



Universidade do Minho
Licenciatura em Engenharia Informática

Unidade Curricular de Bases de Dados

Ano Letivo de 2023/2024

Agência Veritatis

Grupo 3

Ana Cerqueira, Humberto Gomes, Ivo Vieira, José
Lopes, José Matos

Maio, 2024

BD

Data de Receção	
Responsável	
Avaliação	
Observações	

Agência *Veritatis*

Grupo 3

**Ana Cerqueira, Humberto Gomes, Ivo Vieira, José
Lopes, José Matos**

Maio, 2024

Resumo

O Prof. Doutor Elias Ribeiro, chefe da agência de detetives Agência *Veritatis*, encomendou a um grupo de alunos da Universidade do Minho o desenvolvimento de uma base de dados (BD) relacional para a gestão de casos, clientes e funcionários na sua agência. Este documento descreve o processo de desenvolvimento realizado pela equipa de alunos de Engenharia Informática.

Em primeiro lugar, o chefe da agência justificou a sua necessidade de uma BD e definiu o contexto em que esta seria desenvolvida, reunindo-se com a equipa informática para a discussão do plano de execução de trabalhos. Após a definição do método de levantamento de requisitos, estes foram levantados com a agência e posteriormente organizados. Seguiu-se a construção e documentação de um diagrama Entidade-Relacionamento (ER), que foi convertido num modelo lógico relacional, normalizado até à terceira forma normal (3FN) e posteriormente validado através da implementação de interrogações com recurso a álgebra relacional. Este modelo foi convertido para um modelo físico no Sistema de Gestão de Bases de Dados (SGBD) MySQL. Foram criados vários utilizadores da BD com diferentes privilégios, a BD foi povoada, e foram implementadas as interrogações e alguns procedimentos, funções e gatilhos essenciais. Após estar funcional, a BD viu o seu desempenho melhorado através de indexação. A equipa deve agora continuar a prestar serviços à Agência *Veritatis*, garantido a operacionalidade da BD e procurando melhorar o seu funcionamento com base no comentário crítico dos funcionários que a utilizam.

Área de Aplicação: Desenho e Arquitetura de Bases de Dados

Palavras-Chave: Bases de Dados, Diagramas ER, Modelo Relacional, Normalização de Bases de Dados Relacionais, Álgebra Relacional, SQL, MySQL, InnoDB

Índice

1. Definição do Sistema	1
1.1. Contexto de Aplicação	1
1.2. Motivação e Objetivos do Trabalho	1
1.3. Análise da Viabilidade do Processo	2
1.4. Recursos e Equipa de Trabalho	3
1.5. Plano de Execução do Projeto	3
1.6. Revisão e Aprovação do Projeto	4
2. Levantamento e Análise de Requisitos	6
2.1. Método de Levantamento e de Análise de Requisitos Adotado	6
2.2. Organização dos Requisitos Levantados	8
2.3. Análise e Validação Geral dos Requisitos	10
3. Modelação Conceitual	11
3.1. Apresentação da Abordagem de Modelação Realizada	11
3.2. Identificação e Caracterização das Entidades	11
3.3. Identificação e Caracterização dos Relacionamentos	13
3.4. Identificação e Caracterização dos Atributos das Entidades	15
3.5. Apresentação e Explicação do Diagrama ER Produzido	21
3.6. Validação do Modelo Conceitual	22
4. Modelação Lógica	23
4.1. Construção do Modelo de Dados Lógico	23
4.2. Apresentação e Explicação do Modelo Lógico Produzido	23
4.3. Normalização de Dados	29
4.4. Validação do Modelo com Interrogações do Utilizador	33
5. Implementação Física	43
5.1. Apresentação e Explicação da Base de Dados Implementada	43
5.2. Criação de Utilizadores da Base de Dados	49
5.3. Povoamento da Base de Dados	55
5.4. Cálculo do Espaço da Base de Dados	61
5.5. Definição e Caracterização de Vistas de Utilização em SQL	68
5.6. Tradução das interrogações do utilizador para SQL	70

5.7. Indexação do Sistema de Dados	76
5.8. Implementação de Procedimentos, Funções e Gatilhos	78
6. Conclusões e Trabalho Futuro	85
Referências	88
Lista de Siglas e Acrónimos	89

Anexos

I. Diagrama de Gantt – Planeamento de Tarefas	91
II. Requisitos Levantados	94
III. Requisitos Categorizados	97
IV. Relações RelaX	101
V. Expressões de Álgebra Relacional	105
VI. Modelo Físico	107
VII. Script de criação de utilizadores	111
VIII. Povoamento da BD usando a LMD de SQL	116
IX. Povoamento da BD usando um programa externo	120
X. Medições de espaço resultantes de inserções de registo	123
XI. Índices Criados	125
XII. Procedimentos, Funções e Gatilhos	127
XIII. Diagrama de Gantt – Tarefas Realizadas	132

Índice de Figuras

Figura 1 - Miniatura do diagrama de Gantt relativo ao planeamento de tarefas.....	4
Figura 2 - Cabeçalho da ata da reunião de construção do cronograma para o desenvolvimento da BD.....	4
Figura 3 - Cabeçalho da ata da reunião de revisão do projeto da BD.....	5
Figura 4 - Cabeçalho da ata da primeira reunião de levantamento de requisitos.....	6
Figura 5 - Inquérito ao Professor Doutor Elias Ribeiro.....	7
Figura 6 - Primeiros requisitos da tabela global.....	8
Figura 7 - Primeiros requisitos de descrição.....	9
Figura 8 - Primeiros requisitos de manipulação.....	9
Figura 9 - Primeiros requisitos de controlo.....	9
Figura 10 - Cabeçalho da ata da reunião de validação de requisitos.....	10
Figura 11 - Requisitos de descrição que dão origem aos atributos de “Cliente”.....	16
Figura 12 - Diagrama conceitual após a realização da sua primeira fase de construção.....	21
Figura 13 - Versão final do diagrama ER produzido pela equipa de alunos.....	22
Figura 14 - Exemplo da conversão automática de um atributo multivalorado (“Procedimento.Provas”).	24
Figura 15 - Versão final do modelo lógico produzido pela equipa de alunos.....	25
Figura 16 - Relação “Exerce”.....	30
Figura 17 - Relação “Provas”	31
Figura 18 - Relações “Funcao”, “Emails” e “Telefones”	31
Figura 19 - Relação “Procedimento”.....	32
Figura 20 - Relações “Cliente” e “Funcionario”.....	32
Figura 21 - Relações “TipoProcedimento” e “Caso”	33
Figura 22 - Requisitos de manipulação que deram origem às interrogações.....	34
Figura 23 - Árvore da interrogação do RM19 aplicada aos dados de teste.....	35
Figura 24 - Árvore da interrogação do RM2 aplicada aos dados de teste.....	36
Figura 25 - Árvore da interrogação do RM4 aplicada aos dados de teste.....	36
Figura 26 - Árvore da interrogação do RM7 aplicada aos dados de teste, para o “Cliente” de “Id” 1.....	37
Figura 27 - Árvore da interrogação do RM15 aplicada aos dados de teste, para a “Funcao” de “Id” 4.....	37
Figura 28 - Árvore da interrogação do RM10 aplicada aos dados de teste, para a “Caso” de “Id” 6.....	38

Figura 29 - Árvore da interrogação do RM17 aplicada aos dados de teste, para pesquisar “Cliente”s de “Nome” “Miguel”.....	39
Figura 30 - Árvore da interrogação do RM13 aplicada aos dados de teste, para datas superiores a 2020-01-01.....	40
Figura 31 - Árvore da interrogação do RM14 aplicada aos dados de teste.....	41
Figura 32 - Árvore da interrogação do RM18 aplicada aos dados de teste, para o “Caso” de “Id” 6.....	42
Figura 33 - Grafo do diagrama de dependências de criação da BD.....	44
Figura 34 - Mapa de gatilhos da BD <i>Veritatis</i>	84
Figura 35 - Miniatura do diagrama de Gantt relativo às atividades realizadas na primeira fase.....	86
Figura 36 - Miniatura do diagrama de Gantt relativo às atividades realizadas na segunda fase.....	87

Índice de Tabelas

Tabela 1 - Entidades identificadas pela equipa de desenvolvimento.....	13
Tabela 2 - Relacionamentos identificados pela equipa de desenvolvimento.....	15
Tabela 3 - Atributos da entidade “Cliente”	17
Tabela 4 - Atributos da entidade “Funcionario”.....	18
Tabela 5 - Atributos da entidade “Funcao”.....	19
Tabela 6 - Atributos da entidade “Caso”.....	19
Tabela 7 - Atributos da entidade “Procedimento”.....	20
Tabela 8 - Atributos da entidade “TipoProcedimento”.....	21
Tabela 9 - Dicionário de dados da relação “Cliente”.....	26
Tabela 10 - Dicionário de dados da relação “Emails”	26
Tabela 11 - Dicionário de dados da relação “Telefones”	26
Tabela 12 - Dicionário de dados da relação “Caso”.....	27
Tabela 13 - Dicionário de dados da relação “Funcionario”.....	27
Tabela 14 - Dicionário de dados da relação “Funcao”	28
Tabela 15 - Dicionário de dados da relação “Exerce”	28
Tabela 16 - Dicionário de dados da relação “TipoProcedimento”	28
Tabela 17 - Dicionário de dados da relação “Procedimento”	29
Tabela 18 - Dicionário de dados da relação “Provas”	29
Tabela 19 - Mapeamento de domínios entre MySQL e RelaX.....	33
Tabela 20 - Operações de execução permitidas na BD.....	54
Tabela 21 - Associação entre tipos de dados de MySQL e espaço consumido pelo InnoDB.....	61
Tabela 22 - Espaço ocupado pelos atributos de cada registo da relação “Funcao”	63
Tabela 23 - Espaço ocupado pelos atributos de cada registo da relação “Funcionario”	63
Tabela 24 - Espaço ocupado pelos atributos de cada registo da relação “Exerce”	63
Tabela 25 - Espaço ocupado pelos atributos de cada registo da relação “Cliente”	63
Tabela 26 - Espaço ocupado pelos atributos de cada registo da relação “Emails”	64
Tabela 27 - Espaço ocupado pelos atributos de cada registo da relação “Telefones”	64
Tabela 28 - Espaço ocupado pelos atributos de cada registo da relação “Caso”	64
Tabela 29 - Espaço ocupado pelos atributos de cada registo da relação “TipoProcedimento”	64

Tabela 30 - Espaço ocupado pelos atributos de cada registo da relação “Procedimento”	65
Tabela 31 - Espaço ocupado pelos atributos de cada registo da relação “Provas”.....	65
Tabela 32 - Tamanho ocupado por cada relação na BD <i>Veritatis</i> (usando o tamanho teórico dos registos).	65
Tabela 33 - Evolução do tamanho da BD <i>Veritatis</i> (tamanho teórico dos registos).....	66
Tabela 34 - Alguns dos atributos do resultado de SHOW TABLE STATUS IN <i>Veritatis</i>	66
Tabela 35 - Tamanho ocupado por cada relação na BD <i>Veritatis</i> (tamanho dos registos medido experimentalmente).....	68
Tabela 36 - Medições experimentais dos tamanhos médios de cada registo, tendo em conta os tamanhos das relações medidos.....	124

1. Definição do Sistema

1.1. Contexto de Aplicação

Em 2008, o Prof. Doutor Elias Ribeiro, professor Catedrático de Criminologia na Universidade do Minho, sofreu uma tentativa de homicídio após atribuir más classificações aos maiores rufiões de Braga. A polícia descobriu os planos dos alunos antes de porem o seu delito em prática, mas o Prof. Doutor Elias foi motivado a fundar a agência de detetives Agência *Veritatis*, para investigar casos semelhantes que possam ocorrer com os seus colegas.

A agência, sediada em Gualtar, tem vindo a crescer desde então, e conta já com o secretário Jacinto Fonseca e seis detetives, chefiados por Orlando Feio. Com um foco especial na região Norte, a agência trabalha para garantir um ambiente seguro para o pessoal docente em diversas instituições de ensino superior. O Prof. Doutor Elias Ribeiro procurou a ajuda de um grupo de alunos confiáveis de Engenharia Informática para desenvolver um Sistema de Bases de Dados (SBD) para a sua agência.

1.2. Motivação e Objetivos do Trabalho

O sucesso do projeto *Veritatis* levou ao esforço crescente dos alunos de criminologia, que por sua vez se aplicaram aos estudos de forma a produzir melhores tentativas de assassinato. A palavra espalhou-se rapidamente, e até alunos de outros cursos pediam auxílio aos estudantes de criminologia para a orquestração dos seus planos malévolos. Como consequência, a agência cresceu exponencialmente de forma a suportar todas as tentativas de homicídio que se desencadeavam pelas universidades.

Agora mais do que nunca, qualquer mínima falha de informação pode pôr em causa a vida das vítimas. Esta agravante da dificuldade de investigação foi acompanhada de limitações de recursos humanos na Agência *Veritatis*, principalmente de detetives, pelo que maximizar a rentabilidade do seu tempo se torna essencial. Além do mais, as aplicações Microsoft Office usadas na agência até então provaram-se inviáveis, não só devido à dificuldade de gestão de dados com a crescente expansão do projeto, mas também a uma preocupação acrescida na recolha indevida de dados privados pela Microsoft. Assim, o Prof. Doutor Elias Ribeiro julga que um SBD seja essencial para a sua agência, tendo em vista o seu crescimento e o do número de tentativas (e sucessos) de homicídios de colegas docentes.

Após o recrutamento dos cinco alunos de Engenharia Informática, o Prof. Doutor Elias pediu-lhes que desenvolvessem um SBD que se adaptasse ao tipo de casos pelos quais a sua agência é responsável, investigações de assassinatos de professores. Com a implementação do SBD, o Prof. Doutor Elias pretende:

- Facilitar a gestão e registo de provas. Uma organização eficiente facilita a recuperação e análise dos dados quando necessário, reduz o risco de serem perdidos, e diminui o tempo gasto pelos detetives nesta tarefa.
- Armazenar de um modo estruturado o estado de uma investigação (que tarefas já foram e não foram feitas), pois tal tornará mais fácil a organização para os detetives, tornando-os mais eficientes na resolução de um caso.
- Obter uma ferramenta útil para a gestão eficiente da agência, tanto dos recursos financeiros como humanos, que acelere o seu crescimento.
- Tornar mais simples a gestão e contacto dos clientes, para evitar erros e de secretaria.

1.3. Análise da Viabilidade do Processo

A agência de investigação, com funcionamento com base em folhas de cálculo *Excel* e documentos *Word*, apresenta várias falhas e problemas de produtividade, que foram documentados pelos funcionários através da sua experiência e da análise de documentação interna anterior. Deste modo, a Agência *Veritatis* estima que, ao ter um sistema mais eficiente de armazenamento e gestão de dados, conseguirá:

- Aumentar a produtividade de todos os agentes em cerca de 20%, devido a menores tempos de pesquisa de informação (em comparação com as consultas dos vários documentos em vigor);
- Diminuir em 10% os erros de dados relativos a casos (que se devem a perdas e incorreções de informação com o sistema atual);
- Aumentar em cerca de 3% os casos corretamente investigados, consequência do ponto anterior;
- Diminuir em 15% as despesas com serviços externos (como os serviços na nuvem da Microsoft).
- Aumentar a segurança dos seus sistemas (implementação de uma base de dados fechada sob controlo da agência).

Assim, o Prof. Doutor Elias concluiu que a implementação de um SBD é viável e rentável para a agência, desde que não acarrete muitos custos com o seu desenvolvimento. Foi neste momento que decidiu que mão de obra barata de engenheiros em formação seria ideal para a construção da BD.

1.4. Recursos e Equipa de Trabalho

Os recursos humanos necessários para o projeto, divididos pelas categorias “pessoal interno” e “pessoal externo”, são constituídos por funcionários da Agência *Veritatis* e pela equipa de desenvolvimento informático.

O “pessoal interno” é composto pelos seguintes indivíduos, que representam a equipa da Agência *Veritatis* participante no processo de desenvolvimento do SBD.

- Prof. Doutor Elias Ribeiro: fundador, líder e gestor da agência. Tem uma vasta experiência em investigação criminal e conhecimento profundo das necessidades da sua empresa.
- Jacinto Fonseca: secretário da agência, responsável pelo atendimento aos clientes. Apresenta habilidades organizacionais e de comunicação.
- Orlando Feio: detetive principal, com vasta experiência em criminologia. É encarregue pela coordenação das operações em campo.

O “pessoal externo” é constituído pela equipa de alunos de Engenharia Informática escolhida pelo chefe da agência, responsável pelo desenvolvimento do SBD, pela sua implementação e testagem. Apresentam conhecimentos sólidos de programação, *design* de bases de dados, e metodologias de desenvolvimento. Os nomes dos integrantes desta equipa são Ana Cerqueira, Humberto Gomes, Ivo Vieira, José Lopes e José Matos.

Já os recursos materiais podem dividir-se em duas categorias, *software* e *hardware*. Em termos de *software*, é necessária uma licença para uma aplicação de gestão de projetos, uma aplicação de construção de diagramas ER, outra para a conceção de modelos lógicos, e um sistema de gestão de base de dados (SGBD). Em termos de *hardware*, é fundamental existirem 5 computadores pessoais, um por cada elemento da equipa de desenvolvimento, com o propósito de servirem a elaboração do SBD. Estes, tal como o sistema operativo nos mesmos (parte do *software*), serão providenciadas pelos próprios alunos.

1.5. Plano de Execução do Projeto

A equipa de alunos reuniu-se com o Prof. Doutor Elias Ribeiro para planear o desenvolvimento do SBD. Dessa reunião resultou o diagrama de Gantt no [primeiro anexo](#), onde se definiu o período de execução e os vários intervenientes de cada tarefa, tendo em conta os prazos de entrega solicitados pelo Prof. Doutor Elias e os períodos em que os alunos não se poderiam dedicar a tempo inteiro ao projeto da agência. Temendo os atrasos que muitos projetos de engenharia sofrem, a planificação conta com margem de manobra, caso uma das etapas não fique pronta a tempo.

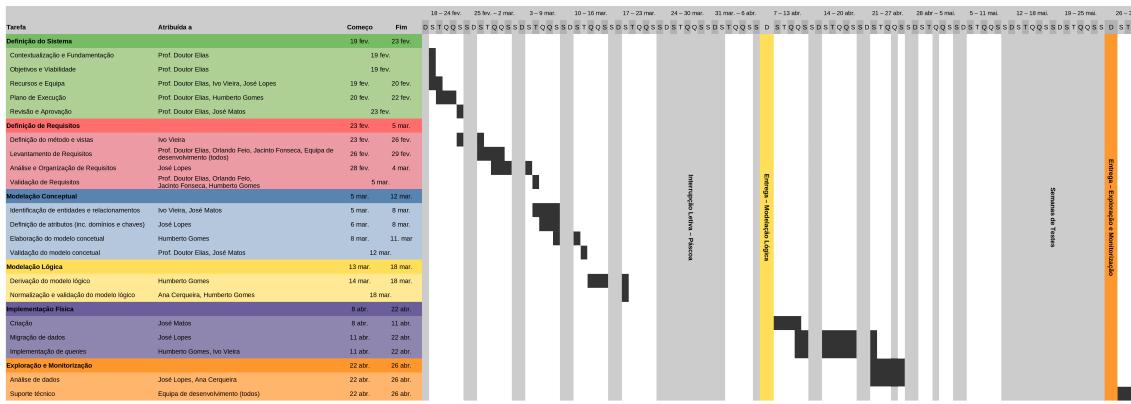


Figura 1 - Miniatura do diagrama de Gantt relativo ao planeamento de tarefas



Cronograma para a BD

DATA: Terça-feira, 20/02/2024

HORÁRIO: 10h00

1. OBJETIVO DA REUNIÃO

Construção do cronograma de tarefas para o desenvolvimento da BD, e a sua atribuição a elementos da equipa de desenvolvimento.

2. PARTICIPANTES PRESENTES

NOME	POSIÇÃO
Prof. Doutor Elias Ribeiro	Líder da Agência Veritatis
Humberto Gomes	Membro da equipa de desenvolvimento

Figura 2 - Cabeçalho da ata da reunião de construção do cronograma para o desenvolvimento da BD.

1.6. Revisão e Aprovação do Projeto

Após melhorar o grafismo do plano de execução, a equipa de engenheiros informáticos da Universidade do Minho agendou uma reunião com o chefe da agência de detetives, onde foi reiterado o plano do projeto a desenvolver. O Prof. Doutor Elias Ribeiro confirmou tudo o que foi definido. O gestor da agência decidiu então fechar o contrato com a equipa de engenheiros num valor de 0€, para desenvolver o SBD proposto.



Revisão do projeto da BD

DATA: Sexta-feira, 23/02/2024
HORÁRIO: 10h00

1. OBJETIVO DA REUNIÃO

Revisão e possível aprovação do desenvolvimento da base de dados para a agência.

2. PARTICIPANTES PRESENTES

NOME	POSIÇÃO
Prof. Doutor Elias Ribeiro	Líder da Agência <i>Veritatis</i>
José Matos	Membro da equipa de desenvolvimento

Figura 3 - Cabeçalho da ata da reunião de revisão do projeto da BD.

2. Levantamento e Análise de Requisitos

2.1. Método de Levantamento e de Análise de Requisitos Adotado

Para o levantamento de requisitos, a equipa de desenvolvimento adotou diversas estratégias. Desde cedo os seus integrantes se aperceberam que dois métodos normalmente utilizados estariam fora de questão: investigação do estado da arte (devido à inexistência de outras agências de investigação próximas de Braga), e observação do funcionamento da agência (devido à confidencialidade dos dados nesta tratados). Inicialmente, estavam planeadas reuniões com elementos da Agência *Veritatis* e a análise de documentação antiga providenciada pela mesma, mas acabou por ser necessária também a realização de um questionário ao Prof. Doutor Elias.

Reuniões com a Equipa da Agência

Foram marcadas duas reuniões com o Prof. Doutor Elias Ribeiro, o secretário Jacinto Fonseca, e o chefe de detetives Orlando Feio. Segue-se o cabeçalho da ata da primeira reunião realizada:

 AGÊNCIA VERITATIS	Levantamento de requisitos										
	DATA: Segunda-feira, 26/02/2024										
	HORÁRIO: 10h00										
1. OBJETIVO DA REUNIÃO											
Comunicação do que é pretendido pela agência com o SBD.											
2. PARTICIPANTES PRESENTES											
<table border="1"><thead><tr><th>NOME</th><th>POSIÇÃO</th></tr></thead><tbody><tr><td>Prof. Doutor Elias Ribeiro</td><td>Líder da Agência <i>Veritatis</i></td></tr><tr><td>Jacinto Fonseca</td><td>Secretário Agência <i>Veritatis</i></td></tr><tr><td>Orlando Feio</td><td>Detetive chefe da Agência <i>Veritatis</i></td></tr><tr><td>José Lopes</td><td>Membro da equipa de desenvolvimento</td></tr></tbody></table>		NOME	POSIÇÃO	Prof. Doutor Elias Ribeiro	Líder da Agência <i>Veritatis</i>	Jacinto Fonseca	Secretário Agência <i>Veritatis</i>	Orlando Feio	Detetive chefe da Agência <i>Veritatis</i>	José Lopes	Membro da equipa de desenvolvimento
NOME	POSIÇÃO										
Prof. Doutor Elias Ribeiro	Líder da Agência <i>Veritatis</i>										
Jacinto Fonseca	Secretário Agência <i>Veritatis</i>										
Orlando Feio	Detetive chefe da Agência <i>Veritatis</i>										
José Lopes	Membro da equipa de desenvolvimento										

Figura 4 - Cabeçalho da ata da primeira reunião de levantamento de requisitos.

Nestas reuniões, a equipa da agência explicitou o que pretendia com o SBD a ser desenvolvido: explicou como a agência operava, o que era necessário armazenar e consultar, e como devia ser controlado o acesso aos dados sensíveis, sendo guiados pelos alunos para fornecerem todos os dados essenciais com correção e completude. Exemplos deste modo de guiar a discussão incluem:

- Sempre que uma operação de manipulação de base de dados é mencionada por um membro da Agência Veritatis, é perguntado que utilizadores a podem executar;
- Sempre que surgem novos conceitos que poderão vir a dar origem a entidades no modelo ER, a equipa de informática pergunta que funcionários as inserem na BD;
- A discussão tem de ser controlada para não as ideias da agência serem abordadas uma de cada vez de acordo com um fio condutor, para que o documento de requisitos não seja uma sequência de pontos nada relacionados com os seus adjacentes;
- Quando necessário, perguntas sobre valores que dadas características podem assumir são colocadas (como, por exemplo, o valor máximo de um salário), de modo a permitir a futura definição dos domínios dos atributos.

A equipa de informáticos anotou a informação necessária para o futuro desenvolvimento da BD, bem como as suas fontes.

Inquérito ao Prof. Doutor Elias Ribeiro

Como o gestor da agência não pôde estar presente na segunda reunião de levantamento de requisitos, foi-lhe enviado um inquérito com perguntas relativas ao seu trabalho: que tipo de tarefas executava, o que julgava que o SBD poderia fazer para o ajudar, etc., que pode ser visto abaixo:



Inquérito para a BD

Para: Professor Doutor Elias Ribeiro

1. Quais são suas expectativas para o SBD, em particular quanto ao seu trabalho de gestor?
2. Descreva o seu dia de trabalho e as tarefas que executa.
3. Quais são as tarefas que lhe consomem mais tempo, mas julga que possam ser automatizadas?
4. Quem tem permissões para executar cada operação que mencionou nas suas respostas?
5. Há algum requisito da reunião anterior que tenha reconsiderado e deva ser alterado?

Figura 5 - Inquérito ao Professor Doutor Elias Ribeiro.

Com base nas respostas do gestor da Agência, o analista Ivo Vieira formulou vários requisitos para o projeto, operações de manipulação da base de dados acompanhadas de informação sobre os utilizadores que tinham permissão para as executar.

Análise de documentação interna

Devido ao secretismo das operações a decorrer na agência, não foi possível aos alunos observarem os funcionários no seu quotidiano. No entanto, foi-lhes dado acesso a documentos relativos a casos já resolvidos. A equipa informática utilizou-os para extrair ainda mais informação sobre o funcionamento interno da agência, e formular modos como um SBD poderia aumentar a eficiência na Agência Veritatis. Em particular, o analista José Matos procurou por erros que ocorreram durante a investigação do caso “Algoritmos e Complexidade”, que resultaram em novas operações de manipulação à BD e restrições de acesso sobre quem as pode executar, dado que a sua inexistência causou problemas durante esta investigação em particular. A documentação analisada não pode ser divulgada aos docentes da Unidade Curricular de Bases de Dados dos alunos, possivelmente os leitores deste relatório, devido ao acordo de não-divulgação assinado.

2.2. Organização dos Requisitos Levantados

À medida que o processo de levantamento de requisitos findava, a equipa informática, partindo da tabela global de requisitos, começou a categorizá-los em requisitos de descrição (RD), de manipulação (RM) e de controlo (RC), conforme o tipo de operações a que estão associados: criação da base de dados, manipulação da mesma, e controlo do acesso aos dados por diferentes utilizadores, respetivamente. Seguem-se os primeiros requisitos da tabela global e de cada tipo. As listas completas podem ser encontradas nos anexos [II](#) e [III](#), respetivamente.

N.º	Data	Hora	Descrição	Vista de Utilização	Fonte	Analista	Substituído
1	26 fev.	10:10	Um cliente é caracterizado pelo seu nome, morada (opcional), NIF, contactos, e local de trabalho	Clientes	Jacinto Fonseca	José Lopes	
2	26 fev.	10:10	Oe contactos de um cliente são constituídos por um endereço de correio eletrónico obrigatório e um número de telefone opcional	Clientes	Jacinto Fonseca	José Lopes	34
3	26 fev.	10:12	Um professor deve ser identificado por um número sequencial	Clientes	Jacinto Fonseca	José Lopes	
4	26 fev.	10:14	Apenas secretários podem registrar novos clientes e alterar os dados dos existentes	Clientes / Gestão	Elias Ribeiro	José Lopes	

Figura 6 - Primeiros requisitos da tabela global.

N.º	Requisitos Originais	Descrição	Vista de Utilização	Revisor
RD1	1	Um cliente é caracterizado pelo seu nome, morada (opcional), NIF, contactos, e local de trabalho	Clientes	Humberto Gomes
RD2	3	Um professor deve ser identificado por um número sequencial	Clientes	Humberto Gomes
RD3	6	Um cliente encomenda a investigação de um ou mais casos, e um caso pode apenas ter um cliente associado	Casos / Clientes	Humberto Gomes
RD4	8	Um funcionário é caracterizado pelo seu nome, um número sequencial, NIF, salário (inferior a 10000€), contactos e apólice de seguro de vida (caso seja um detetive)	Gestão	Humberto Gomes
RD5	9	Os contactos de um funcionário são constituídos por um endereço de e-mail obrigatório e um número de telefone opcional	Gestão	Humberto Gomes

Figura 7 - Primeiros requisitos de descrição.

N.º	Requisitos Originais	Descrição	Vista de Utilização	Revisor
RM1	4	Registrar novos clientes e alterar os dados dos existentes	Clientes	Humberto Gomes
RM2	5	Listar todos os professores com casos atualmente em aberto (identificador do cliente e do caso, nome do cliente, designação do caso e data de abertura, ordenados por nome do cliente)	Clientes	Humberto Gomes
RM3	7	Registrar novos casos	Casos	Humberto Gomes
RM4	12	Enumerar todos os tipos de trabalhador existentes associados a pelo menos um funcionário (identificador e designação)	Gestão	Humberto Gomes
RM5	13	Inserir novas funções de funcionários	Gestão	Humberto Gomes

Figura 8 - Primeiros requisitos de manipulação.

N.º	Requisitos Originais	Descrição	Vista de Utilização	Revisor
RC1	4	Apenas secretários podem registrar novos clientes e alterar os dados dos existentes	Clientes / Gestão	Humberto Gomes
RC2	7	Apenas um administrativo é capaz de registrar novos casos	Casos / Gestão	Humberto Gomes
RC3	13	Apenas gestores podem inserir novas funções de funcionários	Gestão	Humberto Gomes
RC4	14	Apenas gestores podem adicionar novos funcionários e modificar ou remover os funcionários existentes	Gestão	Humberto Gomes
RC5	25	Apenas detetives podem inserir procedimentos, sendo os seus custos derivados do seu tipo	Casos / Gestão	José Lopes

Figura 9 - Primeiros requisitos de controlo.

Ao longo do processo de organização de requisitos, foram estabelecidas três vistas de utilização distintas, que foram denominadas “Casos”, “Clientes” e “Gestão”.

A vista “Casos” está relacionada com a recolha e manipulação de dados necessários para a investigação de casos de assassinato de professores universitários. A maioria das necessidades que compõe esta vista de utilização está relacionada com o trabalho dos detetives. Os dados relativos aos Casos são os mais sigilosos, o que implica restrições à sua consulta e manipulação.

A vista de utilização “Clientes” concerne à interação com os clientes da Agência *Veritatis*. Os clientes da agência não têm acesso direto à BD, mas a interação com clientes exige funcionalidades do SBD que serão utilizadas por funcionários administrativos.

A vista “Gestão” tem que ver com a gestão de funcionários, passando pela sua contratação, pagamento, escalonamento de tarefas, organização horária e permissões de acesso ao SBD. Esta vista também tem em consideração os aspetos financeiros da empresa, de forma a serem tomadas as melhores decisões possíveis para o negócio. São os administradores da empresa (no caso, o Prof. Doutor Elias Ribeiro) os interessados nestas funcionalidades.

2.3. Análise e Validação Geral dos Requisitos

Após uma minuciosa revisão do documento de requisitos por parte da equipa informática, que não resultou em qualquer inconsistência ou classificação errada descoberta, foi realizada uma reunião com todos os intervenientes da Agência *Veritatis* para validar todas as vertentes de utilização e cada requisito.

 AGÊNCIA VERITATIS	Validação de requisitos										
	DATA: Quinta-feira, 07/03/2024										
	HORÁRIO: 10h00										
1. OBJETIVO DA REUNIÃO											
Validação dos requisitos levantados.											
2. PARTICIPANTES PRESENTES											
<table><thead><tr><th>NOME</th><th>POSIÇÃO</th></tr></thead><tbody><tr><td>Prof. Doutor Elias Ribeiro</td><td>Líder da Agência <i>Veritatis</i></td></tr><tr><td>Jacinto Fonseca</td><td>Secretário Agência <i>Veritatis</i></td></tr><tr><td>Orlando Feio</td><td>Detetive chefe da Agência <i>Veritatis</i></td></tr><tr><td>Humberto Gomes</td><td>Membro da equipa de desenvolvimento</td></tr></tbody></table>		NOME	POSIÇÃO	Prof. Doutor Elias Ribeiro	Líder da Agência <i>Veritatis</i>	Jacinto Fonseca	Secretário Agência <i>Veritatis</i>	Orlando Feio	Detetive chefe da Agência <i>Veritatis</i>	Humberto Gomes	Membro da equipa de desenvolvimento
NOME	POSIÇÃO										
Prof. Doutor Elias Ribeiro	Líder da Agência <i>Veritatis</i>										
Jacinto Fonseca	Secretário Agência <i>Veritatis</i>										
Orlando Feio	Detetive chefe da Agência <i>Veritatis</i>										
Humberto Gomes	Membro da equipa de desenvolvimento										

Figura 10 - Cabeçalho da ata da reunião de validação de requisitos.

Todos se dedicaram para garantir que tudo estava correto na documentação. Houve lugar para a discussão de ideias, o que permitiu a identificação e resolução de erros resultantes de má comunicação anterior, e a adição de algumas ideias que tinham ficado por mencionar, já adicionadas ao documento em anexo pelo analista Humberto Gomes. No final deste processo de revisão exaustiva, todos os intervenientes confirmaram de forma unânime os requisitos levantados. Este passo crucial não só assegurou a qualidade e clareza da documentação, mas também estabeleceu uma base sólida para o desenvolvimento subsequente do projeto.

3. Modelação Concretual

3.1. Apresentação da Abordagem de Modelação Realizada

Após a recolha e análise dos requisitos, a equipa de desenvolvimento iniciou o planeamento da estrutura e design da BD. Começaram pela construção de um diagrama ER, uma representação visual de entidades num sistema, e como estas se relacionam entre si. Para a sua criação, foi utilizado o *brModelo* (Cândido, 2020), que produz diagramas numa variante da notação Chen, e com o qual os integrantes da equipa tinham experiência prévia. A equipa começou por identificar as entidades e, de seguida, os relacionamentos entre elas. Posteriormente, descreveu os atributos das entidades e caracterizou os relacionamentos.

É importante destacar que foi utilizada uma abordagem monovista. Esta metodologia implica a construção de um modelo único, tendo em conta todas as vistas de utilização em simultâneo, algo que se considerou adequado devido ao número de requisitos relativamente baixo.

Será já descrito um processo aplicado a todas as entidades e relacionamentos identificados, bem como aos seus atributos: as designações destes foram escritas sem espaços, diacríticos ou preposições, para originar nomes mais simples, iguais aos do modelo lógico descrito posteriormente e futuro modelo físico, onde caracteres especiais exigem notação SQL inconveniente.

3.2. Identificação e Caracterização das Entidades

O desenvolvimento de uma BD exige uma identificação precisa das entidades que a compõem. Estas servem como base para a organização da BD, sendo a sua definição necessária antes da identificação de relacionamentos e de atributos. Uma entidade refere-se a um conjunto de objetos com as mesmas propriedades, objetos esses com uma existência independente, seja esta física (tangível) ou apenas conceitual (Connolly & Begg, 2015, pp. 406-407). Esta definição vaga conduz a que possa haver diferentes conjuntos de entidades deduzidos a partir do mesmo documento de requisitos, pelo que a existência de cada entidade é devidamente justificada neste documento. O processo de identificação de entidades consistiu na leitura ordenada dos requisitos, cada um acompanhado de uma ponderação

sobre se define ou não uma entidade. Foi necessária a consideração de requisitos que não de descrição, pois, como será abordado, estes também tiveram papeis esclarecedores para a definição de entidades. A primeira entidade, identificada logo em RD1, é “Cliente”, uma entidade tangível relativa aos indivíduos que fazem uso dos serviços da Agência Veritatis. O “Cliente” é o pilar fundamental da vista de utilização “Clientes”. Dado que vários clientes existem independentemente e partilham as várias propriedades enumeradas pelo requisito de descrição mencionado, o “Cliente” foi considerado uma entidade. Ao longo dos requisitos, como ocorre por exemplo em RD2, verifica-se que o termo “professor” é utilizado como sinónimo de “Cliente”.

De seguida, em RD4 identifica-se uma entidade à qual se dá o nome “Funcionario”, peça vital para a gestão de recursos humanos (RH) na agência (vista “Gestão”), de existência tangível e independente, cujas características, enumeradas em RD4, são comuns aos trabalhadores na agência.

A entidade “Funcao”, descrita em RD7, é introduzida para caracterizar as tarefas / cargos de cada “Funcionario” (por exemplo, “Detetive”, “Gestor de Recursos Humanos”, etc.). Apesar da sua existência intangível, não se justifica que “Funcao” seja um mero atributo de “Funcionario”. Em primeiro lugar, a linguagem da caracterização desta última entidade, em RD4, não indica que um funcionário seja caracterizado por uma função. Ademais, o requisito RM4, a enumeração de funções com funcionários associados, sugere fortemente a existência de funções sem funcionários associados, que por sua vez permite a conclusão da existência de “Funcao” independente da existência de “Funcionario”. A linguagem deste mesmo requisito mostra que “tipo de trabalhador” é utilizado como sinónimo de “Funcao”. Por último, apesar de, neste momento, a “Funcao” apenas se caracterizar por um identificador e uma designação, a equipa de desenvolvimento acredita veementemente que, no futuro, com o crescimento da agência e o aumento da complexidade da gestão de RH (vista “Gestão”), as características desta entidade poderão vir a ser muitas mais, sendo a sua existência como entidade justificável desde já.

De seguida, “Caso” foi uma entidade identificada em RD8, que representa o único tipo de serviço prestado pela Agência Veritatis, tendo um papel de elevadíssima importância na BD e, em particular, na vista de utilização “Casos”. Foi considerada uma entidade dado que vários casos existem de forma independente e partilham entre si as mesmas características, descritas em RD8. Ao longo dos requisitos, como se verifica em RD11, “investigação” apresenta o mesmo significado que “Caso”.

Foi também identificada a entidade “Procedimento”, mencionada em RD12: procedimentos são distintamente identificáveis e todos os procedimentos partilham o mesmo conjunto de características, enunciadas pelo requisito de descrição referenciado.

Por outro lado, menciona-se o exemplo da prova, referida em RD15, que não foi considerada uma entidade, mas sim, como se verá adiante, um atributo de “Procedimento”, pois uma prova depende unicamente desta entidade para a sua existência, sendo uma característica da mesma.

Por último, a entidade “TipoProcedimento” surge em RD22 para categorizar instâncias de “Procedimento”. Todo o “TipoProcedimento” é caracterizado do mesmo modo, existindo de forma independente, pelo que “TipoProcedimento” constitui uma entidade. Deve ser notado que a existência

de “TipoProcedimento” é independente da de “Procedimento” (algo explicado na secção abaixo, [Identificação e Caracterização dos Relacionamentos](#), evidenciado por RC5 e RC6 em conjunto), pelo que “TipoProcedimento” não dá origem a meros atributos em “Procedimento”. A entidade “TipoProcedimento” encaixa-se na vista de utilização “Casos”, mas tal como “Procedimento”, alguns dos seus atributos ir-se-ão referir a dados cruciais para as vistas “Gestão” e “Clientes”.

Segue-se uma tabela que sumariza a informação apresentada acima:

Entidade	Definição	Sinónimos	Requisito
Cliente	Pessoa que solicita os serviços da Agência Veritatis. Acolhe o registo de dados base, de faturação e contactos.	Professor	RD1
Funcionario	Trabalhador na Agência Veritatis. Acolhe o registo de dados base, dados para o pagamento de salários, e contactos.		RD4
Funcao	Tipo de trabalho / cargo que um funcionário executa, como detetive, administrativo, ou administrador.	Tipo de trabalhador	RD7
Caso	Único produto comercializado pela agência, que envolve o trabalho de detetives para a sua resolução.	Investigação	RD8
Procedimento	Operação executada por um detetive no processo de resolução de um caso. Acolhe o registo de dados para a resolução de casos e relativos a custos (internos e para o cliente).		RD12
TipoProcedimento	Tipo de ação realizada num “Procedimento”. Acolhe dados descritivos e relativos a custos (internos e para o cliente).		RD14

Tabela 1 - Entidades identificadas pela equipa de desenvolvimento.

3.3. Identificação e Caracterização dos Relacionamentos

A identificação dos relacionamentos também é feita com base nos requisitos, processo descrito nesta secção. Após determinar quais as entidades existentes, é possível procurar requisitos que estabeleçam associações entre elas, dando origem a relacionamentos. O processo de leitura de requisitos à procura de relacionamentos, tal como ocorre com a identificação de entidades, deve envolver requisitos além dos de descrição pois, como será abordado, estes contêm informação útil para determinar características de um relacionamento, como a participação das suas entidades. É de notar que esta secção do documento teve de ser adaptada após a construção do diagrama ER, onde o posicionamento espacial das entidades associadas por relacionamentos condiciona os nomes desses mesmos relacionamentos.

Em primeiro lugar, deteta-se que o requisito RD3 enuncia um relacionamento, pois associa “Cliente” a “Caso” através do verbo encomendar. É explicitado que um cliente pode encomendar vários casos, mas, em contrapartida, um caso apenas pode ter apenas um cliente associado. Assim, tem-se o relacionamento “Caso” “Encomendado” (por) “Cliente”, de cardinalidade muitos-para-um. Além disso,

dado que não faz sentido a existência de um cliente que nunca encomendou um caso, ou de um caso que não foi encomendado por ninguém, este relacionamento binário tem participação total dos dois lados.

De seguida, pode ter-se em conta o requisito RD6, que trivialmente mostra a existência de um relacionamento entre as entidades “Funcao” e “Funcionario” (“Funcionario” “Exerce” “Funcao”). Além disso, por este mesmo requisito, pode concluir-se que este relacionamento tem cardinalidade muitos-para-muitos, dado que um funcionário pode exercer várias funções, e uma função pode ser exercida por vários funcionários (por exemplo, o Prof. Doutor Elias Ribeiro contextualizou a sua empresa como tendo seis detetives, ou seja, a mesma função é exercida por seis pessoas). Logicamente, um “Funcionario” tem de exercer pelo menos uma “Funcao”, mas não é necessário que todas as funções sejam exercidas por um funcionário. Isto é evidenciado por RM4, onde “tipos de trabalhador associados a pelo menos um funcionário” são mencionados, pelo que se assume a existência de instâncias de “Funcao” sem qualquer “Funcionario” associado. Assim, conclui-se que se trata de um relacionamento com participação total do lado de “Funcionario” e parcial do lado de “Funcao”.

Ainda sobre a entidade “Funcionario”, como se pode concluir pelos requisitos RD9 e RD10, pode trivialmente relacionar-se essa mesma entidade com “Caso” (“Funcionario” “Investiga” “Caso”). Note-se que uma instância de um detetive é simplesmente uma instância de “Funcionario” relacionada à “Funcao” correta. Os requisitos explicitam que a cardinalidade deste relacionamento é de um-para-muitos, dado que um caso pode apenas ser investigado por um único funcionário (RD9), mas um funcionário pode investigar vários casos (RD10). Um detetive recentemente contratado, por exemplo, pode ainda não ter nenhum caso associado, pelo que apenas faz sentido que a participação do lado do “Funcionario” seja parcial. Por outro lado, RC2 aponta que apenas administrativos podem registar novos casos, mas RC9 indica que um funcionário atribuído a um caso é resultado da ação de um administrador. Consequentemente, irá haver momentos entre a criação de um caso e a atribuição de um detetive em que o “Caso” não tem um “Funcionario” associado, dado que as duas operações de manipulação devem ser levadas a cabo por pessoas distintas. Assim, a participação do “Caso” no relacionamento é parcial. Um “Caso” está associado a “Procedimento”s, relacionamento que pode ser identificado no requisito RD11 e ao qual foi dado o nome “Exige”. O requisito indica que um “Caso” pode estar associado a vários (ou nenhum) “Procedimento”. A definição desta entidade é a de uma instância de procedimento, ou seja, um procedimento executado uma vez para um dado caso, pelo que um “Procedimento” pode estar associado a um e apenas um caso. Com base nesta informação, conclui-se que o relacionamento “Caso” “Exige” “Procedimento” é de cardinalidade um-para-muitos e participação parcial-total.

Por último, o requisito RD13, “Um procedimento tem um tipo associado”, indica claramente a associação entre “Procedimento” e “TipoProcedimento”, que foi denominada “DoTipo”. Ademais, conclui-se que um “Procedimento” é de apenas um tipo, não se podendo privar dele, enquanto que logicamente um “TipoProcedimento” poderá ter vários “Procedimento”s associados. Disto se conclui a cardinalidade muitos-para-um de “DoTipo”, e a participação total de “Procedimento” neste relacionamento. A participação parcial de “TipoProcedimento”, tal como em “Investiga”, deve-se aos

requisitos de controlo que condicionam quem pode inserir estas entidades distintas, RC5 e RC6: apenas detetives inserem procedimentos, mas apenas administradores criam novos tipos, existindo períodos de tempo em que um tipo de procedimento inserido por um administrador ainda não foi utilizado por um detetive.

De modo a dar suporte à criação e leitura do diagrama ER, a equipa informática compilou a informação descrita acima na seguinte tabela. Note-se que na coluna Requisitos, não só foram incluídos os requisitos principais que deram origem a cada associação, mas também os requisitos de manipulação e controlo usados para se tirarem conclusões sobre cardinalidades e participações.

Entidade	Relacionamento	Cardinalidade	Participação	Entidade	Requisitos	Descrição
Caso	Encomendado	N : 1	T : T	Cliente	RD3	Cliente pede à agência a resolução de uma tentativa de homicídio
Funcionario	Exerce	N : M	T : P	Funcao	RD6, (RM4)	Funcionário executa tarefas de uma função
Funcionario	Investiga	1 : N	P : P	Caso	RD9, RD10, (RC2, RC9)	Detetive trabalha para descobrir culpados numa investigação
Caso	Exige	1 : N	P : T	Procedimento	RD11	Procedimento é realizado no âmbito da resolução de um caso
Procedimento	DoTipo	N : 1	T : P	Tipoprocedimento	RD13, (RC5, RC6)	Características de um procedimento têm origem no seu tipo (instância / classe).

Tabela 2 - Relacionamentos identificados pela equipa de desenvolvimento.

3.4. Identificação e Caracterização dos Atributos das Entidades

Assim como nas secções anteriores, a documentação relativa aos atributos teve como base os requisitos recolhidos pela equipa de informática. A identificação das diferentes entidades, às quais serão concedidos os atributos, já foi realizada na secção [Identificação e Caracterização das Entidades](#), pelo que apenas se descreverá o processo de identificação e caracterização dos atributos. Nos requisitos presentes, não foram identificados atributos em relacionamentos. Assim, o processo de identificação de atributos resultou da leitura sequencial dos requisitos, associando campos de dados a cada entidade já identificada.

N.º	Requisitos Originais	Descrição	Vista de Utilização	Revisor
RD1	1	Um cliente é caracterizado pelo seu nome, morada (opcional), NIF, contactos, e local de trabalho	Clientes	Humberto Gomes
RD2	3	Um professor deve ser identificado por um número sequencial	Clientes	Humberto Gomes
RD17	34	Os contactos de um cliente podem ser diversos: números de telefone e endereços de e-mail. É obrigatório pelo menos um endereço eletrónico.	Clientes	José Lopes

Figura 11 - Requisitos de descrição que dão origem aos atributos de “Cliente”.

Tomando como exemplo os requisitos de descrição RD1, RD2 e RD17, que podem ser consultados na figura acima, são trivialmente obtidos todos os atributos relativos à entidade “Cliente”. Tanto RD1 como RD2 descrevem de forma explícita características inerentes à entidade em questão, nomeadamente “Nome”, “Morada”, “NIF”, “Contactos”, “LocalTrabalho” e um número sequencial, que representámos com o atributo de nome “Id”. É de notar que foi feita a remoção de espaços, diacríticos e, nos atributos multi-palavra, de preposições, para se obterem nomes iguais aos dos futuros modelos lógico e físico, onde caracteres especiais exigem notação SQL inconveniente. Ademais, os números sequenciais internos da agência foram identificados como bons identificadores para cada entidade, que poderão vir a constituir chaves primárias no modelo lógico, sendo apresentados em sublinhado. Este último processo de determinação de identificadores é descrito em muito maior detalhe em [Apresentação e Explicação do Modelo Lógico Produzido](#), onde estão presentes descrições meticolosas da origem da chave primária de cada relação.

Os requisitos contêm também outras propriedades relevantes para a caracterização de atributos. Por exemplo, o requisito RD17 descreve a característica “Contactos”, especificando que esta pode ser constituída por diversos “números de telefone” e “endereços de e-mail”, sendo obrigatória a existência de pelo menos um endereço de e-mail. Logo, ao contrário dos restantes atributos, “Contactos” será composto. Vai possuir dois atributos multivvalorados: “Emails”, que tem de conter pelo menos um valor, e “Telefones”, que pode não ter valores. Ademais, RD1 identifica a “Morada” como sendo “opcional”, sendo mapeada para um atributo simples capaz de assumir valores nulos, ao contrário dos restantes atributos simples em “Cliente”. Quanto a este aspeto, a equipa considerou obrigatório qualquer atributo que nos requisitos não tenha sido mencionado como opcional, ou afirmado apenas assumir valores em determinadas circunstâncias. Assim, dá-se origem à tabela de atributos da entidade “Cliente”:

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
<u>Id</u>	Identificador sequencial do cliente	INT	N	104541	RD2
Nome	Nome completo do cliente	VARCHAR(75)	N	Jacinto Dias Ribeiro	RD1
Morada	Nome da localidade, da rua, número de porta e código postal	VARCHAR(200)	S	Ponte de Lima, Rua do Pinheiro Manso, nº12, 4990-289	RD1
NIF	Número de identificação fiscal (Ministério das Finanças, 2013)	INT(9)	N	243785343	RD1
LocalTrabalho	Designação da instituição de	VARCHAR(75)	N	Universidade do	RD1

	ensino superior onde o cliente trabalha			Minho	
Contactos	Contactos do cliente	-	-	-	RD1
Emails [1..n]	Lista de endereços de eletrónicos	VARCHAR (255)	-	a@sapo.pt , b@hotmail.com	RD17
Telefones [0..n]	Lista de números de telefone	VARCHAR(20)	-	+351 964111111	RD17

Tabela 3 - Atributos da entidade “Cliente”.

O leitor pode perguntar-se qual a origem do domínio de cada atributo, o que define o seu tipo de dados e limita os seus valores, pelo que é relevante apresentar o processo de tomada de decisões para tal. Após a leitura da documentação do MySQL (Oracle, 2024a), sistema de gestão de bases de dados que viria a ser utilizado, consideraram-se os seguintes critérios para a escolha dos domínios dos atributos:

- Para atributos onde os requisitos especificam que serão guardados valores inteiros, optou-se por se atribuir o tipo de dados INT (sinónimo de INTEGER). À medida que se caracterizavam os atributos, julgou-se que nenhum inteiro precisaria de assumir valores superiores ao máximo que este tipo comporta, cerca de dois mil milhões (Oracle, 2024a, p. 2186). Ademais, restringiu-se a gama de valores de atributos nos quais é garantido um valor máximo para os inteiros, como é o caso do "NIF" do "Cliente", que é sempre constituídos por 9 dígitos. (Ministério das Finanças, 2013).
- Para se guardarem valores decimais, optou-se pelo tipo DECIMAL, que assume valores decimais exatos, ao contrário dos tipos de vírgula flutuante, sujeitos a aproximações resultantes de conversões entre as bases 2 e 10 (Oracle, 2024a, p. 2187), o que não é ideal para o armazenamento de preços e salários, os únicos tipos de atributo no modelo desenvolvido que admitem valores numéricos não inteiros. Sabendo que salários são menores do que 10000€ (RD4) e que outros custos são inferiores de 1000€ (RD18), foram escolhidos os tipos DECIMAL(6,2) e DECIMAL(5,2), respetivamente, onde o número dois representa as duas casas decimais para armazenar céntimos, e os restantes dígitos (diferença entre primeiro número e dois) são utilizados para a parte inteira.
- Para guardar datas, como por exemplo a “DataInicio” e “DataTermino” da entidade “Caso”, foi trivial a escolha do tipo DATE, já que é um tipo de dados construído para o armazenamento de datas e não é necessário o armazenamento de informação horária, para o qual se utilizaria DATETIME.
- Para o armazenamento de longos campos textuais, identificados nos requisitos como “campos textuais” ou “detalhados”, utilizou-se o tipo TEXT, que permite textos de comprimentos de até 2¹⁶ caracteres (Oracle, 2024a, p. 2254). São exemplos de atributos com este domínio “Descrição” de “TipoProcedimento”, ou “Notas” de “Procedimento”.
- Para guardar referências textuais menores, utilizou-se o tipo VARCHAR. Os critérios de definição de comprimento máximo foram vagos, atribuídos de modo a suportar o que a equipa de informática julgou ser um número razoável de caracteres, dependendo da informação que se

pretendia guardar. Mesmo assim, procurou-se manter a consistência entre atributos de diferentes entidades com a escolha de critérios para definir o comprimento em função do tipo de informação textual a armazenar. Alguns exemplos destas escolhas incluem VARCHAR(200) para localizações (como a “Morada” de um “Cliente”) e VARCHAR(75) para nomes e designações (como o “Nome” de um “Cliente”). Casos especiais são VARCHAR(255), para endereços eletrónicos (Klensin, 2015) e VARCHAR(20) para números de telefone. Estes últimos são armazenados como informação textual com um comprimento máximo generoso, para suporte de qualquer número estrangeiro.

De seguida, é descrito o processo de caracterização de atributos para as restantes entidades, mas em menor detalhe. O leitor deve considerar que foram utilizadas as regras até este ponto enunciadas de atribuição de nomes, identificadores e domínios, a não ser que algo em contrário seja mencionado. O leitor é livre para consultar a tabela de requisitos de descrição anexada: estes não serão colocados no corpo do relatório como na figura 11, para manter a sua brevidade.

Têm-se como base os requisitos RD4 e RD5 para a entidade “Funcionario”. O requisito RD4 enumera os atributos imediatamente associados à entidade “Funcionario”, nomeadamente “Nome”, um número sequencial (“Id”), “NIF”, “Salario”, “Contactos” e “SeguroVida”. Este último será mapeado para um atributo simples capaz de ser nulo, uma vez que é referido que apenas detetives poderão ter um seguro de vida associado, pelo que este campo não pode ser preenchido para outros funcionários. Todos os outros atributos serão mapeados para atributos simples não nulos, com exceção de “Contactos”, atributo constituído por um endereço eletrónico e um número de telefone (RD5), dando origem a um atributo composto. Os subatributos de “Contactos” serão “Email”, que é referido como “obrigatório”, e “Telefone”, que é referido como “opcional” (poderá assumir valores nulos). Tendo em conta os critérios de escolha de chave e de atribuição de domínios aos atributos, tem-se a seguinte tabela sumário:

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
<u>Id</u>	Identificador sequencial do funcionário	INT	N	321	RD4
Nome	Nome completo	VARCHAR(75)	N	Gonçalo Rocha Figueiredo	RD4
NIF	Número de identificação fiscal (Ministério das Finanças, 2013)	INT(9)	N	165823324	RD4
Salario	Valor do salário bruto mensal, em euros	DECIMAL(6,2)	N	0820.20	RD4
SeguroVida	Número de apólice do seguro de vida presente em apenas detetives	INT(9)	S	951729803	RD4
Contactos	Contactos	-	-	-	RD4
Email	Endereço eletrónico institucional	VARCHAR(255)	N	f321@email.institucional.pt	RD5
Telefone	Número de telefone da empresa	VARCHAR(20)	S	+351 253 000 000	RD5

Tabela 4 - Atributos da entidade “Funcionario”.

A entidade “Funcao” é simples de caracterizar, constituída somente por um identificador sequencial (“Id”) e uma designação (“Designacao”), como consta no requisito RD7. Nenhum dos atributos é dado como opcional e nada evidencia a sua possível decomposição.

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
<u>Id</u>	Identificador sequencial da função	INT	N	3	RD7
Designacao	Designação da função	VARCHAR(75)	N	Detetive Júnior	RD7

Tabela 5 - Atributos da entidade “Funcao”.

As características da entidade “Caso” são descritas pelo requisito RD8, dando origem aos atributos simples “Id” (identificador sequencial), “Designacao”, “CustoAbertura”, “DataInicio”, “DataTermino” e “DataPagamento” após a remoção de caracteres especiais. Os últimos dois atributos podem assumir valores nulos, ao contrário dos restantes, pois apenas assumem um valor quando se aplica uma dada condição: a investigação já terminou e o cliente já a pagou, respetivamente. Ademais, a informação em RD18 (limite superior dos preços) é crucial para a escolha do domínio de “CustoAbertura”.

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
<u>Id</u>	Identificador sequencial do caso	INT	N	613	RD8
Designacao	Nome interno atribuído ao caso	VARCHAR(75)	N	Caso das Meias Rotas	RD8
DataInicio	Data em que se deu início à investigação do caso	DATE	N	2024/03/19	RD8
DataTermino	Data em que o caso foi encerrado (ou um valor nulo caso ainda em aberto)	DATE	S	2025/04/19	RD8
DataPagamento	Data em que o cliente pagou a investigação do caso (ou um valor nulo caso ainda por pagar)	DATE	S	2024/04/20	RD8
CustoAbertura	Custo de abertura do caso em euros	DECIMAL(5,2)	N	231.12	RD8; RD18

Tabela 6 - Atributos da entidade “Caso”.

Os atributos da entidade “Procedimento” são de derivação um pouco mais complexa. RD12 identifica as seguintes características, que darão origem a atributos com as seguintes designações: “Id” (identificador sequencial), “Data”, “CustoAgencia”, “CustoCliente”, “Notas” (o único opcional) e “Provas”. O uso do plural em RD12 para a referência às “provas”, juntamente com o requisito RD16, conduzem à conclusão de que “Provas” é um atributo multivlorado que pode não conter valores. O requisito RD15 define uma prova como sendo caracterizada por uma descrição e o local onde foi encontrada, levando a equipa a

concluir que o atributo provas, além de multivalorado, é composto por dois atributos, “Descrição” e “Local”. Ademais, a informação em RD18 (limite superior dos preços) é crucial para a escolha do domínio de “CustoAgencia” e “CustoCliente”.

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
<u>Id</u>	Identificador sequencial do procedimento	INT	N	19	RD12
Notas	Notas que o detetive acha útil incluir sobre a execução do procedimento	TEXT	S	Aplicação de recolha de amostras de DNA bem sucedida, (...)	RD12
Data	Data de execução	DATE	N	2024/04/20	RD12
CustoAgencia	Custo acarretado pela agência, em euros, na aplicação do procedimento	DECIMAL(5,2)	N	111.11	RD12; RD18
CustoCliente	Custo cobrado ao cliente pela aplicação do procedimento	DECIMAL(5,2)	N	222.22	RD12; RD18
Provas [0..n]	Provas descobertas através da aplicação do procedimento	-	-	-	RD12; RD16
Descricao	Descrição da prova	TEXT	N	Faca do talho	RD15
Local	Local de recolha da prova	VARCHAR(200)	N	Pneu do carro da vítima, parque do CP1, Universidade do Minho	RD15

Tabela 7 - Atributos da entidade “Procedimento”.

A única entidade em falta, “TipoProcedimento” é caracterizada pelo requisito RD14, dos quais se derivam os atributos “Id” (identificador sequencial), “Designacao”, “CustoAgencia”, “CustoCliente” e “Descricao”, todos simples (nenhum requisito menciona a sua decomposição) e de natureza obrigatória. Mais uma vez, a informação em RD18 (limite superior dos preços) é crucial para a escolha do domínio de “CustoAgencia” e “CustoCliente”.

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
<u>Id</u>	Identificador sequencial do tipo de procedimento	INT	N	96	RD14
Designacao	Designação do tipo de procedimento	VARCHAR(75)	N	Interrogatório à vítima	RD14
CustoAgencia	Custo acarretado pela agência, em euros, no momento de aplicação de um procedimento	DECIMAL(5,2)	N	231.21	RD14; RD18
CustoCliente	Custo acarretado pelo cliente, em euros, no momento de aplicação de um	DECIMAL(5,2)	N	432.96	RD14; RD18

	procedimento				
Descrição	Descrição do tipo de procedimento	TEXT	N	Um interrogatório deve decorrer de acordo com padrões estabelecidos (...)	RD14

Tabela 8 - Atributos da entidade “TipoProcedimento”.

3.5. Apresentação e Explicação do Diagrama ER Produzido

Após a identificação e caracterização de entidades, relacionamentos e os seus atributos, a equipa de alunos procedeu à esquematização de um diagrama ER. Este processo foi realizado com recurso ao software brModelo, para a geração de um diagrama conceitual em formato digital. Como mencionado anteriormente, foi utilizada uma abordagem monovista, pelo que o diagrama foi construído na íntegra de uma só vez.

O processo de construção do diagrama ER foi dividido em quatro etapas. Primeiro, um aluno da equipa de desenvolvimento (no caso, Humberto Gomes) colocou todas as entidades e os seus atributos na tela do software de construção do diagrama, fazendo uso da simbologia adequada. Nesta fase, todos os domínios dos atributos também foram adicionados ao modelo, bem como a informação de quais dos atributos eram os identificadores de cada entidade e quais eram opcionais. O resultado da execução desta fase pode ser visto na figura abaixo:

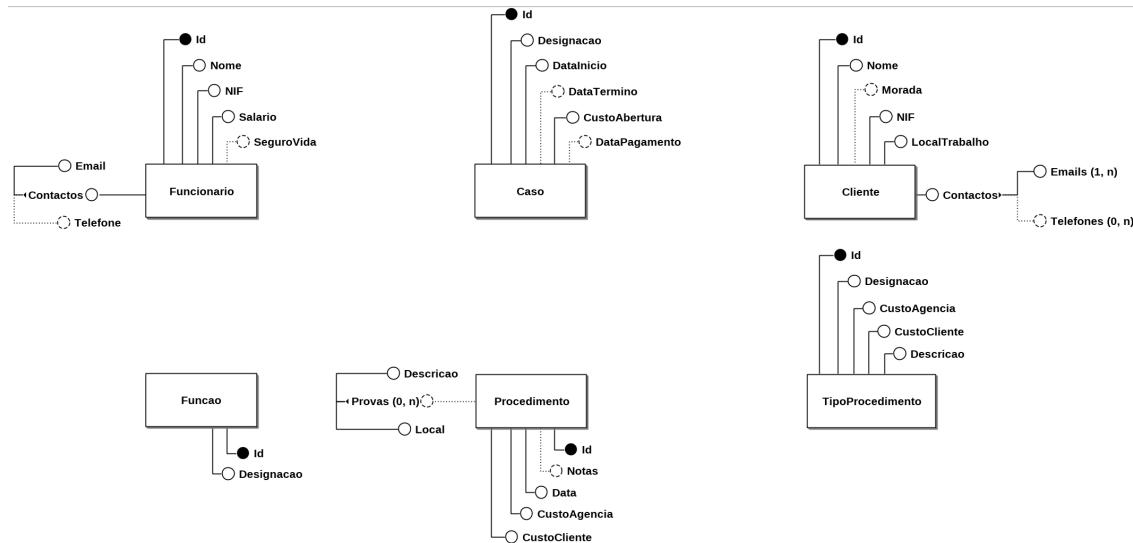


Figura 12 - Diagrama conceitual após a realização da sua primeira fase de construção.

De seguida, foram adicionados todos os relacionamentos entre entidades, bem como as suas cardinalidades e participações (não há atributos), fazendo uso da notação do brModelo, que difere da notação padrão de Chen neste aspecto. Em terceiro lugar, releram-se todos os requisitos, verificando-se

que este modelo não impediria nenhum dos requisitos de ser cumprido. Por último, ajustou-se o posicionamento dos elementos gráficos do diagrama, para se obter uma representação mais fácil de ler. Segue-se o diagrama ER final produzido, após uma revisão pedida pela Agência Veritatis.

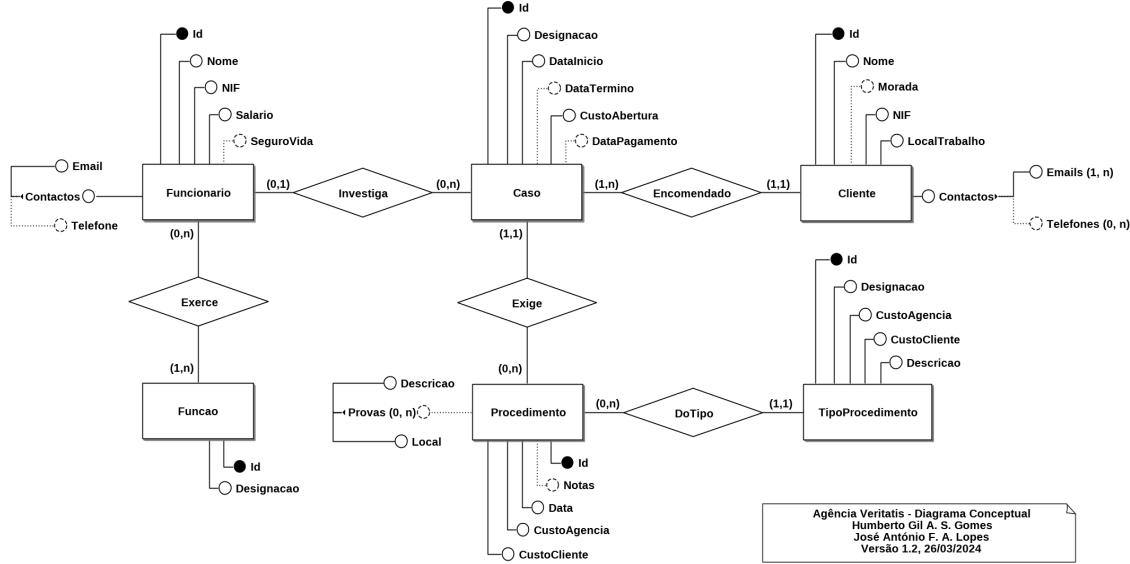


Figura 13 - Versão final do diagrama ER produzido pela equipa de alunos.

3.6. Validação do Modelo Conceitual

Após terminar a elaboração do modelo conceitual e da sua documentação, a equipa de alunos apresentou o seu trabalho ao Professor Doutor Elias Ribeiro, que requisitou mudanças no modo como as provas eram representadas. Anteriormente, estas eram erroneamente consideradas entidades, não coincidindo exatamente com os requisitos, mas tal foi prontamente alterado para provas serem atributos de “Procedimento”.

Um segundo pedido do fundador da Agência Veritatis foi um maior detalhe na documentação, principalmente no processo de dedução das entidades, relacionamentos e atributos, dado que a equipa de informática apenas tinha documentado os resultados, mas não o método usado e o processo para os atingir. Esta mudança seria demorada, e teria de ser feita em paralelo com a construção do modelo lógico, o seguinte passo do ciclo de vida da base de dados em desenvolvimento. Esta documentação melhorada já se encontra presente neste relatório.

4. Modelação Lógica

4.1. Construção do Modelo de Dados Lógico

A equipa manteve a mesma abordagem monovista que utilizou para a construção do modelo conceitual. No contexto da modelação lógica, isto significa que o modelo lógico foi construído na sua íntegra de uma só vez, com base no diagrama ER único previamente concebido.

Foi utilizada uma abordagem de construção redundante, que procurava evitar erros que poderiam surgir na conversão entre modelos. Independentemente, foram feitas duas conversões, uma manual por um aluno, e outra automática por *software*. No final, os resultados de ambas foram comparados, e qualquer discrepância entre eles, causada tanto pela conversão manual como automática, pôde ser retificada. Para a construção automática, foi utilizado o *software brModelo*, já uma ferramenta na conceção do diagrama conceitual, enquanto que para a construção manual do modelo lógico foi utilizado o *MySQL Workbench* (Oracle, 2024b), que futuramente poderia vir a simplificar a implementação física da BD, devido à sua integração com o SGBD MySQL.

Posteriormente, a equipa verificou se o modelo se encontrava na 3FN, verificando se cada tabela cumpria os requisitos necessários para tal. Por último, a validação do modelo foi feita através da implementação de interrogações dos requisitos de manipulação, com recurso a álgebra relacional, para ver se a seria possível defini-las sem alterações ao modelo.

4.2. Apresentação e Explicação do Modelo Lógico Produzido

Para a construção do esquema lógico, a equipa começou por criar uma relação para cada entidade. Para determinar alguns dos atributos de cada uma destas tabelas, não só os atributos simples da entidade foram considerados, como também os atributos compostos após a sua decomposição.

O passo seguinte foi o mapeamento dos atributos multivvalorados, cada um dando origem a uma tabela. Estas tabelas contêm cada uma atributos chave estrangeira, que identificam a relação que corresponde à entidade de onde o atributo multivvalorado pertencia originalmente. Os restantes atributos destas relações correspondem aos valores dos atributos multivvalorados (por exemplo, o valor de um endereço eletrónico).

De seguida, procedeu-se ao mapeamento dos relacionamentos, todos binários, neste caso. Conforme a cardinalidade de cada relacionamento, pôde ser gerada uma nova relação com duas chaves estrangeiras (muitos-para-muitos), ou um novo atributo na tabela da entidade de cardinalidade múltipla, chave estrangeira para a relação correspondente à outra entidade no relacionamento (um-para-muitos).

Agora que todas as tabelas continham todos os seus atributos, as várias chaves candidatas foram determinadas para cada relação, conforme a sua unicidade e minimalidade. Começando com atributos singulares, determinaram-se quais identificavam um registo, sendo dados como chaves candidatas. De seguida, construiram-se conjuntos de dois elementos que não superconjuntos de chaves candidatas já determinadas, e repetiu-se o processo. Continuou-se a aumentar o número de atributos por coleção até este atingir o grau da relação. No final, tinha-se o conjunto total de chaves candidatas, das quais geralmente se escolheram para serem chaves primárias as que tinham valores controlados pela Agência *Veritatis*.

Antes da normalização e validação do modelo lógico, comparou-se o modelo gerado manualmente com o resultado do *brModelo*. Encontrou-se uma discrepancia no modelo gerado automaticamente: seguindo as definições de conversão por omissão, a forma de gerar os atributos multivalorados era distinta da que tinha sido ensinada à equipa de informáticos. A equipa manteve o resultado obtido através da sua conversão manual.

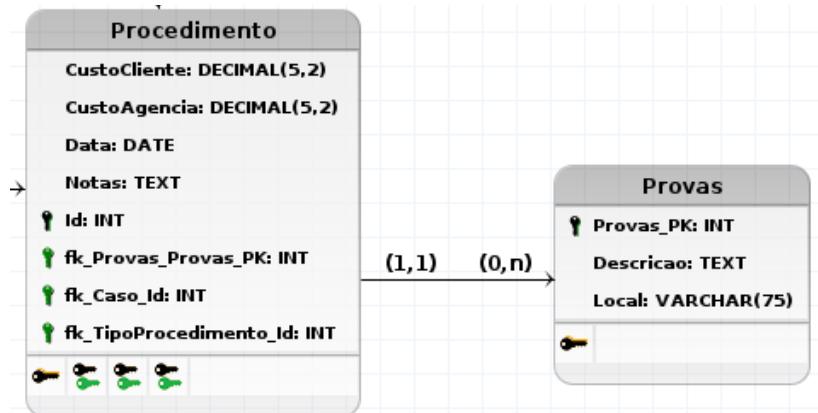


Figura 14 - Exemplo da conversão automática de um atributo multivalorado (“Procedimento.Provas”).

Segue-se a versão final do modelo lógico. É de notar que as relações foram organizadas espacialmente, de modo a ser trivial a análise de dependências entre elas quando o diagrama é lido da esquerda para a direita (a relação A está à esquerda da tabela B se B tem uma chave estrangeira validada na chave primária de A). Tal virá a ser útil para a futura construção da modelo físico.

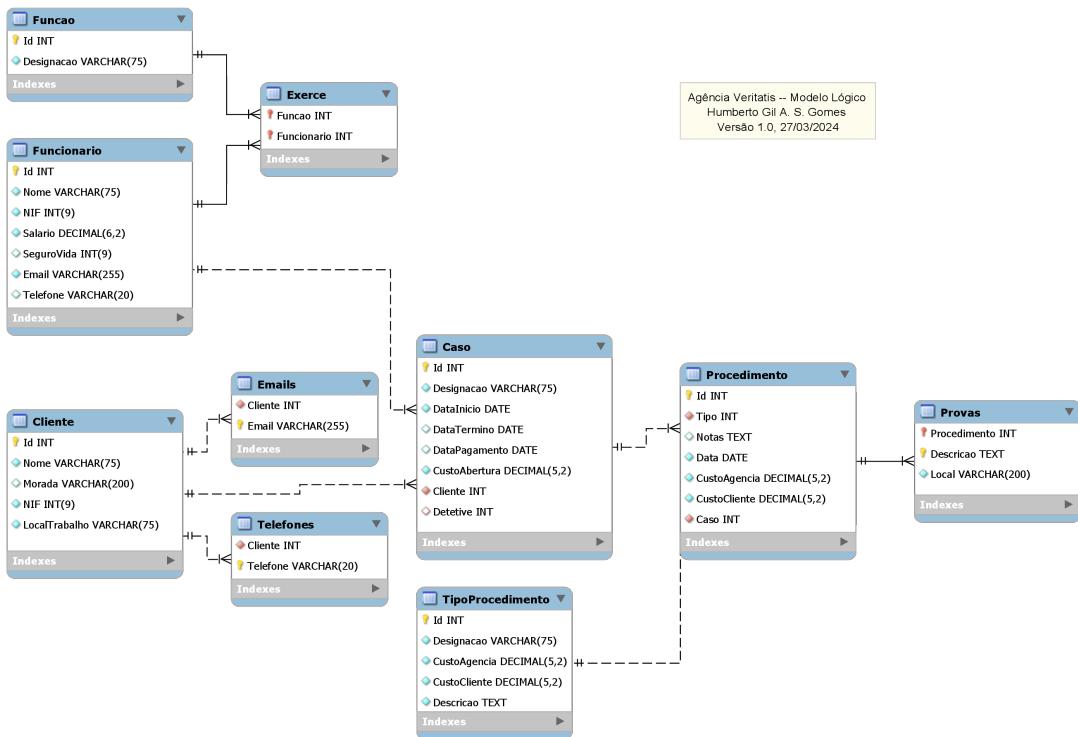


Figura 15 - Versão final do modelo lógico produzido pela equipa de alunos.

Segue-se o dicionário de dados, acompanhado da descrição do processo de criação de cada relação. Seguiu-se a convenção de se apresentar a sublinhado os atributos constituintes da chave primária de uma relação.

A tabela “Cliente” originou da entidade “Cliente”. Os seus atributos simples, “Id”, “Nome”, “Morada”, “NIF” e “LocalTrabalho” todos deram origem a atributos na relação. Note-se que a “Morada”, podendo ser opcional, dá origem a um atributo que pode assumir um valor nulo (com um ícone representado apenas pelo seu contorno), ao contrário dos restantes. O atributo composto “Contactos”, como é apenas constituído por atributos multivvalorados, não dá origem a qualquer atributo na relação. O único relacionamento que envolve a entidade “Cliente”, “Encomenda”, não tem o seu lado de cardinalidade múltipla em “Cliente”, não dando origem a qualquer atributo na tabela. Determinaram-se duas chaves candidatas, o “Id” (número sequencial atribuído pela agência) e o “NIF” do “Cliente”, das quais a primeira foi escolhida para ser a chave primária, dado que o seu valor está sob completo controlo da Agência Veritatis.

Atributo	Domínio	Nulo	Chave Estrangeira
<u>Id</u>	INT	Não	Não
Nome	VARCHAR(75)	Não	Não
Morada	VARCHAR(200)	Sim	Não
NIF	INT(9)	Não	Não
LocalTrabalho	VARCHAR(75)	Não	Não

Tabela 9 - Dicionário de dados da relação “Cliente”.

A tabela “Emails” foi derivada do atributo multivvalor de mesmo nome da entidade “Cliente”. Cada registo nesta tabela deve conter o valor do e-mail em si e fazer referência ao cliente a que este pertence, pelo que esta relação tem dois atributos, “Email” e “Cliente”, este último uma chave estrangeira para a relação “Cliente”. Logo, este atributo deverá ter o mesmo domínio que a chave primária de “Cliente”, INT. Usualmente, seria chave candidata o par (Cliente, Email), mas como o requisito RD17 afirma a unicidade dos endereços eletrónicos por si só, a única chave candidata (e consequentemente primária) fica constituída apenas pelo atributo “Email”. Por último, não faz sentido associar um endereço eletrónico nulo a um utilizador (para tal, não se adiciona outro registo a esta tabela).

Atributo	Domínio	Nulo	Chave Estrangeira
Cliente	INT	Não	Sim (“Cliente”)
<u>Email</u>	VARCHAR (255)	Não	Não

Tabela 10 - Dicionário de dados da relação “Emails”.

Exatamente o mesmo processo seguido para “Emails” é seguido para o atributo “Telefones”, também de “Cliente”, dando origem à seguinte tabela:

Atributo	Domínio	Nulo	Chave Estrangeira
Cliente	INT	Não	Sim (“Cliente”)
<u>Telefone</u>	VARCHAR(20)	Não	Não

Tabela 11 - Dicionário de dados da relação “Telefones”.

A tabela “Caso” tem origem na entidade do mesmo nome, cujos atributos simples são mapeados para atributos da relação (“Id”, “Designação”, “DataInício”, “DataTermino”, “DataPagamento”, dos quais os últimos dois podem assumir valores nulos, e “CustoAbertura”). Há mais dois atributos resultantes de relacionamentos de cardinalidade um-para-muitos, resultantes de “Encomendado” e “Investiga”, a que demos a estes os nomes “Cliente” e “Detetive”, respetivamente. Devido às suas naturezas de chaves estrangeiras, estes atributos devem ter o mesmo domínio que as chaves primárias das tabelas a que se referem, INT para ambos. É de notar que a entidade “Caso” tem uma participação parcial no relacionamento “Investigado”, pelo que o atributo “Detetive” deve poder assumir valores nulos, ao

contrário de “Cliente”. Devido à sua unicidade, a única chave candidata (e consequentemente primária) é o “Id”.

Atributo	Domínio	Nulo	Chave Estrangeira
<u>Id</u>	INT	Não	Não
Designacao	VARCHAR(75)	Não	Não
DataInicio	DATE	Não	Não
DataTermino	DATE	Sim	Não
DataPagamento	DATE	Sim	Não
CustoAbertura	DECIMAL (5, 2)	Não	Não
Cliente	INT	Não	Sim (“Cliente”)
Detetive	INT	Sim	Sim (“Funcionario”)

Tabela 12 - Dicionário de dados da relação “Caso”.

A relação “Funcionario” tem origem na entidade do mesmo nome. Os seus atributos simples foram mapeados diretamente para atributos da relação: “Id”, “Nome”, “NIF”, “Salario” e “SeguroVida”. O atributo composto “Contactos” deu origem aos atributos “Email” e “Telefone”. O “SeguroVida” e “Telefone” podem assumir valores nulos, devido à sua natureza opcional especificada no modelo conceitual. Devido à inexistência de relacionamentos binários de cardinalidade muitos-para-um envolvendo funcionários como as suas primeiras entidades, a relação a construir não apresenta mais atributos. Tal como ocorre com “Cliente”, são chaves candidatas o “Id”, o “NIF” e o “Email” (“SeguroVida” e “Telefone” são desqualificados, pois podem assumir valores nulos), das quais o “Id” foi escolhido para chave primária, devido ao controlo que a Agência Veritatis exerce sobre o seu valor.

Atributo	Domínio	Nulo	Chave Estrangeira
<u>Id</u>	INT	Não	Não
Nome	VARCHAR(75)	Não	Não
NIF	INT(9)	Não	Não
Salario	DECIMAL (6, 2)	Não	Não
SeguroVida	INT(9)	Sim	Não
Email	VARCHAR(255)	Não	Não
Telefone	VARCHAR(20)	Sim	Não

Tabela 13 - Dicionário de dados da relação “Funcionario”.

A tabela “Funcao” deriva da entidade do mesmo nome. Os atributos simples originam dois atributos na tabela, “Id” e “Designacao”, nenhum dos quais pode assumir valores nulos. Não há relacionamentos que adicionem outros atributos a esta relação. Ambos os atributos constituem, por si só, chaves candidatas (não deve haver funções com o mesmo identificador ou com a mesma designação), mas escolhe-se o “Id” para ser chave primária, devido à simplicidade do seu domínio, que conduzirá a um melhor desempenho do SBD.

Atributo	Domínio	Nulo	Chave Estrangeira
<u>Id</u>	INT	Não	Não
Designacao	VARCHAR(75)	Não	Não

Tabela 14 - Dicionário de dados da relação “Funcao”.

Para a representação do relacionamento “Exerce”, devido à sua cardinalidade muitos-para-muitos, é necessária a criação de uma tabela com pelo menos dois atributos (devido às chaves primárias das entidades no relacionamento serem simples), uma chave estrangeira para “Funcionario” e outra para “Funcao”, do tipo das chaves primárias destas relações, INT para ambas. Não há quaisquer outros atributos na relação, visto que o relacionamento “Exerce” não apresenta atributos. A única chave candidata (e consequentemente primária) é composta pelos únicos dois atributos da tabela: remover um destes da chave conduziria ao impedimento de um funcionário exercer várias funções, ou vice-versa. Como ambos os atributos constituem uma chave primária, por definição, nenhum pode assumir valores nulos.

Atributo	Domínio	Nulo	Chave Estrangeira
<u>Funcao</u>	INT	Não	Sim (“Funcao”)
<u>Funcionario</u>	INT	Não	Sim (“Funcionario”)

Tabela 15 - Dicionário de dados da relação “Exerce”.

A entidade “TipoProcedimento” dá origem a uma tabela do mesmo nome, com os mesmos atributos (simples) da entidade: “Id”, “Designacao”, “CustoAgencia”, “CustoCliente” e “Descricao”, nenhum dos quais pode não assumir um valor. Não há relacionamentos nas condições de adicionar outros atributos à tabela. Tal como em “Funcao”, são chaves candidatas o “Id” e a “Designacao” (únicos e mínimos), mas escolhe-se o “Id” para chave primária devido à simplicidade do seu domínio.

Atributo	Domínio	Nulo	Chave Estrangeira
<u>Id</u>	INT	Não	Não
Designacao	VARCHAR(75)	Não	Não
CustoAgencia	DECIMAL(5,2)	Não	Não
CustoCliente	DECIMAL(5,2)	Não	Não
Descricao	TEXT	Não	Não

Tabela 16 - Dicionário de dados da relação “TipoProcedimento”.

A tabela “Procedimento” origina da entidade do mesmo nome, com os seus atributos simples diretamente mapeados para atributos da relação: “Id”, “Notas”, “Data”, “CustoAgencia” e “CustoCliente”, dos quais apenas “Notas” pode assumir valores nulos, pois assim especifica o diagrama ER. Os relacionamentos binários um-para-muitos “Exige” e “DoTipo” têm os seus lados de cardinalidade

múltipla na entidade “Procedimento”, pelo que a sua relação correspondente, esta, terá de ter mais dois atributos, “Caso” e “Tipo”, chaves estrangeiras para as relações “Caso” e “TipoProcedimento”, respetivamente. Mais uma vez, o domínio destes é o mesmo do das chaves primárias das relações a que se referem, INT. Devido à participação total de “Procedimento” nestes relacionamentos, nenhum dos atributos das chaves estrangeiras pode assumir valores nulos. Foi escolhido “Id” para chave primária desta tabela, a única chave candidata devido à sua unicidade e minimalidade.

Atributo	Domínio	Nulo	Chave Estrangeira
<u>Id</u>	INT	Não	Não
Tipo	INT	Não	Sim (“TipoProcedimento”)
Notas	TEXT	Sim	Não
Data	DATE	Não	Não
CustoAgencia	DECIMAL (5, 2)	Não	Não
CustoCliente	DECIMAL (5, 2)	Não	Não
Caso	INT	Não	Sim (“Caso”)

Tabela 17 - Dicionário de dados da relação “Procedimento”.

Por último, o atributo multivalorado “Provas”, da entidade “Procedimento”, dá origem a uma nova tabela do mesmo nome. Esta contém três atributos: a obrigatória chave estrangeira para a entidade pai (“Procedimento”), e os atributos que caracterizam a prova, “Descrição” e “Local”. O tipo de procedimento é INT, pois é o tipo da chave primária de “Procedimento”, onde a chave estrangeira constituída somente pelo atributo “Procedimento” é validada. Usualmente, a chave primária da relação “Provas” seria constituída por todos os atributos da relação, mas o requisito RD15 afirma que a referência ao “Procedimento” e a “Descrição” são suficientes para a identificação de um registo nesta tabela. A equipa tem noção da degradação do desempenho proveniente de uma chave composta utilizando um atributo do tipo TEXT, mas nada pode fazer, considerando os requisitos para o SBD.

Atributo	Domínio	Nulo	Chave Estrangeira
Procedimento	INT	Não	Sim(“Procedimento”)
Descrição	TEXT	Não	Não
Local	VARCHAR (200)	Não	Não

Tabela 18 - Dicionário de dados da relação “Provas”.

4.3. Normalização de Dados

Já foi abordado na secção 3.4, [Identificação e Caracterização dos Atributos das Entidades](#), o processo de escolha de domínios para cada atributo, pelo que se garante que qualquer atributo do modelo conceitual, depois mapeado para o modelo lógico, em qualquer que seja a relação, apenas permite que uma dada célula assuma apenas um valor. Ao analisar cada relação individualmente, valida-se a

correção do mapeamento feito e conclui-se que todas as relações no modelo lógico (e, por consequência, o modelo) estão na Primeira Forma Normal (1FN), dado que não há qualquer atributo que não seja atómico (Connolly & Begg, 2015, p. 466).

De modo a suportar a verificação da normalização do modelo para a segunda e terceira formas normais (2FN e 3FN), a equipa recorreu à construção de diagramas de dependências, representações das dependências funcionais entre os atributos de uma relação. Considera-se que B depende de A ($A \rightarrow B$) se cada valor de A está associado a um e apenas um valor de B (Connolly & Begg, 2015, p. 457). Nem todas as dependências foram representadas, mas apenas aquelas necessárias para concluir que cada relação se encontra na segunda e terceira formas normais (exemplos disto serão abordados).

Foi construído um diagrama de dependências para cada uma das relações presentes no modelo, que permite identificar trivialmente dependências funcionais que façam com que uma relação não esteja normalizada. É necessário garantir, para a 2FN, que não há atributos dependentes de um subconjunto próprio da chave primária, denominadas dependências parciais (Connolly & Begg, 2015, p. 470). Para a 3FN, não pode haver atributos dependentes da chave primária apenas por transitividade (Connolly & Begg, 2015, p. 472). Caso tais padrões fossem reconhecidos, algo que não sucedeu, a equipa de informática teria de proceder à normalização do modelo.

Imediatamente, todas as relações com chaves primárias simples encontram-se na 2FN, pois já se confirmou que se encontram na 1FN e chaves primárias constituídas por um conjunto de atributos singular não admitem subconjuntos próprios. Procurou-se garantir a normalização das restantes tabelas, começando por “Exerce”, onde não existe qualquer relação de dependência, pois nem “Funcionario” é conhecido a partir de “Funcao” nem vice-versa (esta tabela resulta de um relacionamento muitos-para-muitos, pelo que um funcionário pode estar associado a várias funções e vice-versa). Também assim se conclui que esta relação está na 3FN pois, não havendo dependências funcionais, também não há dependências funcionais transitivas.



Figura 16 - Relação “Exerce”.

De seguida, o diagrama gerado para a tabela “Provas” é constituído pela seguinte dependência funcional única: {Procedimento, Descricao} → Local. “Local” é uma característica de cada prova, unicamente identificada pelo procedimento a que está associada e a sua descrição, mas nenhum destes identifica uma prova individualmente. Não havendo dependências parciais, a relação está na 2FN, algo que juntamente com a ausência de dependências transitivas permite a conclusão de que “Provas” está na 3FN.

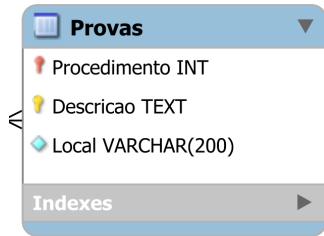


Figura 17 - Relação “Provas”.

As restantes tabelas com apenas dois atributos estão automaticamente na 3FN, devido à impossibilidade da existência de dependências funcionais transitivas. Para as tabelas “Funcao”, “Emails” e “Telefones”, têm-se as respetivas dependências funcionais únicas $Id \rightarrow Designacao$, $Email \rightarrow Cliente$ e $Telefone \rightarrow Cliente$.



Figura 18 - Relações “Funcao”, “Emails” e “Telefones”.

Um exemplo de uma relação de maior grau é “Procedimento”, onde “Id” está associado a um único valor de qualquer outro atributo: $Id \rightarrow$ Tipo, Notas, Data, CustoAgencia, CustoCliente, Caso. É de notar que $Tipo \rightarrow CustoAgencia$, $CustoCliente$ não é uma dependência funcional. Como mencionado em RM8, “CustoAgencia” e “CustoCliente” são atributos cujo valor deriva de “TipoProcedimento” apenas na inserção de um “Procedimento”, mas, após atualizações em “TipoProcedimento”, os valores em “Procedimento” mantêm-se iguais. Logo, por exemplo, um “Procedimento” inserido há dez anos poderá não ter o mesmo custo que um novo procedimento do mesmo tipo hoje em dia, pelo que haverá procedimentos do mesmo tipo com diferentes custos. Ademais, como os custos podem mudar a meio de um dia em que foram realizados dois procedimentos, não se tem que $Data \rightarrow CustoAgencia$, $CustoCliente$. Conclui-se então que a relação está na 3FN.



Figura 19 - Relação “Procedimento”.

De seguida, em “Cliente” e “Funcionario”, os atributos “Id” estão diretamente associados a valores únicos dos restantes atributos das relações: respetivamente $\text{Id} \rightarrow \text{Nome}$, Morada, NIF, LocalTrabalho e $\text{Id} \rightarrow \text{Nome}$, NIF, Salario, SeguroVida, Email, Telefone. Concluem-se que estas relações estão na 3FN. Também é verdade, por exemplo para “Cliente”, que $\text{NIF} \rightarrow \text{Nome}$, Morada, Id, LocalTrabalho. No entanto, estas dependências funcionais são redundantes, pois são incapazes de gerar dependências transitivas, dado que todos os atributos que não “Id” dependem diretamente de “Id”.

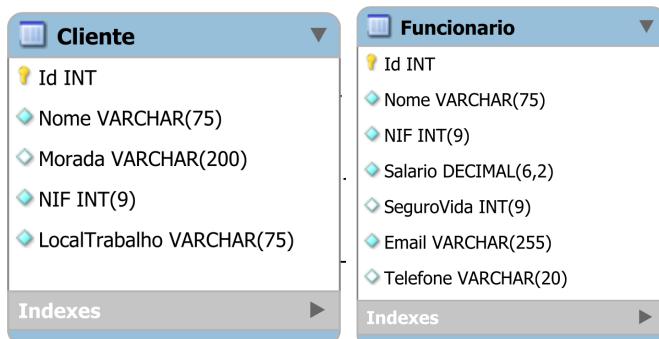


Figura 20 - Relações “Cliente” e “Funcionario”.

A tabela “TipoProcedimento” também está na 3FN, devido às dependências funcionais que impedem a existência de dependências transitivas: diretamente se tem que $\text{Id} \rightarrow \text{Designacao}$, CustoAgencia, CustoCliente, Descricao. O mesmo pode ser dito para “Caso” e as dependências $\text{Id} \rightarrow \text{Designacao}$, DataInicio, DataTermino, DataPagamento, CustoAbertura, Cliente, Detetive.

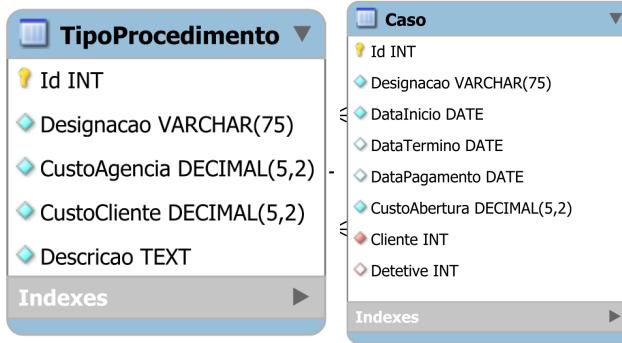


Figura 21 - Relações “TipoProcedimento” e “Caso”.

Como se conclui que todas as tabelas do modelo estão na 3FN, o modelo está normalizado até à 3FN.

4.4. Validação do Modelo com Interrogações do Utilizador

Como já mencionado, a equipa validou o seu modelo através da implementação, utilizando álgebra relacional, das interrogações pedidas pelos requisitos de manipulação. Devido ao espírito de engenheiro dos integrantes da equipa, estes procuraram testar as suas interrogações, para garantir que não havia erros no raciocínio por detrás das mesmas. Para tal, utilizaram a calculadora de álgebra relacional RelaX (dbis-uibk, 2024a).

Antes de testar qualquer interrogação, é necessário especificar à calculadora o esquema das tabelas. Como, nesta calculadora, chaves primárias e estrangeiras não são especificadas, sendo consequências do conteúdo das interrogações introduzidas, a implementação de uma relação é tão simples como enumerar todos os seus atributos fazendo uso da sintaxe adequada, visto que a calculadora suporta todos os tipos de dados utilizados no modelo lógico: numéricos, textuais e datas (todos capazes de assumir valores nulos). Segue-se a tabela de conversão utilizada entre tipos do MySQL e do RelaX.

MySQL	RelaX
DATE	date
DECIMAL(X,Y)	number
INT, INT(X)	number
TEXT	string
VARCHAR(X)	string

Tabela 19 - Mapeamento de domínios entre MySQL e RelaX.

Para povoar a BD de testes, os alunos adicionaram manualmente regtos com dados aleatórios, utilizando a interface gráfica do RelaX para esse propósito. Segue-se, como exemplo, a relação “Cliente”, sendo que o ficheiro contendo todas as tabelas pode ser encontrado no [anexo IV](#).

```

Cliente = {
    Id:number, Nome:string, Morada:string, NIF:number,
    LocalTrabalho:string
    1, 'Orlando Manuel Oliveira Belo', null, 128129130, 'Universidade do
    Minho'
    2, 'João Miguel Lobo Fernandes', null, 155200200, 'Universidade do
    Minho'
    3, 'Vitor Francisco Mendes Gomes Fonte', null, 177188199,
    'Universidade do Minho'
    4, 'João Alexandre Baptista Vieira Saraiva', null, 221196677,
    'Universidade do Minho'
    5, 'Pedro Miguel Silva Paiva António', 'Ed.7, Sala 1.04, Rua da
    Universidade, Braga 4710-057', 111222333, 'Universidade do Minho'
    6, 'Paulo Manuel Martins Carvalho', null, 314782421, 'Universidade
    do Minho'
}

```

De seguida, procedeu-se à implementação de interrogações, i.e., operações de manipulação de dados que correspondem a leituras da BD, cujos requisitos correspondentes se encontram na figura abaixo:

N.º	Requisitos Originais	Descrição	Vista de Utilização	Revisor
RM2	5	Listar todos os professores com casos atualmente em aberto (identificador do cliente e do caso, nome do cliente, designação do caso e data de abertura, ordenados por nome do cliente)	Clientes	Humberto Gomes
RM4	12	Enumerar todos os tipos de trabalhador existentes associados a pelo menos um funcionário (identificador e designação)	Gestão	Humberto Gomes
RM7	16	Listar todos os casos associados a um cliente com base no seu identificador (identificador, designação e data de encomenda do caso, por ordem de data de encomenda)	Clientes	Humberto Gomes
RM10	27	Enumerar todas as provas encontradas no decorrer de uma dada investigação com base no seu identificador (descrição, local e data do procedimento que resultou na prova, ordenados por data)	Casos	José Lopes
RM13	30	Calcular o número de casos investigados e de procedimentos executados por cada detetive com pelo menos um caso desde uma data dada (considerar data de término e data de execução, respetivamente)	Gestão	José Lopes
RM14	31	Gerar uma lista contendo todos os casos, com a receita e despesa resultantes de cada um (identificador, designação do caso, receita, despesa e balanço)	Gestão	José Lopes
RM15	32	Listar por ordem alfabética o nome de todos os funcionários (identificador e nome) que exercem uma função dada pelo seu identificador	Gestão	José Lopes
RM17	35	Pesquisar um cliente pelo seu nome e listar todos os seus contactos. No caso de várias correspondências de clientes, mostrar todas (com identificador e nome, por ordem alfabética)	Clientes	José Lopes
RM18	36	Gerar uma fatura com o preço de abertura de um caso (dado o seu identificador) e todos os procedimentos executados no seu âmbito (designação e preço)	Clientes	José Lopes
RM19	37	Enumerar os casos atualmente em aberto (identificador, designação e data de abertura)	Casos	José Lopes

Figura 22 - Requisitos de manipulação que deram origem às interrogações.

As expressões deduzidas estão presentes ao longo do documento em forma matemática, e no [anexo V](#) em texto plano, caso o leitor pretenda testá-las na calculadora RelaX. Ademais, as árvores sintáticas das expressões são apresentadas com os nós coloridos, algo que não é relevante para esta secção, mas que será usado posteriormente para associação das operações de álgebra relacional às partes das instruções SQL correspondentes a cada expressão.

A equipa começou por implementar as interrogações mais simples, como a que é pedida pelo RM19. A informação sobre os casos em aberto está contida na tabela “Caso”: qualquer caso em aberto tem uma

data de término nula. Para responder à interrogação, basta selecionar esses casos e, através de uma projeção, gerar uma tabela apenas com os atributos pedidos: identificador (“Id”), designação (“Designacao”) e data de abertura (“DataInicio”).

$$\pi_{\text{Id}, \text{Designacao}, \text{DataInicio}} (\sigma_{\text{DataTermino} = \text{null}} (\text{Caso}))$$

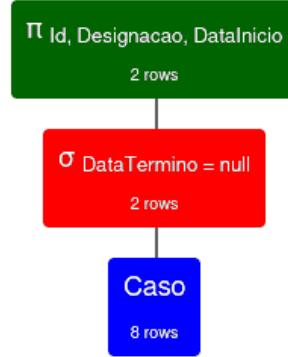


Figura 23 - Árvore da interrogação do RM19 aplicada aos dados de teste.

Depois, começaram-se a abordar requisitos que envolviam várias relações, como RM2. Para se listarem todos os professores com casos atualmente em aberto, começam-se por selecionar os casos em aberto, como foi visto na interrogação anterior. É depois necessário consultar o nome de um “Cliente”, pelo que se pode usar uma equijunção para associar a cada “Caso” as colunas do seu “Cliente” correspondente (“Caso.Cliente” constitui uma chave estrangeira validada em “Cliente.Id”). Para apresentar os dados ao utilizador como estipulado pelo requisito, usa-se uma projeção para a escolha dos atributos pedidos, seguida de uma ordenação pelo “Nome” do “Cliente”.

$$\tau_{\text{Cliente.Nome asc}} (\pi_{\text{Cliente.Id}, \text{Cliente.Nome}, \text{Caso.Id}, \text{Caso.Designacao}, \text{Caso.DataInicio}} (\sigma_{\text{Caso.DataTermino} = \text{null}} (\text{Caso}) \bowtie_{\text{Caso.Cliente} = \text{Cliente.Id}} \text{Cliente}))$$

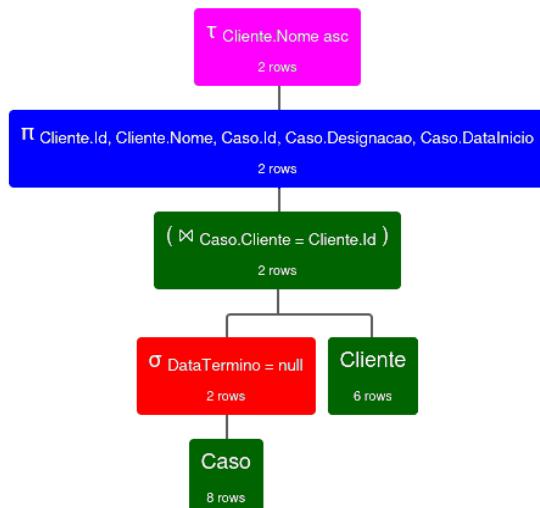


Figura 24 - Árvore da interrogação do RM2 aplicada aos dados de teste.

Antes da apresentação de mais expressões de álgebra relacional, é de notar que a notação para atributos Entidade.Atributo está a ser utilizada quando há mais do que uma tabela envolvida numa expressão, por vezes mesmo quando tal é redundante, para tornar a leitura mais fácil e garantir que não há ambiguidades quando duas relações contêm atributos do mesmo nome.

De RM4 se deriva uma interrogação semelhante à anterior na complexidade e no raciocínio: enumerar as funções com pelo menos um funcionário associado. Estes dados estão na tabela “Exerce”, e as funções podem ser isoladas com uma projeção do atributo “Funcao” (a projeção não gera registos, i.e., funções, repetidas). Associar as designações às funções pode ser feito com uma equijunção à tabela “Funcao”, a única que contém essa informação (associa-se a chave estrangeira constituída por “Exerce.Funcao” à chave primária “Funcao.Id”). Para apresentação ao utilizador, remove-se um atributo repetido, “Funcao.Id” ou “Exerce.Funcao”, através de uma projeção.

$$\pi_{\text{Funcao.Id}, \text{Funcao.Designacao}} (\pi_{\text{Exerce.Funcao}} (\text{Exerce}) \bowtie_{\text{Exerce.Funcao} = \text{Funcao.Id}} \text{Funcao})$$

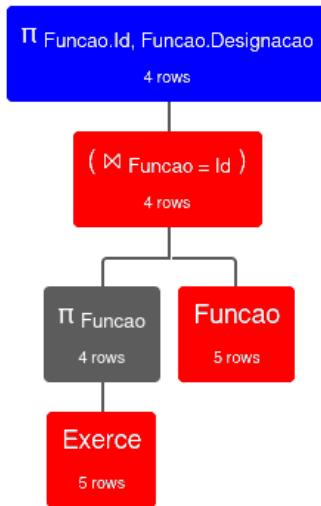


Figura 25 - Árvore da interrogação do RM4 aplicada aos dados de teste.

Há certas interrogações que exigem argumentos, como RM7, a listagem de todos os casos associados a um cliente. Esse cliente tem de ser especificado, no caso, pelo seu identificador (“Cliente.Id”). A interrogação tem de ser gerada através da substituição de uma parte da expressão por um em valor em concreto. Para se representar a variável numa expressão, utilizar-se-á um ponto de interrogação. Após este esclarecimento de notação, a implementação deste requisito torna-se trivial: através de uma seleção em “Caso”, filtram-se os casos cujo atributo “Cliente” assume o valor desejado. Antes da apresentação ao utilizador, é necessária a projeção de apenas alguns atributos pedidos pelo requisito (“Id”, “Designacao”, “DataInicio”) e uma ordenação ascendente por “DataInicio”.

$$\tau_{\text{DataInicio asc}} \left(\pi_{\text{Id, Designacao, DataInicio}} \left(\sigma_{\text{Cliente=2}} (\text{Caso}) \right) \right)$$

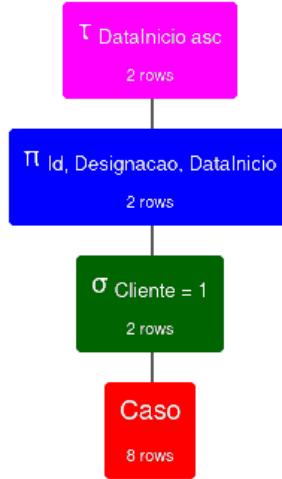


Figura 26 - Árvore da interrogação do RM7 aplicada aos dados de teste, para o “Cliente” de “Id” 1.

Também o RM15, a listagem dos funcionários que exercem uma dada função, é parametrizado. Selecionam-se da relação “Exerce” todos os registo que se referem à função desejada (associações “Funcionario”-“Funcao”), resultado que é junto à tabela “Funcionario” para a associação dos nomes dos trabalhadores (“Exerce.Funcionario” é chave estrangeira validada em “Funcionario.Id”). Por último, para a apresentação final, faz-se uma projeção dos atributos desejados e uma ordenação ascendente pelo nome.

$$\tau_{\text{Funcionario.Nome asc}} \left(\pi_{\text{Funcionario.Id, Funcionario.Nome}} \left(\sigma_{\text{Exerce.Funcao=4}} (\text{Exerce}) \bowtie_{\text{Exerce.Funcionario=Funcionario.Id}} (\text{Funcionario}) \right) \right)$$

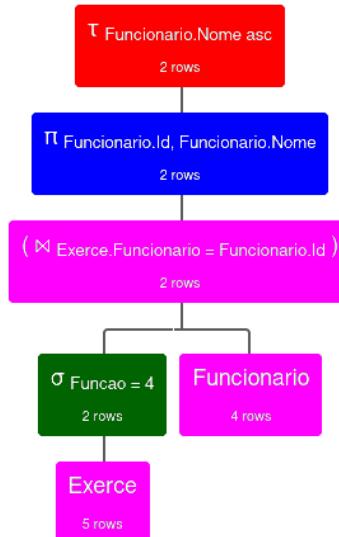


Figura 27 - Árvore da interrogação do RM15 aplicada aos dados de teste, para a “Funcao” de “Id” 4.

Outra interrogação parametrizada é a correspondente ao requisito RM10, a enumeração de todas as provas associadas a um caso, que é identificado pelo seu “Id”. Por outras palavras, procuram-se todas as provas associadas a procedimentos por sua vez associados ao caso desejado. Então, a tabela “Provas” é junta ao resultado de uma seleção de “Procedimento” com o atributo “Caso” desejado (equijunção onde “Provas.Procedimento” é chave estrangeira validada em “Procedimento.Id”). Para a apresentação ao utilizador, é necessária uma projeção do conjunto de atributos pedido pelo requisito, e uma ordenação por “Procedimento.Data”. Devido ao grande comprimento de expressões como esta, incapazes de caber numa única linha, serão utilizadas variáveis para representar as suas partes.

$$P = \text{Provas} \bowtie_{\text{Provas.Procedimento} = \text{Procedimento.Id}} (\sigma_{\text{Procedimento.Caso} = ?}(\text{Procedimento}))$$

$$\tau_{\text{Procedimento.Data asc}} (\pi_{\text{Provas.Descricao}, \text{Provas.Local}, \text{Procedimento.Data}}(P))$$

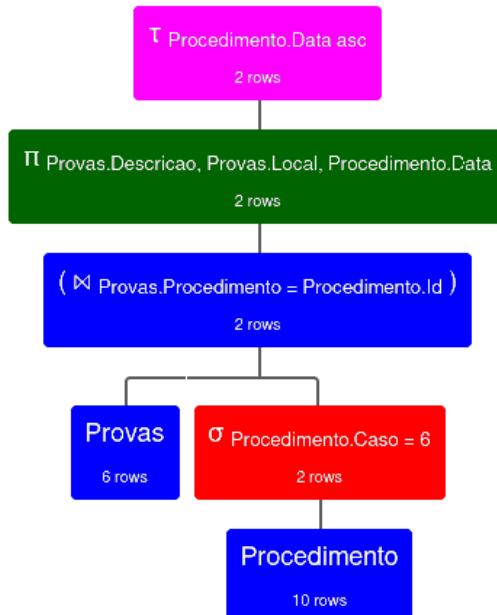


Figura 28 - Árvore da interrogação do RM10 aplicada aos dados de teste, para a “Caso” de “Id” 6.

A última interrogação implementada sem recurso a agregações é relativa ao RM17, a pesquisa de um cliente pelo seu nome e a enumeração dos seus contactos. É necessária uma extensão à álgebra relacional para suportar pesquisa em campos textuais. O RelaX suporta a palavra reservada LIKE de SQL para este feito, podendo ser usada numa seleção, por exemplo. Assim, a equijunção de “Emails” com “Cliente” (onde “Emails.Cliente” é chave estrangeira validada em “Cliente.Id”) gera uma visão que contém tanto os endereços eletrónicos como os nomes dos clientes, que pode ser sujeita a uma seleção para isolar a informação relativa ao utilizador com o nome desejado. A mesma operação pode ser aplicada aos “Telefones”. Uma projeção é utilizada em ambos os casos para o isolamento dos atributos pedidos e a adição de um atributo “Tipo”, que assume o valor “Email” ou “Telefone”. Uma união lógica

das tabelas geradas, seguida de uma ordenação por nome (para o caso de haver várias correspondências), dá origem à relação a apresentar ao utilizador, lembrando que é necessária uma renomeação de “Emails.Email” para “Contacto” na tabela final, para que os telefones não sejam identificados com o nome de atributo “Emails.Email”.

$$E = \pi_{\text{Cliente.Id}, \text{Cliente.Nome}, \text{Emails.Email}, \text{'Email'} \rightarrow \text{Tipo}} (\text{Emails} \bowtie_{\text{Emails.Cliente} = \text{Cliente.Id}} (\sigma_{\text{Nome} \text{ LIKE } \% ? \%} (\text{Cliente})))$$

$$T = \pi_{\text{Cliente.Id}, \text{Cliente.Nome}, \text{Telefones.Telefone}, \text{'Telefone'} \rightarrow \text{Tipo}} (\text{Telefones} \bowtie_{\text{Telefones.Cliente} = \text{Cliente.Id}} (\sigma_{\text{Nome} \text{ LIKE } \% ? \%} (\text{Cliente})))$$

$$\rho_{\text{Contacto} \leftarrow \text{Emails.Email}} (\tau_{\text{Cliente.Nome asc}} (T \cup U))$$

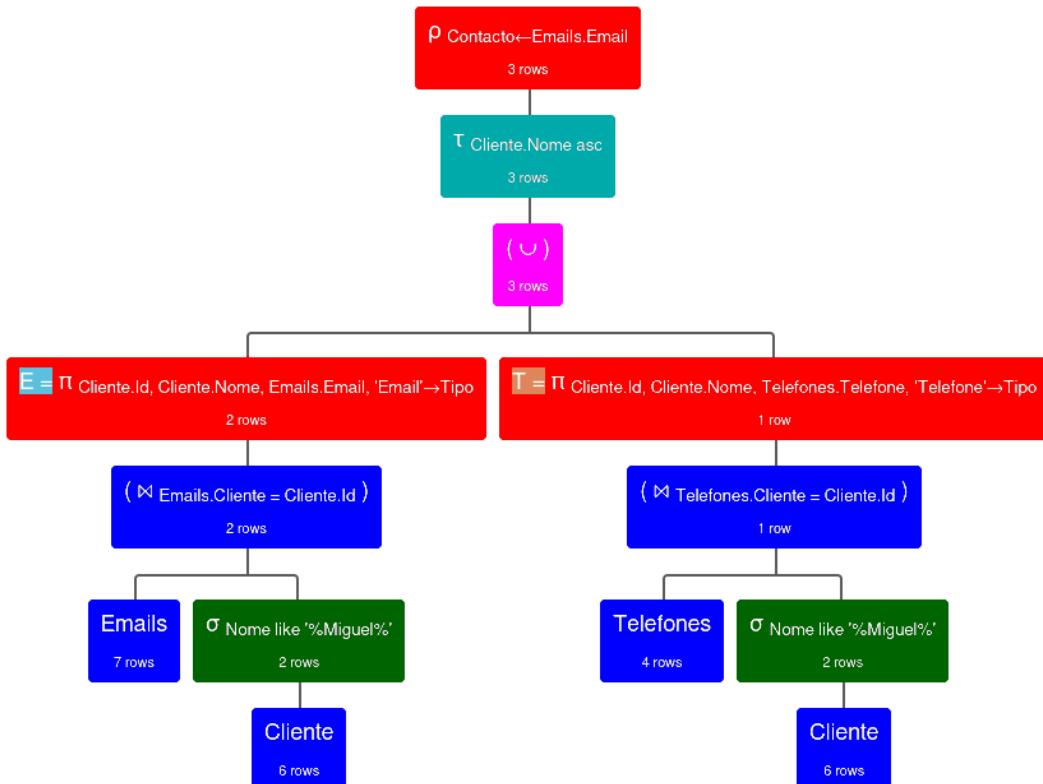


Figura 29 - Árvore da interrogação do RM17 aplicada aos dados de teste, para pesquisar “Cliente”s de “Nome” “Miguel”.

O requisito RM13, o cálculo de casos investigados e procedimentos realizados por cada detetive desde uma data dada, foi o primeiro requisito abordado que envolve agregações. Trivialmente, os casos no último ano podem ser isolados com uma simples seleção em “Caso” por desigualdade de “DataTermino”. Os procedimentos, após ﬁltrados por data, exigem uma equijunção com “Caso”, para associar cada registo à informação sobre o detetive que o executou. Depois, cada um dos resultados das operações anteriores sofre uma agregação por “Caso.Detetive”, criando um atributo novo através de uma contagem de registos (função COUNT). Uma junção externa à esquerda entre a tabela de casos

calculada e a relação de procedimentos calculada faz associar o identificador de um detetive a tanto o número de casos como de procedimentos. A junção externa à esquerda é necessária, dado que pode haver detetives que começaram a investigação de casos mas ainda não executaram qualquer procedimento, algo que não dará origem a registos na segunda tabela. Os valores nulos na relação resultante da junção externa podem ser substituídos por zero através de uma projeção com recurso à função COALESCE.

$$C = \gamma_{\text{Detetive}; \text{COUNT}(\text{*}) \rightarrow \text{Casos}} \left(\sigma_{\text{DataInicio} > \text{DATE}(?)} (\text{Caso}) \right)$$

$$P = \gamma_{\text{Caso.Detetive}; \text{COUNT}(\text{*}) \rightarrow \text{Procedimentos}} \left(\left(\sigma_{\text{Procedimento.Data} > \text{DATE}(?)} (\text{Procedimento}) \right) \bowtie_{\text{Procedimento.Caso} = \text{Caso.Id}} \text{Caso} \right)$$

$$\pi_{\text{Caso.Detetive}, \text{Casos}, \text{COALESCE}(\text{Procedimentos}, 0) \rightarrow \text{Procedimentos}} (C \bowtie P)$$

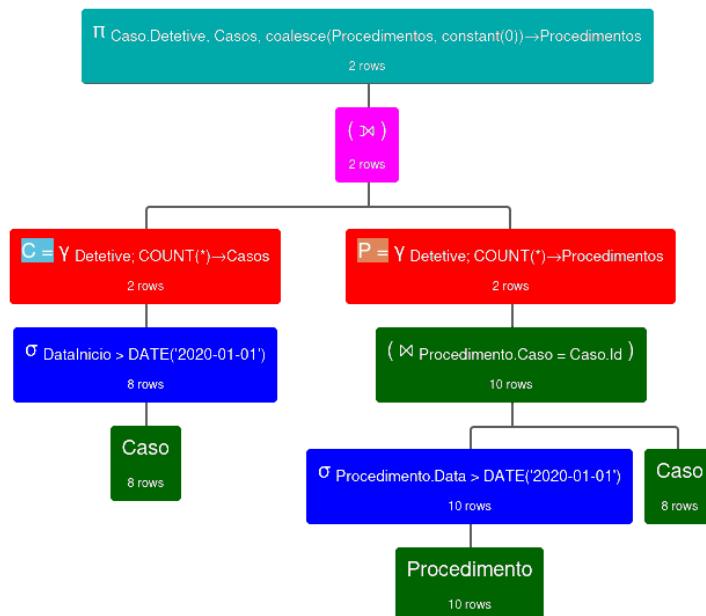


Figura 30 - Árvore da interrogação do RM13 aplicada aos dados de teste, para datas superiores a 2020-01-01.

O requisito RM14 é outro que implica o uso de uma agregação para a sua implementação. Pretende saber-se a receita e a despesa de cada caso, informação contida na relação “Procedimento”, pois são os procedimentos executados para uma investigação que definem os custos e as receitas presentes ao longo do “Caso”. Logo agrupa-se a tabela “Procedimento” pelo atributo “Caso”, fazendo-se a soma dos valores “CustoCliente” para o cálculo da receita e “CustoAgencia” para o cálculo da despesa. É de notar que se usam os atributos de “Procedimento”, e não de “TipoProcedimento”, pois estes últimos contêm os preços atuais, enquanto que os de “Procedimento” contêm os preços na data de execução de cada procedimento. Faz-se uma junção externa à direta desta relação com “Caso”, para cada caso na relação

calculada seja associado ao seu custo de abertura. A junção externa à direita é necessária para manter casos sem procedimentos associados. Uma projeção com operações aritméticas é utilizada para o cálculo da receita total e do balanço, lembrando a necessidade de transformar em zeros os valores nulos nos campos relativos à receita e despesa com procedimentos, algo resultante da agregação quando não há procedimentos associados a um caso. Tal pode ser feito com recurso à função de RelaX COALESCE.

$$S = \gamma_{\text{Caso}; \text{sum}(\text{CustoAgencia}) \rightarrow \text{Despesa}, \text{sum}(\text{CustoCliente}) \rightarrow \text{ReceitaProc}} (\text{Procedimento})$$

$$\pi_{\text{Caso.Id}, \text{Caso.Designacao}} \text{COALESCE}(\text{Despesa}, 0) \rightarrow \text{Despesa}, \text{Caso.CustoAbertura} + \text{COALESCE}(\text{ReceitaProc}, 0) \rightarrow \text{Receita}, \\ \text{COALESCE}(\text{ReceitaProc}, 0) + \text{Caso.CustoAbertura} - \text{COALESCE}(\text{Despesa}, 0) \rightarrow \text{Balanco} (S \bowtie_{\text{Procedimento.Caso} = \text{Caso.Id}} \text{Caso})$$

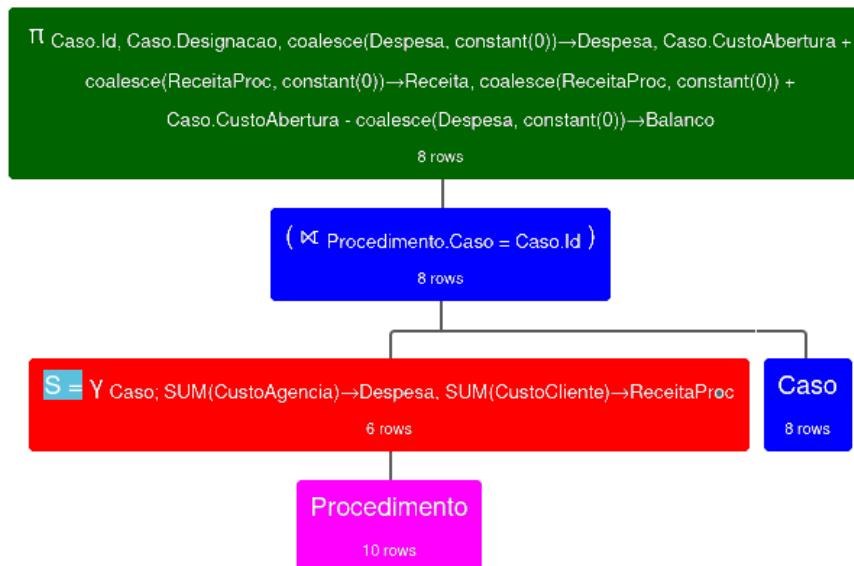


Figura 31 - Árvore da interrogação do RM14 aplicada aos dados de teste.

Por último, será abordado o RM18, a geração de uma fatura relativa a um caso, dado o seu identificador. Uma equijunção de “Procedimento” com “TipoProcedimento” (“Procedimento.Tipo” associado a “TipoProcedimento.Id”) associa a cada procedimento à sua designação. Esta relação calculada pode sofrer uma seleção para isolar procedimentos do caso desejado, e uma projeção para manter apenas a designação e o custo para o cliente, como pedido pelo requisito (tabela *F*). A soma dos custos dos procedimentos pode ser calculada com recurso a uma agregação, mas esta resultará num valor nulo na ausência de procedimentos. Com uma projeção que substitui valores nulos por zero (obtenível através da função COALESCE), este problema é resolvido. O total da fatura é dado pela soma (calculável com uma projeção) do custo de abertura (adquirido com uma simples seleção e projeção em “Caso”) com a soma dos custos dos procedimentos previamente descrita. Devem ser apresentadas ao utilizador as tabelas *F* (procedimentos), *C_a* (custo de abertura) e *T* (total).

$$C_a = \pi_{\text{CustoAbertura}}(\sigma_{\text{Id}=?}(\text{Caso}))$$

$$F = \pi_{\text{TipoProcedimento.Designacao}, \text{Procedimento.CustoCliente}}(\sigma_{\text{Caso}=?}(\text{Procedimento} \bowtie_{\text{Procedimento.Tipo} = \text{TipoProcedimento.Id}} \text{TipoProcedimento}))$$

$$C_p = \pi_{\text{COALESCE}(\text{TotalProcedimentos}, 0) \rightarrow \text{TotalProcedimentos}}(\gamma; \text{SUM}(\text{Procedimento.CustoCliente}) \rightarrow \text{TotalProcedimentos}(F))$$

$$T = \pi_{\text{Total} \leftarrow \text{TotalProcedimentos} + \text{CustoAbertura}}(C_p \bowtie C_a)$$

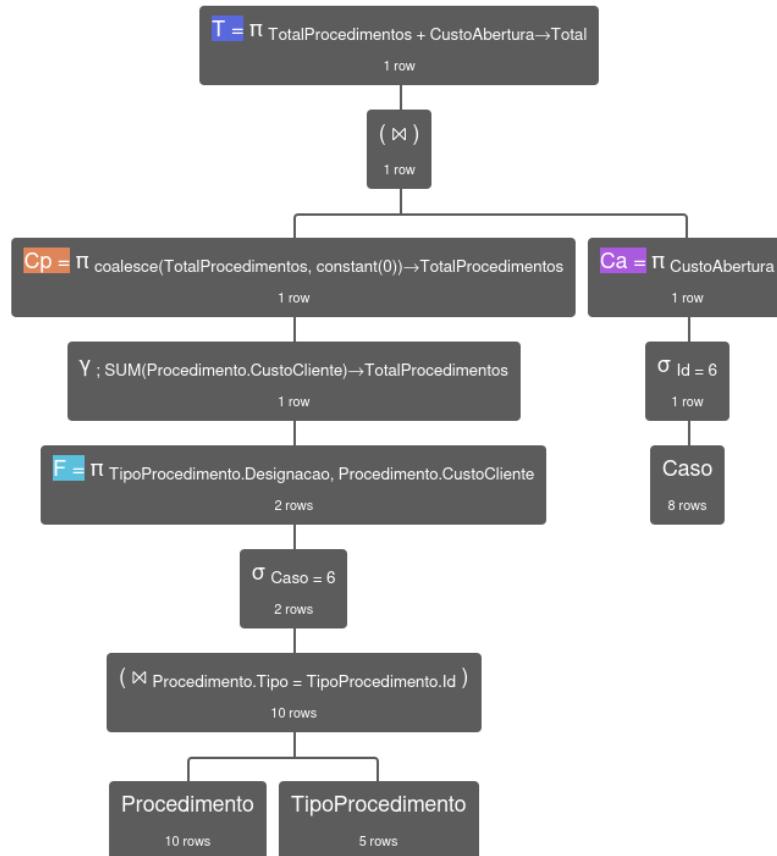


Figura 32 - Árvore da interrogação do RM18 aplicada aos dados de teste, para o “Caso” de “Id” 6.

Como se verificou possível a implementação de todas as interrogações, o modelo lógico foi considerado validado e pronto a ser implementado fisicamente.

5. Implementação Física

5.1. Apresentação e Explicação da Base de Dados Implementada

O modelo físico da BD é definido por um *script* SQL (*Structured Query Language*) criado pela equipa de alunos. Este é lido pelo SGBD MySQL (Oracle, 2024a), que cria, de acordo com as instruções no *script*, a BD para a Agência *Veritatis* e todas as tabelas que esta comporta. O primeiro passo para a escrita deste *script* foi a criação da BD, feita através das seguintes instruções:

```
-- DROP DATABASE IF EXISTS Veritatis;
CREATE DATABASE Veritatis
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;
USE Veritatis;
```

Neste excerto do *script* observam-se três instruções, separadas por pontos-e-vírgula. A primeira, `DROP DATABASE IF EXISTS Veritatis`, elimina a BD “*Veritatis*” caso ela já exista, permitindo que as instruções seguintes a criem do zero (Oracle, 2024a, p. 2777). O uso de `IF EXISTS` garante que este *script* pode ser corrido uma primeira vez sem erros, mesmo se a instrução estivesse fora do comentário em que se encontra (não se dá um erro de remoção de uma BD inexistente). Esta instrução encontra-se comentada porque não deve ser utilizada em produção, mas é de grande utilidade num ambiente de desenvolvimento, pois garante um ambiente determinista onde a BD criada coincide sempre com a descrição da última versão do *script*. Uma alternativa seria dispensar esta instrução e utilizar diretivas `CREATE TABLE IF NOT EXISTS` para a criação das tabelas. No entanto, estas apenas garantiriam a ausência de erro na criação de uma tabela já existente, mas não a alterariam em caso de mudanças ao SQL, fazendo com que a BD criada mantivesse características de versões anteriores do *script* previamente executadas (Oracle, 2024a, p. 2707).

De seguida, a segunda instrução cria a BD de nome “*Veritatis*” (Oracle, 2024a, p. 2673). Não é necessário, através do uso de `IF NOT EXISTS`, evitar um erro caso uma BD do mesmo nome já tenha sido criada, pois esta teria sido eliminada com a instrução anterior. A equipa escolheu que os campos textuais na BD deviam ser armazenados em UTF-8, para permitir que fossem guardadas letras portuguesas com diacríticos e caracteres de línguas estrangeiras. Esta codificação de caracteres apenas

utiliza um octeto para letras, o tipo de carácter mais esperado, apenas utilizando mais espaço para caracteres especiais, ao contrário de outras codificações Unicode, como UTF-16 ou UTF-32. Para tal, foi adicionada a parte CHARACTER SET utf8mb4 à instrução (Oracle, 2024a, p. 2673). Note-se que usar utf8 apenas, sinónimo de utf8mb3, não permite armazenar caracteres que não no *Basic Multilingual Plane* (Oracle, 2024a, p. 2135). Na criação da BD, é também importante definir como valores textuais são comparados, tanto para igualdades como para ordenações. Adicionou-se COLLATE utf8mb4_unicode_ci à instrução, para as comparações não serem sensíveis a maiúsculas / minúsculas e a acentos (Oracle, 2024a, pp. 2006-2107).

Por último, a diretiva USE Veritatis permite informar o MySQL que a BD “Veritatis” é à qual as próximas instruções se irão aplicar (Oracle, 2024a, p. 3152).

Agora, deve começar-se a criação das tabelas. No entanto, surgem questões sobre a ordem em que tal deve ser feito. Durante a criação de uma relação, é necessário mencionar que certos conjuntos de atributos constituem chaves estrangeiras validadas em outras tabelas, algo que requer que essas tabelas já tenham sido criadas. É claro que é possível fazer recurso a instruções ALTER para adicionar as chaves estrangeiras posteriormente, mas procura-se construir um esquema físico sucinto e elegante, pelo que o uso destas instruções será evitado. Logo, para se descobrir a ordem de criação das tabelas, gerou-se um grafo em que cada nodo corresponde a uma relação. Depois, definiu-se que o grafo tem uma aresta A → B caso B tenha uma chave estrangeira cujo valor dos atributos é validado em A. Assim, definiu-se o seguinte grafo na linguagem DOT (Graphviz, 2022):

```

digraph {
    Funcao -> Exerce
    Funcionario -> Exerce
    Cliente -> Emails
    Cliente -> Telefones
    Cliente -> Caso
    Funcionario -> Caso
    Caso -> Procedimento
    TipoProcedimento -> Procedimento
    Procedimento -> Provas
}
  
```

Quando renderizado graficamente, obtém-se o seguinte resultado:

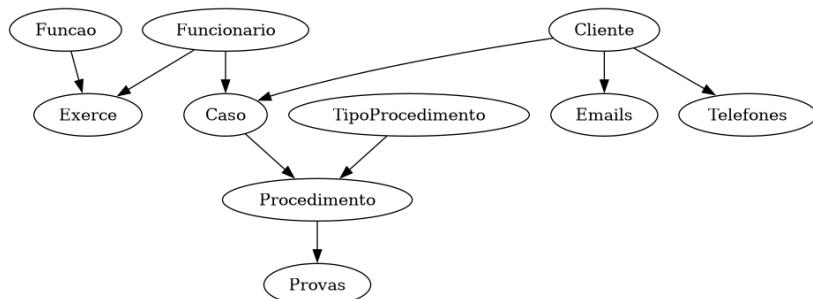


Figura 33 - Grafo do diagrama de dependências de criação da BD.

Considerando-se a definição de ordem topológica de um grafo, uma ordenação dos nodos tal que para cada aresta $A \rightarrow B$, A surge antes de B na ordem, é trivial concluir-se que uma ordenação topológica do grafo anterior dá origem a uma ordem de criação de tabelas onde nunca é necessário que uma chave estrangeira referece uma tabela ainda inexistente. Uma possível ordem topológica do grafo construído é Funcao - Funcionario - Exerce - Cliente - Emails - Telefones - Caso - TipoProcedimento - Procedimento - Provas.

Seguindo a ordem topológica apresentada acima, a equipa começou pela criação da tabela “Funcao”, possível através do uso da diretiva CREATE TABLE, que permite a especificação de tabelas físicas com um dado nome (Oracle, 2024a, pp. 2704-2730). A aplicação desta diretiva para o caso particular da tabela “Funcao” pode ser consultada de seguida:

```
CREATE TABLE Funcao (
    Id INT NOT NULL AUTO_INCREMENT,
    Designacao VARCHAR(75) NOT NULL,
    PRIMARY KEY(Id)
);
```

A designação da tabela é igual ao seu nome no modelo lógico, o que dá origem à parte da instrução antes dos parênteses, CREATE TABLE Funcao. Tal como o nome da relação, também os atributos foram diretamente mapeados do modelo lógico para o modelo físico, sem alterações às suas designações e domínios. Para definir os atributos na sintaxe de SQL, estes são separados por vírgulas, e cada um é definido pela sua designação seguida do seu domínio. Note-se que este domínio é constituído por um tipo de dados (como INT, TEXT, DATE, etc.) e pela possibilidade de um campo assumir ou não um valor nulo, NULL e NOT NULL, respetivamente (Oracle, 2024a, p. 2709). De seguida, foi aplicada a propriedade AUTO_INCREMENT à coluna “Id”, pois este atributo é definido como sendo um inteiro sequencial em RD7. Deste modo, será automaticamente atribuído um número a qualquer registo inserido sem um identificador especificado. Em particular, será atribuído o sucessor do “Id” de maior valor (Oracle, 2024a, p. 2709). A última parte da instrução, PRIMARY KEY (Id), informa o SGBD que a chave primária da tabela é constituída pelo conjunto singular de atributos formado por “Id”. O MySQL garantirá automaticamente a unicidade destes valores (Oracle, 2024a, p. 2712). Como ocorrerá para todas as tabelas, como não existe nenhum requisito que justifique o uso de um motor de armazenamento específico em prol de outro, será utilizada a definição por omissão, InnoDB (Oracle, 2024a, pp. 3552-3553).

Todas as tabelas apresentadas nesta secção seguem o padrão de criação apresentado anteriormente, pelo que aspectos já abordados, como a definição básica de atributos e a denominação das tabelas, não serão apresentados em detalhe. A próxima relação criada foi “Funcionario”, através da seguinte instrução SQL:

```

CREATE TABLE Funcionario(
    Id INT NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(75) NOT NULL,
    NIF INT NOT NULL UNIQUE CHECK (LENGTH(NIF) = 9),
    Salario DECIMAL(6,2) NOT NULL CHECK (Salario >= 0),
    SeguroVida INT NULL CHECK (LENGTH(SeguroVida) = 9),
    Email VARCHAR(255) UNIQUE NOT NULL,
    Telefone VARCHAR(20) UNIQUE NULL,
    PRIMARY KEY(Id)
);

```

Tal como na relação anterior, a propriedade AUTO_INCREMENT está presente em “Id” para respeitar a natureza de número sequencial deste atributo, definida em RD4. Adicionalmente, foi aplicada aos atributos “NIF”, “Email” e “Telefone” a restrição UNIQUE, que impede que dois registos distintos apresentem o mesmo valor do atributo, ou seja, dois funcionários apresentem o mesmo Número de Identificação Fiscal, o mesmo endereço eletrónico ou o mesmo número de telefone (Oracle, 2024a, p. 2712). Adicionalmente, as colunas “NIF” e “SeguroVida” apresentam as restrições CHECK(LENGTH(SeguroVida) = 9) e CHECK(LENGTH(NIF) = 9). Desta modo, garante-se que os números submetidos durante inserções e atualizações de registo são de 9 dígitos (Oracle, 2024a, p. 2339, pp. 2743-2745). É de notar que, durante a criação desta tabela, a equipa de desenvolvimento apercebeu-se de um erro cometido na secção [Identificação e Caracterização dos Atributos das Entidades](#). Uma falha na leitura da documentação levou a equipa a acreditar que estas restrições eram automaticamente verificadas ao especificar o domínio dos atributos como INT(9). No entanto, este atributo de tipo apenas influencia o número de dígitos com que os números são apresentados (Oracle, 2024a, pp. 2187-2188). Também foi definido que salários apenas podem assumir valores positivos através da restrição CHECK(Salario >= 0). Esta decisão foi tomada devido à definição da palavra salário: não faz qualquer sentido este ser negativo. Por último, a chave primária da relação, formada pelo atributo “Id”, é identificada com PRIMARY KEY.

A criação da tabela “Exerce” é semelhante às anteriores em diversos aspectos, como pode ser visto pela sua instrução de criação abaixo:

```

CREATE TABLE Exerce(
    Funcao INT NOT NULL,
    Funcionario INT NOT NULL,
    PRIMARY KEY(Funcao, Funcionario),
    FOREIGN KEY(Funcao) REFERENCES Funcao(Id),
    FOREIGN KEY(Funcionario) REFERENCES Funcionario(Id)
);

```

No entanto, esta relação apresenta uma chave primária constituída por mais de um atributo. Os atributos que a constituem surgem separados por vírgulas nos parênteses de PRIMARY KEY. Assim, a unicidade é garantida para o par de atributos, e não para cada atributo em si. Ademais, esta é a primeira

tabela em que alguns atributos fazem parte de chaves estrangeiras, cujos valores são validados nas chaves primárias de outras tabelas para garantir a integridade referencial. Estas chaves estrangeiras são especificadas com as restrições FOREIGN KEY (Funcao) REFERENCES Funcao (Id) e FOREIGN KEY (Funcionario) REFERENCES Funcionario (Id). Tomando como exemplo a segunda restrição, a chave estrangeira formada pelo único atributo “Funcionario” referencia a tabela “Funcionario”, e deve ser validada na sua chave primária, no caso, “Id”. Durante uma inserção ou atualização de um registo de “Exerce”, caso não exista correspondência entre o valor dos atributos da chave estrangeira e o valor da chave primária de um dos registo da tabela “Funcionario”, a inserção na tabela “Exerce” resulta em erro (Oracle, 2024a, pp. 2736-2742).

De seguida, foi criada a tabela “Cliente”, que apresenta características semelhantes às das relações apresentadas até ao momento. Tal como ocorre em “Funcionario”, foi utilizadas restrições CHECK e UNIQUE para o “NIF”, e uma propriedade AUTO_INCREMENT para assegurar a sequencialidade de “Id” (requisitada por RD2). Segue-se a instrução da tabela “Cliente”:

```
CREATE TABLE Cliente(
    Id INT NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(75) NOT NULL,
    Morada VARCHAR(200) NULL,
    NIF INT NOT NULL UNIQUE CHECK (LENGTH(NIF) = 9),
    LocalTrabalho VARCHAR(75) NOT NULL,
    PRIMARY KEY(Id)
);
```

As tabelas “Emails” e “Telefones”, muito semelhantes, foram criadas de seguida. Não apresentam nem novos conceitos nem escolhas da equipa de desenvolvimento que exijam justificações. Seguem-se as instruções SQL utilizadas na sua criação:

```
CREATE TABLE Emails(
    Cliente INT NOT NULL,
    Email VARCHAR(255) NOT NULL,
    PRIMARY KEY (Email),
    FOREIGN KEY (Cliente) REFERENCES Cliente (Id)
);

CREATE TABLE Telefones(
    Cliente INT NOT NULL,
    Telefone VARCHAR(20) NOT NULL,
    PRIMARY KEY (Telefone),
    FOREIGN KEY (Cliente) REFERENCES Cliente (Id)
);
```

De seguida, a equipa procedeu com a criação da tabela “Caso”, cuja instrução SQL de criação se encontra abaixo:

```

CREATE TABLE Caso(
    Id INT NOT NULL AUTO_INCREMENT,
    Designacao VARCHAR(75) NOT NULL,
    DataInicio DATE NOT NULL,
    DataTermino DATE NULL,
    DataPagamento DATE NULL,
    CustoAbertura DECIMAL(5,2) NOT NULL CHECK(CustoAbertura >= 0),
    Cliente INT NOT NULL,
    Detetive INT NULL,

    PRIMARY KEY (Id),
    FOREIGN KEY (Cliente) REFERENCES Cliente (Id),
    FOREIGN KEY (Detetive) REFERENCES Funcionario (Id),
    CONSTRAINT chk_datas CHECK ((DataInicio <= DataTermino)
        AND (DataTermino <= DataPagamento))
);

```

Tal como já feito várias vezes, a propriedade AUTO_INCREMENT foi utilizada no atributo “Id” devido à sua natureza de número sequencial explicitada por RD8. Ademais, não faz sentido o custo de abertura de um caso ser negativo, pelo que se adicionou um CHECK que garante a sua não-negatividade, CHECK(CustoAbertura >= 0). Por fim, aplicou-se uma restrição utilizando uma sintaxe distinta da que se tem vindo a usar, que faz uso da palavra reservada CONSTRAINT. Esta sintaxe permite atribuir um nome à restrição, que foi escolhido para ser “chk_datas”, dado que esta verifica se as datas em cada registo estão na ordem correta: “DataInicio” não deve ser posterior a “DataTermino”, e esta não deve ser posterior a “DataPagamento” (Oracle, 2024a, p. 2339, pp. 2743-2745). Apesar desta restrição poder ter sido representada como as anteriores, um CHECK depois de um atributo, tal não foi feito visto que a restrição é dependente dos valores de vários atributos, sendo global à relação.

De seguida, criou-se a tabela “TipoProcedimento” com a instrução que se vê abaixo. Novamente, para se implementar um número sequencial (“Id”, de acordo com RD14), utilizou-se a propriedade AUTO_INCREMENT. Ademais, não faz sentido os custos de um procedimento para a agência e para o cliente serem negativos, daí o surgimento de restrições CHECK semelhantes às última descritas.

```

CREATE TABLE TipoProcedimento(
    Id INT NOT NULL AUTO_INCREMENT,
    Designacao VARCHAR(75) NOT NULL,
    CustoAgencia DECIMAL(5,2) NOT NULL CHECK (CustoAgencia >= 0),
    CustoCliente DECIMAL(5,2) NOT NULL CHECK (CustoCliente >= 0),

    Descricao TEXT NOT NULL,
    PRIMARY KEY (Id)
);

```

A tabela “Procedimento” também apresenta um “Id” sequencial (especificado por RD12), pelo que também se utilizou AUTO_INCREMENT nesse atributo. Também estão presentes verificações de não-negatividade para os custos para agência e para o cliente, tal como em “TipoProcedimento”. Note-se que estes custos têm um valor por omissão de 0, definido através da expressão DEFAULT 0 (Oracle, 2024a,

p. 2709), para ser possível inserir procedimentos sem custos, e um gatilho automaticamente os definir conforme o tipo de procedimento utilizado. Segue-se a instrução SQL que cria a tabela “Procedimento”:

```
CREATE TABLE Procedimento(
    Id INT NOT NULL AUTO_INCREMENT,
    Tipo INT NOT NULL,
    Notas TEXT NULL,
    Data DATE NOT NULL,
    CustoAgencia DECIMAL (5,2) NOT NULL DEFAULT 0 CHECK (CustoAgencia >= 0),
    CustoCliente DECIMAL (5,2) NOT NULL DEFAULT 0 CHECK (CustoCliente >= 0),
    Caso INT NOT NULL,
    PRIMARY KEY (Id),
    FOREIGN KEY (Tipo) REFERENCES TipoProcedimento (Id),
    FOREIGN KEY (Caso) REFERENCES Caso (Id)
);
```

Por último, foi criada a tabela “Provas”, com a instrução SQL abaixo. O único aspeto que faz esta relação diferir das restantes ocorre na sua chave primária. Os motores de armazenamento suportados por MySQL apresentam limites de comprimento para chaves. Para InnoDB, este valor é de 767 octetos (Oracle, 2024a, pp. 3766 – 3767). Todavia, apesar da insistência da equipa de desenvolvimento, como evidenciado pelo requisito RD15, a Agência Veritatis não desejou ter numeração nas provas. Como a descrição textual de comprimento possivelmente elevado faz parte da chave primária, é necessário truncá-la no processo de verificação de unicidade de registos. O número de caracteres a utilizar é colocado entre parênteses após o nome do atributo, e a equipa de desenvolvimento escolheu o maior valor possível. Os alunos estão cientes do desempenho pouco ideal desta solução, mas nada podem fazer caso desejem respeitar o documento de requisitos.

```
CREATE TABLE Provas(
    Procedimento INT NOT NULL,
    Descricao TEXT NOT NULL,
    Local VARCHAR(200) NOT NULL,
    PRIMARY KEY (Procedimento, Descricao(767)),
    FOREIGN KEY (Procedimento) REFERENCES Procedimento(Id)
);
```

5.2. Criação de Utilizadores da Base de Dados

A BD “Veritatis” será utilizada por vários utilizadores com permissões distintas, que precisam de ser criados pela equipa de desenvolvimento, como especificado pelos requisitos de controlo no [anexo III](#). A leitura destes requisitos levou a equipa à conclusão de que as permissões atribuídas a cada utilizador estavam relacionadas com as funções executadas por cada funcionário. Logo, em vez de se atribuírem privilégios individualmente a cada funcionário, criaram-se *roles* (papéis), conjuntos de privilégios que

podem ser associados a cada utilizador. A leitura dos requisitos de controlo evidencia a existência de três tipos de funcionário, que darão origem a três papéis: administrativo/secretário, detetive, e gestor/administrador. Note-se que, nesta secção, o termo administrador não será utilizado para fazer referência ao administrador da BD, um elemento da equipa de desenvolvimento que tem todos os privilégios, mas sim a um administrador da Agência Veritatis. Foram também criados três utilizadores de exemplo, cada um associado a um papel distinto. A criação dos três papéis é feita com as seguintes duas instruções SQL:

```
-- DROP ROLE IF EXISTS
-- 'administrativo'@'localhost',
-- 'detetive'@'localhost',
-- 'administrador'@'localhost';
CREATE ROLE
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
```

Tal como ocorre com a criação da BD, a primeira instrução, do tipo `DROP ROLE`, elimina os papéis previamente criados (Oracle 2024a, pp. 3024-3025), para se garantir um ambiente de desenvolvimento determinista onde as permissões no MySQL correspondem sempre à descrição providenciada pela versão mais recente do *script*. A instrução encontra-se em comentário para não ser executada acidentalmente num ambiente de produção. O uso de `IF EXISTS` permite que o *script* seja executado pela primeira vez mesmo se a instrução estivesse fora do comentário em que se encontra, sem a ocorrência de um erro devido à inexistência dos papéis. De seguida, através da diretiva `CREATE ROLE`, criam-se os papéis de administrativo, detetive, e administrador, todos no domínio local (Oracle, 2024a, p. 3011).

Para a definição de privilégios, é necessário terem-se em mente não só os requisitos de controlo, mas também os de manipulação. Alguns destes não têm qualquer requisito de controlo associado, definindo uma operação que pode ser executada por todos os utilizadores, que requer contabilização para a definição de permissões. A cada requisito estará associado um conjunto de instruções SQL, que atribui as permissões estritamente necessárias para a execução da operação que define. Deste modo, garante-se que cada utilizador tem o conjunto mínimo de permissões necessárias. Ademais, facilmente se pode alterar o conjunto de utilizadores com permissão para a execução de uma operação, alterando-se os papéis (*roles*) referidos pela instrução SQL de privilégios correspondente à operação.

O primeiro requisito implementado foi RM1, associado ao requisito de controlo RC1: “apenas secretários podem registar novos clientes e alterar os dados dos existentes”. A permissão de inserção (`INSERT`) e de atualização (`UPDATE`) precisa de ser dada ao papel de administrativo em todos os atributos de todas as relações que constituem a entidade “Cliente”: “Cliente”, “Emails” e “Telefones”. Ademais, a equipa de desenvolvimento estipulou que se um utilizador tem permissão de inserção numa tabela, também deve ser capaz ler os dados nela inseridos (permissão `SELECT`). Considerou-se que o facto de funcionários com o mesmo cargo serem capazes de ler os dados inseridos por si mesmos ou

pelos seus colegas não constitui um problema para a segurança. Por último, modificar um cliente pode implicar a remoção de um ou mais dos seus contactos, pelo que é necessária a permissão de eliminação de registo (DELETE) nas tabelas “Emails” e “Telefones” para o cumprimento deste requisito. A atribuição das permissões mencionadas ao papel de administrativo nas três tabelas pode ser feita com as seguintes instruções GRANT, criadas com base na documentação da Oracle (2024a, pp. 3026-3039):

```
-- RC1 / RM1
GRANT INSERT, UPDATE, SELECT ON Cliente TO
    'administrativo'@'localhost';
GRANT INSERT, UPDATE, DELETE, SELECT ON Emails TO
    'administrativo'@'localhost';
GRANT INSERT, UPDATE, DELETE, SELECT ON Telefones TO
    'administrativo'@'localhost';
```

De seguida, implementou-se RM2, um requisito de manipulação sem qualquer requisito de controlo associado. Logo, as permissões necessárias para a sua execução deverão ser atribuídas a todos os papéis previamente definidos. RM2 define uma operação de interrogação já implementada em álgebra relacional. A leitura da expressão previamente construída informa a equipa de desenvolvimento não só das tabelas para as quais os utilizadores necessitam de permissões de leitura (SELECT), mas também os atributos em particular que precisam de ser consultados. Esta granularidade de controlo permite que não se atribuam permissões de leitura de todos os atributos de uma relação quando tal não é necessário. Logo, para a implementação deste requisito, é necessária a atribuição, a todos os papéis, de permissões de consulta dos atributos “Id” e “Nome” de “Cliente” e “Id”, “Designacao”, “DataInicio”, “DataTermino” e “Cliente” de “Caso”, resultando nas instruções SQL com a seguinte sintaxe:

```
-- RM2
GRANT SELECT(Id, Nome) ON Cliente TO
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
GRANT SELECT(Id, Designacao, DataInicio, DataTermino, Cliente) ON Caso TO
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
```

O mesmo foi feito para os restantes requisitos de manipulação sem requisitos de controlo associados: RM4, RM7, RM15, RM17, RM18 e RM19. Todos correspondem a interrogações previamente implementadas em álgebra relacional, pelo que a implementação do controlo de permissões necessário consistiu na leitura das expressões previamente construídas, a identificação dos atributos utilizados e a construção das instruções SQL que atribuem, a todos os papéis criados, a permissão SELECT para esses atributos. As instruções SQL relativas a estes e aos outros requisitos podem ser encontradas no *script* SQL de criação de utilizadores, no [anexo VIII](#). Foi semelhante o processo de definição de privilégios para os requisitos de controlo correspondentes às restantes interrogações já implementadas: RM10 (RC7), RM13 (RC10) e RM14 (RC11). Também se consultaram os atributos mencionados nas expressões de

álgebra relacional previamente construídas, mas em vez de se atribuírem permissões de SELECT a todos os utilizadores, estas apenas foram atribuídas aos utilizadores que cada requisito de controlo mencionava, como se pode observar na implementação das permissões para RC7, “apenas detetives podem enumerar todas as provas encontradas no decorrer de uma dada investigação [...]”:

```
-- RM10 / RC7
GRANT SELECT(Id, Data, Caso) ON Procedimento T0
    'detetive'@'localhost';
GRANT SELECT(Procedimento, Descricao, Local) ON Provas T0
    'detetive'@'localhost';
```

De seguida, implementou-se o RM3 (correspondente a RC2), “apenas um administrativo é capaz de registar novos casos”. No entanto, como os requisitos RC8 (RM11), RC9 (RM12) e RC12 (RM16) afirmam que apenas detetives podem definir a data de término de um caso, apenas administradores podem associar detetives a casos e apenas administrativos podem dar um caso como pago, não é possível que administrativos tenham controlo sobre todos os atributos durante uma inserção em “Caso”. Logo, excluíram-se os atributos “DataTermino”, “Detetive” e “DataPagamento” da permissão INSERT dada ao papel de administrativo. Quando for feita uma inserção sem valores para estes atributos, estes assumirão o valor NULL (Oracle, 2024a, p. 2250). Note-se que permissões de leitura (SELECT) dos dados inseridos também foram atribuídas pelos motivos já descritos. Segue-se a instrução SQL usada para a atribuição destas permissões:

```
-- RM3 / RC2
GRANT
    INSERT(Id, Designacao, DataInicio, CustoAbertura, Cliente),
    SELECT(Id, Designacao, DataInicio, CustoAbertura, Cliente) ON Caso
    TO 'administrativo'@'localhost';
```

Os requisitos mencionados, RC8, RC9 e RC12, requerem a atribuição das permissões UPDATE(DataTermino), UPDATE(Detetive) e UPDATE(DataPagamento) na tabela “Caso” aos detetives, administradores e administrativos, respetivamente. No entanto, para estes saberem que caso devem modificar, é necessário que sejam capazes de consultar alguns atributos da relação, pelo que a equipa de desenvolvimento também lhes concedeu a permissão SELECT(Id, Designacao, Cliente) para a tabela “Caso”, resultando nas seguintes instruções SQL:

```

-- RM11 / RC8
GRANT UPDATE(DataTermino), SELECT(Id, Designacao, Cliente) ON Caso T0
    'detetive'@'localhost';

-- RM12 / RC9
GRANT UPDATE(Detetive), SELECT(Id, Designacao, Cliente) ON Caso T0
    'administrador'@'localhost';

-- RM16 / RC12
GRANT UPDATE(DataPagamento), SELECT(Id, Designacao, Cliente) ON Caso T0
    'administrativo'@'localhost';

```

Os requisitos RM5 (RC3, “apenas gestores podem inserir novas funções de funcionários”) e RM6 (RC4, “apenas gestores podem adicionar novos funcionários e modificar ou remover funcionários existentes”) exigem controlos de permissões semelhantes. A administradores devem ser concedidas permissões de inserção (INSERT) de todos os atributos em “Funcao” e “Funcionario”. Como já mencionado, a equipa de desenvolvimento também atribui a permissão de leitura (SELECT) quando a inserção é permitida. Ademais, para RM6, é também necessário dar a administradores permissão para atualizar (UPDATE) e eliminar (DELETE) regtos em “Funcionario”. Por último, considerou-se que criar/modificar um funcionário implica a modificação do conjunto das suas funções. Logo, atribuíram-se também permissões INSERT e DELETE em “Exerce” (e consequentemente SELECT) para administradores. Têm-se, assim, as seguintes três instruções SQL:

```

-- RM5 / RC3
GRANT INSERT, SELECT ON Funcao T0
    'administrador'@'localhost';

-- RM6 / RC4
GRANT INSERT, UPDATE, DELETE, SELECT ON Funcionario T0
    'administrador'@'localhost';
GRANT INSERT, DELETE, SELECT ON Exerce T0
    'administrador'@'localhost';

```

O requisito RM8 (RC5) permite que detetives insiram procedimentos. É necessário dar-lhes a permissão INSERT na relação “Procedimento”, e já foi explicado que nestes casos também é dada a permissão SELECT na mesma tabela. Os atributos “CustoCliente” e “CustoAgencia” podem ser derivados do “TipoProcedimento” escolhido, pelo que não é necessário atribuir aos detetives permissões para inserção destes atributos, que assumirão os valores por omissão definidos na secção anterior (zero), posteriormente substituídos pela ação de um gatilho a implementar. Segue-se a instrução SQL que atribui ao papel de detetive as permissões desejadas:

```

-- RM8 / RC5
GRANT INSERT(Id, Tipo, Notas, Data, Caso), SELECT ON Procedimento T0
    'detetive'@'localhost';

```

De seguida, o requisito RC6 (RM9) afirma que “apenas gestores podem registar tipos de procedimento”. Apenas é necessário atribuir as permissões INSERT e SELECT para todos os atributos de “TipoProcedimento” aos administradores:

```
-- RM9 / RC6
GRANT INSERT, SELECT ON TipoProcedimento TO
    'administrador'@'localhost';
```

Por último, note-se que, de momento, ainda há permissões por definir. Estas estão associadas à execução de procedimentos, funções e gatilhos, que precisam de estar definidos antes da definição dos privilégios necessários para a sua execução. Logo, a atribuição dessas permissões será abordada na secção [Implementação de Procedimentos, Funções e Gatilhos](#).

Antes da criação dos utilizadores, a equipa de desenvolvimento criou uma tabela a associar cada relação às operações que nela podiam ser executadas:

	Cliente	Emails	Telefones	Funcionario	Funcao
INSERT	X	X	X	X	X
UPDATE	X	X	X	X	
DELETE		X	X	X	
SELECT	X	X	X	X	X
	Exerce	Caso	TipoProcedimento	Procedimento	Provas
INSERT	X	X	X	X	
UPDATE		X			
DELETE	X				
SELECT	X	X	X	X	X

Tabela 20 - Operações de execução permitidas na BD.

Detetaram-se algumas anomalias, funcionalidades que deviam estar presentes nos requisitos mas que se encontravam em falta. Após contactar o Professor Doutor Elias Ribeiro, este informou a equipa de desenvolvimento que administradores deviam ser capazes de atualizar os custos de tipos de procedimento e que detetives deviam ser capazes de inserir provas. Estes novos requisitos resultaram nas seguintes instruções SQL:

```
GRANT UPDATE(CustoAgencia, CustoCliente) ON TipoProcedimento TO
    'administrador'@'localhost';
GRANT INSERT On Provas T0
    'detetive'@'localhost';
```

Após a definição das permissões necessárias para a execução das operações definidas nos requisitos de manipulação, a equipa de desenvolvimento criou três utilizadores, um para cada papel. Tal como foi feito para a criação da BD e para os papéis criados, uma primeira instrução, DROP USER IF EXISTS, elimina os utilizadores anteriores para garantir o determinismo do ambiente de desenvolvimento

(Oracle, 2024a, pp. 3025-3026), encontrando-se em comentário para não ser executada em produção. De seguida, criam-se três utilizadores no domínio local com a instrução CREATE USER (Oracle, 2024a, pp. 3011-3025):

```
DROP USER IF EXISTS
    'elias.ribeiro'@'localhost',
    'orlando.feio'@'localhost',
    'jacinto.fonseca'@'localhost';
CREATE USER
    'elias.ribeiro'@'localhost' IDENTIFIED BY 'donodistotudo',
    'orlando.feio'@'localhost' IDENTIFIED BY 'alterego',
    'jacinto.fonseca'@'localhost' IDENTIFIED BY 'fonmolhada';
```

Note-se que cada utilizador tem uma palavra-chave para aceder à sua conta, que surge depois das palavras reservadas IDENTIFIED BY. Várias opções de segurança poderiam ter sido definidas na criação dos utilizadores (expiração das palavras-chaves, limites superiores de número de conexões, etc.). No entanto, não sendo pedidas pelos requisitos de manipulação, estas medidas de segurança não foram implementadas. De seguida, atribui-se um papel a cada utilizador, que é definido como o seu padrão, algo feito através das diretivas GRANT (Oracle, 2024a, p. 3036) e SET DEFAULT ROLE (Oracle, 2024a, pp. 3046-3047).

```
GRANT 'administrador'@'localhost' TO 'elias.ribeiro'@'localhost';
SET DEFAULT ROLE 'administrador'@'localhost' TO 'elias.ribeiro'@'localhost';

GRANT 'detetive'@'localhost' TO 'orlando.feio'@'localhost';
SET DEFAULT ROLE 'detetive'@'localhost' TO 'orlando.feio'@'localhost';

GRANT 'administrativo'@'localhost' TO 'jacinto.fonseca'@'localhost';
SET DEFAULT ROLE 'administrativo'@'localhost' TO 'jacinto.fonseca'@'localhost';
```

Assim se conclui a criação dos utilizadores da BD. Quando os requisitos de manipulação forem implementados, será necessário ter em conta as permissões definidas nesta secção, por exemplo, para não se consultarem mais atributos do que os permitidos. Uma testagem adequada, descrita posteriormente, será utilizada para averiguar se as operações de manipulação podem ser executadas tendo em conta os privilégios aqui definidos.

5.3. Povoamento da Base de Dados

Há diversas formas de povoar uma BD, das quais a equipa de desenvolvimento implementou duas. Independentemente do método de povoamento escolhido, um dos detalhes que se tem de ter em consideração é a ordem pela qual as relações são povoadas. Por exemplo, não é possível inserir todos os registos de “Caso” antes dos registos de “Cliente”, pois os registos de “Caso” referenciam, através de uma chave estrangeira, os registos de “Cliente”, que ainda não foram adicionados durante a inserção dos casos, conduzindo a uma violação da integridade referencial. Este problema é muito semelhante ao

que está presente na criação da BD e que diz respeito à ordem de criação das relações. Ambos os problemas apresentam a mesma solução: criar/povoar as relações utilizando uma ordem topológica do grafo da figura 33.

O primeiro método implementado consiste no uso da Linguagem de Manipulação de Dados (LMD) de SQL, em particular, o uso das instruções INSERT e UPDATE, para adicionar à BD um conjunto de registo. Considerando o conjunto de dados utilizado para a testagem das expressões de álgebra relacional ([Anexo IV](#)), a inserção de todos os registo de “Funcao”, a primeira tabela a povoar, pode ser feita com a instrução SQL abaixo, que segue a estrutura sintática descrita pela Oracle (2024a, pp. 2798-2802), `INSERT INTO Tabela(<nomes dos atributos separados por vírgulas>) VALUES (<valores dos atributos do primeiro registo separados por vírgulas> , <valores dos atributos do segundo registo separados por vírgulas> , ...)`:

```
INSERT INTO Funcao(Id, Designacao) VALUES
    (1, 'Gestor financeiro'),
    (2, 'Gestor de recursos humanos'),
    (3, 'Administrativo'),
    (4, 'Detetive'),
    (5, 'Assistente operacional');
```

Como se observa, os literais inteiros não são colocados entre plicas (Oracle, 2024a, p. 2036), ao contrário do que acontece com os literais de dados textuais, strings (Oracle, 2024a, pp. 2033-2036). No entanto, é de notar que, se os valores inteiros estivessem entre aspas únicas, a instrução continuaria a funcionar. Para valores decimais, o separador decimal utilizado é o ponto (Oracle, 2024a, p. 2036), e datas encontram-se entre plicas no formato ‘AAAA-MM-DD’, ano-mês-dia (Oracle, 2024a, p. 2037). Exemplos de inserções destes valores podem ser vistos na instrução que insere os registo da tabela “Caso”, apresentada abaixo, onde valores nulos, NULL, também são inseridos:

```
INSERT INTO Caso(Id, Designacao, DataInicio, DataTermino, DataPagamento, CustoAbertura,
    Cliente, Detetive) VALUES
    (1, 'Clube da Luta', '2020-12-05', '2021-05-04', '2021-05-10', 300.00, 1, 2),
    (2, 'Atrasos Suspeitos', '2021-06-12', '2021-09-24', '2021-10-01', 300.00, 2, 2),
    (3, 'Ministros, Coitados', '2022-04-01', '2022-07-07', '2022-08-01', 300.00, 3, 4),
    (4, 'O Silêncio Dos Inocentes', '2023-01-01', '2023-05-16', '2023-05-19', 350.00, 4, 2),
    (5, 'Vermes na Cantina', '2023-09-09', '2024-02-29', '2024-03-13', 400.00, 5, 4),
    (6, 'Meias Rotas', '2024-01-01', '2024-03-30', NULL, 450.00, 1, 2),
    (7, 'Operação Visconde', '2024-02-12', NULL, NULL, 450.0, 6, 4),
    (8, 'O Barulho Dos Culpados', '2024-03-13', NULL, NULL, 450.00, 3, 2);
```

O ficheiro de povoamento completo, com as instruções de povoamento de cada relação, pode ser encontrado no anexo [Povoamento da BD usando a LMD de SQL](#).

É usual ser necessário o uso de instruções UPDATE para a atualização dos valores de atributos derivados. No entanto, na BD Verititis, os únicos atributos derivados, “Procedimento.CustoCliente” e “Procedimento.CustoAgencia” apenas são atualizados durante inserções: quando é inserido, o “Procedimento” herda os valores de custo do seu “TipoProcedimento”. No entanto, estas atualizações não devem ser feitas no processo de povoamento, visto que os registo de “Procedimento” inseridos se

podem referir a, por exemplo, procedimentos executados há vários anos, quando os custos para o cliente e para a agência eram distintos. Logo, não são necessárias instruções de atualização no processo de povoamento da BD *Veritatis*.

O segundo método utilizado para o povoamento da BD é um programa externo, que comunica com o SGBD e lhe envia as operações de inserção mencionadas anteriormente. A agência *Veritatis*, por não possuir técnicos informáticos internos, deixou a cargo da equipa de desenvolvimento a definição da especificação do programa de povoamento da BD e do formato de dados com que este lidaria. Os funcionários da agência organizariam depois os dados a importar conforme o formato estipulado. A equipa de desenvolvimento construiu a seguinte especificação:

1. O programa deve aceitar como argumento um caminho para uma diretoria com os dados de povoamento. Em caso de um número de argumentos inválido, deverá sair com uma mensagem de erro;
2. O programa deve verificar se o ficheiro “Ordem.txt” existe na diretoria passada como argumento. Este contém o nome de todas as relações da BD a povoar, uma em cada linha, na ordem pela qual devem ser povoadas, e deve ser lido para memória central. Note-se que este ficheiro deve ser armazenado em UTF-8 e o separador de linha deve ser LF (*Line Feed*, ‘\n’, U+000A). Se o ficheiro não existir ou não conseguir ser lido, o programa deve ser sair com uma mensagem de erro;
3. O programa deve pedir ao utilizador a palavra-chave de administração do MySQL no domínio local, e conectar-se à BD “*Veritatis*”. Em caso de erro, deverá sair com uma mensagem de erro;
4. O programa deve, para cada uma das relações em “Ordem.txt”, pela ordem especificada, ler os conteúdos de um ficheiro contido na diretoria passada como argumento, com o nome da relação seguido da extensão “.tsv” (*Tab-Separated Values*). Este ficheiro UTF-8 contém vários campos, separados de acordo com formato especificado por Shafranovich (2005), mas considerando, em vez de vírgulas, tabulações horizontais (*Horizontal Tab*, ‘\t’) como separadores, e separadores de linha LF (e não CR+LF, *Carriage Return* e *Line Feed*). Os campos do primeiro registo (linha) do ficheiro correspondem aos nomes dos atributos da relação, pela ordem que surgem nos restantes regtos do ficheiro, correspondentes aos regtos que serão inseridos na relação. Todos os dados inseridos devem ser imprimidos para o ecrã. Caso um campo de um destes últimos regtos contenha o valor textual “NULL”, será considerada a ausência de valor nesse campo (NULL de MySQL), não sendo possível a existência de campos textuais com a string “NULL”. Em caso de erro na leitura ou na interpretação de qualquer um dos ficheiros, uma mensagem de erro deve ser emitida e o programa deve sair sem modificar a BD.
5. Em caso de sucesso na leitura de todos os ficheiros e inserção de todos os regtos, o programa deve tornar as suas mudanças na BD persistentes e terminar.

A principal decisão por detrás desta especificação foi a escolha da localização dos metadados, constituídos pelos nomes das relações e a sua ordem de povoamento, e pelos nomes dos atributos de cada relação. Perante a possibilidade de estes estarem presentes no programa ou nos ficheiros em si, escolheu-se a segunda opção, visto que o acoplamento entre dados e os programas que os modificam constitui um dos principais problemas dos sistemas de ficheiros que precederam as bases de dados (Connolly & Begg, 2015, pp. 60-62).

O Professor Doutor Elias Ribeiro foi consultado sobre a especificação proposta pela equipa, aprovando-a, e permitindo que a equipa de desenvolvimento procedesse à sua implementação. Esta foi feita em Python (Python Software Foundation, 2024a) com recurso à biblioteca MySQL Connector/Python (Oracle, 2024d). A explicação do código do programa que se segue assume que o leitor se encontra familiarizado com a linguagem Python, pelo que não será explicada a sintaxe do código, e apenas se citará a documentação das bibliotecas utilizadas. Devido ao pequeno tamanho do programa, este é constituído por apenas um procedimento, como mostra o código abaixo:

```
#!/usr/bin/env python3

# (C) Agência Veritatis 2024
# Autor: Humberto Gomes
#
# Instale mysql-connector-python antes de executar:
#   pip3 install mysql-connector-python

import csv
from getpass import getpass
import mysql.connector
from os.path import join
import sys

def main(argv: list[str]) -> int:
    # ... código que será descrito de seguida ...

if __name__ == '__main__':
    sys.exit(main(sys.argv))
```

No início do ficheiro, o primeiro cometário, denominado *shebang*, permite que sistemas UNIX executem o script sem o utilizador ter de especificar que o comando python3 deve ser usado para o interpretar (The kernel development community, 2023). De seguida, segue-se um comentário com uma nota de direitos de autor, o autor do programa, e o que deve ser feito antes do seu uso, a instalação da biblioteca necessária para comunicação com o SGBD. Após a importação dos módulos utilizados, segue-se a declaração do procedimento main, cujo código será descrito de seguida, e uma condição que determina que ele só é chamado quando o programa é executado (assim, main não é executado, por exemplo, se outro programa incluir este ficheiro como um módulo). A assinatura do procedimento main é semelhante à do ponto de entrada de um programa C: tem como entrada os argumentos do programa e como saída o valor de saída do programa (usualmente 0 para a ausência de erros, e outro valor quando erros ocorrem).

A primeiras três linhas do código de `main` asseguram a implementação do primeiro ponto da especificação do programa, a verificação da validade dos argumentos do programa e a saída com uma mensagem de erro caso estes sejam inválidos:

```
if len(argv) != 2:  
    print('Utilização: ./povoamento [diretoria]', file=sys.stderr)  
    return 1
```

De seguida, aborda-se o segundo item da especificação, a leitura do ficheiro de ordem das relações. Estas são lidas para uma lista que conserva a ordem no ficheiro. Além disso, linhas vazias são removidas, pois relações não podem ter um nome vazio e é comum, particularmente em sistemas UNIX, os ficheiros terminarem com um separador de linha, deixando uma linha vazia no seu final. Em caso de erro (código de resposta à exceção), é imprimida uma mensagem de erro e o programa termina.

```
try:  
    with open(join(argv[1], 'Ordem.txt')) as ficheiro:  
        tabelas = [linha for linha in ficheiro.read().split('\n') if linha != '']  
    except OSError as e:  
        print(e, file=sys.stderr)  
    return 1
```

O passo seguinte é a conexão ao servidor MySQL, implementada no pedaço de código abaixo. Em primeiro lugar, pede-se a palavra-chave ao utilizador usando a biblioteca `getpass` (Python Software Foundation, 2024b). A criação de um contexto MySQL é feita com base nos exemplos da documentação da Oracle (2024d, p. 13), tendo em conta que a conexão pode resultar num erro, tendo sido escrito código que termina o programa nesse caso, como pedido pela especificação. De seguida, é necessário criar-se um cursor (distinto dos cursores de SQL), para se poderem enviar instruções SQL ao SGBD, que por omissão terá a funcionalidade de `autocommit` desligada (Oracle, 2024d, p. 18), permitindo reverter qualquer alteração à BD antes desta se tornar permanente, o que é necessário para o cumprimento dos pontos seguintes da especificação.

```
password = getpass(prompt=  
    'Password de administrador (root@localhost) do MySQL: ')  
try:  
    contexto = mysql.connector.connect(user='root',  
                                         password=password,  
                                         host='127.0.0.1',  
                                         database='Veritatis')  
    cursor = contexto.cursor()  
except mysql.connector.Error as e:  
    print(e, file=sys.stderr)  
    return 1
```

De seguida, é feita a leitura de cada ficheiro usando a biblioteca `csv` (Python Software Foundation, 2024c), como se pode observar no código abaixo. Os parâmetros providenciados ao método

`csv.reader` asseguram a compatibilidade com o formato de separadores e a deteção de erros especificados pelo ponto 4 da especificação do programa.

Com base nos dados da primeira linha de cada ficheiro (`i == 0`), gera-se uma instrução SQL `INSERT` tendo em conta os nomes dos campos da primeira linha. Por exemplo, a primeira linha do ficheiro “Telefones.tsv” contém os valores “Cliente” e “Telefone”, pelo que é gerada a instrução `INSERT INTO Telefones(Cliente, Telefone) VALUES (%s, %s)`, onde `%s` representa um campo que será substituído durante a execução da instrução.

Para os restantes registos no ficheiro (`i != 0`), valores textuais “NULL” são substituídos pelo valor de Python `None` (na variável `processado`), que será interpretado como a ausência de valor por parte do MySQL quando a instrução construída é executada com o método `cursor.execute` (Oracle, 2024d, p. 18). Como especificado, todas as inserções são imprimidas para o ecrã.

Em caso de erro, a transação é revertida (Oracle, 2024d, p. 63) e o programa termina com uma mensagem de erro, como pede a especificação.

```
for tabela in tabelas:
    caminho = join(argv[1], tabela + '.tsv')
    try:
        ficheiro = open(caminho)
        leitor = csv.reader(ficheiro,
                            delimiter='\t',
                            quotechar="'",
                            strict=True)
        for i, tuplo in enumerate(leitor):
            if i == 0:
                instrução = f'INSERT INTO {tabela} VALUES ({", ".join(["%s"] * len(tuplo))})'
            else:
                print(tabela + str(tuple(tuplo)))
                processado = tuple(
                    (t if t != 'NULL' else None) for t in tuplo)
                cursor.execute(instrução, processado)

        ficheiro.close()
    except (OSError, csv.Error, mysql.connector.Error) as e:
        print(e, file=sys.stderr)
        contexto.rollback()
        cursor.close()
        contexto.close()
        if ficheiro is not None:
            ficheiro.close()
    return 1
```

Por último, em caso de sucesso, o programa torna persistentes os resultados da transação (Oracle, 2024d, p. 57), fecha o cursor (Oracle, 2024d, p. 74), fecha a conexão (Oracle, 2024d, pp. 56-57) e termina com sucesso:

```
contexto.commit()
cursor.close()
contexto.close()
return 0
```

O código completo deste programa pode ser encontrado no anexo [Povoamento da BD usando um programa externo](#).

5.4. Cálculo do Espaço da Base de Dados

Para conhecer os requisitos de *hardware* necessários para o armazenamento da BD desenvolvida, determinou-se o seu tamanho inicial e como este evoluiria com a passagem do tempo, à medida que a BD cresceria. Em primeiro lugar, procurou-se calcular o tamanho inicial da BD. Foi determinado o espaço utilizado pelo motor de armazenamento InnoDB, escolhido para todas as tabelas, para armazenar os tipos de dados dos domínios dos vários atributos de cada relação. A equipa de desenvolvimento compilou a seguinte tabela, que associa todos os tipos de dados utilizados aos seus tamanhos em disco.

Tipo de dados	Tamanho (octetos)	Fonte
INT	4	(Oracle, 2024a, p. 2186)
DECIMAL(6,2)	3	(Oracle, 2024a, p. 2622)
DECIMAL(5,2)	3	(Oracle, 2024a, p. 2622)
DATE	3	(Oracle, 2024a, p. 2253)
VARCHAR(N), N ≤ 64	1 + N	(Oracle, 2024a, pp. 2254-2555)
VARCHAR(N), N > 64	2 + N	(Oracle, 2024a, pp. 2254-2555)
TEXT	2 + N	(Oracle, 2024a, p. 2254)

Tabela 21 - Associação entre tipos de dados de MySQL e espaço consumido pelo InnoDB.

Note-se que a determinação do tamanho dos números decimais (DECIMAL) exigiu algum cálculo com base na informação providenciada pela documentação do MySQL. Os dígitos da parte inteira são armazenados separadamente dos da parte decimal. Tomando como exemplo DECIMAL(6,2), há dois dígitos na parte decimal, e $6 - 2 = 4$ dígitos na parte inteira. Em cada parte, cada nove dígitos ocupam quatro octetos. Por exemplo, se uma das partes tivesse 20 dígitos, formar-se-iam dois grupos de nove dígitos, cada um com quatro octetos, e um restante de dois. Nos tipos decimais utilizados na BD para a Agência Veritatis, todos os grupos de dígitos são de dimensão inferior a nove, pelo que o seu tamanho pode ser consultado em Oracle (2024a, p. 2622). Para o exemplo dado, os quatro dígitos inteiros consomem ao todo dois octetos, sendo mais um octeto ocupado pelos dois dígitos da parte decimal.

Outro aspeto a ter em atenção é que é possível que um carácter num campo textual ocupe mais do que um octeto (dado que é utilizada a codificação UTF-8 para armazenamento de informação textual). A equipa de desenvolvimento não foi capaz de encontrar uma fonte que estimasse, com base num conjunto de dados de grande dimensão, a percentagem de letras com diacríticos na língua portuguesa, pois estas constituem a maior parte dos caracteres que em UTF-8 consomem mais do que um octeto. No entanto, Stankevičius et al. (2022) mede percentagens de diacríticos em várias outras línguas, entre as quais castelhano e francês, outras línguas românicas que podem ser comparadas com a língua portuguesa para uma aproximação grosseira da percentagem de caracteres com mais de um octeto

(apenas se procura uma estimativa do espaço utilizado, não sendo necessária uma medição exata). Como as duas línguas mencionadas no artigo apresentam percentagens de diacríticos de 2,33% e 3,72%, pode considerar-se, sem piorar significativamente a estimativa do espaço usado por cada campo na BD, que cada carácter português ocupa apenas um octeto.

Outra consideração a ter é que o número de octetos usado para armazenar o tamanho de um campo textual pode ser 1, quando o tamanho máximo do campo é inferior ou igual a 255 octetos, ou 2, quando o tamanho máximo é superior a 255 octetos. Como um carácter UTF-8 tem um comprimento máximo de quatro octetos, a partir de um limite de caracteres de 64, é necessário utilizarem-se dois octetos para o armazenamento do tamanho do campo.

Seguindo a metodologia proposta pelo enunciado do trabalho prático, a equipa de desenvolvimento calculou o tamanho de cada registo através da soma dos tamanhos dos seus atributos. Esta informação pode ser depois utilizada para calcular o tamanho da BD. Para uma simulação mais realista, consideraram-se os seguintes critérios para se escolherem os comprimentos dos campos variáveis:

- Como o comprimento médio de um endereço de correio eletrónico (“Emails.Email” e “Funcionario.Email”) não se aproxima do seu valor máximo, definido como 255 para acomodar o limite estipulado pela especificação dos protocolos de correio eletrónico, considerou-se que cada endereço era composto por 30 caracteres, valor médio de comprimento calculado com base na execução da interrogação `SELECT AVG(LENGTH(email)) FROM customer` na BD Sakila (Oracle, 2024c). Reitere-se que não se procura um valor médio exato, apenas uma aproximação boa o suficiente.
- Também os campos de morada (“Cliente.Morada”, “Cliente.LocalTrabalho” e “Provas.Local”) foram planificados para suportarem moradas extremamente longas, apesar da morada média ser mais curta. O comprimento médio das moradas foi também extraído da Sakila (Oracle, 2024c), onde um valor próximo de 35 caracteres foi o resultado da seguinte interrogação:

```
SELECT AVG(LENGTH(
    CONCAT(address, ' ', city, ' ', district, ' ', postal_code)))
FROM address INNER JOIN city ON address.city_id = city.city_id;
```

- Telefones (“Telefones.Telefone” e “Funcionario.Telefone”) foram definidos com um grande comprimento de modo a suportar números telefónicos estrangeiros. No entanto, para o cálculo do tamanho da BD, considerou-se que todos os números eram portugueses e compostos por nove dígitos.
- Devido à origem anglo-saxónica dos nomes na BD Sakila (Oracle, 2024c), o comprimento utilizado para os atributos “Cliente.Nome” e “Funcionario.Nome” foi 30 caracteres, próximo da média do comprimento dos nomes dos integrantes da equipa de desenvolvimento.
- Por último, sem uma boa referência para o comprimento dos valores do atributo “Designacao” das relações “Caso”, “Funcao” e “TipoProcedimento”, a equipa de desenvolvimento optou por

utilizar 20 caracteres. O mesmo se tem para as descrições textuais “Provas.Descricao”, “TipoProcedimento.Descricao” e “Procedimento.Notas”, onde se considerou que os textos rondariam os 2000 caracteres.

Note-se que, nos critérios acima, não se descreve o processo de raciocínio por detrás das interrogações utilizadas na Sakila, visto que interrogações semelhantes foram implementadas para a BD desenvolvida para a Agência Veritatis, que serão posteriormente descritas. Assim, pode-se calcular o tamanho médio dos registo das diversas tabelas, cujos resultados se encontram nas tabelas abaixo:

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Id	INT	-	4
Designacao	VARCHAR(75)	20	22
Total: 26			

Tabela 22 - Espaço ocupado pelos atributos de cada registo da relação “Funcao”.

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Id	INT	-	4
Nome	VARCHAR(75)	30	32
NIF	INT	-	4
Salario	DECIMAL(6,2)	-	3
SeguroVida	INT	-	4
Email	VARCHAR(255)	30	32
Telefone	VARCHAR(20)	9	10
Total: 89			

Tabela 23 - Espaço ocupado pelos atributos de cada registo da relação “Funcionario”.

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Funcao	INT	-	4
Funcionario	INT	-	4
Total: 8			

Tabela 24 - Espaço ocupado pelos atributos de cada registo da relação “Exerce”.

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Id	INT	-	4
Nome	VARCHAR(75)	30	32
Morada	VARCHAR(200)	35	37
NIF	INT	-	4
LocalTrabalho	VARCHAR(75)	35	37
Total: 114			

Tabela 25 - Espaço ocupado pelos atributos de cada registo da relação “Cliente”.

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Cliente	INT	-	4
Email	VARCHAR(255)	30	32
Total: 36			

Tabela 26 - Espaço ocupado pelos atributos de cada registo da relação “Emails”.

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Cliente	INT	-	4
Telefone	VARCHAR(20)	9	10
Total: 14			

Tabela 27 - Espaço ocupado pelos atributos de cada registo da relação “Telefones”.

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Id	INT	-	4
Designacao	VARCHAR(75)	20	22
DataInicio	DATE	-	3
DataTermino	DATE	-	3
DataPagamento	DATE	-	3
CustoAbertura	DECIMAL(5,2)	-	3
Cliente	INT	-	4
Detetive	INT	-	4
Total: 46			

Tabela 28 - Espaço ocupado pelos atributos de cada registo da relação “Caso”.

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Id	INT		4
Designacao	VARCHAR(75)	20	22
CustoAgencia	DECIMAL(5,2)		3
CustoCliente	DECIMAL(5,2)		3
Descricao	TEXT	2000	2002
			Total: 2034

Tabela 29 - Espaço ocupado pelos atributos de cada registo da relação “TipoProcedimento”.

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Id	INT	-	4
Tipo	INT	-	4
Notas	TEXT	2000	2002
Data	DATE	-	3
CustoAgencia	DECIMAL(5,2)	-	3
CustoCliente	DECIMAL(5,2)	-	3
Caso	INT	-	4

		Total: 2023
--	--	-------------

Tabela 30 - Espaço ocupado pelos atributos de cada registo da relação “Procedimento”.

Atributo	Tipo de dados	Número variável	Tamanho (octetos)
Procedimento	INT	-	4
Descricao	TEXT	2000	2002
Local	VARCHAR(200)	35	37
Total: 2043			

Tabela 31 - Espaço ocupado pelos atributos de cada registo da relação “Provas”.

Seguindo a metodologia do enunciado, com o tamanho esperado de cada registo e o número de regtos por relação, é possível estimar o tamanho da BD. Tendo em conta os dados utilizados no povoamento da BD, tem-se a seguinte tabela abaixo. Note-se que este não é o tamanho da BD, mas o tamanho de uma BD com o mesmo número de regtos em cada relação, regtos estes das dimensões esperadas enunciadas anteriormente.

Relação	Tamanho do registo (octetos)	Número de regtos (octetos)	Tamanho (octetos)
Funcao	26	5	130
Funcionario	89	4	356
Exerce	8	6	48
Cliente	114	5	570
Emails	36	7	252
Telefones	14	4	54
Caso	46	8	368
TipoProcedimento	2034	5	10170
Procedimento	2023	10	20230
Provas	2043	6	12258
Total: 44436 ≈ 44.4 kB			

Tabela 32 - Tamanho ocupado por cada relação na BD Veritatis (usando o tamanho teórico dos regtos).

Com o tamanho da BD calculado, 44.4 kiloBytes (kB), pode analisar-se como este aumentará ao longo do tempo, considerando diferentes taxas de crescimento anuais possíveis, como se vê na tabela abaixo.

Taxa de crescimento anual	Ano 0 (kB)	Ano 1 (kB)	Ano 2 (kB)	Ano 3 (kB)	Ano 4 (kB)	Ano 5 (kB)
10 %	44.4	44.8	53.7	59.1	65.0	71.5
15 %	44.4	51.1	58.7	67.5	77.7	89.3
20 %	44.4	53.3	63.9	76.7	92.1	110.5
25 %	44.4	55.5	69.4	86.7	108.4	135.5
50 %	44.4	66.6	99.9	149.9	224.7	337.2

Tabela 33 - Evolução do tamanho da BD *Veritatis* (tamanho teórico dos registo).

Conclui-se que, mesmo considerando uma taxa de crescimento muito alta, como 50%, o tamanho da BD é sempre muito pequeno mesmo após cinco anos, nunca ultrapassando sequer 1MB. Logo, os requisitos de armazenamento da BD *Veritatis* são muito baixos, e qualquer disco rígido atual é capaz de a armazenar sem qualquer problema. A sua dimensão é tão baixa que até poderia ser totalmente armazenada numa disquete de 3½" de alta densidade!

No entanto, deve ser notado que esta metodologia de cálculo do espaço ocupado é completamente errada, sendo necessário terem-se em conta muitos mais fatores para se fazer um cálculo teórico e *a priori* do espaço ocupado pela BD. Em primeiro lugar, é necessário mencionar que as tabelas são armazenadas em páginas de tamanho fixo (Oracle, 2024a, p. 3326), pelo que é impossível ter uma relação com, por exemplo, 200 octetos, sendo este número sempre arredondo por excesso à página seguinte. Abaixo, são mostrados alguns dos atributos do resultado da instrução SQL SHOW TABLE STATUS IN *Veritatis*. Observa-se que todas as tabelas, de pequena dimensão, ocupam 16kB, o tamanho de uma página.

Name	Row_format	Rows	Avg_row_length	Data_length	Index_length
Caso	Dynamic	8	2048	16384	32768
Cliente	Dynamic	6	2730	16384	16384
Emails	Dynamic	7	2340	16384	16384
Funcao	Dynamic	5	3276	16384	16384
Emails	Dynamic	5	3276	16384	0
Funcionario	Dynamic	4	4096	16384	49152
Procedimento	Dynamic	10	1638	16384	32768
Provas	Dynamic	6	2730	16384	0
Telefones	Dynamic	4	4096	16384	16384
TipoProcedimento	Dynamic	5	3276	16384	0

Tabela 34 - Alguns dos atributos do resultado de SHOW TABLE STATUS IN *Veritatis*.

No entanto, pode ser contra-argumentado que, apesar de uma página inteira ser usada em disco, é possível armazenar mais informação nela se não se encontrar cheia, pelo que o número de registo ainda pode aumentar sem aumento do tamanho da BD. Todavia, a não-consideração da paginação utilizada pelo SGBD não é o maior problema da metodologia utilizada para o cálculo do tamanho da BD. Esta também não considera o tamanho ocupado pelos índices que, como se observa na tabela anterior (*Index_length*), não é nulo. Ademais, esta metodologia não considera as diversas formas como um registo pode estar estruturado em disco (coluna *Row_format* na tabela anterior). O MySQL armazena informação em cada registo que não os valores dos campos de dados em si. Por exemplo, um atributo com domínio INT NULL precisa de mais de quatro bytes para ser representado (como se distinguiria o valor NULL de um valor numérico sem sacrificar o alcance de valores que podem ser representados?).

Logo, o MySQL, antes dos dados do registo, armazena alguns *bits* que informam se cada atributo que pode ser nulo ou não, sendo gastos mais octetos para armazenar muitos mais outros dados (Oracle, 2024a, pp. 3228-3330).

Por exemplo, considere-se a tabela “Telefones”, onde cada registo supostamente ocupa 14 octetos (assumindo telefones de nove dígitos). A inspeção dos conteúdos do ficheiro onde é armazenada a relação revela que cada registo ocupa 19 bytes, como se vê na parte da saída do comando hexdump -C /var/lib/mysql/Veritatis/Telefones.ibd apresentada abaixo (execução num sistema Linux).

```
00014080  00 01 32 35 33 36 30 34  34 37 36 08 20 00 18 00 | ..253604476. ...
00014090  00 80 00 00 01 39 31 36  30 30 39 30 30 09 00 00 | .....91600900...
000140a0  20 00 13 80 00 00 03 32  35 33 36 30 34 34 38 35 | .....253604485
000140b0  09 00 00 28 ff ba 80 00  00 06 32 35 33 36 30 34 | ....(.....253604
000140c0  34 33 32 09 00 00 30 ff  da 80 00 00 01 39 31 30 | 432...0.....910|
```

Como se observa, cada registo contém o identificador do cliente, 1, 3, ou 6. Nos octetos a vermelho, verde e azul, este encontra-se nos últimos 31 bits dos primeiros quatro bytes coloridos. O número de telefone é representado nos nove octetos seguintes. Entre os dados de diferentes registos, aparentam sempre existir outros seis octetos (a preto), um possivelmente correspondente ao comprimento do atributo VARCHAR, a negrito, e os cinco restantes um cabeçalho de registo, o que coincide com a documentação da Oracle (2024a, p. 3328). Note-se que um número de telefone mais curto (oito dígitos) foi inserido para procurar o byte de comprimento e distinguir a posição deste valor.

Devido à complexidade do motor de armazenamento InnoDB, é mais adequada uma abordagem prática ao problema da estimativa do espaço ocupado: a inserção de um elevado número de registos e, após uma medição do espaço ocupado, a estimativa do tamanho médio por registo. Com um procedimento SQL descrito na secção [Implementação de Procedimentos, Funções e Gatilhos](#), milhares de registos foram inseridos, e foram medidos os seus comprimentos médios para diferentes números de inserções, de modo reduzir a influência do arredondamento devido à paginação. A medição foi realizada com a instrução SHOW TABLE STATUS IN Veritatis, previamente apresentada. Ademais, note-se que foram criados menos registos que contêm atributos do tipo TEXT, pois o seu maior tamanho torna as suas medições menos sensíveis à variação causada pelo arredondamento à página seguinte.

A partir das medições no anexo [Medições de espaço resultantes de inserções de registos](#), conclui-se que o tamanho real de cada registo, e consequentemente o tamanho da BD, são superiores às estimativas feitas anteriormente. Calcula-se então o novo tamanho da BD:

Relação	Tamanho do registo (octetos)	Número de registos (octetos)	Tamanho (octetos)
Funcao	41.8	5	209
Funcionario	175.3	4	702.1
Exerce	43.3	6	259.8
Cliente	140.7	5	703.5
Emails	89.5	7	626.5

Telefones	60.8	4	243.2
Caso	102.0	8	616.0
TipoProcedimento	2307.8	5	11539.0
Procedimento	2332.6	10	23326.0
Provas	3454.5	6	20727.0
			Total: 58952.1 ≈ 59.0 kB

Tabela 35 - Tamanho ocupado por cada relação na BD *Veritatis* (tamanho dos registos medido experimentalmente).

Apesar do crescimento da BD em 35% quando são considerados os novos tamanhos dos registo, o seu novo tamanho (não considerando paginação) ainda está ordens de grandeza abaixo das capacidades dos meios de armazenamento atuais, pelo que, mesmo que fossem consideradas taxas de crescimento elevadas como foi feito para os tamanhos teóricos dos registo, os requisitos de *hardware* de armazenamento para a Agência *Veritatis* são desprezáveis.

5.5. Definição e Caracterização de Vistas de Utilização em SQL

O uso de vistas em SQL está geralmente associado ao controlo das permissões atribuídas a cada utilizador. Uma vista é uma tabela temporária, cujos dados têm origem em tabelas regulares ou em outras vistas. Por exemplo, pode utilizar-se uma vista para se armazenar o resultado de uma projeção seguida de uma seleção na tabela “Caso”. Pode dar-se permissão de consulta desta tabela temporária a um utilizador, e este apenas poderá ler alguns dos seus atributos (projeção) e alguns dos seus registo (seleção). Considere-se que o detetive Orlando Feio, funcionário de identificador 2, apenas pode consultar os casos que lhe foram atribuídos. Considere-se ainda que este não pode aceder aos atributos “DataPagamento” e “CustoAbertura”. Logo, pode criar-se uma vista que é o resultado de uma seleção dos casos com “Detetive” = 2, seguida de uma projeção de todos os atributos com exceção dos dois excluídos. Em SQL, tal pode ser conseguido com a seguinte instrução:

```
CREATE VIEW vwCasosDetetive2 AS
    SELECT Id, Designacao, DataInicio, DataTermino, Cliente, Detetive
    FROM Caso
    WHERE Detetive = 2;
```

A diretiva CREATE VIEW permite a criação de uma nova vista (Oracle, 2024a, pp. 2773-2777), a que foi atribuída a designação “vwCasosDetetive2”. O prefixo vw foi utilizado para facilmente se identificar que esta tabela criada é temporária. A operação SELECT FROM WHERE é responsável pela seleção e pela projeção, mas apenas será abordada em detalhe na secção seguinte. Como foi abordado para a criação de utilizadores, pode atribuir-se ao detetive em questão permissões para consultar esta tabela temporária:

```
GRANT SELECT ON vwCasosDetetive2 TO 'orlando.feio'@'localhost';
```

Apesar da sua utilidade, a equipa de desenvolvimento optou por não criar qualquer vista, fazendo todo o controlo de permissões no processo de criação dos papéis e dos utilizadores. A atribuição de privilégios apenas permite controlar a que atributos cada utilizador tem acesso, sem possibilidade de qualquer controlo ao nível dos registo. No entanto, os requisitos de controlo apresentados pela Agência Veritatis podem todos ser implementados no processo de criação dos utilizadores, dado que todos podem ser reduzidos à forma “funcionários do tipo F podem executar as operações X, Y e Z nos atributos A, B e C da relação R”. Estes requisitos também poderiam ser implementados com recurso a vistas. Por exemplo, para a tabela “Caso”, podem considerar-se todos os requisitos de controlo e concluir-se que um administrativo apenas pode consultar os atributos “Id”, “Designacao”, “DataInicio”, “DataTermino”, “CustoAbertura” e “Cliente”. Logo, uma vista com estes atributos pode ser criada para consulta por administrativos:

```
CREATE VIEW vwCasosAdministrativo AS  
SELECT Id, Designacao, DataInicio, DataTermino, Cliente, CustoAbertura  
FROM Caso;
```

```
GRANT SELECT ON vwCasosAdministrativo TO 'administrativo'@'localhost';
```

No entanto, esta metodologia apresenta alguns problemas. Em primeiro lugar, torna mais difícil alterar o conjunto de utilizadores que podem executar cada operação. Por exemplo, para se remover uma permissão a um utilizador considerando a implementação de privilégios atual, basta remover-se esse utilizador da instrução SQL correspondente ao requisito de controlo desejado. No entanto, com o uso de vidas, é necessário recalcular-se todos os atributos a que um utilizador precisa de acesso antes de se fazer qualquer modificação: é necessário confirmar-se se os atributos a que o acesso foi desautorizado não são necessários para outra operação que o utilizador ainda tenha permissão para executar. Uma forma de endereçar este problema seria criar vidas para cada requisito, mas isso conduziria a um número absurdamente alto de vidas. Se o MySQL é capaz de gerir as permissões escondendo esta complexidade do administrador da BD, o método de gestão de permissões na criação de papéis e utilizadores é preferível.

Ademais, o controlo de permissões durante a criação dos papéis e utilizadores traz a vantagem de tornar transparente a implementação das interrogações. O programador de SQL não precisa de ter em conta que vidas deve usar em cada interrogação: pode usar a tabela real diretamente (ex.: “Caso”), e caso tente executar uma operação em atributos para os quais não tem permissões, encontrará um erro descritivo do que fez incorretamente.

Por outro lado, o uso de vidas é mais versátil. Como já foi mencionado, a criação de vidas permite um maior controlo sobre os privilégios, sendo capaz de fazer com que um utilizador apenas tenha acesso a determinados registo. Ademais, utilizando-se agregações na criação de vidas, é também possível dar a um utilizador permissão para consultar dados estatísticos calculados pela agregação, sem este conhecer

os dados crus que lhes deram origem. Por exemplo, se a agência desejasse ter um contador no seu sítio web a mostrar o número de casos resolvidos, poderia criar a seguinte vista que seria consultada pela API do sítio:

```
CREATE VIEW vwWebsite AS
    SELECT COUNT(Id) AS Casos
    FROM Caso
    WHERE Caso.DataTermino IS NOT NULL;
```

Isto é só um exemplo ilustrativo do que pode ser feito com vistas e agregações, mas note-se que há modos melhores de se implementar esta funcionalidade (como, por exemplo, com um gatilho e uma tabela regular dedicada ao sítio web), que não exigem uma iteração pela tabela “Caso” sempre que um utilizador realiza um pedido ao sítio web. O uso de agregações nos requisitos RC10 e RC11 faz com que estes pudessem ter sido implementados com recurso a duas vistas, às quais apenas administradores teriam acesso para consulta. No entanto, os privilégios atribuídos na criação dos papéis não revelam aos administradores nenhuma informação sensível que apenas deveria ser acedida por detetives, pelo que se julga desnecessário um maior controlo das permissões utilizando vistas.

Como não é necessária granularidade de controlo de privilégios ao nível dos registo, a equipa de desenvolvimento optou por fazer o controlo de permissões na criação dos papéis dos utilizadores. No entanto, informou-se sobre o funcionamento de vistas de utilização, que podem ser necessárias para implementar requisitos de controlo mais complexos que surjam no futuro.

Outras utilidades de vistas não justificaram a sua utilização. Vistas de utilização permitem estabelecer uma camada de abstração entre a implementação da BD e os subesquemas dos utilizadores, algo que a equipa de desenvolvimento não julgou necessário numa BD de tão baixa complexidade como a *Veritatis*. Ademais, vistas podem ser utilizadas para simplificar a implementação de interrogações, permitindo, por exemplo, reutilização de código. No entanto, isto não se verificou necessário ao longo deste processo de implementação, que é descrito na secção que se segue.

5.6. Tradução das interrogações do utilizador para SQL

Estando já implementadas em álgebra relacional as interrogações à BD pedidas pelos requisitos de manipulação, a sua conversão para SQL é quase sempre imediata, constituindo apenas uma simples mudança de sintaxe para expressar a mesma semântica. Nesta secção, serão apresentadas as instruções SQL na mesma ordem que as suas expressões de álgebra relacional correspondentes, começando com as interrogações mais simples e apresentando novos conceitos um a um. Ademais, não será justificado por que motivos se utilizaram seleções, projeções, junções, etc., devendo o leitor consultar a secção [Validação do Modelo com Interrogações do Utilizador](#) para essas justificações. Nessa secção, os nodos das árvores sintáticas das expressões de álgebra relacional encontram-se coloridos, tal como várias

partes das instruções SQL aqui apresentadas, permitindo estabelecer uma associação entre as duas linguagens e informar o leitor de como surgiu cada parte de cada instrução SQL.

Começando com o requisito RM19, referente à consulta de todos os casos em aberto, este pode ser implementado com uma seleção dos casos pedidos, seguida de uma projeção dos atributos pedidos. Em SQL, consultas são feitas com a instrução SELECT, que pode ser seguida do conjunto atributos a consultar, separados entre vírgulas, para ser feita uma projeção. A seguir a estes atributos, FROM Caso especifica a relação que constitui o argumento da seleção feita em WHERE DataTermino IS NULL, que conserva apenas os registos com datas de término nulas, ou seja, os casos ainda em investigação (Oracle 2024a, pp. 2831-2837). Segue-se a instrução SQL que implementa RM19:

```
SELECT Id, Designacao, DataInicio  
      FROM Caso  
     WHERE DataTermino IS NULL;
```

De seguida, RM2, a enumeração dos clientes com casos em aberto, é um requisito cuja implementação requer uma junção e uma ordenação. Observando o código SQL abaixo, mantém-se a estrutura SELECT <atributos> FROM <relação> WHERE <condição>, correspondente à projeção de alguns atributos dos casos abertos selecionados. Note-se, no entanto, que devido à presença de várias relações nesta interrogação, os atributos são especificados com a notação Tabela.Atributo (Oracle, 2024a, pp. 2832). No final da instrução, ORDER BY Cliente.Nome ASC é responsável pela ordenação de acordo com o nome do cliente por ordem natural ascendente (Oracle, 2024a, pp. 2833-2834). Esta instrução SQL permite apresentar como é feita a tradução de junções de álgebra relacional. No caso, a θ-junção (em específico, a equijunção) da expressão de álgebra relacional é traduzida para a estrutura <tabela> INNER JOIN <tabela> ON <condição> (Oracle 2024a, p. 2842). Segue-se a instrução SQL que implementa RM2:

```
SELECT Cliente.Id, Cliente.Nome, Caso.Id, Caso.Designacao, Caso.DataInicio  
      FROM Caso INNER JOIN Cliente ON Caso.Cliente = Cliente.Id  
     WHERE Caso.DataTermino IS NULL  
     ORDER BY Cliente.Nome ASC;
```

O requisito RM4, a enumeração das funções executadas por pelo menos um funcionário, é de simples implementação com os conceitos já abordados. No entanto, é necessário o uso da diretiva SELECT DISTINCT, dado que SELECT, ao contrário da projeção de álgebra relacional, não remove registos repetidos, pelo que, sem o uso da palavra reservada DISTINCT, que faz esta remoção (Oracle, 2024a, pp. 2836-2837), funções executadas por mais do que um funcionário surgiriam na resposta da interrogação mais do que uma vez. O uso de DISTINCT não é necessário nas restantes projeções, visto que estas incluem sempre o conjunto de atributos constituintes de uma chave candidata da relação a projetar, assegurando a unicidade de cada registo. Ademais, note-se que esta expressão não é a mesma que a sua correspondente de álgebra relacional, apesar de ser sua equivalente, pois a projeção

π_{Funcao} (Exerce) na expressão original é completamente desnecessária, podendo esta parte da expressão ser substituída pela relação Exerce. Também deve ser notado que muitas outras expressões SQL não são iguais (mas são equivalentes) às suas correspondentes de álgebra relacional, devido à ordem de aplicação das operações distintas. Segue-se a instrução SQL que implementa RM4:

```
SELECT DISTINCT Funcao.Id, Funcao.Designacao
  FROM Exerce INNER JOIN Funcao ON Exerce.Funcao = Funcao.Id;
```

A calculadora RelaX, utilizada para a testagem das expressões de álgebra relacional, não suporta parametrização das expressões. O mesmo não pode ser dito em relação a SQL, onde instruções parametrizadas podem ser pré-preparadas, permitindo a substituição dos valores dos parâmetros em tempo de execução. Estas instruções podem ser utilizadas para implementar interrogações como RM7, a enumeração de todos os casos associados a um professor especificado pelo seu identificador. A diretiva PREPARE <nome> FROM <instrução> permite a pré-compilação de instruções (Oracle, 2024a, pp. 2956-2958), como se vê na implementação de RM7 abaixo:

```
PREPARE RM7 FROM
  'SELECT Caso.Id, Caso.Designacao, Caso.DataInicio
   FROM Caso
  WHERE Caso.Cliente = ?
 ORDER BY Caso.DataInicio ASC;';
```

A interrogação SQL é uma simples projeção de alguns atributos do resultado da ordenação por data de registo dos casos, casos estes selecionados com base no cliente a que estão associados (SELECT <atributos> FROM <tabela> WHERE <condição> ORDER BY <atributo> ASC). O identificador de cliente na condição é representado com um ponto de interrogação (Oracle, 2024a, p. 2956), e será substituído quando se pede a execução da instrução SQL. A interrogação pode ser executada com a diretiva EXECUTE, que apenas toma variáveis como argumentos (Oracle, 2024a, p.2958), sendo necessário carregar valores para estas utilizando a diretiva SET (Oracle, 2024a, p. 3072), como se vê no código abaixo, correspondente à execução da interrogação anterior para o utilizador de identificador 1:

```
SET @RM7Cliente = 1;
EXECUTE RM7 USING @RM7Cliente;
```

Note-se que instruções pré-preparadas devem ser dealocadas quando não são mais necessárias, ação que pode ser desempenhada com a diretiva DEALLOCATE, que no caso é utilizada na instrução DEALLOCATE PREPARE RM7.

O requisito RM15, a listagem dos funcionários que exercem uma função especificada pelo seu identificador, tal como o requisito anterior, é parametrizado, pelo que se recorre a instruções pré-preparadas para a sua implementação. Para todas as operações de álgebra relacional utilizadas, já foi

enunciada a sua forma de conversão para SQL. Segue-se, então, a criação desta interrogação, a sua execução para a “Funcao” de identificador 4, e a sua dealocação:

```

PREPARE RM15 FROM
    'SELECT Funcionario.Id, Funcionario.Nome
     FROM Exerce
      INNER JOIN Funcionario
      ON Exerce.Funcionario = Funcionario.Id
     WHERE Exerce.Funcao = ?
     ORDER BY Funcionario.Nome ASC;';

SET @RM15Funcao = 4;
EXECUTE RM15 USING @RM15Funcao;
DEALLOCATE PREPARE RM15;

```

Estando implementada em álgebra relacional a interrogação de RM10, a enumeração de todas as provas associadas a um caso especificado pelo seu identificador, é trivial a sua tradução para SQL fazendo uso de instruções pré-alocadas, projeções, seleções, junções e ordenações, conceitos todos já previamente apresentados. Seguem-se as instruções SQL correspondentes à preparação, execução (“Caso” de identificador 6) e dealocação desta interrogação:

```

PREPARE RM10 FROM
    'SELECT Provas.Descricao, Provas.Local, Procedimento.Data
     FROM Provas
      INNER JOIN Procedimento
      ON Provas.Procedimento = Procedimento.Id
     WHERE Procedimento.Caso = ?
     ORDER BY Procedimento.Data ASC;';

SET @RM10Caso = 6;
EXECUTE RM10 USING @RM10Caso;
DEALLOCATE PREPARE RM10;

```

Ao contrários das duas interrogações anteriores, a implementação do requisito RM17, a enumeração dos contactos de um utilizador pesquisado pelo seu nome, implica a apresentação de vários novos conceitos de SQL. Em primeiro lugar, esta interrogação requer o uso de uma união, representada em SQL com a estrutura <tabela> UNION <tabela> (Oracle, 2024a, pp. 2872-2873), podendo estas tabelas ser resultados de instruções SELECT. Nas duas instruções SELECT internas no código abaixo (a vermelho), observam-se duas diferenças quando comparadas com as instâncias anteriores do uso desta diretiva. Em primeiro lugar, os valores textuais “Email” e “Telefone” são colocados em atributos de nome “Tipo” com recurso à palavra reservada AS (Oracle, 2024a, p. 2833). Note-se que duas plicas são utilizadas em vez de uma, dado que essa é a forma de escapar este caráter quando dentro de um valor textual (a instrução da interrogação). Ademais, tal como na expressão de álgebra relacional, a operação LIKE é utilizada na comparação dos nomes dos clientes (Oracle, 2024a, pp. 2354-2356). Todavia, o valor ‘%?’ após o LIKE, utilizado na expressão álgebra relacional, não teria o efeito desejado aqui, pois o ponto de interrogação seria interpretado como um caráter e não como a referência a um argumento. Consequentemente, para construir o valor textual desejado, utiliza-se a função CONCAT (Oracle, 2024a,

p. 2343). Além disso, devido à necessidade de ordenação do resultado da união, é necessário colocar esta operação entre parênteses, constituindo uma subinterrogação à qual é dado o nome “Contactos” com o uso da palavra reservada AS. Uma projeção de todos os atributos desta relação pode ser feita com a estrutura SELECT * FROM (Oracle, 2024a, p. 2832), seguida da cláusula de ordenação ORDER BY. Note-se que a instrução SQL não faz a renomeação presente no topo da árvore sintática da expressão de álgebra relacional correspondente, optando por fazê-la durante as projeções dos endereços eletrónicos e dos números de telefone. Seguem-se as instruções correspondentes à preparação, execução (para o nome “Miguel”) e dealocação da interrogação, notando que é necessário providenciar dois argumentos durante a execução, dado que a expressão pré-compilada contém dois pontos de interrogação:

```

PREPARE RM17 FROM
'SELECT * FROM (
    SELECT Cliente.Id, Cliente.Nome, Emails.Email AS Contacto, ''Email'' AS Tipo
        FROM Emails INNER JOIN Cliente ON Emails.Cliente = Cliente.Id
        WHERE Cliente.Nome LIKE CONCAT('''%', ?, '%''')
UNION
    SELECT Cliente.Id, Cliente.Nome, Telefones.Telefone AS Contacto, ''Telefone'' AS Tipo
        FROM Telefones INNER JOIN Cliente ON Telefones.Cliente = Cliente.Id
        WHERE Cliente.Nome LIKE CONCAT('''%', ?, '%''')
) AS Contactos
ORDER BY Contactos.Nome ASC;';

SET @RM17Nome = 'Miguel';
EXECUTE RM17 USING @RM17Nome, @RM17Nome;
DEALLOCATE PREPARE RM17;

```

A implementação de RM13, o cálculo do número de casos investigados e de procedimentos executados pelos detetives desde uma dada data, envolve a junção de duas relações que são resultados de execuções de interrogações. Tal como feito para RM13, estas são colocadas em parênteses e denominam-se *subqueries* (subinterrogações). A junção destas tabelas, ao contrário de todas as junções feitas até ao momento, não é uma equijunção, mas sim uma junção externa à esquerda (\bowtie), representada em SQL com a estrutura sintática <tabela> LEFT JOIN <tabela>. (Oracle, 2024a, pp. 2842-2843). Note-se que, na projeção do resultado desta junção (SELECT exterior do código abaixo), a função COALESCE é usada para substituir valores nulos pela constante 0, tal como já era feito na expressão de álgebra relacional (Oracle, 2024a, pp. 2295-2296). Quanto às tabelas que sofrem a junção externa, estas são os primeiros exemplos de resultados de agregações. Tomando a primeira subinterrogação como exemplo, os casos são agrupados pelo atributo “Caso.Detetive”, algo expresso pela cláusula GROUP BY Caso.Detetive. Esta agregação apenas é aplicada ao resultado da seleção dos casos com a data posterior à desejada (cláusula WHERE anterior). Note-se que nesta agregação é feita uma contagem. Isto é indicado pelos atributos projetados em SELECT: a função COUNT é usada para contar identificadores (casos únicos), resultado ao qual se dá o nome “Casos” (Oracle, 2024a, p. 2568). Tem-se algo semelhante para a agregação de procedimentos na segunda subinterrogação. Seguem-se as instruções correspondentes à preparação, execução (para a data 2020-01-01) e

dealocação da interrogação, notando que é necessário providenciar dois argumentos durante a execução, dado que a expressão pré-compilada contém dois pontos de interrogação:

```

PREPARE RM13 FROM
    'SELECT Casos.Detetive, Casos.Casos, COALESCE(Procedimentos.Procedimentos, 0) AS
        Procedimentos
    FROM (
        SELECT Caso.Detetive, COUNT(Caso.Id) AS Casos
        FROM Caso
        WHERE Caso.DataInicio > ?
        GROUP BY Caso.Detetive
    ) AS Casos LEFT JOIN (
        SELECT Caso.Detetive, COUNT(Procedimento.Id) AS Procedimentos
        FROM Procedimento INNER JOIN Caso ON Procedimento.Caso = Caso.Id
        WHERE Procedimento.Data > ?
        GROUP BY Caso.Detetive
    ) AS Procedimentos ON Casos.Detetive = Procedimentos.Detetive';

SET @RM13Data = '2020-01-01';
EXECUTE RM13 USING @RM13Data, @RM13Data;
DEALLOCATE PREPARE RM13;

```

De seguida, RM14, o cálculo da receita e despesa de cada caso, pode ser trivialmente implementado com recurso aos vários aspetos da sintaxe SQL usados em RM13: agregações, subinterrogações, a função COALESCE e junções exteriores. A nova adição é que, em vez de ser feita uma junção à esquerda, é feita uma junção à direta com uma condição (\bowtie), representada pela estrutura <tabela> RIGHT JOIN <tabela> ON <condição> (Oracle, 2024a, pp. 2840-2841). Segue-se a instrução SQL correspondente à implementação de RM14:

```

SELECT
    Caso.Id,
    Caso.Designacao,
    COALESCE(CustosProcedimentos.CustoCliente, 0) + Caso.CustoAbertura AS CustoCliente,
    COALESCE(CustosProcedimentos.CustoAgencia, 0) AS CustoAgencia,
    COALESCE(CustosProcedimentos.CustoCliente, 0) + Caso.CustoAbertura -
        COALESCE(CustosProcedimentos.CustoAgencia, 0) AS Balanco
FROM (
    SELECT
        Procedimento.Caso,
        SUM(Procedimento.CustoCliente) AS CustoCliente,
        SUM(Procedimento.CustoAgencia) AS CustoAgencia
        FROM Procedimento
        GROUP BY Procedimento.Caso
    ) AS CustosProcedimentos
RIGHT JOIN Caso ON CustosProcedimentos.Caso = Caso.Id;

```

Por último, RM18 exige a saída de várias relações incompatíveis (diferente número de argumentos e/ou diferentes domínios dos atributos), pelo que a equipa de desenvolvimento julgou melhor a sua implementação numa função, descrita em detalhe em [Implementação de Procedimentos, Funções e Gatilhos](#).

Para a testagem da correção das instruções SQL derivadas, estas foram executadas e os seus resultados foram comparados com os resultados das expressões de álgebra relacional correspondentes (usando a calculadora RelaX). Ademais, cada interrogação foi executada por um utilizador de MySQL que apenas tinha as permissões necessárias para a executar a interrogação a ser testada (ver [Criação de Utilizadores](#)

[da Base de Dados](#)), de modo a se testar também a correção dos privilégios considerados necessários para a execução de cada interrogação. Esta testagem não revelou qualquer incorreção, tanto nas interrogações traduzidas como nos privilégios atribuídos aos utilizadores para as executar.

5.7. Indexação do Sistema de Dados

Se corretamente utilizados, índices podem ser altamente benéficos para o desempenho de uma BD. Um índice é uma estrutura de dados que organiza os valores de um ou mais atributos de uma tabela, evitando uma iteração completa pela relação quando se procuram encontrar registo com base nos valores de determinados atributos. Se esses atributos se encontrarem indexados, pode ser possível a obtenção de um melhor desempenho (Oracle, 2024a, pp. 1892-1893). No entanto, é necessário notar que o uso de índices pode tornar as operações de inserção e de manipulação mais lentas, dado que estas podem passar a exigir atualizações ao índice. Ademais, é necessário notar que estes índices ocupam espaço de armazenamento, pelo que o seu uso deve ser ponderado em BDs de grande dimensão, onde o tamanho dos índices é considerável.

Devido ao pequeno número de registo na BD *Veritatis*, a equipa de desenvolvimento não utilizou nenhum índice no produto entregue ao cliente, visto que todas as interrogações podiam ser executadas em frações de segundos. No entanto, foi escrito o código de criação de vários índices (mas este não foi executado), que pode vir a ser utilizado futuramente caso o desempenho de alguma interrogação se torne pouco desejável à medida que a BD cresce.

Em SQL, a diretiva `CREATE INDEX` é utilizada para criar um índice para um conjunto de atributos numa relação (Oracle 2024a, pp. 2678-2692):

```
CREATE INDEX <nome> ON <Tabela>(<atributo 1>, <atributo 2>, ...);
```

Para cada interrogação na secção anterior, analisou-se a expressão SQL desenvolvida e procuraram-se instâncias onde ocorria identificação de registo numa tabela com base nos valores de alguns dos seus atributos (por exemplo, a identificação de registo de “Caso” com base em “Caso.Cliente”). É de notar que certos conjuntos de atributos já se encontram indexados: as chaves primárias e os atributos definidos como `UNIQUE` (Oracle, 2024a, pp. 2711-2712). Tendo isso em conta, seguem-se alguns exemplos de índices que podem vir a melhorar o desempenho de interrogações específicas.

Para RM2, a equijunção entre “Caso” e “Cliente” beneficiaria de uma identificação rápida de casos com base no seu identificador de cliente. Logo, esta interrogação beneficia do índice criado pela seguinte instrução:

```
CREATE INDEX idxCasoCliente ON Caso(Cliente);
```

Para RM4, a equijunção entre “Exerce” e “Funcao” beneficiaria de uma identificação rápida dos registo de “Exerce” com base na sua “Funcao”:

```
CREATE INDEX idxExerceFuncao ON Exerce(Funcao);
```

Começa-se a reparar que é usualmente benéfica a indexação de uma relação pelas suas chaves estrangeiras, justificando este comportamento por parte de ferramentas de conversão de modelos lógicos para físicos (ex.: MySQL Workbench).

Devido à seleção utilizada em RM7, seria melhor para o desempenho da interrogação ser rápida a identificação de casos com base no cliente a eles associado, sendo benéfico o uso do índice definido para RM2, idxCasoCliente. A ordenação utilizada também pode beneficiar de uma estrutura ordenada de referências a registos (um índice) para melhorias de desempenho (Oracle, 2024a, pp. 1858-1860), podendo o seguinte índice ser utilizado: CREATE INDEX idxCasoDataInicio ON Caso(DataInicio);.

RM15 pode beneficiar, por exemplo, de um índice em “Funcionario.Nome” (para a ordenação), e outro em “Exerce.Funcao” (como RM4, para a seleção). No entanto, o índice mais relevante seria o utilizado na equijunção, “Exerce.Funcionario”, pois é necessário consultar os registos de “Exerce” associados a cada um dos funcionários, mas apenas uma destas consultas é efetuada na execução da seleção, por exemplo. Tem-se algo muito semelhante para a interrogação RM10, a mesma expressão SQL mas utilizando outras tabelas e outros atributos.

Note-se, por exemplo, que a pesquisa textual pelo nome de um cliente, em RM17, não beneficiaria dos índices regulares descritos até agora. Seria necessário o uso de um índice FULLTEXT, e também necessárias modificações à comparação textual da interrogação (Oracle, 2024a, pp. 2368-2370), visto que a pesquisa textual é completamente diferente da ordenação de valores (por exemplo, o subtexto que se deseja procurar pode estar no meio do nome do cliente). No entanto, índices em “Emails.Cliente” e “Telefones.Cliente” ainda seriam benéficos para a pesquisa nestas duas relações, feita por RM17:

```
CREATE INDEX idxEmailsCliente ON Emails(Cliente);
CREATE INDEX idxTelefonesCliente ON Telefones(Cliente);
```

Mantendo a estratégia que se tem vindo a seguir, este processo de indexação poderia ser continuado para as restantes relações. No entanto, deve reiterar-se que a indexação não deve ser uma preocupação nesta etapa de desenvolvimento da BD, visto que o desempenho será excelente devido ao pequeno número de registos. Ademais, a equipa de desenvolvimento não foi capaz de detetar diferenças de desempenho significativas, mesmo inserindo na BD milhares de registos com dados falsos, tal como foi feito para a medição do tamanho real de cada registo. A equipa acredita que um número de registos de uma ordem de grandeza muito maior seria necessário para tal. Num grande ambiente de produção, a decisão do uso de cada índice necessita de uma avaliação prática de desempenho, o que não se conseguiu replicar. No entanto, algo pode ser concluído: o desempenho de momento já é excelente sem indexação.

Encontra-se, no anexo ???, um ficheiro com a criação dos índices descritos nesta secção.

5.8. Implementação de Procedimentos, Funções e Gatilhos

MySQL suporta a definição e execução de procedimentos, funções e gatilhos, que permitem, dentro do próprio SGBD, o controlo de fluxo de execução de várias interrogações, sem a necessidade de software aplicacional para gerir operações condicionais e cíclicas. Procedimentos e funções constituem geralmente ferramentas úteis para a administração de uma BD, enquanto que gatilhos, executados na alteração dos conteúdos de uma tabela, são utilizados para assegurar um estado da BD consistente, por exemplo, através da atualização de atributos derivados.

Já foi mencionado, em [Cálculo do Espaço da Base de Dados](#), que um procedimento foi utilizado para se povoar a BD com milhares de registo, para depois se medir o seu tamanho final. Como procedimentos, funções e gatilhos são constituídos por várias instruções separadas por ponto e vírgula, é necessário permitir que o MySQL reconheça o fim do procedimento em si. Logo de inicio é aplicado o comando DELIMITER, que permite redefinir o delimitador por omissão de MySQL (Oracle, 2024a, pp. 4940-4941). Depois da declaração do procedimento, o delimitador é redefinido como o ponto e vírgula, como se observa no código abaixo:

```
DELIMITER $$  
-- Criação do procedimento, função ou gatilho $$  
DELIMITER ;
```

Entre as alterações de delimitador, o procedimento pode ser criado do seguinte modo:

```
CREATE PROCEDURE EncheBD(IN nrRegistros INT)  
BEGIN  
    -- Código do procedimento  
END $$
```

A instrução CREATE PROCEDURE cria um procedimento de nome “EncheBD” com um parâmetro de entrada (IN) denominado “nrRegistros”, cujo domínio é inteiro (Oracle, 2024a, pp. 2693-2699). O bloco BEGIN <corpo> END define o corpo do procedimento, onde são declaradas variáveis locais e definidas, por ordem de execução, todas as instruções a executar (Oracle, 2024a, p. 2959).

Como algumas das inserções de dados falsos feitas ao longo do procedimento exigem que seja fornecida uma data, a data de execução do procedimento é obtida através da função CURDATE (Oracle, 2024a, p. 2320) e armazenada numa variável dt de domínio DATE, declarada através do uso da diretiva DECLARE. O valor por omissão por omissão da variável (DEFAULT) é igual ao resultado da função mencionada (Oracle, 2024a, p. 2961), como se vê abaixo:

```
DECLARE dt DATE DEFAULT CURDATE();
```

De seguida, para melhor desempenho, a instrução START TRANSACTION inicia uma transação, que é terminada no final do procedimento com a instrução COMMIT. Sem esta transação, todas as inserções

exigiriam escoamento de dados para o meio de armazenamento, de modo a garantir a persistência dos dados inseridos pela operação, degradando o desempenho do procedimento (Oracle, 2024a, pp. 2887-2890). O código de transação pode ser visto abaixo:

```
START TRANSACTION;
-- Código que será analisado
COMMIT;
```

O outro dos dois motivos para o uso de transações é a possibilidade de reversão de operações de manipulação caso ocorram falhas (instrução ROLLBACK), garantindo a consistência dos dados na BD (Oracle, 2024a, pp. 2887-2890). No que foi necessário implementar para a Agência Veritatis, não houve necessidade de utilização desta funcionalidade.

Dentro da transação, para se repetir o mesmo conjunto de instruções diversas vezes (inserção de regtos), é utilizada a estrutura REPEAT <instruções> UNTIL <condição> END REPEAT, que repete as instruções no seu corpo até se verificar a condição especificada (Oracle, 2024a, p. 2965). Para garantir que são feitas nrRegistros iterações, o valor desta variável é decrementado em uma unidade a cada iteração com recurso à instrução SET (Oracle, 2024a, pp. 3071-3072):

```
REPEAT
    -- Operações de inserção a analisar
    SET nrRegistros = nrRegistros - 1;
UNTIL nrRegistros = 0
END REPEAT;
```

As inserções feitas dentro do ciclo não são de grande complexidade, tendo a sua estrutura já sido explicada na secção [Povoamento da Base de Dados](#). Apenas foram utilizadas duas novas funções: REPEAT, para a geração de valores textuais com um determinado comprimento através da repetição de um valor textual (no caso, de um único carácter) várias vezes, com base nos comprimentos médios definidos em [Cálculo do Espaço da Base de Dados](#) (Oracle, 2024a, p. 2348), e LPAD, que adiciona um carácter à esquerda de um valor textual o número de vezes necessário para se formar um outro valor textual com o comprimento desejado (Oracle, 2024a, p. 2347). Os valores inseridos que precisam de ser únicos são definidos com base em nrRegistros (número do registo), sendo usada esta segunda função quando o valor deve ser textual com e ter comprimento específico. Segue-se o código utilizado para as primeiras inserções:

```

INSERT INTO Funcao(Id, Designacao) VALUES
    (nrRegistros, REPEAT('A', 20));
INSERT INTO Funcionario(Id, Nome, NIF, Salario, SeguroVida, Email, Telefone) VALUES
    (nrRegistros, REPEAT('A', 30), 100000000 + nrRegistros, 0000.00, 111111111,
     LPAD(nrRegistros, 30, '0'), LPAD(nrRegistros, 9, '0'));
INSERT INTO Exerce(Funcao, Funcionario) VALUES
    (nrRegistros, nrRegistros);
INSERT INTO Cliente(Id, Nome, Morada, NIF, LocalTrabalho) VALUES
    (nrRegistros, REPEAT('A', 30), REPEAT('A', 35), 100000000 + nrRegistros,
     REPEAT('A', 35));
INSERT INTO Emails(Cliente, Email) VALUES
    (nrRegistros, LPAD(nrRegistros, 30, '0'));
INSERT INTO Telefones(Cliente, Telefone) VALUES
    (nrRegistros, LPAD(nrRegistros, 9, '0'));
INSERT INTO Caso(Id, Designacao, DataInicio, DataTermino, DataPagamento,
               CustoAbertura, Cliente, Detetive) VALUES
    (nrRegistros, REPEAT('A', 20), dt, dt, dt, 000.00, nrRegistros, nrRegistros);

```

Por último, para apenas se inserir um décimo das entradas de registo com atributos de domínio TEXT, utiliza-se uma estrutura condicional IF <condição> THEN <instruções> END IF para as instruções dentro do bloco condicional apenas serem executadas quando a condição especificada é cumprida. No código abaixo, apenas se inserem os registo mencionados quando o número de registo é um múltiplo de 10:

```

IF nrRegistros % 10 = 0 THEN
    INSERT INTO TipoProcedimento(Id, Designacao, CustoAgencia, CustoCliente, Descricao)
        VALUES (nrRegistros, REPEAT('A', 20), 000.00, 000.00, REPEAT('A', 2000));
    INSERT INTO Procedimento(Id, Tipo, Notas, Data, CustoAgencia, CustoCliente, Caso)
        VALUES
            (nrRegistros, nrRegistros, REPEAT('A', 2000), dt, 000.00, 000.00, nrRegistros);
    INSERT INTO Provas(Procedimento, Descricao, Local) VALUES
        (nrRegistros, REPEAT('A', 2000), REPEAT('A', 35));
END IF;

```

Como este procedimento apenas deve ser executado pelo administrador da BD, não é necessária a atribuição do privilégio para a sua execução a nenhum utilizador. O código do procedimento pode ser encontrado no [penúltimo anexo](#), juntamente com a função e o gatilho que serão apresentados de seguida. A execução deste procedimento, por exemplo, para 10000 registo, pode ser feita com a instrução CALL EncheBD(10000) (Oracle, 2024a, pp. 2787-2788).

Para a implementação do requisito RM18, a geração de uma fatura para um “Caso” especificado pelo seu identificador, foi criada uma função que devolve um valor textual (para impressão, por exemplo). Dentro de uma troca de delimitadores igual à descrita anteriormente, pode definir-se a assinatura da função:

```

CREATE FUNCTION RM18(IdCaso INT)
RETURNS TEXT
NOT DETERMINISTIC
SQL SECURITY INVOKER
READS SQL DATA
BEGIN
    -- Código a analisar
END $$
```

Com base na análise da documentação da Oracle (2024a, pp. 2693-2699), pode ser afirmado que esta função, de nome “RM18” aceita como único argumento de entrada um inteiro denominado “IdCaso”, devolve um valor textual, apresenta um comportamento não-determinístico (o valor devolvido não depende apenas dos valores dos argumentos) e pode consultar dados da BD. A parte SQL SECURITY INVOKER será explicada em detalhe quando for abordado como é feito o controlo de permissões para a execução de funções.

Dentro do corpo da função, começa-se por se declarar todas as variáveis que serão necessárias, do mesmo modo que é feito nos procedimentos (Oracle, 2024a, p. 2961):

```
DECLARE Abertura DECIMAL(8, 2);
DECLARE Total DECIMAL(8, 2);
DECLARE Fatura TEXT;

DECLARE Fim INTEGER DEFAULT 0;
DECLARE Item VARCHAR(100);
DECLARE Itens CURSOR FOR
    SELECT CONCAT(TipoProcedimento.Designacao,
                  ': ',
                  Procedimento.CustoCliente, '€')
    FROM Procedimento
    INNER JOIN TipoProcedimento
    ON Procedimento.Tipo = TipoProcedimento.Id
    WHERE Procedimento.Caso = IdCaso;
```

Os domínios decimais das variáveis “Abertura” e “Total” foram declarados com mais dígitos do que os valores decimais armazenados na BD. Tal foi feito para permitir a computação de adições das quais resultam valores elevados, fora do menor domínio dos tipos usados para armazenamento. A variável “Itens” é diferente das restantes: é um cursor, que permite a iteração pelos valores do resultado de uma interrogação. Este foi implementado com base no exemplo da Oracle (2024a, pp. 2966-2967). A interrogação criada gera uma linha de texto para cada procedimento executado para o caso, com o seu nome e o seu custo.

A instrução seguinte permite a execução de uma outra instrução quando o fim de um cursor é atingido (Oracle, 2024a, pp. 2996-2973). Deste modo, definindo a variável de nome “Fim” como 1, é possível detetar o fim do cursor e parar qualquer iteração que esteja a ser feita:

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET Fim = 1;
```

De seguida, no pedaço de código abaixo, dá-se a inicialização de várias variáveis, como o valor textual de saída (“Fatura”), o custo de abertura do caso (“Abertura”), e o custo total a cobrar ao cliente (“Total”). Note-se o uso da diretiva `SELECT <valor> INTO <variável>`, que permite o carregamento para uma variável do resultado de uma interrogação (Oracle 2024a, pp. 2837-2840).

```

SET Fatura = '';
SELECT Caso.CustoAbertura INTO Abertura
  FROM Caso
 WHERE Caso.Id = IdCaso;
SELECT COALESCE(SUM(Procedimento.CustoCliente), 0) + Abertura INTO Total
  FROM Procedimento
 WHERE Procedimento.Caso = IdCaso;

```

As interrogações acima são simples, sendo o único detalhe a ser notado o uso de COALESCE no total da soma de custos de procedimentos associados ao caso pedido, visto que este valor pode ser nulo se nenhum “Procedimento” tiver sido executado, sendo necessário tornar o valor resultante da agregação no elemento neutro da adição, zero.

De seguida, verifica-se se o caso pedido pelo utilizador existe. Se este não existir, o custo de abertura (“Abertura”) será nulo, podendo sair-se da função com a devolução do valor de “Fatura”, definido como uma mensagem de erro, através da instrução RETURN (Oracle, 2024a, p. 2996):

```

IF Abertura IS NULL THEN
  SET Fatura = CONCAT('Caso ', IdCaso, ' não encontrado!');
  RETURN Fatura;
END IF;

```

Agora, deve iterar-se por todos os procedimentos, adicionando-os um a um ao valor textual de Fatura. Depois do cursor ser aberto (Oracle, 2024a, p. 2968), segue-se um ciclo expresso pela estrutura LOOP <instruções> END LOOP, que não termina até se sair dele com a instrução LEAVE (Oracle, 2024a, pp. 2964-2965). Este ciclo encontra-se etiquetado com a etiqueta processaItem para ser possível identificá-lo na instrução LEAVE. Logo, o código abaixo abre o cursor, itera por todos os procedimentos até não existirem mais (Fim = 1), adicionando uma linha a “Fatura” usando a função CONCAT, e termina com o fecho do cursor (Oracle, 2024a, p. 2967).

```

OPEN Itens;
processaItem: LOOP
  FETCH Itens INTO Item;
  IF Fim = 1 THEN
    LEAVE processaItem;
  END IF;

  SET Fatura = CONCAT(Fatura, Item, '\n');
END LOOP processaItem;
CLOSE Itens;

```

No final da função, no caso da ausência de procedimentos, uma mensagem é colocada na fatura a assinalar isso. Depois, são adicionadas duas linhas para o custo de abertura do caso e o valor total a pagar. O texto da fatura é devolvido com a instrução RETURN:

```

IF Fatura = '' THEN
    SET Fatura = 'Sem procedimentos a apresentar!\n';
END IF;

-- Mostrar valores finais
SET Fatura = CONCAT(Fatura, '\nCustoAbertura: ', Abertura, '€');
SET Fatura = CONCAT(Fatura, '\nTotal: ', Total, '€');
RETURN Fatura;

```

Apesar da função estar concluída, utilizadores que não o administrador não têm permissão para a executar. A parte SQL SECURITY INVOKER na assinatura da função define que quem executa a função deve também ter as permissões necessárias para a execução das operações nela contidas. Deste modo, é possível atribuir a todos os utilizadores/papéis locais permissões para a execução da função, e apenas os que têm permissão de consulta das tabelas necessárias a poderão executar. Assim, é possível separar a definição de privilégios da definição da função (os papéis abaixo são todos os existentes, independentemente de quem possa executar a função):

```

GRANT EXECUTE ON FUNCTION RM18 TO
    'administrador'@'localhost',
    'administrativo'@'localhost',
    'detetive'@'localhost';

```

O resultado da função pode ser obtido como se faz para qualquer outra função demonstrada até agora, através de uma operação SELECT. No exemplo que se segue, gera-se a fatura para o caso de identificador 1: SELECT RM18(1);

Em último lugar, foi implementado um gatilho para, com base nos preços em “TipoProcedimento”, se atribuírem preços a cada “Procedimento” inserido. Tal como ocorre para a criação do procedimento (*stored procedure*) e da função anteriores, a criação do gatilho deve ser feita dentro de um ambiente de troca de delimitadores. O código de implementação do gatilho é o seguinte:

```

CREATE TRIGGER ProcedimentoHerdarCusto
    BEFORE INSERT ON Procedimento
    FOR EACH ROW
BEGIN
    DECLARE Custo DECIMAL(5, 2);

    IF NEW.CustoAgencia = 0 THEN
        SELECT CustoAgencia INTO Custo
        FROM TipoProcedimento
        WHERE Id = NEW.Tipo;
        SET NEW.CustoAgencia = Custo;
    END IF;

    IF NEW.CustoCliente = 0 THEN
        SELECT CustoCliente INTO Custo
        FROM TipoProcedimento
        WHERE Id = NEW.Tipo;
        SET NEW.CustoCliente = Custo;
    END IF;
END $$
```

A instrução CREATE TRIGGER é utilizada para a criação de um gatilho de nome “ProcedimentoHerdarCusto”, que é chamado durante a inserção de registos em “Procedimento”, antes dos resultados da operação de inserção se tornarem persistentes, o que é evidenciado pelas palavras reservadas BEFORE INSERT. (Oracle, 2024a, pp. 2771-2773). Tal como ocorre com procedimentos e funções, o corpo do gatilho encontra-se entre BEGIN e END. A única diferença do gatilho quando comparado com as estruturas imperativas anteriores é a adição da relação NEW, que contém os registo que foram inseridos (a relação OLD está definida para gatilhos sobre outras operações que não inserções). A primeira estrutura condicional, em caso de, durante a inserção, não ter sido especificado o custo do procedimento para a agência, carrega para uma variável o valor do custo do “TipoProcedimento” correspondente ao registo inserido. Depois, atualiza o valor a inserir para o valor consultado. A segunda estrutura condicional faz o mesmo, mas para “CustoCliente”.

Note-se que a integridade referencial ainda não foi verificada durante a execução do gatilho (BEFORE INSERT), pelo que podem ser providenciados tipos de procedimento não existentes. Nesse caso, o gatilho falhará ao tentar carregar o valor NULL para “CustoAgencia” ou “CustoCliente”, mas nunca permitirá a violação da integridade referencial.

Este gatilho é a peça final da solução que permite a implementação do requisito RC5, “apenas detetives podem inserir procedimentos, sendo os seus custos derivados do seu tipo”. Os detetives não têm permissão para inserir valores nos campos “CustoAgencia” e “CustoCliente”, sendo o valor por omissão, zero, escolhido. Este gatilho substituirá esse valor pelo valor consultado na relação “TipoProcedimento”. A equipa de desenvolvimento julga que este gatilho não interferirá com o povoamento da BD, visto que não é esperado haver procedimentos com custos de 0€. Além disso, a existência de apenas um gatilho faz com que este não possa interferir com outros gatilhos, formando cadeias potencialmente perigosas. Mesmo assim, como a BD Veritatis pode vir a ser expandida no futuro, um mapa de gatilhos foi criado para facilmente se poder analisar se a introdução de um novo gatilho conduziria ou não a um risco de ciclo infinito. No seguinte grafo, as relações da BD surgem como nodos, e os gatilhos como arestas, partindo da tabela onde estão definidos para a(s) tabela(s) onde causam modificações. De momento, tem-se o seguinte simples mapa de gatilhos:

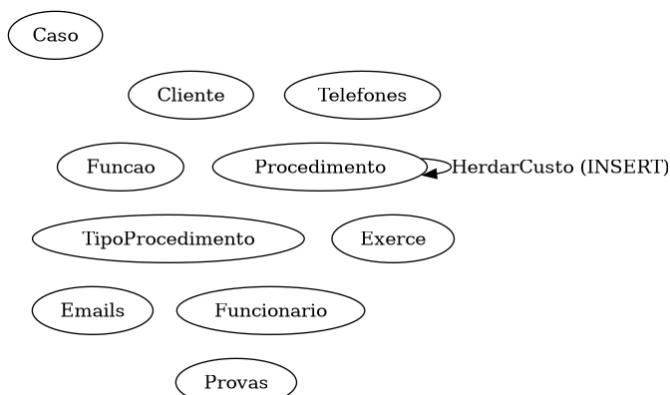


Figura 34 - Mapa de gatilhos da BD Veritatis.

6. Conclusões e Trabalho Futuro

A equipa de alunos, ao longo do último semestre, desenvolveu uma BD para a Agência Veritatis seguindo o ciclo de desenvolvimento de uma BD desde suas fases iniciais, a definição do sistema e dos requisitos, passando pela modelação conceitual e lógica, até a implementação física da BD e a sua exploração. Neste processo, a equipa pôde explorar e ganhar experiência em diversas áreas, como a gestão de projetos, a comunicação com clientes, diversas formas de modelação de uma BD, interação com um SGBD e principalmente como documentar adequadamente cada processo de desenvolvimento. Foi ainda possível explorar certos tópicos em maior detalhe, como o funcionamento do motor de armazenamento de um SGBD, e como isso se traduz no espaço utilizado por uma BD, tendo sido conduzida uma experiência prática para aprofundar o conhecimento da equipa nesta área.

Após a conclusão da segunda fase deste trabalho, a equipa de informáticos ponderou sobre os aspectos positivos e negativos do processo de desenvolvimento. Concluiu que, a seu ver, as tarefas propostas foram realizadas na sua íntegra e com correção, apesar do processo de desenvolvimento poder ter sido melhor documentado.

Há certas etapas de desenvolvimento cuja documentação poderia estar mais detalhada, mas onde a falta de documentação não causou problemas na execução dos passos seguintes do ciclo de vida da BD. Um exemplo disto ocorre com a etapa de definição de requisitos, onde os métodos utilizados e passos intermédios não são descritos em detalhe. Ademais, muitos documentos envolvidos nesta etapa encontram-se em falta (atas de reunião completas, documentação providenciada pela Agência Veritatis, etc.). Esta forma de falta de documentação teve origem numa má organização do tempo na primeira fase deste trabalho, deixando poucos recursos temporais e humanos para a melhoria da parte em questão.

Todavia, há instâncias de documentação em falta que se mostraram problemáticas para a realização de passos posteriores do ciclo de vida da BD. A título de exemplo, na definição dos atributos das relações do modelo lógico, não foram mencionados aspectos essenciais da documentação do modelo conceitual, como a definição de cada coluna e as restrições de cada relação que envolvem um ou mais atributos. Isto conduziu a que, na definição do modelo físico, fosse necessária referência cruzada das documentações dos modelo conceitual e lógico, ou até mesmo dos requisitos. Ademais, várias decisões (como a decisão de que todos os valores de custo não podem ser negativos) deveriam ter sido tomadas e documentadas na construção do modelo conceitual, visto que derivam das definições dos atributos nos requisitos, mas apenas foram definidas durante a implementação da BD com base em documentação

anterior. O motivo por detrás destas lacunas na documentação foi a inexperiência da equipa de desenvolvimento. Sendo a BD *Veritatis* a primeira que este grupo de alunos desenvolveu, várias necessidades de documentação não eram conhecidas quando a modelação da BD foi feita, apenas se detetando as lacunas descritas durante a implementação física. A equipa pôde aprender com estes erros cometidos, e já sabe com exatidão a documentação que precisa de escrever nas fases de modelação de bases de dados de projetos futuros.

Os aspetos que deveriam ter sido abordados em diferentes etapas do ciclo de vida da BD não se restringem à melhor documentação do processo de desenvolvimento. Pelo motivo enunciado anteriormente, também a verificação da correção dos requisitos de controlo (tabela 20) apenas foi realizada durante a implementação física, implicando a adição de novos requisitos para que a BD tivesse o funcionamento desejado pelo cliente. Este exemplo constitui outro aspeto a manter em mente para correção em projetos futuros.

Apesar de não terem sido sentidas grandes dificuldades na aplicação dos conhecimentos adquiridos em aula pelos alunos, houve-as na gestão do elevado número de tarefas a realizar, que conduziu à fraca organização temporal.

Em primeiro lugar, tendo em conta apenas a primeira fase do trabalho, foi concordado de forma unânime que a planificação de atividades original não foi respeitada. A distribuição temporal real das tarefas encontra-se no [último anexo](#), mas também em miniatura na figura abaixo. Demoras no processo de levantamento de requisitos conduziram a atrasos na sua validação e nas fases de desenvolvimento subsequentes. Ademais, a documentação do modelo conceitual exigida pelo Prof. Doutor Elias Ribeiro era mais detalhada do que a que a equipa lhe apresentou, pelo que foram necessárias mudanças que desviaram recursos humanos da construção do modelo lógico, atrasando também o seu desenvolvimento. Mesmo com estes atrasos, o projeto esteve pronto atempadamente, pois a planificação original deixava margem de manobra para situações exatamente como esta. No âmbito da distribuição de pessoas pelas tarefas, as diferenças da planificação original não foram significativas.



Figura 35 - Miniatura do diagrama de Gantt relativo às atividades realizadas na primeira fase.

De seguida, dizendo respeito à planificação da segunda fase do trabalho prático, a equipa de desenvolvimento reconhece que a planificação construída inicialmente, na ausência de experiência do que realmente se tratava a implementação de uma BD, não corresponde à realidade das tarefas que precisam de ser executadas. Logo, são poucas as comparações que se admitem entre a planificação e o mapa temporal de atividades mostrado abaixo, presente em ponto grande no [último anexo](#). A principal diferença a ter em conta é o atraso em uma semana no começo do desenvolvimento, de modo a que a equipa de alunos pudesse frequentar a primeira aula teórica de Bases de Dados sobre SQL. Quando comparado com o diagrama da figura 34, o diagrama abaixo demonstra maior paralelismo na execução das tarefas, algo possível devido à independência das tarefas entre si, ao contrário do que ocorria na primeira fase do trabalho. Além disso, houve oportunidade para aprimorar algumas das partes, algo evidenciado pela sua execução interrupta no diagrama abaixo.



Figura 36 - Miniatura do diagrama de Gantt relativo às atividades realizadas na segunda fase.

A equipa de alunos deve agora continuar a prestar serviços à Agência Veritatis, garantido a operacionalidade da BD e procurando melhorar o seu funcionamento com base no comentário crítico dos funcionários que a utilizam.

Referências

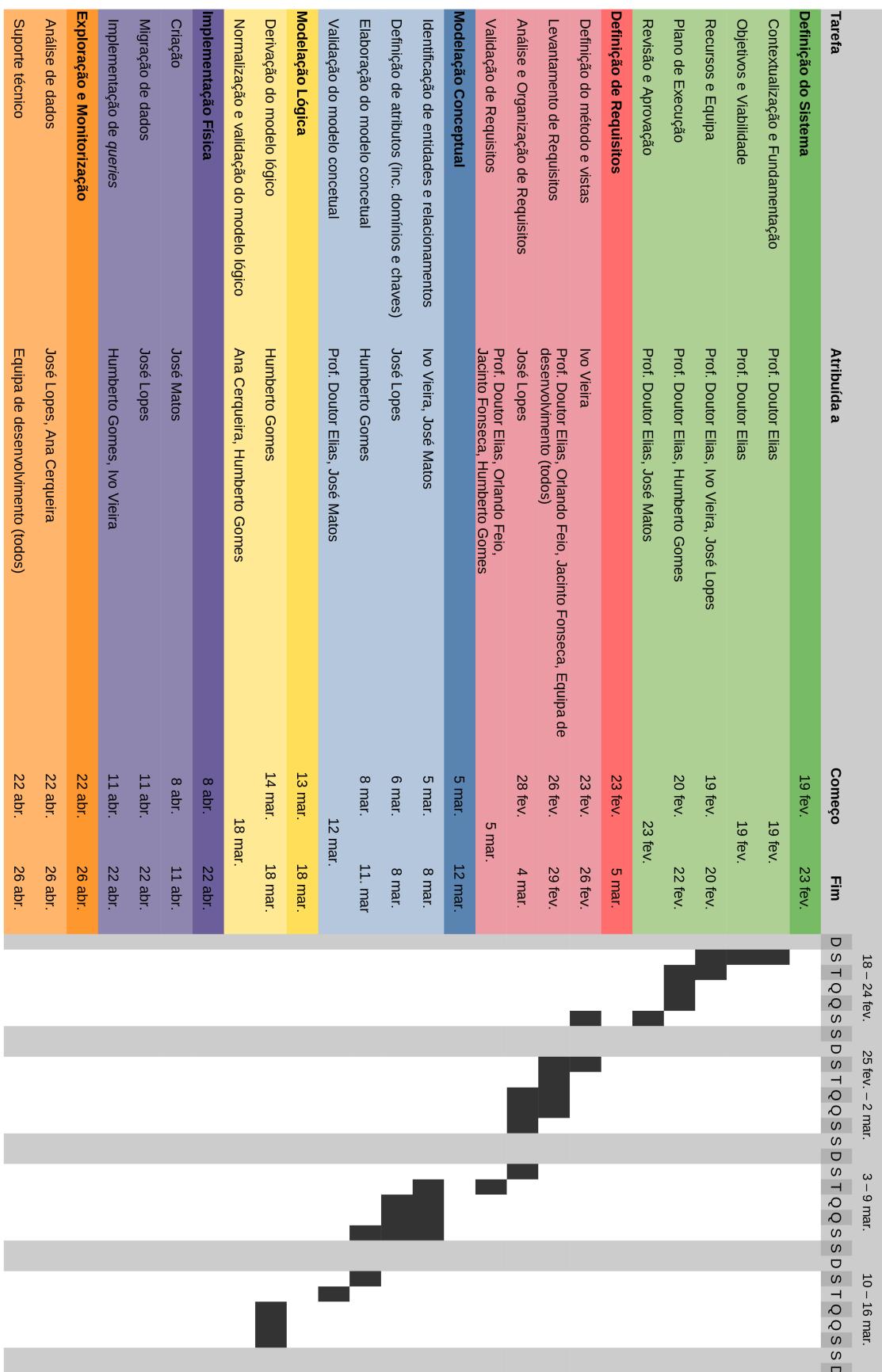
- Cândido, C. (2020). *brModelo*. SIS4.com. <http://www.sis4.com/brModelo/>
- Connolly, T., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed. – Global ed.). Pearson Education Limited.
- dbis-uibk. (2024a). *RelaX – relational algebra calculator*. <https://dbis-uibk.github.io/relax/>
- dbis-uibk. (2024b). *RelaX – Help*. <https://dbis-uibk.github.io/relax/help>
- GraphViz. (2022, October). *DOT Language*. <https://graphviz.org/doc/info/lang.html>
- Kleinsin, J. (2015, June). *Simple Mail Transfer Protocol*. Request For Comments. <doi:10.17487/RFC7504>
- The kernel development community. (2023), *execve(2)*. <https://linux.die.net/man/2/execve>
- Ministério das Finanças. (2013, January 28). *Decreto-Lei n.º 14/2013, de 28 de janeiro*. Diário Da República. <https://diariodarepublica.pt/dr/detalhe/decreto-lei/14-2013-257001>
- Oracle. (2024a, May 4). *MySQL 8.0 Reference Manual: Including MySQL NDB Cluster 8.0*. MySQL. <https://downloads.mysql.com/docs/refman-8.0-en.a4.pdf>
- Oracle. (2024b). *MySQL Workbench*. MySQL. <https://www.mysql.com/products/workbench/>
- Oracle. (2024c). *Sakila Sample Database*. MySQL. <https://dev.mysql.com/doc/sakila/en/>
- Oracle. (2024d). *MySQL Connector/Python Developer Guide*. <https://downloads.mysql.com/docs/connector-python-en.a4.pdf>
- Python Software Foundation (2024a), *Python 3.12.3 documentation*. <https://docs.python.org/3/>
- Python Sofware Foundation (2024b, May 26), *getpass — Portable password input*. <https://docs.python.org/3/library/getpass.html>
- Python Software Foundation (2024c, May 26), *csv — CSV File Reading and Writing*. <https://docs.python.org/3/library/csv.html>
- Shafranovich, Y. (2005). *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. Network Working Group. <https://datatracker.ietf.org/doc/html/rfc4180.html>
- Stankevičius, L., Lukoševičius, M., Kapočiūtė-Dzikiienė, J., Briediene, M. & Krilavičius, T. (2022). Correcting diacritics and typos with ByT5 transformer model. *Applied Sciences*, 2022; 12(5), 2636. <https://doi.org/10.3390/app12052636>

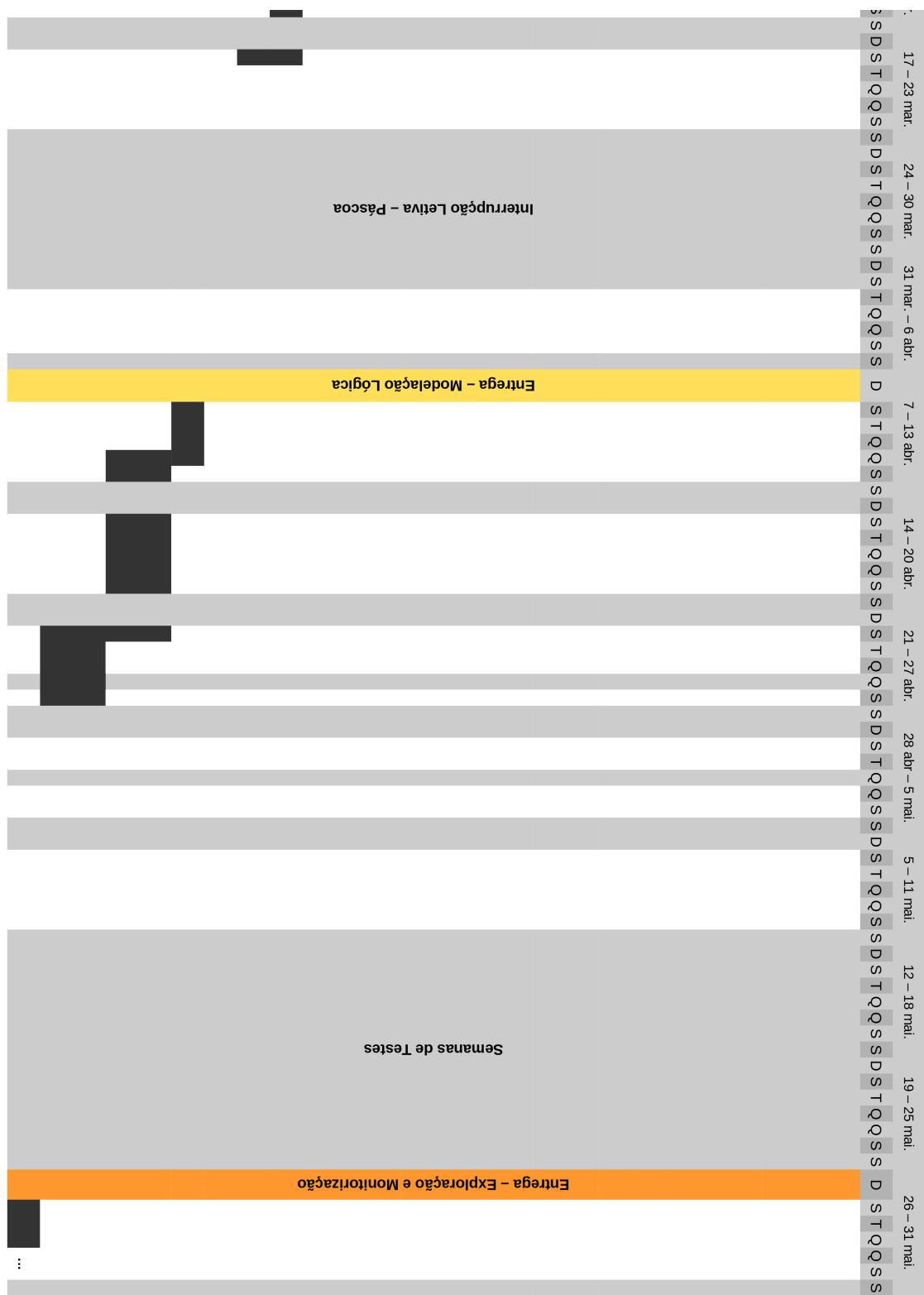
Lista de Siglas e Acrónimos

1FN	Primeira Forma Normal
2FN	Segunda Forma Normal
3FN	Terceira Forma Normal
BD	Base de Dados
CR	<i>Carriage Return</i>
ER	Entidade-Relacionamento
LF	<i>Line Feed</i>
LMD	Linguagem de Manipulação de Dados
RC	Requisito de Controlo
RD	Requisito de Descrição
RH	Recursos Humanos
RM	Requisito de Manipulação
SBD	Sistema de Bases de Dados
SGBD	Sistema de Gestão de Bases de Dados
SQL	<i>Structured Query Language</i>
TSV	<i>Tab-Separated Values</i>

Anexos

I. Diagrama de Gantt – Planeamento de Tarefas





II. Requisitos Levantados

Agência Veritatis

Requisitos

Nº	Data	Hora	Descrição	Vista de Utilização	Fonte	Analista	Substituído
1	26 fev.	10:10	Um cliente é caracterizado pelo seu nome, morada (opcional), NIF, contactos e local de trabalho	Clients	Jacinto Fonseca	José Lopes	
2	26 fev.	10:10	O contacto de um cliente são constituidos por um endereço de e-mail-elettronico-obrigatório e um número de telefone opcional	Clients	Jacinto Fonseca	José Lopes	34
3	26 fev.	10:12	Um professor deve ser identificado por um número sequencial	Clients	Jacinto Fonseca	José Lopes	
4	26 fev.	10:14	Apenas secretários podem registar novos clientes, e alterar os dados dos existentes	Clients / Gestão	Elias Ribeiro	José Lopes	
5	26 fev.	10:18	Listar todos os professores com casos atualmente em aberto (Identificador do cliente e do caso, nome do cliente, designação do caso e data de abertura, ordenados por nome do cliente)	Clients	Jacinto Fonseca	José Lopes	
6	26 fev.	10:22	Um cliente encerra a investigação de um ou mais casos, e um caso pode apenas ter um cliente associado	Casos / Clients	Orlando Feio	José Lopes	
7	26 fev.	10:25	Apenas um administrativo é capaz de registar novos casos	Casos / Gestão	Elias Ribeiro	José Lopes	
8	26 fev.	10:33	Um funcionário é caracterizado pelo seu nome, um número sequencial, NIF, salário (inferior a 10000€), contactos e apólice de seguro de vida (caso seja um detetive)	Gestão	Elias Ribeiro	José Lopes	
9	26 fev.	10:35	Os contactos de um funcionário são constituídos por um endereço de e-mail/obrigatório e um número de telefone opcional	Gestão	Elias Ribeiro	José Lopes	
10	26 fev.	10:38	Devido à pequena dimensão da empresa, um funcionário pode exercer mais do que uma função	Gestão	Jacinto Fonseca	José Lopes	
11	26 fev.	10:40	Uma função é definida por um identificador sequencial e uma designação	Gestão	Jacinto Fonseca	José Lopes	
12	26 fev.	10:45	Enumerar todos os tipos de trabalhador existentes associados a pelo menos um funcionário (identificador e designação)	Gestão	Jacinto Fonseca	José Lopes	
13	26 fev.	10:46	Apenas gestores podem inserir novas funções de funcionários	Gestão	Jacinto Fonseca	José Lopes	
14	26 fev.	10:50	Apenas gestores podem adicionar novos funcionários e modificar ou remover os funcionários existentes	Gestão	Elias Ribeiro	José Lopes	
15	28 fev.	10:10	Um caso é caracterizado por um identificador sequencial, uma designação, custo de abertura, data de inicio, data de término (caso fechado) e data de pagamento (caso pago)	Casos / Clients / Gestão	Orlando Feio	Ana Cerqueira	
16	26 fev.	10:15	Listar todos os casos associados a um cliente com base no seu identificador (identificador, designação e data de encerramento do caso, por ordem de data de encerramento)	Casos / Clients	Jacinto Fonseca	Ana Cerqueira	
17	26 fev.	10:20	Um caso é investigado por apenas um detetive, que permanecerá na investigação até o seu término	Casos / Gestão	Orlando Feio	Ana Cerqueira	
18	26 fev.	10:21	Um detetive pode investigar mais do que um caso em simultâneo	Casos / Gestão	Orlando Feio	Ana Cerqueira	
19	28 fev.	10:30	Uma investigação exige a execução de procedimentos, sendo que após à sua abertura, pode não ter nenhum associado	Casos	Orlando Feio	Ana Cerqueira	

20	28 fev.	10:34	Um procedimento é caracterizado por um identificador sequencial, uma data de execução, um custo para a agência, um custo para o cliente, um campo textual opcional para notas do detetive, e as provas resultantes	Casos / Clientes / Gestão	Orlando Feio	Ana Cerqueira
21	28 fev.	10:35	Um procedimento tem um tipo associado	Casos	Orlando Feio	Ana Cerqueira
22	28 fev.	10:39	O tipo de um procedimento é constituído por um identificador sequencial, uma designação, custos para a agência e cliente e uma descrição detalhada	Casos / Clientes / Gestão	Jacinto Fonseca	Ana Cerqueira
23	28 fev.	10:42	Uma prova é caracterizada por uma descrição detalhada única por caso, e o local onde foi encontrada	Casos	Orlando Feio	Ana Cerqueira
24	28 fev.	10:45	A aplicação de um procedimento pode não resultar em qualquer prova	Casos	Orlando Feio	Ana Cerqueira
25	28 fev.	10:50	Apenas detetives podem inserir procedimentos, sendo os seus custos derivados do seu tipo	Casos / Gestão	Orlando Feio	Ana Cerqueira
26	28 fev.	-	Apenas gestores podem registrar tipos de procedimento	Casos / Gestão	Documentação do caso AlgC	José Matos
27	28 fev.	-	Apenas detetives podem enumerar todas as provas encontradas no decorrer de uma dada investigação (descrição, local e data do procedimento que resultou na prova, ordenados por data)	Casos / Gestão	Documentação do caso AlgC	José Matos
28	28 fev.	-	Apenas detetives podem dar um caso como terminado (atribuir uma data de término a uma investigação)	Casos / Gestão	Documentação do caso AlgC	José Matos
29	28 fev.	-	Apenas gestores podem associar detetives a investigações	Casos / Gestão	Documentação do caso AlgC	José Matos
30	06 mar.	-	Apenas administradores podem calcular o número de casos investigados e de procedimentos executados por cada detetive com pelo menos um caso desde uma data dada (considerar data de término e data de execução, respectivamente)	Gestão	Elias Ribeiro	Ivo Vieira
31	06 mar.	-	Apenas administradores podem gerar uma lista contendo todos os casos, com a receita e despesa resultantes de cada um (identificador, nome) que exercem uma função dentro do seu identificador	Gestão	Elias Ribeiro	Ivo Vieira
32	06 mar.	-	Listar por ordem alfabética todos os funcionários (identificador e nome) que exercem uma função dentro do seu identificador	Gestão	Elias Ribeiro	Ivo Vieira
33	06 mar.	-	Apenas um administrativo pode dar um caso como pago (atribuir-lhe uma data de pagamento)	Casos / Clientes / Gestão	Elias Ribeiro	Ivo Vieira
34	05 mar.	10:05	Os contactos de um cliente podem ser diversos: números de telefone e endereços de e-mail distintos entre clientes. É obrigatório pelo menos um endereço eletrónico.	Clientes	Jacinto Fonseca	Humberto Gomes
35	07 mar.	10:08	Pesquisar um cliente pelo seu nome e listar todos os seus contactos. No caso de várias correspondências de clientes, mostrar todas (com identificador e nome, por ordem alfabética)	Clientes	Jacinto Fonseca	Humberto Gomes
36	07 mar.	10:30	Gerar uma fatura com o preço de abertura de um caso (dado o seu identificador) e todos os procedimentos executados no seu âmbito (designação e preço)	Clientes	Jacinto Fonseca	Humberto Gomes
37	07 mar.	10:40	Enumarar os casos atualmente em aberto (identificador, designação e data de abertura)	Casos	Orlando Feio	Humberto Gomes
38	07 mar.	10:56	Procedimentos, tipos de procedimento e aberturas de casos têm custos, tanto para a agência como para o cliente, inferiores a 1000€	Casos / Clientes / Gestão	Jacinto Fonseca	Humberto Gomes

III. Requisitos Categorizados

Agência Veritatis

Requisitos de Descrição

N.º	Requisitos Originais	Descrição	Vista de Utilização	Revisor
RD1	1	Um cliente é caracterizado pelo seu nome, morada (opcional), NIF, contactos, e local de trabalho	Clientes	Humberto Gomes
RD2	3	Um professor deve ser identificado por um número sequencial	Clientes	Humberto Gomes
RD3	6	Um cliente encomenda a investigação de um ou mais casos, e um caso pode apenas ter um cliente associado	Casos / Clientes	Humberto Gomes
RD4	8	Um funcionário é caracterizado pelo seu nome, um número sequencial, NIF, salário (inferior a 10000€), contactos e apólice de seguro de vida (caso seja um detetive)	Gestão	Humberto Gomes
RD5	9	Os contactos de um funcionário são constituídos por um endereço de e-mail obrigatório e um número de telefone opcional	Gestão	Humberto Gomes
RD6	10	Devido à pequena dimensão da empresa, um funcionário pode exercer mais do que uma função	Gestão	Humberto Gomes
RD7	11	Uma função é definida por um identificador sequencial e uma designação	Gestão	Humberto Gomes
RD8	15	Um caso é caracterizado por um identificador sequencial, uma designação, custo de abertura, data de início, data de término (caso fechado) e data de pagamento (caso pago)	Casos / Clientes / Gestão	Humberto Gomes
RD9	17	Um caso é investigado por apenas um detetive, que permanecerá na investigação até o seu término	Casos / Gestão	Humberto Gomes
RD10	18	Um detetive pode investigar mais do que um caso em simultâneo	Casos / Gestão	Humberto Gomes
RD11	19	Uma investigação exige a execução de procedimentos, sendo que após a sua abertura, pode não ter nenhum associado	Casos	Humberto Gomes
RD12	20	Um procedimento é caracterizado por um identificador sequencial, uma data de execução, um custo para a agência, um custo para o cliente, um campo textual opcional para notas do detetive, e as provas resultantes	Casos / Clientes / Gestão	Humberto Gomes
RD13	21	Um procedimento tem um tipo associado	Casos	Humberto Gomes
RD14	22	O tipo de um procedimento é constituído por um identificador sequencial, uma designação, custos para a agência e cliente, e uma descrição detalhada	Casos / Clientes / Gestão	Humberto Gomes
RD15	23	Uma prova é caracterizada por uma descrição detalhada única por caso, e o local onde foi encontrada	Casos	Humberto Gomes
RD16	24	A aplicação de um procedimento pode não resultar em qualquer prova	Casos	Humberto Gomes
RD17	34	Os contactos de um cliente podem ser diversos: números de telefone e endereços de e-mail, distintos entre clientes. É obrigatório pelo menos um endereço eletrónico.	Clientes	José Lopes
RD18	38	Procedimentos, tipos de procedimento e aberturas de casos têm custos, tanto para a agência como para o cliente, inferiores a 1000€	Casos / Clientes / Gestão	José Lopes

Agência Veritatis

Requisitos de Manipulação

N.º	Requisitos Originais	Descrição	Vista de Utilização	Revisor
RM1	4	Registrar novos clientes e alterar os dados dos existentes	Clientes	Humberto Gomes
RM2	5	Listar todos os professores com casos atualmente em aberto (identificador do cliente e do caso, nome do cliente, designação do caso e data de abertura, ordenados por nome do cliente)	Clientes	Humberto Gomes
RM3	7	Registrar novos casos	Casos	Humberto Gomes
RM4	12	Enumerar todos os tipos de trabalhador existentes associados a pelo menos um funcionário (identificador e designação)	Gestão	Humberto Gomes
RM5	13	Inserir novas funções de funcionários	Gestão	Humberto Gomes
RM6	14	Adicionar novos funcionários e modificar ou remover os funcionários existentes	Gestão	Humberto Gomes
RM7	16	Listar todos os casos associados a um cliente com base no seu identificador (identificador, designação e data de encomenda do caso, por ordem de data de encomenda)	Clientes	Humberto Gomes
RM8	25	Inserir procedimentos, sendo os seus custos derivados do seu tipo	Casos	José Lopes
RM9	26	Registrar tipos de procedimento	Casos	José Lopes
RM10	27	Enumerar todas as provas encontradas no decorrer de uma dada investigação com base no seu identificador (descrição, local e data do procedimento que resultou na prova, ordenados por data)	Casos	José Lopes
RM11	28	Dar um caso como terminado (atribuir uma data de término a uma investigação)	Casos	José Lopes
RM12	29	Associar detetives a investigações	Casos / Gestão	José Lopes
RM13	30	Calcular o número de casos investigados e de procedimentos executados por cada detetive com pelo menos um caso desde uma data dada (considerar data de término e data de execução, respetivamente)	Gestão	José Lopes
RM14	31	Gerar uma lista contendo todos os casos, com a receita e despesa resultantes de cada um (identificador, designação do caso, receita, despesa e balanço)	Gestão	José Lopes
RM15	32	Listar por ordem alfabética o nome de todos os funcionários (identificador e nome) que exercem uma função dada pelo seu identificador	Gestão	José Lopes
RM16	33	Dar um caso como pago (atribuir-lhe uma data de pagamento)	Casos / Clientes	José Lopes
RM17	35	Pesquisar um cliente pelo seu nome e listar todos os seus contactos. No caso de várias correspondências de clientes, mostrar todas (com identificador e nome, por ordem alfabética)	Clientes	José Lopes
RM18	36	Gerar uma fatura com o preço de abertura de um caso (dado o seu identificador) e todos os procedimentos executados no seu âmbito (designação e preço)	Clientes	José Lopes
RM19	37	Enumerar os casos atualmente em aberto (identificador, designação e data de abertura)	Casos	José Lopes

Agência Veritatis

Requisitos de Controlo

N.º	Requisitos Originais	Descrição	Vista de Utilização	Revisor
RC1	4	Apenas secretários podem registar novos clientes e alterar os dados dos existentes	Clientes / Gestão	Humberto Gomes
RC2	7	Apenas um administrativo é capaz de registrar novos casos	Casos / Gestão	Humberto Gomes
RC3	13	Apenas gestores podem inserir novas funções de funcionários	Gestão	Humberto Gomes
RC4	14	Apenas gestores podem adicionar novos funcionários e modificar ou remover os funcionários existentes	Gestão	Humberto Gomes
RC5	25	Apenas detetives podem inserir procedimentos, sendo os seus custos derivados do seu tipo	Casos / Gestão	José Lopes
RC6	26	Apenas gestores podem registar tipos de procedimento	Casos / Gestão	José Lopes
RC7	27	Apenas detetives podem enumerar todas as provas encontradas no decorrer de uma dada investigação (descrição, local e data do procedimento que resultou na prova, ordenados por data)	Casos / Gestão	José Lopes
RC8	28	Apenas detetives podem dar um caso como terminado (atribuir uma data de término a uma investigação)	Casos / Gestão	José Lopes
RC9	29	Apenas gestores podem associar detetives a investigações	Casos / Gestão	José Lopes
RC10	30	Apenas administradores podem calcular o número de casos investigados e de procedimentos executados por cada detetive com pelo menos um caso desde uma data dada (considerar data de término e data de execução, respetivamente)	Gestão	José Lopes
RC11	31	Apenas administradores podem gerar uma lista contendo todos os casos, com a receita e despesa resultantes de cada um (identificador, designação do caso, receita, despesa e balanço)	Gestão	José Lopes
RC12	33	Apenas um administrativo pode dar um caso como pago (atribuir-lhe uma data de pagamento)	Casos / Clientes / Gestão	José Lopes

IV. Relações RelaX

```

group: Veritatis

Cliente = {
    Id:number, Nome:string, Morada:string, NIF:number, LocalTrabalho:string
    1, 'Orlando Manuel Oliveira Belo', null, 128129130, 'Universidade do Minho'
    2, 'João Miguel Lobo Fernandes', null, 155200200, 'Universidade do Minho'
    3, 'Vitor Francisco Mendes Freitas Gomes Fonte', null, 177188199, 'Universidade do
Minho'
    4, 'João Alexandre Baptista Vieira Saraiva', null, 221196677, 'Universidade do Minho'
    5, 'Pedro Miguel Silva Paiva António', 'Ed.7, Sala 1.04, Rua da Universidade, Braga
4710-057', 111222333, 'Universidade do Minho'
    6, 'Paulo Manuel Martins Carvalho', NULL, 314782421, 'Universidade do Minho'
}

Emails = {
    Cliente:number, Email:string
    1, 'obelo@di.uminho.pt'
    2, 'jmf@di.uminho.pt'
    3, 'vff@di.uminho.pt'
    4, 'saraiva@di.uminho.pt'
    5, 'd12943@di.uminho.pt'
    6, 'pmc@di.uminho.pt'
    6, 'd1542@di.uminho.pt'
}

Telefones = {
    Cliente:number, Telefone:string
    1, '253604476'
    1, '91600900'
    3, '253604485'
    6, '253604432'
}

Funcionario = {
    Id:number, Nome:string; NIF:number, Salario:number, SeguroVida:number, Email:string,
    Telefone:string
    1, 'Elias Ribeiro', 132164128, 4000.00, null, 'ddt@veritatis.pt', 936563339
    2, 'Orlando Feio', 128256512, 2500.00, 753638026, 'feio@veritatis.pt', 253000111
    3, 'Jacinto Fonseca', 170169196, 1200.00, null, 'fonseca@veritatis.pt', null
    4, 'Efigénia Leite', 143233777, 2300.00, 752742000, 'leite@veritatis.pt', 253000222
}

Funcao = {
    Id:number, Designacao:string
    1, 'Gestor financeiro'
    2, 'Gestor de recursos humanos'
    3, 'Administrativo'
    4, 'Detetive'
    5, 'Assistente operacional'
}

```

```

Exerce = {
    Funcao:number, Funcionario:number
    1, 1
    2, 1
    3, 3
    4, 2
    4, 4
}

Caso = {
    Id:number, Designacao:string, DataInicio:date, DataTermino:date, DataPagamento:date,
    CustoAbertura:number, Cliente:number, Detetive:number
    1, 'Clube da Luta', 2020-12-05, 2021-05-04, 2021-05-10, 300.00, 1, 2
    2, 'Atrasos Suspeitos', 2021-06-12, 2021-09-24, 2021-10-01, 300.00, 2, 2
    3, 'Ministros, Coitados', 2022-04-01, 2022-07-07, 2022-08-01, 300.00, 3, 4
    4, 'O Silêncio Dos Inocentes', 2023-01-01, 2023-05-16, 2023-05-19, 350.00, 4, 2
    5, 'Vermes na Cantina', 2023-09-09, 2024-02-29, 2024-03-13, 400.00, 5, 4
    6, 'Meias Rotas', 2024-01-01, 2024-03-30, null, 450.00, 1, 2
    7, 'Operação Visconde', 2024-02-12, null, null, 450.00, 6, 4
    8, 'O Barulho Dos Culpados', 2024-03-13, null, null, 450.00, 3, 2
}

TipoProcedimento = {
    Id:number, Designacao:string, CustoAgencia:number, CustoCliente:number,
    Descricao:string
    1, 'Observação de Manchas de Sangue', 0020.00, 0049.99, 'Através da topografia das
    manchas, é possível (...)'
    2, 'Revistar o Local de Trabalho', 0100.00, 0199.99, 'Abordagem ao cliente, de modo a
    (...)'
    3, 'Monitorização de Suspeitos', 0100.00, 0305.00, 'Consiste na observação detalhada
    do dia a dia (...)'
    4, 'Inquérito', 0000.00, 0160.99, 'Com o objetivo de serem esclarecidas possíveis
    (...)'
    5, 'Pizza Party', 0600.20, 1000.00, 'Os agentes também merecem alguma diversão'
}
}

Procedimento = {
    Id:number, Tipo:number, Notas:string, Data:date, CustoAgencia:number,
    CustoCliente:number, Caso:number
    1, 1, 'Salpicos, projeção do sangue feita sob impulso', 2020-12-10, 0015.00, 0039.99,
    1
    2, 3, 'Suspeitos ocupados a tentar passar às outras cadeiras', 2020-12-19, 0085.00,
    170.00, 1
    3, 2, 'O escritório da vítima contém testes todos com classificações negativas', 2021-
    06-13, 0100.00, 0199.99, 2
    4, 2, 'Dez dias depois, todos os testes tinham classificação de 20', 2021-06-23,
    0100.00, 0199.99, 2
    5, 4, 'Parece que o trabalho de LI3 é fazer uma base de dados para o ministério',
    2022-04-10, 0000.00, 0160.99, 3
}

```

```
6, 2, 'Demasiado barulho...', 2023-01-14, 0100.00, 0199.99, 4
7, 2, 'É impossível isto ser regulado pela ASAЕ...', 2023-09-09, 0100.00, 0199.99, 5
8, 5, 'Isto foi preciso depois de ver a comida', 2023-09-10, 0600.20, 1000.00, 5
9, 2, 'Demasiadas meias, os buracos formam um padrão...', 2024-02-10, 0100.00,
0199.99, 6
10, 3, 'O suspeito é regular de um café na rua vermelha', 2024-02-15, 0100.00,
0305.00, 6
}
```

```
Provas = {
    Procedimento:number, Descricao:string, Local:string
    3, 'Teste do suspeito principal (2.3 valores)', 'Gabinete 3.22, Ed. 7, Universidade do
    Minho'
    3, 'Pauta final da UC', 'Gabinete 3.22, Ed. 7, Universidade do Minho'
    6, 'Nota escrita por uma testemunha inocente (os inocentes quebraram o silêncio)',
    'Edifício de Psicologia, Universidade do Minho, Gualtar'
    7, 'Lasanha envenenada com sementes de Ricinus communis L.', 'Grill, Universidade do
    Minho, Gualtar'
    9, 'Faca usada para romper as meias cravada na secretária', 'Gabinete 3.08, Ed. 7,
    Universidade do Minho'
    10, 'Bilhete rasgado depositado no lixo pelo suspeito', 'Shady Coffee, Rua Vermelha,
    Gualtar'
}
```

V. Expressões de Álgebra Relacional

RM2

$$\tau_{Cliente}.Nome \text{ asc}(\pi_{Cliente}.Id, Cliente.Nome, Caso.Id, Caso.Designacao, Caso.DataInicio(\sigma_{DataTermino=null}(Caso) \bowtie Caso.Cliente=Cliente.Id Cliente))$$
RM4

$$\pi_{Funcao}.Id, Funcao.Designacao((\pi_{Funcao}(Exerce)) \bowtie Funcao=Id Funcao)$$
RM7

$$\tau_{DataInicio} \text{ asc}(\pi_{Id}.Designacao, DataInicio(\sigma_{Cliente=?}(Caso)))$$

RM10 (a variável P não é utilizada, sendo substituída diretamente na expressão)

$$\tau_{Procedimento}.Data \text{ asc}(\pi_{Provas}.Descricao, Provas.Local, Procedimento.Data(Provas \bowtie Provas.Procedimento=Procedimento.Id (\sigma_{Procedimento.Caso=?}(Procedimento))))$$
RM13

$$\begin{aligned} C &= \gamma_{Detetive}; \text{count}(* \rightarrow Casos}(\sigma_{DataInicio > DATE(?)}(Caso)) \\ P &= \gamma_{Detetive}; \text{count}(* \rightarrow Procedimentos}(\sigma_{Procedimento.Data > DATE(?)}(Procedimento) \bowtie Procedimento.Caso=Caso.Id Caso) \\ \pi_{Caso.Detetive, Casos, COALESCE(Procedimentos, 0)} &\rightarrow Procedimentos(C \bowtie P) \end{aligned}$$
RM14

$$\begin{aligned} S &= \gamma_{Caso}; \text{sum}(CustoAgencia) \rightarrow Despesa, \text{sum}(CustoCliente) \rightarrow ReceitaProc (Procedimento) \\ \pi_{Caso.Id, Caso.Designacao, COALESCE(Despesa, 0)} &\rightarrow Despesa, Caso.CustoAbertura + COALESCE(ReceitaProc, 0) \rightarrow Receita, COALESCE(ReceitaProc, 0) + Caso.CustoAbertura - COALESCE(Despesa, 0) \rightarrow Balanco (S \bowtie Procedimento.Caso=Caso.Id Caso) \end{aligned}$$
RM15

$$\tau_{Funcionario.Nome} \text{ asc}(\pi_{Funcionario.Id}, Funcionario.Nome(\sigma_{Funcao=?}(Exerce) \bowtie Exerce.Funcionario=Funcionario.Id Funcionario))$$
RM17

$$\begin{aligned} E &= \pi_{Cliente.Id, Cliente.Nome, Emails.Email, 'Email'} \rightarrow Tipo (Emails \bowtie Emails.Cliente=Cliente.Id (\sigma_{Nome LIKE '%?%'(Cliente)})) \\ T &= \pi_{Cliente.Id, Cliente.Nome, Telefones.Telefone, 'Telefone'} \rightarrow Tipo (Telefones \bowtie Telefones.Cliente=Cliente.Id (\sigma_{Nome LIKE '%?%'(Cliente)})) \\ \rho_{Contacto} &\leftarrow Emails.Email(\tau_{Cliente.Nome} \text{ asc}(E \cup T)) \end{aligned}$$
RM18

$$\begin{aligned} Ca &= \pi_{CustoAbertura}(\sigma_{Id=6}(Caso)) \\ F &= \pi_{TipoProcedimento.Designacao, Procedimento.CustoCliente}(\sigma_{Caso=6}(Procedimento \bowtie Procedimento.Tipo=TipoProcedimento.Id TipoProcedimento)) \\ Cp &= \pi_{COALESCE(TotalProcedimentos, 0)} \rightarrow TotalProcedimentos(\gamma; \text{sum}(Procedimento.CustoCliente) \rightarrow TotalProcedimentos(F)) \\ T &= \pi_{Total} \leftarrow TotalProcedimentos + CustoAbertura (Cp \bowtie Ca) \end{aligned}$$
RM19

$$\pi_{Id, Designacao, DataInicio(\sigma_{DataTermino=null}(Caso))}$$

VI. Modelo Físico

```

-- (C) Agência Veritatis 2024
-- Autores: Ana Cerqueira e José Lopes

-- DROP DATABASE IF EXISTS Veritatis;
CREATE DATABASE Veritatis
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;
USE Veritatis;

CREATE TABLE Funcao(
    Id INT NOT NULL AUTO_INCREMENT,
    Designacao VARCHAR(75) NOT NULL,
    PRIMARY KEY (Id)
);

CREATE TABLE Funcionario(
    Id INT NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(75) NOT NULL,
    NIF INT NOT NULL UNIQUE CHECK(LENGTH(NIF) = 9),
    Salario DECIMAL(6,2) NOT NULL CHECK(Salario >= 0),
    SeguroVida INT NULL CHECK(LENGTH(SeguroVida) = 9),
    Email VARCHAR(255) UNIQUE NOT NULL,
    Telefone VARCHAR(20) UNIQUE NULL,
    PRIMARY KEY(Id)
);

CREATE TABLE Exerce(
    Funcao INT NOT NULL,
    Funcionario INT NOT NULL,
    PRIMARY KEY(Funcao, Funcionario),
    FOREIGN KEY(Funcao) REFERENCES Funcao(Id),
    FOREIGN KEY(Funcionario) REFERENCES Funcionario(Id)
);

CREATE TABLE Cliente(
    Id INT NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(75) NOT NULL,
    Morada VARCHAR(200) NULL,
    NIF INT NOT NULL UNIQUE CHECK(LENGTH(NIF) = 9),
    LocalTrabalho VARCHAR(75) NOT NULL,

```

```

        PRIMARY KEY(Id)
);

CREATE TABLE Emails(
    Cliente INT NOT NULL,
    Email VARCHAR(255) NOT NULL,

    PRIMARY KEY (Email),
    FOREIGN KEY (Cliente) REFERENCES Cliente (Id)
);

CREATE TABLE Telefones(
    Cliente INT NOT NULL,
    Telefone VARCHAR(20) NOT NULL,

    PRIMARY KEY (Telefone),
    FOREIGN KEY (Cliente) REFERENCES Cliente (Id)
);

CREATE TABLE Caso(
    Id INT NOT NULL AUTO_INCREMENT,
    Designacao VARCHAR(75) NOT NULL,
    DataInicio DATE NOT NULL,
    DataTermino DATE NULL,
    DataPagamento DATE NULL,
    CustoAbertura DECIMAL(5,2) NOT NULL CHECK (CustoAbertura >= 0),
    Cliente INT NOT NULL,
    Detetive INT NULL,

    PRIMARY KEY (Id),
    FOREIGN KEY (Cliente) REFERENCES Cliente (Id),
    FOREIGN KEY (Detetive) REFERENCES Funcionario (Id),
    CONSTRAINT chk_datas CHECK ((DataInicio <= DataTermino)
        AND (DataTermino <= DataPagamento))
);

CREATE TABLE TipoProcedimento(
    Id INT NOT NULL AUTO_INCREMENT,
    Designacao VARCHAR(75) NOT NULL,
    CustoAgencia DECIMAL(5,2) NOT NULL CHECK (CustoAgencia >= 0),
    CustoCliente DECIMAL(5,2) NOT NULL CHECK (CustoCliente >= 0),

    Descricao TEXT NOT NULL,
    PRIMARY KEY (Id)
)

```

```

);

CREATE TABLE Procedimento(
    Id INT NOT NULL AUTO_INCREMENT,
    Tipo INT NOT NULL,
    Notas TEXT NULL,
    Data DATE NOT NULL,
    CustoAgencia DECIMAL (5,2) NOT NULL DEFAULT 0 CHECK (CustoAgencia >= 0),
    CustoCliente DECIMAL (5,2) NOT NULL DEFAULT 0 CHECK (CustoCliente >= 0),
    Caso INT NOT NULL,
    PRIMARY KEY (Id),
    FOREIGN KEY (Tipo) REFERENCES TipoProcedimento (Id),
    FOREIGN KEY (Caso) REFERENCES Caso (Id)
);

CREATE TABLE Provas(
    Procedimento INT NOT NULL,
    Descricao TEXT NOT NULL,
    Local VARCHAR(200) NOT NULL,
    PRIMARY KEY (Procedimento, Descricao(767)),
    FOREIGN KEY (Procedimento) REFERENCES Procedimento(Id)
);

```

VII. *Script* de criação de utilizadores

```

-- (C) Agência Veritatis 2024
-- Autores: Humberto Gomes

-- CRIAR ROLES
DROP ROLE IF EXISTS
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
CREATE ROLE
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';

-- RC1 / RM1
GRANT INSERT, UPDATE, SELECT ON Cliente TO
    'administrativo'@'localhost';
GRANT INSERT, UPDATE, DELETE, SELECT ON Emails TO
    'administrativo'@'localhost';
GRANT INSERT, UPDATE, DELETE, SELECT ON Telefones TO
    'administrativo'@'localhost';

-- RM2
GRANT SELECT(Id, Nome) ON Cliente TO
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
GRANT SELECT(Id, Designacao, DataInicio, DataTermino, Cliente) ON Caso TO
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';

-- RM3 / RC2
GRANT
    INSERT(Id, Designacao, DataInicio, CustoAbertura, Cliente),
    SELECT(Id, Designacao, DataInicio, CustoAbertura, Cliente) ON Caso
    TO 'administrativo'@'localhost';

-- RM4
GRANT SELECT(Id, Designacao) ON Funcao TO
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
GRANT SELECT(Funcao) ON Exerce TO
    'administrativo'@'localhost',

```

```

'detective'@'localhost',
'administrador'@'localhost';

-- RM5 / RC3
GRANT INSERT, SELECT ON Funcao T0
'administrador'@'localhost';

-- RM6 / RC4
GRANT INSERT, UPDATE, DELETE, SELECT ON Funcionario T0
'administrador'@'localhost';
GRANT INSERT, DELETE, SELECT ON Exerce T0
'administrador'@'localhost';

-- RM7
GRANT SELECT(Id, Designacao, DataInicio, Cliente) ON Caso T0
'administrativo'@'localhost',
'detective'@'localhost',
'administrador'@'localhost';

-- RM8 / RC5
GRANT INSERT, SELECT ON Procedimento T0
'detective'@'localhost';

-- RM9 / RC6
GRANT INSERT, SELECT ON TipoProcedimento T0
'administrador'@'localhost';

-- RM10 / RC7
GRANT SELECT(Id, Data, Caso) ON Procedimento T0
'detective'@'localhost';
GRANT SELECT(Procedimento, Descricao, Local) ON Provas T0
'detective'@'localhost';

-- RM11 / RC8
GRANT UPDATE(DataTermino), SELECT(Id, Designacao, Cliente) ON Caso T0
'detective'@'localhost';

-- RM12 / RC9
GRANT UPDATE(Detective), SELECT(Id, Designacao, Cliente) ON Caso T0
'administrador'@'localhost';

-- RM13 / RC10
GRANT SELECT(Id, DataInicio, Detective) ON Caso T0
'administrador'@'localhost';

```

```

GRANT SELECT(Id, Caso, Data) ON Procedimento T0
    'administrador'@'localhost';

-- RM14 / RC11
GRANT SELECT(Id, Designacao, CustoAbertura) ON Caso T0
    'administrador'@'localhost';
GRANT SELECT(Caso, CustoCliente, CustoAgencia) ON Procedimento T0
    'administrador'@'localhost';

-- RM15
GRANT SELECT(Id, Nome) ON Funcionario T0
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
GRANT SELECT(Funcionario, Funcao) ON Exerce T0
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';

-- RM16 / RC12
GRANT UPDATE(DataPagamento), SELECT(Id, Designacao, Cliente) ON Caso T0
    'administrativo'@'localhost';

-- RM17
GRANT SELECT(Id, Nome) ON Cliente T0
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
GRANT SELECT(Cliente, Email) ON Emails T0
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
GRANT SELECT(Cliente, Telefone) ON Telefones T0
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';

-- RM18
GRANT SELECT(Id, CustoAbertura) ON Caso T0
    'administrativo'@'localhost',
    'detetive'@'localhost',
    'administrador'@'localhost';
GRANT SELECT(Tipo, CustoCliente, Caso) ON Procedimento T0
    'administrativo'@'localhost',

```

```

'detetive'@'localhost',
'administrador'@'localhost';
GRANT SELECT(Id, Designacao) ON TipoProcedimento TO
'administrativo'@'localhost',
'detetive'@'localhost',
'administrador'@'localhost';

-- RM19
GRANT SELECT(Id, Designacao, DataInicio, DataTermino) ON Caso TO
'administrativo'@'localhost',
'detetive'@'localhost',
'administrador'@'localhost';

-- EXTRAS
GRANT UPDATE(CustoAgencia, CustoCliente) ON TipoProcedimento TO
'administrador'@'localhost';
GRANT INSERT On Provas TO
'detetive'@'localhost';

-- FIX MySQL BUG IN RM3
REVOKE INSERT ON Caso FROM
'administrador'@'localhost',
'detetive'@'localhost';

-- CRIAR UTILIZADORES
DROP USER IF EXISTS
'elias.ribeiro'@'localhost',
'orlando.feio'@'localhost',
'jacinto.fonseca'@'localhost';
CREATE USER
'elias.ribeiro'@'localhost' IDENTIFIED BY 'donodistotudo',
'orlando.feio'@'localhost' IDENTIFIED BY 'alterego',
'jacinto.fonseca'@'localhost' IDENTIFIED BY 'fonmolhada';

GRANT 'administrador'@'localhost' TO 'elias.ribeiro'@'localhost';
SET DEFAULT ROLE 'administrador'@'localhost' TO 'elias.ribeiro'@'localhost';

GRANT 'detetive'@'localhost' TO 'orlando.feio'@'localhost';
SET DEFAULT ROLE 'detetive'@'localhost' TO 'orlando.feio'@'localhost';

GRANT 'administrativo'@'localhost' TO 'jacinto.fonseca'@'localhost';
SET      DEFAULT      ROLE      'administrativo'@'localhost'      TO
'jacinto.fonseca'@'localhost';

```

VIII. Povoamento da BD usando a LMD de SQL

```
-- (C) Agência Veritatis 2024
```

```
-- Autores: José Matos
```

```
INSERT INTO Funcao(Id, Designacao) VALUES
```

```
(1, 'Gestor financeiro'),  
(2, 'Gestor de recursos humanos'),  
(3, 'Administrativo'),  
(4, 'Detetive'),  
(5, 'Assistente operacional');
```

```
INSERT INTO Funcionario(Id, Nome, NIF, Salario, SeguroVida, Email, Telefone)
```

```
VALUES
```

```
(1, 'Elias Ribeiro', 132164128, 4000.00, NULL, 'ddt@veritatis.pt',  
936563339),  
(2, 'Orlando Feio', 128256512, 2500.00, 753638026, 'feio@veritatis.pt',  
253000111),  
(3, 'Jacinto Fonseca', 170169196, 1200.00, NULL, 'fonseca@veritatis.pt',  
null),  
(4, 'Efigénia Leite', 143233777, 2300.00, 752742000, 'leite@veritatis.pt',  
253000222);
```

```
INSERT INTO Cliente(Id, Nome, Morada, NIF, LocalTrabalho) VALUES
```

```
(1, 'Orlando Manuel Oliveira Belo', NULL, 128129130, 'Universidade do  
Minho'),  
(2, 'João Miguel Lobo Fernandes', NULL, 155200200, 'Universidade do  
Minho'),  
(3, 'Vitor Francisco Mendes Freitas Gomes Fonte', NULL, 177188199,  
'Universidade do Minho'),  
(4, 'João Alexandre Baptista Vieira Saraiva', NULL, 221196677,  
'Universidade do Minho'),  
(5, 'Pedro Miguel Silva Paiva António', 'Ed.7, Sala 1.04, Rua da  
Universidade, Braga 4710-057', 111222333, 'Universidade do Minho'),  
(6, 'Paulo Manuel Martins Carvalho', NULL, 314782421, 'Universidade do  
Minho');
```

```
INSERT INTO Exerce(Funcao, Funcionario) VALUES
```

```
(1, 1),  
(2, 1),  
(3, 3),  
(4, 2),  
(4, 4);
```

```
INSERT INTO Caso(Id, Designacao, DataInicio, DataTermino, DataPagamento,
```

```
CustoAbertura, Cliente, Detetive) VALUES
```

```

(1, 'Clube da Luta', '2020-12-05', '2021-05-04', '2021-05-10', 300.00, 1,
2),
(2, 'Atrasos Suspeitos', '2021-06-12', '2021-09-24', '2021-10-01', 300.00,
2, 2),
(3, 'Ministros, Coitados', '2022-04-01', '2022-07-07', '2022-08-01',
300.00, 3, 4),
(4, 'O Silêncio Dos Inocentes', '2023-01-01', '2023-05-16', '2023-05-19',
350.00, 4, 2),
(5, 'Vermes na Cantina', '2023-09-09', '2024-02-29', '2024-03-13', 400.00,
5, 4),
(6, 'Meias Rotas', '2024-01-01', '2024-03-30', NULL, 450.00, 1, 2),
(7, 'Operação Visconde', '2024-02-12', NULL, NULL, 450.0, 6, 4),
(8, 'O Barulho Dos Culpados', '2024-03-13', NULL, NULL, 450.00, 3, 2);

```

```
INSERT INTO TipoProcedimento (Id, Designacao, CustoAgencia, CustoCliente,
Descricao) VALUES
```

```

(1, 'Observação de Manchas de Sangue', 0020.00, 0049.99, 'Através da
topografia das manchas, é possível (...)'),
(2, 'Revistar o Local de Trabalho', 0100.00, 0199.99, 'Abordagem ao
cliente, de modo a (...)'),
(3, 'Monitorização de Suspeitos', 0100.00, 0305.00, 'Consiste na
observação detalhada do dia a dia (...)'),
(4, 'Inquérito', 0000.00, 0160.99, 'Com o objetivo de serem esclarecidas
possíveis (...)'),
(5, 'Pizza Party', 0600.20, 0999.00, 'Os agentes também merecem alguma
diversão');

```

```
INSERT INTO Emails (Cliente, Email) VALUES
```

```

(1, 'obelo@di.uminho.pt'),
(2, 'jmf@di.uminho.pt'),
(3, 'vff@di.uminho.pt'),
(4, 'saraiva@di.uminho.pt'),
(5, 'd12943@di.uminho.pt'),
(6, 'pmc@di.uminho.pt'),
(6, 'd1542@di.uminho.pt');

```

```
INSERT INTO Telefones (Cliente, Telefone) VALUES
```

```

(1, '253604476'),
(1, '91600900'),
(3, '253604485'),
(6, '253604432');

```

```
INSERT INTO Procedimento (Id, Tipo, Notas, Data, CustoAgencia, CustoCliente,
Caso) VALUES
```

(1, 1, 'Salpicos, projeção do sangue feita sob impulso', '2020-12-10', 0015.00, 0039.99, 1),
(2, 3, 'Suspeitos ocupados a tentar passar às outras cadeiras', '2020-12-19', 0085.00, 170.00, 1),
(3, 2, 'O escritório da vítima contém testes todos com classificações negativas', '2021-06-13', 0100.00, 0199.99, 2),
(4, 2, 'Dez dias depois, todos os testes tinham classificação de 20', '2021-06-23', 0100.00, 0199.99, 2),
(5, 4, 'Parece que o trabalho de LI3 é fazer uma base de dados para o ministério', '2022-04-10', 0000.00, 0160.99, 3),
(6, 2, 'Demasiado barulho...', '2023-01-14', 0100.00, 0199.99, 4),
(7, 2, 'É impossível isto ser regulado pela ASAE...', '2023-09-09', 0100.00, 0199.99, 5),
(8, 5, 'Isto foi preciso depois de ver a comida', '2023-09-10', 0600.20, 0999.00, 5),
(9, 2, 'Demasiadas meias, os buracos formam um padrão...', '2024-02-10', 0100.00, 0199.99, 6),
(10, 3, 'O suspeito é regular de um café na rua vermelha', '2024-02-15', 0100.00, 0305.00, 6);

INSERT INTO Provas (Procedimento, Descricao, Local) VALUES
(3, 'Teste do suspeito principal (2.3 valores)', 'Gabinete 3.22, Ed. 7, Universidade do Minho'),
(3, 'Pauta final da UC', 'Gabinete 3.22, Ed. 7, Universidade do Minho'),
(6, 'Nota escrita por uma testemunha inocente (os inocentes quebraram o silêncio)', 'Edifício de Psicologia, Universidade do Minho, Gualtar'),
(7, 'Lasanha envenenada com sementes de Ricinus communis L.', 'Grill, Universidade do Minho, Gualtar'),
(9, 'Faca usada para romper as meias cravada na secretária', 'Gabinete 3.08, Ed. 7, Universidade do Minho'),
(10, 'Bilhete rasgado depositado no lixo pelo suspeito', 'Shady Coffee, Rua Vermelha, Gualtar');

IX. Povoamento da BD usando um programa externo

```

#!/usr/bin/env python3

# (C) Agência Veritatis 2024
# Autor: Humberto Gomes
#
# Instale mysql-connector-python antes de executar:
#   pip3 install mysql-connector-python

import csv
from getpass import getpass
import mysql.connector
from os.path import join
import sys

def main(argv: list[str]) -> int:
    if len(argv) != 2:
        print('Utilização: ./povoamento [diretoria]', file=sys.stderr)
        return 1

    try:
        with open(join(argv[1], 'Ordem.txt')) as ficheiro:
            tabelas = [ linha for linha in ficheiro.read().split('\n') if
linha != '' ]
    except OSError as e:
        print(e, file=sys.stderr)
        return 1

    password = getpass(prompt='Password de administrador (root@localhost) do
MySQL: ')
    try:
        contexto = mysql.connector.connect(user='root',
                                            password=password,
                                            host='127.0.0.1',
                                            database='Veritatis')

        cursor = contexto.cursor()
    except mysql.connector.Error as e:
        print(e, file=sys.stderr)
        return 1

    for tabela in tabelas:
        caminho = join(argv[1], tabela + '.tsv')
        try:
            ficheiro = open(caminho)

```

```

        leitor = csv.reader(ficheiro, delimiter='\t', quotechar='',
strict=True)
        for i, tuplo in enumerate(leitor):
            if i == 0:
                atributos = ', '.join(tuplo)
                substituições = ', '.join(['%s'] * len(tuplo))
                instrução = f'INSERT INTO {tabela}({atributos}) VALUES
({substituições})'
            else:
                print(tabela + str(tuple(tuplo)))
                processado = tuple((t if t != 'NULL' else None) for t in
tuplo)
                cursor.execute(instrução, processado)

        ficheiro.close()
    except (OSError, csv.Error, mysql.connector.Error) as e:
        print(e, file=sys.stderr)
        contexto.rollback()
        cursor.close()
        contexto.close()
        if ficheiro is not None:
            ficheiro.close()
        return 1

    contexto.commit()
    cursor.close()
    contexto.close()
    return 0

if __name__ == '__main__':
    sys.exit(main(sys.argv))

```

X. Medições de espaço resultantes de inserções de registos

Relação	Registros (octetos)	Dados (octetos)	Índices (octetos)	Total (octetos)	Por registo (octetos)	Média (octetos)
Funcao	100000	3686400	0	3686400	36,86	41,75
Funcao	200000	8929280	0	8929280	44,65	41,75
Funcao	300000	13123584	0	13123584	43,75	41,75
Funcionario	100000	9977856	7913472	17891328	178,91	175,33
Funcionario	200000	18366464	13189120	31555584	157,78	175,33
Funcionario	300000	33095680	23691264	56786944	189,29	175,33
Exerce	100000	2637824	1589248	4227072	42,27	43,31
Exerce	200000	5783552	2637824	8421376	42,11	43,31
Exerce	300000	8929280	4734976	13664256	45,55	43,31
Cliente	100000	12075008	1589248	13664256	136,64	140,7
Cliente	200000	23642112	2637824	26279936	131,4	140,7
Cliente	300000	41484288	4734976	46219264	154,06	140,7
Emails	100000	5783552	4734976	10518528	105,19	89,49
Emails	200000	7913472	5783552	13697024	68,49	89,49
Emails	300000	16318464	12124160	28442624	94,81	89,49
Telefones	100000	3686400	1589248	5275648	52,76	60,78
Telefones	200000	7880704	4734976	12615680	63,08	60,78
Telefones	300000	9977856	9977856	19955712	66,52	60,78
Caso	100000	6832128	3178496	10010624	100,11	101,99
Caso	200000	14172160	6324224	20496384	102,48	101,99
Caso	300000	21544960	9469952	31014912	103,38	101,99
TipoProcedimento	10000	23642112	0	23642112	2364,21	2307,78
TipoProcedimento	20000	44630016	0	44630016	2231,5	2307,78
TipoProcedimento	30000	69828608	0	69828608	2327,62	2307,78
Procedimento	10000	23642112	327680	23969792	2396,98	2332,63
Procedimento	20000	47792128	622592	48414720	2420,74	2332,63
Procedimento	30000	64585728	819200	65404928	2180,16	2332,63
Provas	10000	35651584	0	35651584	3565,16	3454,48
Provas	20000	70254592	0	70254592	3512,73	3454,48

Tabela 36 - Medições experimentais dos tamanhos médios de cada registo, tendo em conta os tamanhos das relações medidos.

XI. Índices Criados

```
-- (C) Agência Veritatis 2024
-- Autores: Ivo Vieira

-- RM2 e RM7
CREATE INDEX idxCasoCliente ON Caso(Cliente);

-- RM4 e RM15
CREATE INDEX idxExerceFuncao ON Exerce(Funcao);

-- RM7
CREATE INDEX idxCasoDataInicio ON Caso(DataInicio);

-- RM15
CREATE INDEX idxFuncionarioNome ON Funcionario(Nome);
CREATE INDEX idxExerceFuncionario ON Exerce(Funcionario);

-- RM10
CREATE INDEX idxProcedimentoData ON Procedimento(Data);
CREATE INDEX idxProcedimentoCaso ON Procedimento(Caso);
CREATE INDEX idxProvasProcedimento ON Provas(Procedimento);

-- RM17
CREATE INDEX idxEmailsCliente ON Emails(Cliente);
CREATE INDEX idxTelefonesCliente ON Telefones(Cliente);
```

XII. Procedimentos, Funções e Gatilhos

```

-- (C) Agência Veritatis 2024
-- Autores: José Lopes, José Matos e Humberto Gomes

-- EncheBD

-- Ter a BD sempre povoada antes de correr este procedimento
-- SOURCE Criação.sql;

DROP PROCEDURE IF EXISTS EncheBD;

DELIMITER $$

CREATE PROCEDURE EncheBD(IN nrRegistros INT)
BEGIN
    DECLARE dt DATE DEFAULT CURDATE();

    START TRANSACTION;

    REPEAT
        INSERT INTO Funcao(Id, Designacao) VALUES
        (nrRegistros, REPEAT('A', 20));
        INSERT INTO Funcionario(Id, Nome, NIF, Salario, SeguroVida, Email,
        Telefone) VALUES
        (nrRegistros, REPEAT('A', 30), 100000000 + nrRegistros, 0000.00,
        1111111111, LPAD(nrRegistros, 30, '0'), LPAD(nrRegistros, 9, '0'));
        INSERT INTO Exerce(Funcao, Funcionario) VALUES
        (nrRegistros, nrRegistros);
        INSERT INTO Cliente(Id, Nome, Morada, NIF, LocalTrabalho) VALUES
        (nrRegistros, REPEAT('A', 30), REPEAT('A', 35), 100000000 +
        nrRegistros, REPEAT('A', 35));
        INSERT INTO Emails(Cliente, Email) VALUES
        (nrRegistros, LPAD(nrRegistros, 30, '0'));
        INSERT INTO Telefones(Cliente, Telefone) VALUES
        (nrRegistros, LPAD(nrRegistros, 9, '0'));
        INSERT INTO Caso(Id, Designacao, DataInicio, DataTermino,
        DataPagamento, CustoAbertura, Cliente, Detetive) VALUES
        (nrRegistros, REPEAT('A', 20), dt, dt, dt, 000.00, nrRegistros,
        nrRegistros);

        IF nrRegistros % 10 = 0 THEN
            INSERT INTO TipoProcedimento(Id, Designacao, CustoAgencia,
            CustoCliente, Descricao) VALUES
            (nrRegistros, REPEAT('A', 20), 000.00, 000.00, REPEAT('A',
            2000));
    END REPEAT;
END

```

```

        INSERT INTO Procedimento(Id, Tipo, Notas, Data, CustoAgencia,
CustoCliente, Caso) VALUES
        (nrRegistros, nrRegistros, REPEAT('A', 2000), dt, 000.00,
000.00, nrRegistros);
        INSERT INTO Provas(Procedimento, Descricao, Local) VALUES
        (nrRegistros, REPEAT('A', 2000), REPEAT('A', 35));
    END IF;

    SET nrRegistros = nrRegistros - 1;
    UNTIL nrRegistros = 0
    END REPEAT;

    COMMIT;
END $$

DELIMITER ;

CALL EncheBD(1000);

-- RM18

DROP FUNCTION IF EXISTS RM18;

DELIMITER $$

CREATE FUNCTION RM18(IdCaso INT)
RETURNS TEXT
NOT DETERMINISTIC
SQL SECURITY INVOKER
READS SQL DATA
BEGIN
    DECLARE Abertura DECIMAL(8, 2);
    DECLARE Total DECIMAL(8, 2);
    DECLARE Fatura TEXT;

    DECLARE Fim INTEGER DEFAULT 0;
    DECLARE Item VARCHAR(100);
    DECLARE Itens CURSOR FOR
        SELECT CONCAT(TipoProcedimento.Designacao, ' : ',
Procedimento.CustoCliente, '€')
        FROM Procedimento INNER JOIN TipoProcedimento ON Procedimento.Tipo
= TipoProcedimento.Id
        WHERE Procedimento.Caso = IdCaso;

    DECLARE CONTINUE HANDLER
        FOR NOT FOUND SET Fim = 1;

```

```

-- Inicialização
SET Fatura = '';
SELECT Caso.CustoAbertura INTO Abertura
  FROM Caso
 WHERE Caso.Id = IdCaso;
SELECT COALESCE(SUM(Procedimento.CustoCliente), 0) + Abertura INTO Total
  FROM Procedimento
 WHERE Procedimento.Caso = IdCaso;

-- Verificar se o caso existe
IF Abertura IS NULL THEN
    SET Fatura = CONCAT('Caso ', IdCaso, ' não encontrado!');
    RETURN Fatura;
END IF;

-- Mostrar itens da fatura
OPEN Itens;
processaItem: LOOP
    FETCH Itens INTO Item;
    IF Fim = 1 THEN
        LEAVE processaItem;
    END IF;

    SET Fatura = CONCAT(Fatura, Item, '\n');
END LOOP processaItem;
CLOSE Itens;

IF Fatura = '' THEN
    SET Fatura = 'Sem procedimentos a apresentar!\n';
END IF;

-- Mostrar valores finais
SET Fatura = CONCAT(Fatura, '\nCustoAbertura: ', Abertura, '€');
SET Fatura = CONCAT(Fatura, '\nTotal: ', Total, '€');

RETURN Fatura;
END $$

DELIMITER ;

```

GRANT EXECUTE ON FUNCTION RM18 TO
 'administrador'@'localhost',
 'administrativo'@'localhost',
 'detetive'@'localhost';

```

-- Herança de custo dos procedimentos

DROP TRIGGER IF EXISTS ProcedimentoHerdarCusto;

DELIMITER $$

CREATE TRIGGER ProcedimentoHerdarCusto
    BEFORE INSERT ON Procedimento
    FOR EACH ROW
BEGIN
    DECLARE Custo DECIMAL(5, 2);

    IF NEW.CustoAgencia = 0 THEN
        SELECT CustoAgencia INTO Custo
        FROM TipoProcedimento
        WHERE Id = NEW.Tipo;
        SET NEW.CustoAgencia = Custo;
    END IF;

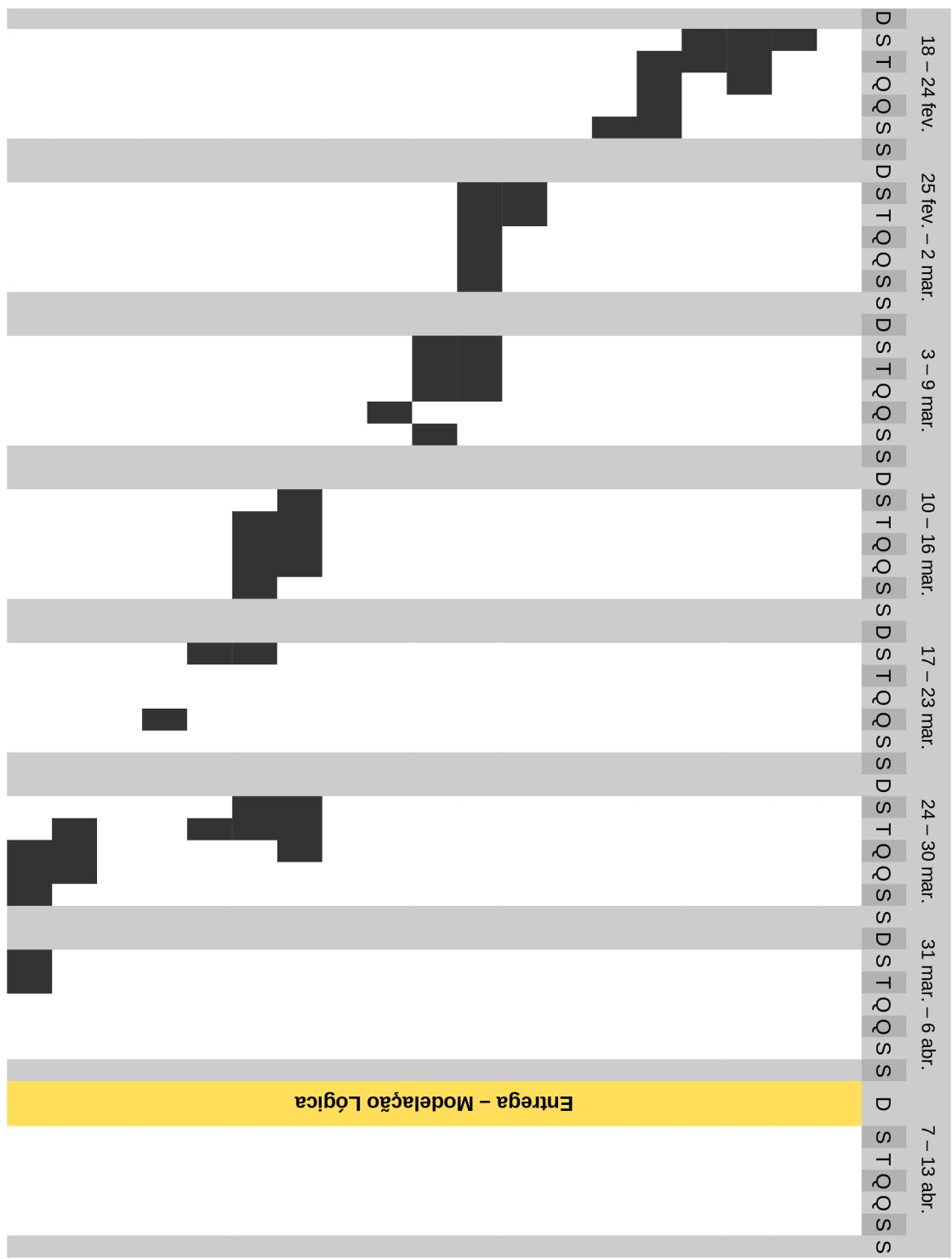
    IF NEW.CustoCliente = 0 THEN
        SELECT CustoCliente INTO Custo
        FROM TipoProcedimento
        WHERE Id = NEW.Tipo;
        SET NEW.CustoCliente = Custo;
    END IF;
END $$

DELIMITER ;

```

XIII. Diagrama de Gantt – Tarefas Realizadas

Tarefa	Atribuída a	Começo	Fim
Definição do Sistema			
Contextualização e Fundamentação	Prof. Doutor Elias	19 fev.	23 fev.
Objetivos e Viabilidade	Prof. Doutor Elias	19 fev.	19 fev.
Recursos e Equipa	Prof. Doutor Elias, Ivo Vieira, José Lopes	19 fev.	20 fev.
Plano de Execução	Prof. Doutor Elias, Humberto Gomes	20 fev.	22 fev.
Revisão e Aprovação	Prof. Doutor Elias, José Matos	23 fev.	
Definição de Requisitos		26 fev.	5 mar.
Definição do método e vistas	Ivo Vieira	26 fev.	27 fev.
Levantamento de Requisitos	Prof. Doutor Elias, Orlando Feio, Jacinto Fonseca, Equipa de desenvolvimento (todos)	26 fev.	6 mar.
Análise e Organização de Requisitos	José Lopes	3 fev.	6 mar.
Validação de Requisitos	Prof. Doutor Elias, Orlando Feio, Jacinto Fonseca, Humberto Gomes	7 mar.	
Modelação Conceptual		11 mar.	27 mar.
Identificação de entidades e relacionamentos	Ivo Vieira, José Matos	11 mar.	27 mar.
Definição de atributos (inc. domínios e chaves)	José Lopes	12 mar.	26 mar.
Elaboração do modelo conceitual	Humberto Gomes	18 mar. + 26 mar.	
Validação do modelo conceitual	Prof. Doutor Elias, José Matos	21 mar.	
Modelação Lógica		26 mar.	2 abr.
Derivação do modelo lógico	Humberto Gomes	26 mar.	28 mar.
Normalização e validação do modelo lógico	Ana Cerqueira, Humberto Gomes	27 mar.	2 abr.



Tarefa	Atribuída a	Começo	Fim	D
Implementação Física				
Criação da base de dados	Ana Cerqueira e José Lopes	18 abr.	27 mai.	
Criação de utilizadores e atribuição de privilégios	Humberto Gomes	18 abr.	30 abr.	
Criação de vistas de utilização	Humberto Gomes	29 abr.	10 mai.	
Povoamento da base de dados	José Matos e Humberto Gomes	29 abr.	2 mai.	
Exploração e Monitorização				
Cálculo do espaço da base de dados	Ivo Vieira e Humberto Gomes	8 mai.	24 mai.	
Implementação de interrogações	Humberto Gomes	13 mai.	15 mai.	
Indexação do sistema de dados	Ivo Vieira	22 mai.	27 mai.	
Implementação de procedimentos, funções e gatilhos	José Lopes	6 mai.	13 mai.	
Suporte técnico	Equipa de desenvolvimento (todos)	28 mai.	-	
Entregue - Modelação Lógica				