lista de Siglas e Acrónimos

BD	Base de Dados
SBD	Sistema de Base de Dados
SGBD	Sistema de Gestão de Base de Dados
SGBDR	Sistema de Gestão de Base de Dados Relacional
SGBDNR	Sistema de Gestão de Base de Dados Não Relacional

Anexos