

Universidade do Minho

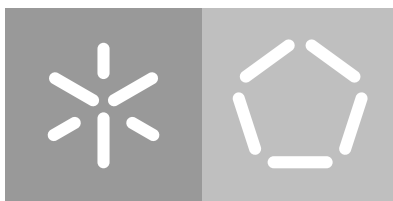
Escola de Engenharia

Departamento de Informática

Bárbara Nadine Freitas Oliveira

Incorporação de um Sistema Analítico numa Plataforma Computacional de Avaliação

Novembro 2021



Universidade do Minho

Escola de Engenharia

Departamento de Informática

Bárbara Nadine Freitas Oliveira

Incorporação de um Sistema Analítico numa Plataforma Computacional de Avaliação

Dissertação de Mestrado

Mestrado Integrado em Engenharia Informática

Trabalho efectuado sob orientação de

Orlando Belo

Novembro 2021

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.



Atribuição
CC BY

<https://creativecommons.org/licenses/by/4.0/>

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

ABSTRACT

It is now possible to prove that technology has proven to be a strong ally in the most diverse areas, from economics and management to health or banking. Education is therefore no exception. The insertion of technology and software that provide students with educational and motivational support for their learning has been a major challenge. It is based on this lack of support that a project called “Leonardo” emerged at the University of Minho. This project considers the development of educational software, which aims to provide students with a supervised learning method that will allow students to improve their knowledge and also observe their results. All of this will allow you to draw lessons that support your progress in a given area of study. This dissertation proposes, therefore, the development of a data warehousing system, which allows the collection of information about users and their interactions with the referred system. In this way, it will be possible to obtain a multidimensional data analysis platform, which allows to monitor the student’s state of knowledge over time, thus allowing him / her to evaluate his / her progress, in a given field of study, during the interaction with a student. the system.

Keywords: Educational Software, Educational Systems, Data Warehousing, Multidimensional Data Systems, Data Analysis, Data Visualization, Dashboarding.

RESUMO

Atualmente, é possível comprovar que a tecnologia tem-se demonstrado um forte aliado nas mais diversas áreas, desde a economia e gestão até à saúde ou à banca. A área da educação não é, pois, uma exceção. A inserção de tecnologia e de software que forneçam ao aluno um suporte de ensino e de apoio à sua aprendizagem, de forma educativa e motivadora, tem sido um grande desafio. É com base nesta carência de suporte, que surgiu na Universidade do Minho um projeto denominado “Leonardo”. Este projeto considera o desenvolvimento de software educacional, cujo objetivo é fornecer aos alunos um método de aprendizagem supervisionado, que permitirá aos alunos melhorarem o seu conhecimento e também observar os seus resultados. Tudo isto permitirá retirar ilações que suportem o seu progresso numa dada área de estudo. Esta dissertação propõe assim, o desenvolvimento de um sistema de data warehousing, que permita a recolha de informações acerca dos utilizadores e das suas interações com o sistema referido. Desta forma, será possível obter uma plataforma de análise de dados, multidimensional, que permita acompanhar o estado do conhecimento de um aluno ao longo do tempo, permitindo desta forma avaliar o seu progresso, num determinado domínio de estudo, ao longo da sua interação com o sistema.

Palavras-chave: Software Educativo, Sistemas Educacionais, Data Warehousing, Sistemas Multidimensionais de Dados, Análise de Dados, Visualização de Dados, Dashboarding.

CONTEÚDO

1	INTRODUÇÃO	1
1.1	Contextualização	1
1.2	Motivação	2
1.3	Objetivos	3
1.4	Trabalho Realizado	4
1.5	Estrutura da Dissertação	4
2	UTILIZAÇÃO DE DATA WAREHOUSES EM SISTEMAS DE ELEARNING	7
2.1	Sistemas de <i>eLearning</i>	8
2.2	ITS e a sua importância num domínio educativo	8
2.3	A Utilização de <i>Data Warehouses</i>	9
2.4	Exemplos de aplicação	11
2.5	Tipos e Modelos de <i>Data Warehouse</i> Utilizados	13
2.6	A Opção <i>Document Store</i>	15
3	DATA WAREHOUSES ORIENTADOS A DOCUMENTOS	16
3.1	Apresentação Geral	16
3.2	Sistemas Orientados a Documentos vs Sistemas Relacionais em Sistemas de <i>Data Warehousing</i>	16
3.3	Modelos de Dados Orientados a Documentos	18
3.4	Modelação Dimensional	23
3.5	Tipos de Tabelas de Dimensão	26
4	DATA WAREHOUSE ORIENTADO A DOCUMENTOS - CASO DE ESTUDO	55
4.1	Implementação de um Sistema Analítico	56
4.2	Estrutura Base do <i>Data Warehouse</i> do Sistema	58
4.3	Dimensões com variação	63
4.4	<i>Queries</i> sobre Dimensões com Variação	67
4.5	Utilização de Vistas Materializadas	73
4.6	Visualização e Análise de Dados	75
5	CONCLUSÕES E TRABALHO FUTURO	85
5.1	Conclusões	85
5.2	Trabalho Futuro	89

LISTA DE FIGURAS

Figura 1	O processo de interação aluno-agente.	9
Figura 2	Esquema representativo de um sistema de <i>data warehousing</i> .	10
Figura 3	Exemplo representativo de um documento com estrutura <i>DFL</i> .	19
Figura 4	Exemplo representativo de documento com estrutura <i>DSH</i> (coleção de factos).	19
Figura 5	Exemplo representativo de documentos com estrutura <i>DSH</i> (coleção das dimensões).	20
Figura 6	Exemplo representativo de uma opção de salvar história num modelo <i>DFL</i> .	21
Figura 7	Exemplo representativo de uma outra opção de salvar história num modelo <i>DFL</i> .	22
Figura 8	Um exemplo de um esquema em estrela.	24
Figura 9	Um exemplo de um esquema em floco-neve.	25
Figura 10	Exemplo de um modelo dimensional integrando uma dimensão " <i>calendário</i> ".	26
Figura 11	Atualização do valor do par chave-valor " <i>categoria</i> " na fonte de dados.	27
Figura 12	Valor do par chave-valor com variação tipo o mantém-se inalterado no <i>data warehouse</i> .	27
Figura 13	Exemplo demonstrativo de uma dimensão com variação - tipo 1.	28
Figura 14	Documento antes da inserção de um novo documento.	30
Figura 15	Documentos atualizados com a nova informação.	30
Figura 16	Exemplo demonstrativo de uma dimensão com variação - tipo 3.	31
Figura 17	Modelo exemplificativo contendo uma dimensão com variação tipo 4.	32
Figura 18	<i>Data Mart</i> exemplificativo relativo a alugueres de filmes.	33
Figura 19	Mini Dimensão relativa à dimensão filmes do <i>data mart</i> alugueres.	33
Figura 20	Um modelo contendo uma dimensão com variação tipo 5.	34
Figura 21	<i>Data Mart</i> exemplificativo relativo a alugueres de filmes.	35
Figura 22	Documento contendo um atributo com variação tipo 6.	37
Figura 23	Documentos contendo atributo com variação tipo 6.	38
Figura 24	Documentos que representam o processo aquando a atualização de um valor.	40
Figura 25	Dimensão onde é armazenado o valor mais recente corresponde ao filme com " <i>filme_id</i> " 'Foo1'.	40
Figura 26	Um esquema dimensional utilizando uma dimensão conforme.	41
Figura 27	<i>Data marts</i> do sistema.	41
Figura 28	Dimensão conforme " <i>Dim Filme</i> ".	42
Figura 29	Modelo exemplificativo contendo uma dimensão degenerada.	43

Figura 30	Um modelo de um documento contendo uma dimensão degenerada.	43
Figura 31	Modelo de dados contendo uma dimensão gigantesca.	45
Figura 32	Documento contendo uma dimensão gigantesca.	46
Figura 33	Modelo exemplificativo de uma dimensão contendo informação pessoal.	47
Figura 34	Modelo exemplificativo de uma subdimensão.	47
Figura 35	Modelo de uma <i>role playing dimension</i> .	48
Figura 36	Modelo contendo as vistas relativas aos diferentes significados da dimensão <i>calendario</i> . ¹	48
Figura 37	Documento contendo os pares chave-valor referentes aos diferentes significados da dimensão <i>calendário</i> .	49
Figura 38	Modelo contendo duas novas dimensões.	50
Figura 39	Modelo contendo <i>junk dimension</i> .	50
Figura 40	Modelo exemplificativo de um documento contendo atributos "indicadores".	51
Figura 41	Arquitetura da implementação do sistema.	57
Figura 42	Exemplo de um documento do <i>Data mart</i> de Respostas.	59
Figura 43	Ilustração do processo de captura, tratamento e registo da informação de acesso.	60
Figura 44	Exemplo de um log gerado pelo servidor <i>Apache</i> .	61
Figura 45	Exemplo de um documento do <i>Data mart</i> do sistema.	61
Figura 46	Exemplo de um documento da dimensão " <i>Dim.User</i> " contendo um novo par chave-valor com variação.	64
Figura 47	Exemplo de um documento da dimensão " <i>Dim.User</i> " contendo o novo par chave-valor " <i>Study_Plan</i> " com variação.	66
Figura 48	Exemplo de um documento contendo valor histórico do chave-valor " <i>Study_Plan</i> ".	66
Figura 49	<i>Query</i> executada para a estrutura 1.	68
Figura 50	<i>Query</i> executada para a estrutura 2.	68
Figura 51	<i>Query</i> executada para a estrutura 1.	69
Figura 52	<i>Query</i> executada para a estrutura 2.	69
Figura 53	<i>Query</i> executada para a estrutura 1.	71
Figura 54	<i>Query</i> executada para a estrutura 2.	71
Figura 55	Número de respostas dadas a questões de um dado domínio de estudo.	73
Figura 56	Processo de criação da vista.	74
Figura 57	Exemplo de documento com informação relativa ao cálculo do intervalo temporal dos registos do sistema para um dado domínio.	74
Figura 58	<i>Dashboard</i> que contém a vista mais geral do sistema.	77
Figura 59	<i>Dashboard</i> que contém vistas direcionadas às respostas armazenadas no sistema por nível de desempenho.	79
Figura 60	<i>Dashboard</i> que contém vistas direcionadas às respostas individuais de um determinado utilizador armazenadas no sistema.	81
Figura 61	<i>Dashboard</i> que contém vistas direcionadas às respostas individuais de um determinado utilizador armazenadas no sistema.	82
Figura 62	<i>Dashboard</i> que contém vistas direcionadas aos acessos da plataforma.	83

ACRÓNIMOS

CBT	Computer Based Training
DFL	Document Flat
DoDBMS	Distributed Object-oriented Database Management System
DSH	Document Shattered
DW	Data warehouse
ETL	Extract, transform, load
ITS	Intelligent tutoring systems
MQL	MongoDB Query Language
RAM	Random-access memory
RDBMS	Relational Database Management System
ROLAP	Relational On Line Analytical Processing
SQL	Structured Query Language

INTRODUÇÃO

1.1 Contextualização

A introdução de soluções tecnológicas no ensino é vista como uma nova forma de disponibilizar ferramentas para auxílio ao processo de aprendizagem do aluno. Hoje, praticamente todos conseguem ter acesso a essas soluções, integrando-as facilmente nas suas atividades mundanas e, obviamente, de ensino e de aprendizagem. Dentro dessas soluções encontramos os sistemas *Computer Based Training (CBT)* e os *Intelligent Tutoring Systems (ITS)*. Os sistemas *CBT* caracterizam-se por terem uma estrutura predefinida para todos os utilizadores, o que inviabiliza a disponibilização de um tipo de ensino individualizado que vá de encontro às dificuldades de cada utilizador [Hamidullah e Samiullah, 2015]. Os *Intelligent Tutoring Systems* dispõem de técnicas mais avançadas, possibilitando um tipo de ensino adaptativo e individualizado, ajustando os conteúdos disponibilizados ao longo do processo de aprendizagem do aluno, de acordo com as suas necessidades. Para que tal seja possível, este tipo de sistemas têm por base as respostas e interações do utilizador ao longo do tempo. Deste modo, o sistema não só regista todas as ações tomadas pelo utilizador, como fornece intervenções pedagógicas e motivacionais, de acordo com a análise dos resultados obtidos. Assim, é possível fornecer ao aluno um processo de ensino personalizado, bem como avaliar o seu conhecimento num determinado domínio, determinar quais as áreas nas quais este apresenta maiores dificuldades e orientar o processo de aprendizagem de modo a que estas lacunas sejam preenchidas [Santos e Jorge, 2013].

No ensino dito tradicional, no qual os conteúdos são lecionados por um tutor humano, não existe recursos ou tempo de proporcionar um ensino individualizado, focado nas dificuldades de cada aluno. Os *ITS* são vistos como complementos para este tipo de ensino tradicional, que proporcionam ao aluno um tipo de acompanhamento personalizado que um tutor humano não consegue fornecer, e ajustam os conteúdos abordados de acordo com as suas necessidades. Além disso, dispõem também de serviços que auxiliam os alunos na resolução de problemas, fornecendo um processo de aprendizagem interativo e motivador, disponibilizando *feedbacks* dos seus resultados e instruções, de modo a que possam melhorar as suas competências nos seus domínios de estudo [Reddy e Sasikumar, 2014]. Atualmente já existem implementados alguns *ITS* com bastante sucesso, em áreas como a matemática ou a aprendizagem de novas linguagens. Durante esta dissertação vão ser apresentados alguns destes exemplos de aplicação.

Geralmente, a arquitetura dos ITS é composta por quatro componentes. O primeiro componente é denominado por *Expert Model* ou *Domain Model*. Neste componente está representado o conhecimento de um especialista em um determinado domínio. É com base neste conhecimento, que o sistema vai comparar as respostas fornecidas pelos utilizadores e avaliar se estas estão corretas ou incorretas. O segundo componente é denominado por *Tutor Model*. Este componente é responsável pela escolha do conteúdo que será apresentado ao utilizador, tendo por base o seu conhecimento, bem como a estratégia de ensino adotada. O terceiro componente é designado por *Student Model*. Como o próprio nome indica, este componente é direcionado ao aluno. É responsável por armazenar toda a informação referente a um aluno, o seu estado de conhecimento e as suas características cognitivas e individuais. Por fim, temos o componente designado por *User Interface*. Este componente representa a interação entre o utilizador e o sistema. É o módulo responsável por apresentar a informação ao utilizador e capturar as respostas inseridas pelo mesmo [Britto et al., 2017].

1.2 Motivação

Nos últimos anos tem-se vindo a observar alterações significativas no sistema de ensino e de aprendizagem, em que, por forma a aumentar a motivação dos alunos e de tentar reforçar os conhecimentos adquiridos no ensino dito tradicional, a introdução de *software* educacional nas instituições de ensino e nos lares tem sido visto como uma boa solução.

Dado o grande número de alunos presentes atualmente no sistema de ensino, e tendo em vista que cada um deles carece de um tipo de acompanhamento distinto, é natural reconhecer que os ITS podem ser bastante úteis no processo de formação de qualquer aluno. Para que a implementação de um sistema deste tipo possa ser realizada com sucesso, é necessário processar a informação obtida a partir das interações do aluno, determinar quais os domínios cujo conhecimento é menor e focar o processo de aprendizagem nesses domínios de modo a melhorar o seu conhecimento. Assim, a incorporação de uma plataforma de análise e de avaliação neste tipo de sistema, que permita acompanhar a evolução do conhecimento adquirido pelos alunos ao longo do tempo, num determinado domínio, é bastante atrativo e interessante. A incorporação de uma plataforma analítica neste tipo de sistemas pretende disponibilizar diferentes *dashboards* com diferentes pontos de análise referentes ao desempenho dos alunos ao longo do tempo. Desta forma, um professor conseguirá, por exemplo, identificar, de uma forma imediata, quais os domínios em que determinado aluno tem melhor ou pior desempenho.

Para que tal plataforma analítica opere de forma eficiente, é imperativo manter um histórico de todos os processos de avaliação realizados pelos alunos e, com base no seu desempenho, definir novas estratégias de avaliação de modo a combater os aspetos mais negativos que foram identificados [Reddy e Sasikumar, 2014]. Para além disso, é também necessário ter em conta as diferentes vertentes de avaliação estabelecidas, fazendo a sua análise de acordo com um conjunto bem estabelecido de critérios base.

O plano de trabalhos desta dissertação foi estabelecido com base na necessidade de conceber e implementar um módulo analítico de dados para um projeto em desenvolvimento na Universidade do Minho, o "Leonardo" [BELO et al., 2019]. O "Leonardo" pode ser classificado como um sistema inteligente de aferição de conhecimento. Este projeto, pretende apoiar o processo de aprendizagem dos alunos em determinados domínios. O sistema "Leonardo" será abordado com mais detalhe no Capítulo 4. Como em qualquer ITS, um módulo analítico torna-se obrigatoriamente um requisito. Este módulo analítico, entre outras coisas, permite que o professor acompanhe a evolução dos seus alunos e que os alunos identifiquem os seus pontos fracos em termos de conhecimento adquirido.

1.3 Objetivos

O objetivo final desta dissertação foi a implementação de um módulo responsável pela análise da informação gerada durante o processo de aprendizagem dos utilizadores num ITS específico, o "Leonardo" [BELO et al., 2019]. Neste módulo é possível analisar todas as interações do utilizador com o sistema, bem como a criação de *dashboards*, que disponibilizem uma vista global da evolução do conhecimento adquirido pelo aluno, ao longo do tempo. Para isso, foi concebido e implementado um sistema multidimensional de dados, assente num *data warehouse*, especificamente orientado para o acolhimento dos dados do processo de avaliação de alunos, no qual foi possível efetuar a procura e análise de informações tendo em conta várias dimensões, como o tipo de domínio de estudo, o nível de dificuldade, entre muitos outros. Posteriormente, foi desenvolvida uma plataforma de visualização de dados, na qual foram apresentados diferentes *dashboards*, tendo por base os dados recolhidos durante o processo de avaliação, orientados por diferentes tópicos.

A implementação deste módulo assentou num sistema de *data warehousing* que foi implementado num sistema de base de dados orientado a documentos. Ao longo do processo de implementação tivemos que estudar como aplicar as diferentes estruturas de um *data warehouse* numa base de dados orientada a documentos. Assim, estudámos várias abordagens para a implementação de um sistema deste tipo, comparando-as, quando possível, com as formas de implementação tradicionais utilizando um sistema de bases de dados relacional e avaliando o desempenho do sistema ao utilizá-las em situações práticas reais. Tendo em consideração que o funcionamento de um sistema orientado por documentos difere bastante do comportamento de um sistema de bases de dados relacional, a adaptação de modelos já conhecidos para sistemas de bases de dados relacionais a sistemas orientados por documentos, poderia não ser suficiente para a obtenção de um bom desempenho. Desta forma, foram abordados outro tipo de estruturas para complementar esta adaptação, sendo elas, vistas materializadas.

Em suma, com este trabalho de dissertação, pretendeu-se fazer a:

- Análise de diferentes estruturas para a implementação de *data warehouse* orientado a documentos;
- Análise do desempenho de diferentes *queries* aplicadas a um *data warehouse* orientado a documentos;

- Determinação de qual a melhor estrutura de dados para a implementação do *data warehouse* aplicado ao caso de estudo em análise (o “Leonardo”);
- Implementação de um *data warehouse* orientado a documentos para o caso de estudo em análise;
- Criação de *dashboards* que permitissem a análise da informação gerada pelo sistema em tempo real.

1.4 Trabalho Realizado

Durante a realização desta dissertação foi efetuado uma análise de várias abordagens para a implementação de um *data warehouse* em bases de dados orientadas a documentos. Esta análise focou-se em determinar diferentes opções para a modelação de um *data warehouse* em bases de dados deste tipo, tendo sido efetuada uma análise crítica em todas as opções apresentadas, realçando as vantagens e desvantagens de cada uma delas.

Numa fase posterior foi concetualizado e implementado um módulo analítico de dados que incorporou um projeto educativo, já referido anteriormente, o “Leonardo”[BELO et al., 2019]. Este módulo analítico assentou num *data warehouse* implementado em *MondoDB*, dividido em dois *data marts* distintos. O *data mart* de respostas, onde foram armazenadas todas as respostas geradas pelo sistema, e o *data mart* de logs, responsável por armazenar a informação dos logs gerados pelo sistema. Estes *data marts* eram alimentados com informação em tempo real, permitindo que a posterior análise desta informação, fosse também em tempo real.

Com esta dissertação conseguiu-se fazer a:

- Análise de possíveis estruturas de dados para modelação de *data warehouses* em sistemas orientados a documentos;
- Implementação de um *data warehouse* em *MongoDB*;
- Implementação de um processo de processamento de ficheiros de logs gerados por um servidor *Apache* e atualização de um *data warehouse* em tempo real;
- Criação de *dashboards* que permitiram analisar a informação em tempo real.

1.5 Estrutura da Dissertação

Para além do presente capítulo, esta dissertação está organizada em mais cinco capítulos, cada um deles abordando uma parte específica do trabalho de dissertação, nomeadamente:

- **Capítulo 2** - Utilização de *Data Warehouses* em Sistemas de *eLearning*. Neste capítulo aborda-se a necessidade da presença de Sistemas de *eLearning* num domínio educativo e qual a importância da utilização de *data warehouses* neste tipo de sistemas. Numa segunda parte deste capítulo, são apresentados três exemplos de aplicações reais de sistemas de *eLearning* aplicados a diferentes áreas de educação, sendo avaliadas e comparadas as características de cada uma delas. Numa fase seguinte, são abordados diferentes tipos de sistemas para a implementação de *data warehouses* conhecidos, nomeadamente sistemas de gestão de base de dados relacionais e sistemas de gestão de base de dados não relacionais, e especificadas possíveis vantagens e desvantagens disponibilizadas por cada um deles. Por fim, é introduzida a opção *document store*, sendo esta o foco desta dissertação.
- **Capítulo 3** - *Data Warehouses* Orientados a Documentos. Neste capítulo é efetuada uma análise comparativa entre sistemas orientados a documentos e sistemas relacionais quando aplicados a sistemas de *data warehousing*. Nesta análise são referenciadas duas abordagens distintas da implementação de dois *data warehouses* em três sistemas de base de dados distintos, sendo feita uma comparação em termos de desempenho de determinadas consultas para diferentes volumes de dados. De seguida, são apresentados dois modelos de dados distintos aplicados a sistemas orientados a documentos. Estes modelos denominam-se como *Document Flat* e *Document Shattered*, sendo que o *Document Flat* corresponde a um modelo desnormalizado de dados, em que todos os campos da estrutura encontram-se no mesmo nível e o *Document Shattered* corresponde a um modelo mais normalizado de dados, onde, à semelhança do que ocorre em sistemas relacionais, são criados relacionamentos entre os dados. Numa fase final, são abordadas diferentes estruturas que possibilitam a modelação dimensional em sistemas orientados a documentos. Nesta parte deste capítulo, foram apresentados os principais conceitos referidos por [Kimball e Caserta, 2011] e efetuadas várias análises da sua possível aplicação a modelos orientados por documentos. Em cada um destes tópicos foi efetuada uma análise crítica e apresentados exemplos representativos que a suportam.
- **Capítulo 4** - *Data Warehouse* Orientado a Documentos - Caso De Estudo. Neste capítulo é apresentado um projeto em desenvolvimento na Universidade do Minho, o "Leonardo" [BELO et al., 2019], no qual se integrou o módulo analítico desenvolvido ao longo desta dissertação. Este módulo analítico teve por base um *data warehouse* implementado em *MongoDB*, composto por dois *data marts* distintos, o *data mart* de respostas e o *data mart* de logs do sistema. Neste Capítulo foi ainda efetuada uma análise de *performance* de algumas *queries* de consulta aplicadas a diferentes volumes de dados, tendo sido estes resultados debatidos de forma a encontrar possíveis motivos que os justificassem. Numa fase final, foram criados diferentes tipos de *dashboards*, assentes no *data warehouse* desenvolvido, cujo propósito principal seria permitir acompanhar o processo de aprendizagem e a evolução de um aluno ao longo do tempo.
- **Capítulo 5** - Conclusões e Trabalho Futuro. Neste último capítulo apresentamos as principais conclusões deste trabalho, abordando resumidamente os principais tópicos apresentados ao longo desta dissertação, referindo as eventuais vantagens e desvantagens de cada um deles e enaltecendo a importância da integração de uma ferramenta analítica de dados em plataformas de suporte à decisão. Para além disto, foram também referidas algumas dificuldades

encontradas durante o desenvolvimento da dissertação, bem como identificados alguns pontos que requerem mais trabalho e exploração no futuro.

UTILIZAÇÃO DE DATA WAREHOUSES EM SISTEMAS DE ELEARNING

A evolução da tecnologia é um processo inevitável e imprescindível no mundo de hoje. O seu enorme desenvolvimento advém de um aumento massivo de informação gerada pelos mais diversos dispositivos. Estimou-se que, já em 2020, existiam mais do que 75 mil milhões de dispositivos conectados à Internet [Beatty, 2017], bem como foram gerados cerca de 40 biliões de *gigabytes* de novos dados [IDC, 2012]. Desta forma, tornou-se imperativo a emergência de novos mecanismos de análise de informação, de forma a dar resposta a este aumento de informação. Para que tal seja possível, é importante avaliar outros paradigmas de estruturação e de armazenamento de informação, para além das convencionais ou das opções mais seguras que atualmente podem ser encontradas no mercado.

Neste capítulo foi apresentada uma primeira abordagem a novas alternativas na criação de sistemas multidimensional de dados em detrimento das soluções mais usuais.

A educação tem vindo a ser sujeita a constantes mudanças requeridas para ultrapassar os diversos desafios que vão surgindo. Um dos principais problemas, e do qual advêm muitos outros, é a dificuldade dos educadores em conseguirem captar a atenção e manter o interesse dos seus alunos ao longo dos seus processos de aprendizagem. Porém, não é fácil encontrar uma justificação válida e consensual para a ocorrência desta situação. Várias questões podem ser levantadas. Por exemplo, Poderá a culpa da falta de interesse demonstrada pelos alunos ser-lhes totalmente atribuída? Não tomarão eles as suas decisões com base na informação que conseguem obter? E não será esta informação insuficiente ou inconsistente?.

Como forma de responder a estas questões, e com o intuito de propor uma solução para estes problemas, surgiu a ideia da incorporação de um sistema de apoio à decisão num sistema de ensino capaz de o alojar, e de o executar como um sistema de *eLearning* [Kim Schildkamp, 2013]. Assim, de seguida, abordaremos os sistemas de *eLearning*, as suas vantagens e a forma como é que estes podem influenciar o mundo educativo, bem com a importância da possível incorporação ou associação de um sistema de suporte à decisão à sua estrutura funcional.

2.1 Sistemas de eLearning

eLearning é definido como sendo “o uso de novas tecnologias de multimídia e da Internet para melhorar a qualidade da aprendizagem, facilitando o acesso a recursos e serviços” [Communities, 2001]. Basicamente, os sistemas de *eLearning* pretendem facilitar o acesso ao ensino, podendo ser integrados em diversos tipos de dispositivos e plataformas, permitindo que a sua utilização seja possível em qualquer altura e lugar. Vários autores têm definido eLearning, utilizando em cada uma das suas definições diferentes perspetivas de caracterização. Veja-se, por exemplo a definição de [Koochang e Harman, 2005]:

“E-learning is the delivery of education (all activities relevant to instructing, teaching, and learning) through various electronic media.”

Elearning pode ser considerado como um projeto educacional, uma vez que pressupõe a disponibilidade de um conjunto de domínios de estudo, estruturados, implementados e apresentados aos utilizadores com o propósito de fornecer instrução e aprendizagem sobre esses domínios de estudo. Este tipo de sistemas, é concebido com foco na aprendizagem individual dos utilizadores, disponibilizando um tipo de aprendizagem personalizada, de acordo com as necessidades de cada indivíduo. Desta forma, a conceção deste tipo de sistemas tem como princípios base, aumentar a eficácia na aprendizagem de determinados conceitos focando-se nas dificuldades individuais dos utilizadores, e fornecendo de informação e *feedback* que permitam orientar o aluno na direção correta para que ele possa melhorar o seu desempenho educativo [Kapounova e Kostolanyova, 2011].

Os sistemas de *eLearning* compreendem não só as funcionalidades relativas a este processo de aprendizagem, como também os domínios disponíveis para estudo ou questões que sobre eles possam ser lançadas. Porém, são sistemas complexos, que realizam muito mais do que estas tarefas. Para além do conteúdo educativo, os sistemas de *eLearning* incluem, infraestruturas e serviços para adicionar, armazenar e apresentar conteúdos, administrar a informação sobre os alunos e para monitorizar o seu processo de aprendizagem ao longo do tempo [Pahl, 2007].

2.2 ITS e a sua importância num domínio educativo

Os ITS são caracterizados como sendo programas que utilizam técnicas de Inteligência Artificial para apoiar alunos nos seus processos de ensino e de aprendizagem. Este tipo de *software* é capaz de complementar o processo de aprendizagem dos alunos de uma forma personalizada em várias áreas do conhecimento, avaliando as interações dos alunos e adaptando o processo de aprendizagem de acordo com as necessidades específicas de cada aluno em questão [Britto et al., 2017].

Com o propósito de proporcionar esta forma de ensino personalizada, este tipo de sistema recorre frequentemente à utilização de agentes inteligentes. Tais agentes pretendem simular o comportamento ou raciocínio de um tutor humano em domínios específicos de aplicação [Alvarado et al., 2017]. O

papel de um agente inteligente é o de fornecer ferramentas educacionais para facilitar a aquisição de conhecimento por parte do aluno, bem como identificar as necessidades que lhe vão surgindo ao longo do processo de aprendizagem, motivando-o, orientando-o e definindo estratégias que possam combater tais falhas, sempre com foco no objetivo principal, isto é, melhorar o aproveitamento e aumentar o conhecimento do aluno.

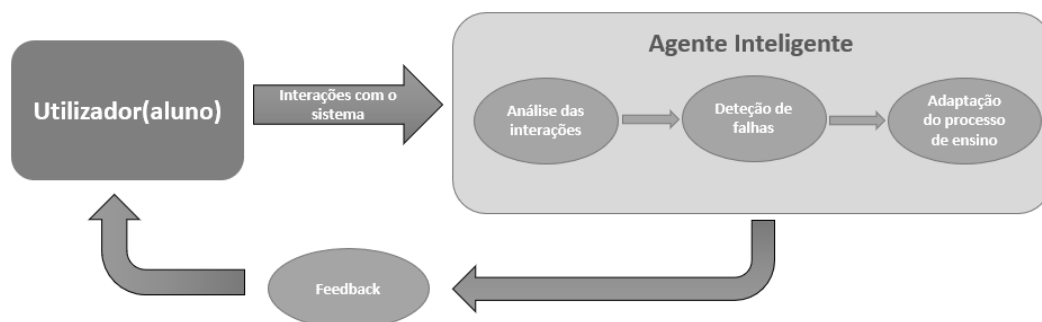


Figura 1: O processo de interação aluno-agente.

Este tipo de sistema prima pela motivação que proporcionam aos alunos, em detrimento do ensino tradicional, que muitos dos jovens consideram aborrecido e sem grande interesse. Assim, os ITS podem ser vistos como uma alternativa inovadora para motivar os alunos nas atividades de ensino e, ao mesmo tempo, transmitir conhecimento, proporcionando um método de ensino entusiasmante e incentivador.

Para além disto, atualmente o ensino à distância tem-se demonstrado cada vez mais necessário, o que faz com que os ITS possam ser vistos como um forte aliado no acompanhamento do processo de aprendizagem do aluno, bem como uma ferramenta de avaliação efetiva do seu desempenho.

Na Figura 1 é apresentada uma visão bastante genérica da intervenção de um agente inteligente num sistema de *eLearning*, sendo exibido as suas principais operações. A partir do momento que um aluno inicia o seu processo de aprendizagem, o sistema é alimentado com nova informação relativa ao seu desempenho. Com os dados gerados a partir dessas interações é definido o perfil do utilizador, o qual está a ser constantemente atualizado com novos dados gerados à medida que o utilizador interage com a plataforma. Por fim, a partir do perfil criado e dos resultados obtidos pelo utilizador, é ajustado o conteúdo das atividades educativas disponibilizadas pela plataforma. Sendo que o conhecimento que o aluno necessita adquirir está em constante mudança, é dever do agente adaptar os conteúdos disponibilizados de modo a focar os mesmos na necessidade individual do aluno. À medida que este processo ocorre, são fornecidos ao aluno *feedbacks* acerca do seu desempenho, bem como, conhecimento que visa complementar as dificuldades identificadas [Heales,2014].

2.3 A Utilização de Data Warehouses

É do conhecimento geral que hoje em dia, qualquer aplicação, plataforma ou sistema computacional, gera imensa informação, a qual é armazenada em repositórios de dados dos mais diversos tipos.

Infelizmente, por melhor que estejam estruturados esses repositórios, a forma como os dados são muitas vezes armazenados não permite, através da sua observação, retirar conhecimento e informação útil de uma forma rápida, eficaz e precisa, tão pouco fornecer uma visão temporal da evolução da informação. Um sistema de *eLearning* não é, pois, exceção. É neste domínio que a implementação de *data warehouses* pode trazer grandes vantagens nesses aspetos.

Os *data warehouses* são repositórios de dados estruturados especificamente para permitir a análise da informação armazenada ao longo do tempo. Têm como objetivo principal o suporte operacional de um sistema de suporte à decisão, que permita ao utilizador retirar conclusões informadas, que permitam a obtenção de melhores resultados da área de negócio em estudo, ou no caso particular dos sistemas de *eLearning*, permitir aos tutores avaliar o desempenho dos seus alunos ao longo do tempo, segundo uma ou mais perspetivas de análise.

No domínio de sistemas *eLearning*, os *data warehouses* podem desempenhar um papel fundamental na vertente analítica da informação. A implementação de um sistema de *data warehousing* pode disponibilizar um conjunto muito diverso de ferramentas analíticas ao professor, que, a partir da informação armazenada ao longo do tempo na plataforma, permite-lhe acompanhar e avaliar o desempenho dos alunos, bem como identificar a interação destes com a plataforma de *eLearning*, tendo acesso a um leque muito diverso de informação como, por exemplo, o número de sessões do utilizador, o tempo despendido em cada sessão, quais as páginas visitadas, entre várias outras opções.

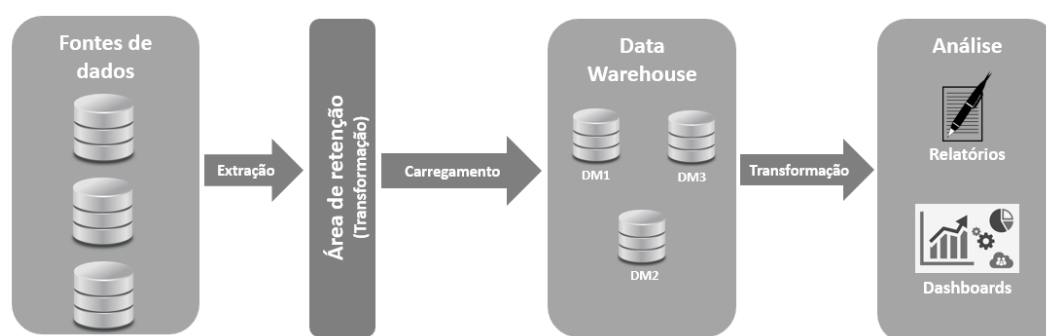


Figura 2: Esquema representativo de um sistema de *data warehousing*.

Na Figura 2 podemos ver uma esquematização da estrutura usual de um sistema de *data warehousing*. Como é possível observar nessa figura, o primeiro passo para a implementação de um sistema de *data warehousing* é fazer a identificação das fontes de dados e da interpretação da informação a ser armazenada. Desta forma, é possível determinar qual a informação que é vital ao *data warehouse* e, assim, partir para a conceitualização do modelo de dados do sistema. Após terminada a fase de conceitualização, na qual se definiu a arquitetura de todo o sistema, bem como a informação a armazenar, parte-se então para a implementação do mesmo. Nesta fase, começa-se por construir a área de retenção. A área de retenção é vista como uma zona intermédia, uma área de estágio, na qual os dados vão ser temporariamente armazenados e processados antes de serem carregados no *data warehouse*. Inicialmente, é feita a extração dos dados das fontes para estruturas localizadas

nesta área, na qual são sujeitos posteriormente a diversas transformações de forma a combinar a informação recolhida das diversas fontes. Aqui, basicamente, o objetivo é limpar e organizar os dados, bem como resolver possíveis inconsistências antes dos dados recolhidos serem carregados no *data warehouse*. Este processo é efetuado regularmente, à medida que nova informação é adicionada às fontes, podendo ser realizado em tempo real ou, então, num determinado período agendado previamente [Ponniah, 2004]. Por fim, e com base na informação armazenada no *data warehouse*, é implementado um módulo analítico, no qual poderão ser gerados relatórios, ou criados *dashboards* que forneçam uma vista clara dos dados ao utilizador e, que lhe possibilitem uma análise concisa da evolução da informação ao longo do tempo, permitindo tirar conclusões justificadas acerca do domínio em análise.

2.4 Exemplos de aplicação

Atualmente existem diversos sistemas de *eLearning* nas mais diversas áreas da educação. Neste capítulo, serão apresentados alguns exemplos, já implementados, de sistemas cujo foco principal é disponibilizar um ensino personalizado e individual dos utilizadores. Todos eles têm um objetivo comum, sendo este melhorar o conhecimento de quem o utiliza.

2.4.1 ZOSMAT

O “ZOSMAT” [Keleş et al., 2009] é um *web-based ITS* para o domínio da matemática. As características do “ZOSMAT” permitem acompanhar um aluno em qualquer ponto do processo de aprendizagem que se encontre, orientando-o acerca do que ele precisa de fazer e aprender. Para suportar este processo foi implementado no “ZOSMAT” um *data warehouse*, no qual são armazenadas as questões e respostas fornecidas pelos utilizadores. Desta forma, é possível, a partir da informação armazenada, inferir e delinear o perfil de um aluno, e a partir deste criar um acompanhamento personalizado tendo por base os seu histórico de resultados. Para consultar toda a informação acerca deste sistema, poderá ser consultado o artigo [Keleş et al., 2009].

2.4.2 MATHIA

O “MATHIA” [Carnegie Learning, 1998] é um *software* desenvolvido e utilizado nos Estados Unidos da América, dirigido para a área da matemática. Este *software* para além de se focar na aprendizagem do aluno, também se adapta às suas necessidades, na medida que redireciona ou reescreve questões para clarificar as mesmas, retorna *feedbacks* e dicas na resolução de problemas, entre outras funcionalidades. Além de permitir um tipo de ensino personalizado, este *software* dispõe também de funcionalidades analíticas de informação e de progresso. Este *software* permite aos alunos acompanhar o seu progresso e disponibiliza aos professores um conjunto variado de *dashboards* com projeções realizadas a partir

dos dados recolhidos ao longo do processo de aprendizagem dos alunos, de modo a fornecer informação para que estes possam identificar onde os alunos mais precisam de ajuda. Para além disto fornece ainda outros *dashboards* com informação acerca do desempenho geral dos alunos, como o tempo despendido a responder às questões, o número de alunos ativos, a média de erros por problema, etc [Wire, 2019].

Este sistema para além de avaliar e testar o conhecimento de aluno, fornece explicações prévias para que o aluno seja capaz de responder aos problemas colocados. Durante o processo de resolução das questões, este sistema auxilia o aluno, na medida que vai retornando dicas e disponibilizando correções aos problemas apresentados. Numa fase final, o sistema analisa as soluções dos alunos e identifica os erros que este cometeu. Depois, com base nestes erros, o sistema sugere a próxima atividade que o aluno deverá realizar.

2.4.3 Duolingo

O "Duolingo" [von Ahn, 2011], lançado em 2011, é um *software* educacional com o propósito de permitir a aprendizagem de novos idiomas de uma forma acessível e divertida. Esta aplicação conta com mais de 30 diferentes linguagens e permite que um utilizador tenha acesso a um ensino individual e personalizado no processo de aprendizagem de uma nova linguagem. Para que isto seja possível, este sistema começa por avaliar o conhecimento de um determinado utilizador à linguagem à qual pretende estudar, e determina o nível em qual se encontra. Desta forma, é possível otimizar o seu processo de ensino, evitando que conhecimento que já foi adquirido seja apresentado. Para além disto, o processo vai-se adaptando de acordo com o desempenho do aluno, contribuindo para uma melhor experiência do utilizador, evitando que perguntas demasiado básicas e que não acrescentem valor ao seu processo de aprendizagem sejam apresentadas. Para além do processo de aprendizagem abordado, o "Duolingo" possui um módulo de gamificação, permitindo ao utilizador ganhar moedas virtuais, desbloquear novos níveis, acompanhar a sua pontuação e comparar com outros utilizadores. Desta forma, este módulo permite motivar os utilizadores, tornando o processo de ensino mais divertido [Marr, 2020].

2.4.4 Comparação das Caraterísticas

Na Tabela 2 está apresentado um pequeno resumo em relação às funcionalidades disponibilizadas pelos três exemplos de sistemas de *eLearning* apresentados anteriormente. Comparando os três exemplos apresentados, é possível concluir que muitas das suas funcionalidades são comuns, pois todas estas são essenciais na composição de um sistema de *eLearning*. Porém, depois de uma breve observação e análise dos sistemas estudados, é possível perceber que existem algumas diferenças que se destacam. A grande diferença nota-se na disponibilização da informação ao professor. Enquanto o "MATHIA" fornece *dashboards* bastante completos do desempenho dos alunos, onde é possível visualizar quais os conteúdos que o aluno completou, tempos despendidos na plataforma, entre

muitos outros, os outros sistemas limitam-se a uma disponibilização de resultados. Também é interessante notar, que o “Duolingo” possui um sistema de gamificação bastante mais complexo que os restantes, sendo dada a sensação que se está a jogar um jogo educativo e a aprender ao mesmo tempo. Desta forma, o “Duolingo” torna-se um *software* bastante atrativo, podendo já contar com cerca de 300 milhões de utilizadores.

Funcionalidades	ZOSMAT	MATHIA	Duolingo
Módulo de gamificação	✓	✓	✓
Comparação de resultados com outros utiizadores	✓	X	✓
Disponibilização de sugestões	✓	✓	✓
Disponibilização de explicações de conteúdo	✓	✓	✓
Dashboards de desempenho para alunos	✓	✓	✓
Disponibilização dos resultados aos professores	✓	✓	✓
Dashboards de desempenho dos alunos para professores	X	✓	X

Tabela 2: Tabela comparativa das caraterísticas.

2.5 Tipos e Modelos de Data Warehouse Utilizados

Um *data warehouse* é um sistema que dispõe de estruturas que possibilitam o armazenamento de dados ao longo do tempo, permitindo manter um histórico dos mesmos para posterior análise. Para a conceção deste tipo de sistemas, é imperativo realizar a identificação dos diferentes tipos de informação que se quer armazenar, e a partir daí, definir a estrutura sobre a qual os dados vão ser organizados e armazenados. Um modelo de dados, pode ser visto como uma parte do *data warehouse* onde podem ser identificadas as estruturas dos dados, os seus atributos e relações necessárias ao armazenamento dos mesmos, tudo isto, para um determinado domínio do *data warehouse*. Porém, para que se consiga determinar qual o modelo de dados que mais se adequa ao problema em causa, não basta apenas determinar quais os dados a armazenar. A idealização de um modelo de dados tem bem mais condicionantes do que apenas o tipo de dados que se pretende albergar. É vital, pois, estar ciente do tipo de sistema para o qual o *data warehouse* se destina, do volume de dados que vai ser gerado, o tipo de base de dados com que vai ser implementado e, ainda, de analisar de que forma o modelo de dados pode ser implementado.

Um sistema de *data warehousing* é tradicionalmente implementado em sistemas de gestão de base de dados relacionais (*RDBMS*). Este tipo de sistemas disponibiliza ferramentas que permitem manter a integridade dos dados. Os dados são armazenados em estruturas no formato de tabelas, e permitem a utilização de chaves para referenciar e relacionar os dados entre diferentes tabelas. Disponibiliza ainda formas de controlo do tipo de dados de cada atributo da estrutura de dados, bem como, a criação de restrições que vetam a inserção de dados replicados [Chickerur et al., 2015]. Para além disto, este tipo de sistemas, disponibiliza uma linguagem intuitiva e de fácil aprendizagem (*SQL*),

que permite aceder e manipular esta informação. Todos estes fatores, tornam **RDBMS**, sistemas apelativos à implementação de um sistema de *data warehousing*.

Apesar de a via relacional ser aquela que é mais natural, um sistema de *data warehousing* pode perfeitamente ser implementado num modelo não relacional. Tendo em conta fatores externos ao *data warehouse*, e considerando todo o sistema em que este se insere, pode haver diversos motivos que impliquem a implementação do *data warehouse* numa base de dados *NoSQL*.

Os sistemas de base de dados *NoSQL* apresentam um conjunto de propriedades que fornecem ao utilizador uma maior flexibilidade em termos de reestruturação dos dados, quando comparado com sistemas relacionais. Desta forma, num sistema no qual a base de dados esteja sujeita a constantes alterações, pode ser mais vantajoso que a sua implementação seja num tipo de base de dados não relacional, tendo em conta que será um processo mais rápido e mais simples de executar. Porém, a modelação dum sistema deste tipo não pode ser influenciada por uma típica modelação feita em moldes relacionais. É necessário haver um distanciamento da abordagem que é feita nesse tipo de sistemas, e considerar as características de um sistema não relacional, de forma a se tirar partido das vantagens do mesmo para se implementar um sistema que cumpra o propósito de um sistema de suporte à decisão o melhor e mais eficiente possível.

2.5.1 Sistemas de Gestão de Bases de Dados

Um sistema de gestão de bases de dados pode ser definido como um *software* cujo propósito é armazenar e gerir a informação, dispondo de funcionalidades que permitem aos utilizadores consultar, inserir, remover ou modificar a informação armazenada [Sharma e Kaushik, 2015].

A arquitetura típica de um sistema de gestão de bases de dados divide-se em três módulos essenciais, nomeadamente:

- **Sistema de bases de dados:** Uma das principais funções de um sistema de gestão de base de dados, é a capacidade de armazenar dados. Existem diferentes formas de a informação ser armazenada, alguma delas apenas em **RAM**, grande parte em disco, garantindo maior persistência dos dados e, ainda, alguns que optam por um sistema híbrido de **RAM** e persistência em disco.
- **Acesso e manipulação da informação:** O segundo módulo é relativo à capacidade de fornecer ao utilizador uma forma de consultar a informação armazenada na base de dados e claro de a manipular. Para que esta comunicação seja possível, são utilizadas *query languages*, entre as quais **SQL** (Structured Query Language) no caso de **RDBMS**, ou então **MQL** (MongoDB Query Language) no caso de **DoDBMS**, em particular MongoDB.
- **Estruturas de armazenamento dos dados:** Por fim a estrutura dos dados armazenados é também uma importante parte dos sistemas de gestão de base de dados. Alguns sistemas de gestão de bases de dados permitem estruturar os dados de forma pouco flexível, sendo possível

definir numa fase inicial como os dados serão estruturados, e posteriormente é definido um esquema a ser aplicado aos mesmos. Este tipo de esquemas existe geralmente em base de dados relacionais, no qual a informação a ser armazenada é especificada e estruturada numa primeira fase. Por outro lado, existem sistemas mais flexíveis, onde não é necessária qualquer especificação deste tipo. Os dados não têm que ter um tipo associado e nova informação pode ser adicionada a qualquer altura sem implicar alterações nos dados armazenados até à altura, temos como exemplo base de dados orientadas a documentos [MongoDB,2009].

2.6 A Opção *Document Store*

Existem diversos tipos de sistemas de base de dados *NoSQL*, entre os quais se incluem os sistemas orientados a documentos - *document stores*. Neste tipo de sistemas, os dados estão estruturados em formato *JSON* e armazenados em estruturas denominadas por documentos. Estando os dados armazenados neste tipo de formato, é disponibilizado um nível de flexibilidade que um sistema relacional não dispõe, permitindo que, caso o sistema esteja sujeito a variações nos seus esquemas de dados, estas possam ser introduzidas no sistema de forma rápida e estruturada.

Porém, a implementação de *data warehouses* em sistemas orientados a documentos é ainda uma prática pouco convencional e, por isso, é uma opção que carece de alguma exploração e análise, tendo em conta as suas diversas vantagens e desvantagens. Devido ao facto deste tipo de sistema de base de dados, não ter sido primordialmente concebido para lidar com operações de junção de dados, nem com relacionamentos entre si, a implementação de um sistema de *data warehousing* orientado a documentos é um desafio bastante interessante. No próximo capítulo abordaremos alguns dos aspetos mais essenciais à sua implementação, bem como evidenciaremos algumas das suas características quando comparadas com os sistemas relacionais.

DATA WAREHOUSES ORIENTADOS A DOCUMENTOS

3.1 Apresentação Geral

As bases de dados *NoSQL* [Schaefer, n.d] surgiram como uma resposta a problemas criados na última década pelo grande crescimento de dados que se verificou a nível empresarial. Muitos desses problemas requereram um armazenamento de dados de uma forma bastante desnormalizada, podendo ser guardados em formatos que não implicavam relacionamentos e que permitiam à sua base de dados de acolhimento escalar rapidamente. Além disso, as bases de dados *NoSQL* têm grande flexibilidade em termos da estruturação dos dados, permitindo armazenar a informação que contêm, por exemplo, em formato *JSON* e manter um esquema de dados de forma bastante dinâmica, não tendo que, obrigatoriamente, manter a sua configuração inicial até ao fim do ciclo de vida do sistema de bases de dados. O tipo de estrutura que estes sistemas de bases de dados apresentam pode, assim, ser alterada a qualquer momento, sem custos em termos de tempo ou de implementação. Este tipo de característica pode ser bastante útil, quando o sistema em questão corre o risco de sofrer alterações no futuro, uma vez que não impõe uma nova implementação ou reestruturação do esquema previamente definido.

Apesar de existirem diversos tipos de base de dados *NoSQL*, por exemplo, base de dados baseadas em grafos ou bases de dados de chave-valor [Schaefer, n.d], o principal foco deste capítulo vai ser aplicado no domínio das bases de dados orientadas a documentos. Neste tipo de base de dados, os dados são armazenados em documentos com o formato *JSON*. Cada documento é constituído por diferentes pares chave-valor referentes à informação que se pretende armazenar [Aggoune e Namoune, 2020].

3.2 Sistemas Orientados a Documentos vs Sistemas Relacionais em Sistemas de *Data Warehousing*

Usualmente, o crescimento do volume de dados gerados por plataformas digitais implica também o crescimento do número de dados que necessitam de ser analisados. Para que isso seja possível

é necessário criar estruturas e serviços que permitam criar e suportar relatórios e *dashboards* que permitam fazer a sua análise ao longo do tempo. Como já referido anteriormente (Secção 3.1), o volume de dados tende a aumentar nos últimos anos, e sistemas relacionais não estão preparados para lidar com a enorme quantidade de dados que se tem presenciado atualmente de forma tão eficaz quanto os sistemas ditos não relacionais. A enorme mutabilidade dos dados é, também, um dos principais fatores que faz com que a opção por sistemas orientados a documentos na implementação de sistemas analíticos tenha um peso muito significativo. Em termos de *performance*, comparar os dois tipos de sistemas é bastante ambíguo, dada a grande diversidade de fatores que têm que ser considerados para que se consiga entender qual dos tipos de sistema é o mais adequado e qual é aquele que apresenta o melhor desempenho.

Em [Pereira et al., 2015] foi implementado um mesmo *data warehouse* em dois sistemas diferentes, de modo a ser possível comparar o desempenho de ambos. Um destes *data warehouses* foi implementado num sistema de base de dados relacional, o *MS SQLServer*, enquanto que o outro num sistema de bases de dados orientado a documentos, nomeadamente o *MongoDB*. O sistema de bases de dados relacional foi implementado com base num esquema em estrela, enquanto que o modelo não relacional foi implementado a partir de um esquema desnormalizado, no qual os documentos armazenavam as medidas e os atributos das dimensões com que estes se relacionavam. Após a implementação destes dois sistemas foi feita uma avaliação do seu desempenho tendo em conta 6 volumes de dados diferentes, que incluíram entre 250 mil até 5 milhões de registos e 4 consultas distintas, com níveis de complexidade distintos, sobre estes dados para ambos os sistemas. Os resultados destas operações, levou a crer que o aumento do volume de dados do sistema impactou negativamente das consultas ao sistema implementado em *MongoDB*. Para os mesmos volumes de dados o sistema implementado em *MongoDB* obteve tempos de execução sempre superiores ao sistema implementado em *SQLServer*, com exceção do volume mais pequeno (250 mil registos), em que ambos obtiveram resultados idênticos. Nos restantes conjuntos de dados, a discrepância entre ambos os sistemas foi bastante acentuada, tendo mesmo ultrapassado os 900 segundos no pior dos casos. Uma outra observação interessante, foi a discrepância nos valores de execução entre as diferentes *queries* feitas ao sistema. Enquanto no sistema implementado em *SQLServer* esta diferença foi bastante acentuada, onde para o máximo volume de dados testado (5 milhões), a diferença entre a *query* mais simples e a mais complexa teve uma diferença de aproximadamente 65 segundos, no caso do sistema implementado em *MongoDB* isto não se verificou. Neste caso, para todos os diferentes volumes de dados abordados, o sistema implementado em *MongoDB* apresentou tempos de execução muito idênticos para todas as *queries* executadas. Em média, para as 4 consultas diferentes, este valor rondou os 15 segundos, onde a consulta com maior tempo teve um valor de aproximadamente 1040 segundos e a de menor tempo de execução teve um valor aproximado de 990 segundos. A partir dessa avaliação, pode-se concluir que o sistema implementado em *MongoDB* não foi muito viável como sistema de suporte à decisão, uma vez que os seus resultados foram significativamente piores quando comparados com os valores obtidos em *SQLServer*.

Outro processo de análise pode ser visto em [Chavalier et al., 2016], no qual se fez uma comparação entre dois sistemas de *data warehousing* implementados em *PostgreSQL* e em *MongoDB*. Neste processo foi realizada uma comparação entre dois modelos de dados que foram implementados tanto no modelo relacional como no modelo não relacional. Para que isso fosse possível, foram implementados

para ambos os sistemas em avaliação dois tipos de esquemas, nomeadamente, um esquema em estrela e um esquema desnormalizado. Depois, realizaram-se diversos testes de *performance* tendo em consideração diversos volumes de dados. Com base nesses testes pudemos concluir que, no caso da implementação realizada com o *MongoDB*, o modelo de dados normalizado requereu menos espaço e o seu carregamento foi mais rápido. Tal como era esperado, o número de operações de junção com que este sistema teve que lidar prejudicou o seu desempenho na execução das *queries*. Porém, no modelo de dados desnormalizado obteve um melhor resultado em termos de desempenho. No caso de *PostgreSQL*, o modelo normalizado de dados teve as mesmas consequências em termos do espaço utilizado. O inverso aconteceu em termos de *performance*, que foi consideravelmente melhor quando o esquema em estrela foi usado em vez do modelo de dados desnormalizado. Por fim, contrariamente aquilo que aconteceu com o exemplo anterior, neste caso, a hipótese da implementação de sistemas de *data warehousing* em sistemas orientados a documentos não foi totalmente descartada.

Assim, a partir dos casos analisados, podemos concluir que a opção por uma solução não relacional poderá ser viável na implementação de um sistema de *data warehousing*. Apesar disso, é necessário realizar mais alguns estudos e análise, envolvendo mais modelos de dados para que se possa avaliar com mais rigor o comportamento deste tipo de sistemas.

3.3 Modelos de Dados Orientados a Documentos

Os sistemas de base de dados orientados a documentos dispõem de diferentes tipos de funcionalidades e estruturas de dados quando comparados com sistemas de base de dados relacionais. Estas diferenças, nomeadamente nas estruturas de dados, onde se podem ter documentos aninhados ou *arrays* de dados, levam a crer que um estudo da viabilidade da implementação de um *data warehouse* neste tipo de sistemas seja vantajoso. A flexibilidade fornecida por este tipo de sistemas possibilita uma grande diversidade na definição de diferentes modelos de dados. Dentro deste leque é possível ter um modelo de dados totalmente desnormalizado, em que as medidas e atributos ou pares chave-valor das dimensões se encontram ao mesmo nível. Uma outra opção, também ela desnormalizada, passa por criar subdocumentos com as diferentes dimensões e factos, fazendo com que as medidas e os pares chave-valor de cada dimensão fiquem agrupadas dentro dos subdocumentos respetivos. Além disso, é ainda possível, ter uma alternativa mais normalizada dos dados, na qual cada dimensão é armazenada em diferentes coleções, bem como o documento de factos que contém o identificador do documento relativo a cada dimensão com que se relaciona [Chavalier et al., 2016]. De seguida vão ser abordados com mais detalhe alguns destes modelos de dados, bem como analisadas algumas questões relativamente à sua aplicação num sistema de *data warehousing*.

[Chavalier et al., 2016] propuseram dois modelos de dados distintos (*Document Flat* e *Document Shattered*), para que fosse possível comparar o comportamento e desempenho de cada um e, consequentemente, conseguir determinar qual a melhor abordagem quando aplicado a um sistema orientado a documentos. No primeiro exemplo é apresentado um modelo desnormalizado de dados, denominado por DFL (*Document Flat*), no qual os factos e as dimensões associadas são armazenados numa única coleção. Nesta coleção cada documento contém as medidas e os pares chave-valor das

dimensões com os quais estas medidas se relacionam. Os documentos são denominados *flat* pois todos os dados estão ao mesmo nível.



Figura 3: Exemplo representativo de um documento com estrutura DFL.

O segundo modelo de dados proposto por [Chavalier et al., 2016], para estruturar os mesmos dados aplicados ao modelo anterior (DFL) e possibilitar a comparação dos resultados de ambos os modelos, foi designado por DSH (*Document Shattered*). Neste modelo os factos estão separados das dimensões de forma a evitar grandes níveis de redundância de dados. Como tal, os factos estão armazenados numa coleção e as dimensões em outras coleções. Para cada elemento de uma dimensão que esteja relacionado com um facto é guardada a correspondente chave nos documentos relativos ao armazenamento das medidas do modelo. Este segundo modelo é uma adaptação do esquema em estrela proposto por Kimball [Kimball e Ross, 2013] aplicado a um modelo orientado a documentos.

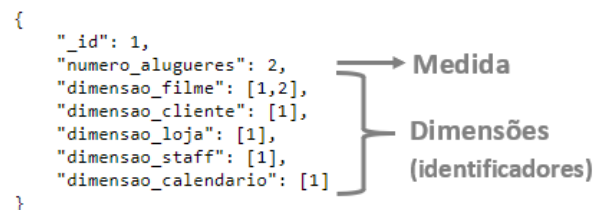


Figura 4: Exemplo representativo de documento com estrutura DSH (coleção de factos).

```

"dimensao_cliente"
{
  "id":1,
  "nome":"João Oliveira",
  "morada": "Rua dos Chãos, Braga"
}

"dimensao_staff"
{
  "id":1,
  "nome":"Maria Ferreira",
  "morada": "Rua de Barros, Braga"
}

"dimensao_loja"
{
  "id":1,
  "nome":"Maria Ferreira",
  "morada": "Avenida da Liberdade, Braga"
}

"dimensao_calendario"
{
  "id":1,
  "data":"18-12-2019",
  "dia_da_semana": "quarta-feira",
  "semestre": 2
}

"dimensao_filme"
{
  "id":1,
  "titulo":"Interstellar",
  "classificacao": 9.8,
  "linguagem": "inglês",
  "categoria": "ficção científica",
  "atores": [
    "Mathew McConaughey",
    "Anne Hathaway"
  ]
}

{
  "id":2,
  "titulo":"Ford vs Ferrari",
  "classificacao": 8.3,
  "linguagem": "inglês",
  "categoria": "biografia",
  "atores": [
    "Matt Damon",
    "Christian Bale"
  ]
}

```

Figura 5: Exemplo representativo de documentos com estrutura DSH (coleção das dimensões).

No modelo DFL facilmente verificamos que teremos um nível de redundância de dados elevado, nomeadamente no conteúdo relativo às dimensões. Em contrapartida, as *queries* de consulta serão mais simples e de execução imediata. Já no modelo DSH, o nível de redundância é mais reduzido. Porém, as *queries* serão bem mais complexas e mais custosas em termos de *performance*.

A próxima questão que temos que analisar é a incorporação de dimensões que integrem pares chave-valor com variação nos modelos apresentados e que requerem a manutenção do historial das modificações dos seus elementos de dados ao longo do tempo.

Para o primeiro modelo (DFL), uma possível abordagem para tratar essas questões é adicionar uma etiqueta temporal (*timestamp*) e um subdocumento, cujos valores sustentem o estado do documento. Assim, quando um registo de algum par chave-valor for atualizado, o estado anterior do documento é “marcado” como ‘inativo’ e um novo documento é criado com a nova informação. Um exemplo deste comportamento, pode ser observado na Figura 6. Esta abordagem tem alguns problemas em termos de escalabilidade e de redundância, que são claramente identificáveis. Por exemplo, se ocorrer a modificação de um registo de uma dada dimensão, que esteja presente em vários documentos na base de dados, será necessário proceder, também, à modificação de todos esses documentos.



Figura 6: Exemplo representativo de uma opção de salvar história num modelo DFL.

Verde: Novo subdocumento com estado e *timestamp*

Cinza: Par chave-valor com variação

Outra possível alternativa é adicionar um novo subdocumento ao documento, com o objetivo de guardar a informação desatualizada do par chave-valor que sofreu alteração num dado registo. Neste caso, o subdocumento adicionado, que é comumente uma lista de objetos, contém a seguinte informação: a data de modificação do registo e o valor do antigo registo. Um exemplo deste comportamento está representado na Figura 7. Neste caso as *queries* orientadas para a consulta desta informação são bastante simples e a questão da redundância referida no caso do exemplo anterior já não se verifica.

```

{
  "_id": 1,
  "numero_alugueres": 2,
  "dimensao_filme": [
    {
      "data_insercao": "19-12-2019",
      "titulo": "Interstellar",
      "classificacao": 8.8,
      "classificacao_historia": [
        {
          "data_insercao": "18-12-2019",
          "valor": 9.8
        }
      ]
    },
    {
      "titulo": "Ford v Ferrari",
      "classificacao": 8.3,
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ],
  "dimensao_cliente": {
    "nome": "João Oliveira",
    "morada": "Rua dos Chãos, Braga"
  },
  "dimensao_loja": {
    "morada": "Avenida da Liberdade, Braga"
  },
  "dimensao_staff": {
    "nome": "Maria Ferreira",
    "morada": "Rua de Barros, Braga"
  },
  "dimensao_calendario": {
    "data": "18-12-2019",
    "dia_da_semana": "quarta-feira",
    "semestre": 2
  }
}

```

Figura 7: Exemplo representativo de uma outra opção de salvar história num modelo DFL.

Preto: Timestamp da última inserção

Cinza: Par chave-valor com variação

Verde: Subdocumento com valores históricos relativos ao atributo com variação

No segundo modelo (DSH) podemos realizar uma outra abordagem: fazer a implementação de mini-dimensões nos documentos. Esta abordagem implica a criação de uma nova coleção para cada uma das dimensões e outra para os factos. Nesta última coleção ficarão todos os documentos que contenham informação relativa aos valores antigos dos atributos modificados e a respetiva data de inserção. Nos documentos atuais far-se-á a inserção de um novo par chave-valor, cujo valor será o identificador do documento correspondente à mini-dimensão que foi criada. Esta mini-dimensão conterá os valores anteriores do par chave-valor que foi modificado. Apesar de esta abordagem ser uma solução viável e que evita redundância de informação, é, porém, uma solução que envolve mais funções de agregação no modelo. Por este motivo esta solução poderá não ser uma abordagem viável a adotar num modelo orientado a documentos pois poderia ser prejudicial em termos de *performance*. Contudo, isso dependerá sempre do problema para o qual o modelo estará a ser implementado.

3.4 Modelação Dimensional

A modelação dimensional pode ser caracterizada como uma técnica que permite a concetualização e estruturação de um sistema dimensional de dados. Estes sistemas são especificamente criados para proporcionar uma leitura e análise da informação armazenada num *data warehouse*, de uma forma rápida e eficiente. Nesta secção vão ser abordados os principais conceitos apresentados por [Kimball e Caserta, 2011], que adaptámos de forma a poderem ser aplicados a sistemas orientados a documentos. Além disso, apresentamos, também, alguns exemplos aplicados a um caso de estudo. De realçar que todos os exemplos que irão ser apresentados seguem uma estrutura em estrela [Kimball e Ross, 2013] e, no caso de modelos orientados a documentos, uma estrutura do género DFL (Secção 3.3).

3.4.1 Estrelas e Flocos-de-neve

No domínio da modelação dimensional é usual referirem-se dois tipos de esquemas: esquemas em estrela e esquemas em floco-de-neve. O esquema em estrela, tal como a sua própria designação indica, caracteriza-se por apresentar uma estrutura em estrela, na qual a tabela de factos encontra-se posicionada ao centro, relacionando-se com as diversas dimensões definidas no sistema que estão localizadas na sua periferia. Neste tipo de esquema, as dimensões estão usualmente desnormalizadas, de forma a conferirem uma melhor *performance* das *queries* lançadas sobre a sua estrutura. Mas, em contrapartida, este tipo de esquema promove uma maior redundância de informação. Na Figura 8 podemos ver um exemplo de um modelo típico de um esquema em estrela.

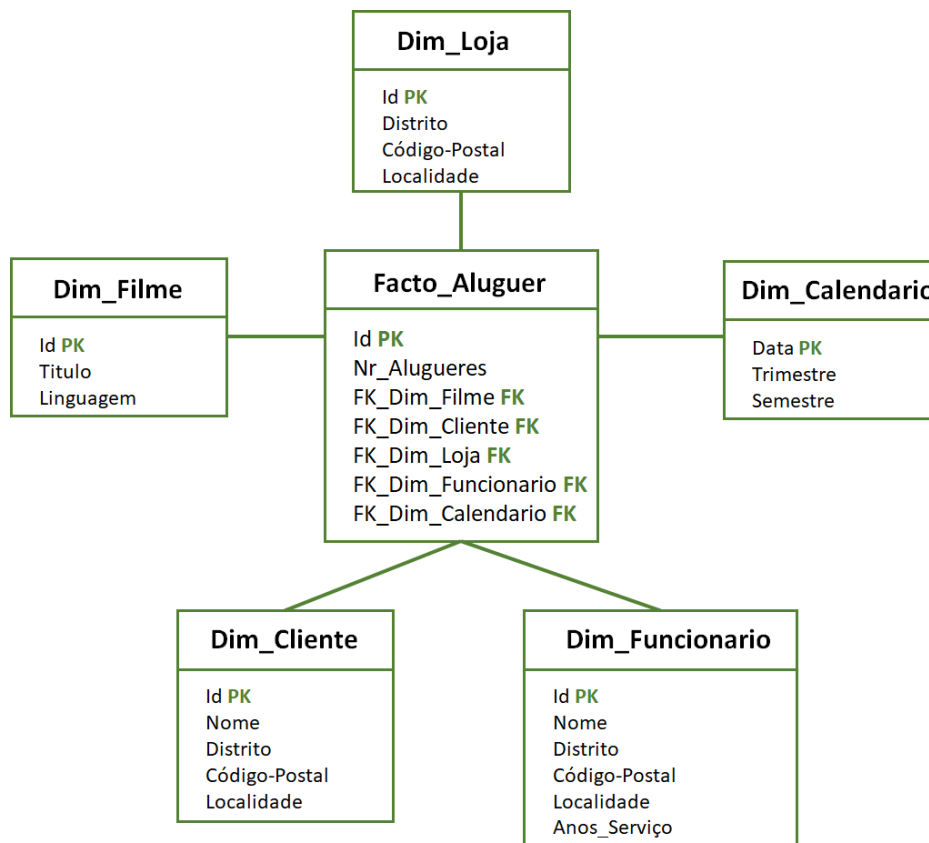


Figura 8: Um exemplo de um esquema em estrela.

Em modelos orientados a documentos a adaptação literal do esquema em estrela seria equivalente a uma estrutura **DSH** (Secção 3.3) (Figuras 5 e 4). Porém, como foi referido, este tipo de estrutura não potencia um bom desempenho ao sistema, uma vez que à medida que o sistema vai escalando, também o número de agregações e o número de dados agregados vai aumentando.

Um esquema em floco-de-neve pode ser composto por uma ou mais tabelas de factos, relacionada com as suas respetivas dimensões. Essencialmente, este esquema difere do esquema em estrela em termos da normalização das dimensões envolvidas. Assim, neste modelo podemos ver dimensões relacionadas com outras dimensões, o que não acontece no caso do esquema em estrela.

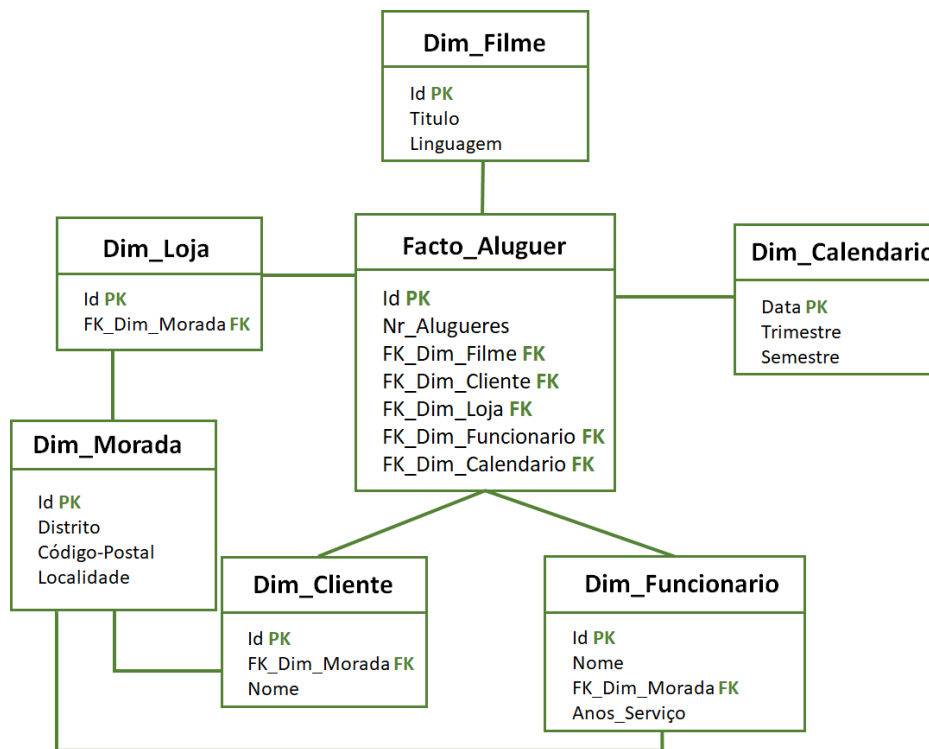


Figura 9: Um exemplo de um esquema em floco-neve.

Como se pode observar na Figura 9, os atributos que caracterizam uma "morada", deram origem a uma nova dimensão. Isto permite diminuir substancialmente a redundância de dados, já que agora uma "morada" apenas ocorre uma única vez, não se repetindo tal como acontecia anteriormente. Em contrapartida, isto vai fazer com que o desempenho das *queries* seja claramente prejudicado, pois vai implicar a utilização de mais operações de junção para obter a mesma informação que anteriormente estava disposta num esquema em estrela.

Se num modelo de dados relacional este tipo de esquema já tem impacto negativo no tempo de execução das *queries*, então num modelo de dados não relacional este impacto vai ser ainda mais acentuado, visto que os sistemas que acolhem estes modelos não foram construídos para lidar com operações de junção entre os dados. Desta forma, em sistemas orientados a documentos, uma possível melhor solução passará por adotar um modelo o mais desnormalizado possível, evitando assim o número de agregações necessárias para obter a informação. Na prática, isto significa materializar *a priori* os resultados de eventuais operações de junção que poderão ser realizadas.

3.4.2 Hierarquias e Agregações

As hierarquias definem a forma como os valores de um ou mais atributos de uma dimensão podem ser agregados (ou desagregados). A sua implementação é um processo bastante exigente, em

particular na construção das estruturas de dados que as suportam. Os relacionamentos de dados que as hierarquias usualmente requerem fazem com que a *performance* das *queries* sobre as tabelas de dimensão nas quais estão definidas se degrade naturalmente, devido ao tipo e número de cálculos que implicam. Assim, é necessário que a definição das hierarquias seja feita de forma metódica e bem suportada, tendo em consideração os requisitos do próprio sistema, para que se possa inferir a quantidade de operações de transformação de dados que serão necessárias para satisfazer a aplicação das hierarquias [Kimball e Ross, 2013].

Na Figura 10 podemos ver um exemplo de uma hierarquia para uma dimensão “calendário”. No diagrama apresentado, segundo a notação de [Golfarelli et al., 1998], os círculos representam os atributos da dimensão e as ligações entre os círculos representam a relação hierárquica entre elas (N-1).

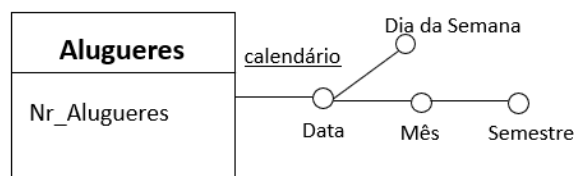


Figura 10: Exemplo de um modelo dimensional integrando uma dimensão “calendário”.

Através do diagrama é possível verificar que o atributo “Data” representa a granularidade mais fina dos elementos de dados desta dimensão. Para além deste atributo, não é possível obter informação temporal mais detalhada. Em oposição, neste caso, o atributo “Semestre” representa o nível de maior agregação em termos de granularidade.

Em sistemas orientados a documentos a definição de uma hierarquia também pode ser implementada. Contudo as agregações de dados têm que ser pensadas *a priori*, cuidadosamente, de forma a que a sua implementação não implique grandes custos em termos de *performance*. Uma possível forma de lidar com este tipo de situação é optar pela implementação de processos de materialização de vistas. Esta opção ajuda a evitar que grandes volumes de dados sejam agregados e, consequentemente, melhorar a agilidade do próprio sistema.

3.5 Tipos de Tabelas de Dimensão

Nesta secção vão ser abordados os diferentes tipos de dimensões com variação com ou sem história tendo como referência [Kimball e Ross, 2013]. Numa fase final, vão ser ainda apresentados outros tipos de dimensões também utilizados na modelação dimensional, como por exemplo, dimensões partilhadas, degeneradas, etc. Para cada diferente tipo apresentado, será feita uma análise comparativa da sua aplicação em dois modelos de dados diferentes, mais concretamente relacionais e orientados a documentos.

3.5.1 Dimensões com variação – *slowly changing dimensions*

Uma das características mais importantes de um sistema de apoio à decisão é a capacidade que este tem para manter a sua informação ao longo do tempo. Porém, uma parte da informação armazenada num *data warehouse* costuma variar ao longo do tempo, podendo ser modificada. Usualmente, este tipo de variação ocorre ao nível das tabelas de dimensão, em alguns dos seus atributos. Estes atributos podem variar de uma forma imprevisível e alheia à dimensão temporal especificada para o *data warehouse*. Desta forma, as dimensões sujeitas a esta variação de informação esporádica, necessitam de ser avaliadas e estruturadas adequadamente, de forma a poderem, de algum modo, suportar as variações que possam ocorrer nos dados das dimensões envolvidas. Costuma-se designar este tipo de dimensões por dimensões com variação. De acordo com [Kimball e Ross, 2013], as dimensões com variação podem dividir-se em diversos tipos. Vejamos cada um deles.

- **Tipo o - Manutenção do valor original**

Apesar de dimensões do tipo o terem sido classificadas por [Kimball e Ross, 2013] como dimensões com variação, esta designação pode ser vista como contraditória. O motivo que me leva a fazer tal afirmação, é que dimensões do tipo o definem-se como sendo dimensões em que o valor do atributo da dimensão nunca é alterado. Ou seja, caso haja uma atualização do valor na fonte de dados do atributo, o valor do atributo correspondente na dimensão do *data warehouse* não vai ser atualizado.

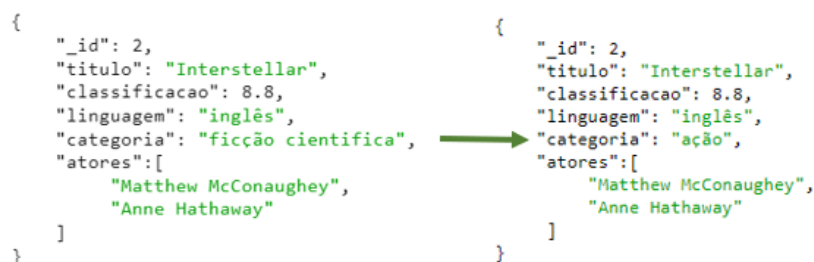


Figura 11: Atualização do valor do par chave-valor "categoria" na fonte de dados.

```

{
  "_id": 2,
  "numero_alugueres": 1,
  "dimensao_filme": [
    {
      "titulo": "Interstellar",
      "classificacao": 8.8,
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ]
}
  
```

Figura 12: Valor do par chave-valor com variação tipo o mantém-se inalterado no *data warehouse*.

Na Figura 11 podemos ver uma representação da atualização do valor do par chave-valor “Categoria”, relativa ao filme “Interstellar”, enquanto que na Figura 12 podemos ver o estado do registo na dimensão para o filme correspondente. Como é possível verificar, o par chave-valor com variação do tipo 0 não sofre qualquer alteração, apesar de ter acontecido uma atualização do respetivo valor na fonte de dados.

• Tipo 1 - Substituição dos Valores Antigos

Nas dimensões com variação do tipo 1 os valores dos atributos envolvidos num processo de modificação de dados são reescritos. Quando alguns elementos de dados da tabela de dimensão são modificados não se preservam os valores anteriores. Estes são, simplesmente, apagados, sendo os novos valores colocados nos seus lugares. Assim, os valores dos atributos com variação (que podem sofrer alterações ao longo do tempo) são sempre os mais recentes, não se mantendo qualquer tipo de histórico. De seguida apresenta-se um pequeno exemplo demonstrativo deste tipo de dimensão. O exemplo escolhido refere-se a um *data warehouse* orientado a documentos. Porém, a lógica de trabalho e de análise será idêntica à que se utilizássemos uma base de dados relacional. Em particular, neste caso, o documento contendo o par chave-valor que se pretende alterar é atualizado, provocando a perda dos valores anteriores. De notar que, enquanto num sistema relacional apenas se atualiza um registo (correspondente a uma linha de uma tabela), no caso de um sistema orientado a documentos poderá não ser tão simples, uma vez que isso dependerá da estrutura do *data warehouse* que esteja envolvida. Por exemplo, se o par chave-valor a atualizar estiver repetido em diferentes documentos, todos esses valores terão que ser atualizados em todos os documentos, de forma a manter a consistência da informação armazenada no *data warehouse*.

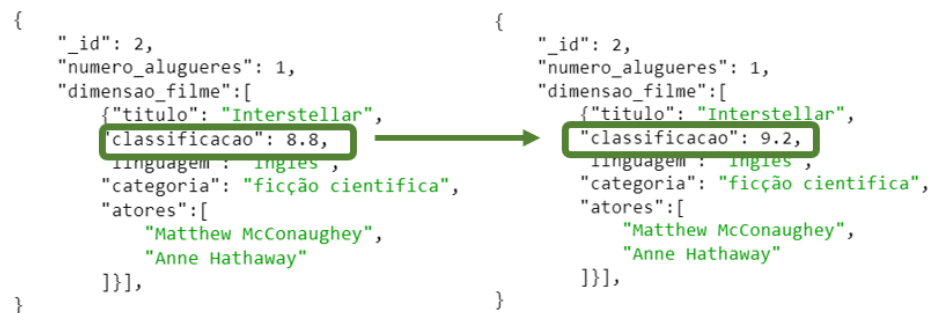


Figura 13: Exemplo demonstrativo de uma dimensão com variação - tipo 1.

• Tipo 2 - Adição de Novos Registos

Nas dimensões com variação do tipo 2 os documentos que sejam afetados por um processo de modificação de dados são mantidos no *data warehouse*, colocando-se os novos valores em novos documentos na dimensão afetada. Além disso, a estrutura destes documentos incluem três novos pares chave-valor, que são utilizados para caracterizar o estado de cada um dos registos da dimensão. Um dos pares adicionado define o estado do registo, por exemplo, se ele está ‘ativo’ ou ‘inativo’, enquanto que os restantes dois definem o intervalo de tempo no qual o valor foi válido. Para isso é

utilizado um par chave-valor no qual é guardado o valor da data na qual o documento foi inserido e um outro que acolhe a data na qual esse documento deverá expirar, isto é, o momento no qual o documento passará ao estado 'inativo' [Wancerz, 2013]. Se estivéssemos a utilizar uma base de dados relacional, este tipo de alteração implicaria a adição de uma nova linha à tabela de dimensão envolvida.

Como foi abordado anteriormente, na Secção 3.4.1, em modelos relacionais a tabela de factos contém as chaves estrangeiras que referenciam os registos das diversas dimensões integradas na *data warehouse*. Como podemos ver nas Tabelas 3 e 4, quando é inserido um novo registo, este vai ter uma nova chave primária, tornando o registo anterior obsoleto. Desta forma, será espectável que os registos na tabela de factos, no intervalo de datas em que foi inserido o registo obsoleto, até à inserção do novo registo, tenham a chave estrangeira que referencia o registo antigo. Por sua vez, os registos posteriores passam a ter a chave estrangeira que referencia o novo registo. Desta forma é possível manter um histórico dos registos desse atributo ao longo do tempo, garantindo que a sua informação seja mantida de forma fidedigna. Na Tabela 4 podemos ver como esse histórico é mantido.

ID	Título	Classificação	Linguagem	Categoria	Data Inserção	Data Ex-pira	Estado
2	Interstellar	8.8	Inglês	Ficção Científica	2020-06-01	2020-08-01	Ativo

Tabela 3: Tabela antes da inserção do novo registo.

ID	Título	Classificação	Linguagem	Categoria	Data Inserção	Data Ex-pira	Estado
2	Interstellar	8.8	Inglês	Ficção Científica	2020-06-01	2020-10-01	Inativo
44	Interstellar	9.2	Inglês	Ficção Científica	2020-10-02	2100-08-01	Ativo

Tabela 4: Tabela após a inserção do novo registo.

Em sistemas de bases de dados não relacionais, em particular em sistemas de bases de dados orientados a documentos, a abordagem é bastante semelhante, diferindo apenas na forma de estruturação do modelo de dados em questão. Neste caso este tipo preocupação, envolvendo relacionamentos entre dimensões e factos, deixa de existir. Cada documento possui toda a informação das dimensões com que se relaciona, bem como das medidas do registo em causa. Desta forma, a implementação de dimensões com variação tipo 2 envolve apenas a inserção de um novo documento com a nova informação atualizada. Contudo, ambos os documentos, o antigo e o novo que será inserido, necessitam dos três pares chave-valor, para que seja possível controlar o estado da dimensão no documento e manter o histórico da dimensão.

Como é possível verificar através das Figuras 14 e 15, após a atualização do valor relativo à chave “*classificacao*”, o valor associado à data de expiração (“*data_expira*”) do documento é atualizado e o valor referente à chave “*estado*” passa a ‘inativo’. Desta forma, é possível identificar que naquele intervalo de tempo, o valor referente à “*classificacao*” era ‘8.8’ e, após a atualização, ficou com o valor ‘9.2’, a partir da data ‘10-01-2020’ até atingir a data de vencimento ou até sofrer uma nova atualização. Note-se que, nas figuras referidas, os novos pares chave-valor, que identificam o estado da dimensão no documento, estão assinalados com cor verde, enquanto que os pares chave-valor com variação com a cor cinza.

```
{
  "_id": 2,
  "numero_alugueres": 1,
  "dimensao_filme": [
    {
      "titulo": "Interstellar",
      "classificacao": 8.8,
      "data_insercao": "19-12-2019",
      "data_expira": "18-10-2100",
      "estado": "ativo",
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ]
}
```

Figura 14: Documento antes da inserção de um novo documento.

<pre>{ "_id": 2, "numero_alugueres": 1, "dimensao_filme": [{ "titulo": "Interstellar", "classificacao": 8.8, "data_insercao": "19-12-2019", "data_expira": "09-01-2020", "estado": "inativo", "linguagem": "inglês", "categoria": "ficção científica", "atores": ["Matthew McConaughey", "Anne Hathaway"] }] }</pre> <p>(a) Documento antigo.</p>	<pre>{ "_id": 3, "numero_alugueres": 1, "dimensao_filme": [{ "titulo": "Interstellar", "classificacao": 9.2, "data_insercao": "10-01-2020", "data_expira": "12-02-2100", "estado": "ativo", "linguagem": "inglês", "categoria": "ficção científica", "atores": ["Matthew McConaughey", "Anne Hathaway"] }] }</pre> <p>(b) Documento com a nova informação.</p>
---	--

Figura 15: Documentos atualizados com a nova informação.

Verde: Pares chave-valor que identificam o estado da dimensão no documento
Cinza: Par chave-valor com variação

• Tipo 3 - Adição de Novos Atributos

As dimensões com variação do tipo 3 implicam a adição de um ou mais atributos à estrutura da própria dimensão. Os atributos adicionados serão os responsáveis pelo armazenamento dos valores que a dimensão continha antes da realização da atualização de dados. Comparando com o tipo 2, apresentado anteriormente, este tipo de variação não acarreta um grande crescimento de registos na tabela de dimensão, uma vez que os dados a preservar são colocados nos atributos que foram

criados especificamente para o efeito. Porém este tipo também não tem as vantagens que o tipo 2 oferece como, por exemplo, preservar um histórico de informação ao longo do tempo praticamente sem limitações. As dimensões com variação do tipo 3 apenas oferecem uma parte da história, isto é, apenas permitem armazenar num novo atributo de história os valores correspondentes ao valor que antecedeu a atualização do atributo que foi atualizado. Assim, quando o registo é atualizado novamente, o valor do atributo responsável por salvar o valor do atributo correspondente no registo antigo será substituído por outro valor, perdendo-se, assim, a informação que foi sobrescrita.

De seguida, apresenta-se um pequeno exemplo para vermos como este tipo de dimensão com variação funciona. Tal como anteriormente, o tipo de abordagem é semelhante àquela que é adotada quando lidamos com sistema de dados relacionais, nos quais a implementação deste tipo de dimensão implica a adição de um ou mais atributos à tabela de dimensão, conforme necessário. No caso de um sistema baseado em documentos adiciona-se um ou mais pares chave-valor ao documento afetado pela operação de modificação de dados.

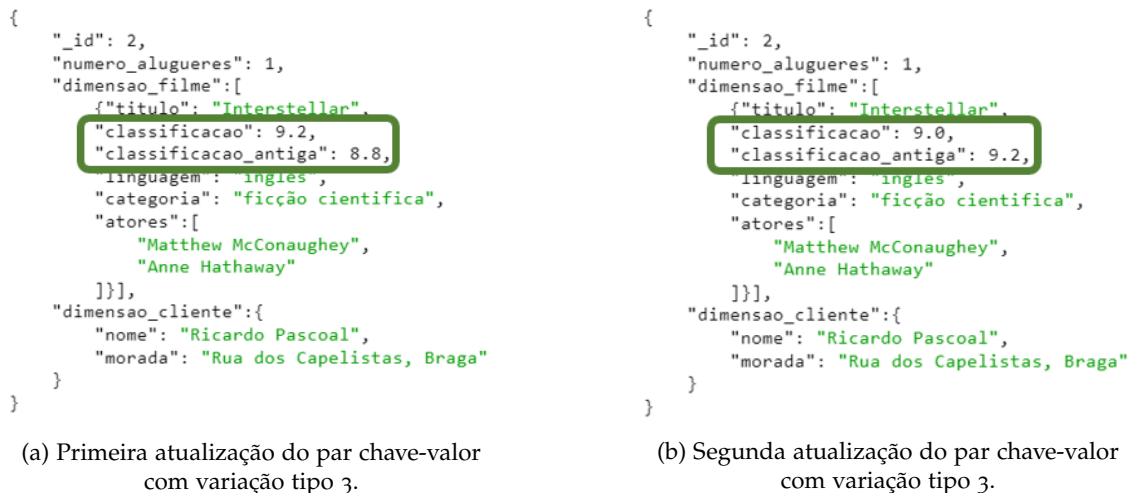


Figura 16: Exemplo demonstrativo de uma dimensão com variação - tipo 3.
 Verde: Pares chave-valor que contêm o valor atual e antigo da "classificacao"

Na Figura 16a podemos ver o resultado de uma primeira atualização do par chave-valor "classificacao: 8.8" com o novo valor '9.2' e, o valor referente à chave "classificacao_antiga" contém o valor anterior do par chave-valor "classificacao: 8.8". Depois na Figura 16b, podemos ver o resultado de uma nova atualização do mesmo par chave-valor. Como se pode observar, o valor que foi anteriormente preservado no exemplo da Figura 16a foi, agora, perdido, tendo sido substituído pelo valor que precedeu esta atualização. Note-se que, na figura referida, o par chave-valor com variação do tipo 3 assim como novo par chave-valor adicionado estão assinalados com cor verde. Como se pode constatar, este exemplo é bastante simples, uma vez que afeta, apenas, de forma isolada, a dimensão "dimensao_filme", na qual foi identificado o atributo com o tipo de variação referido.

Porém, se considerarmos agora também a dimensão "dimensao_cliente" (Figura 16), o processo de atualização da dimensão fica um pouco mais complicado. Se presumirmos que existem 10 clientes

que alugaram o filme “*Interstellar*”, o que envolve 10 documentos diferentes, todos eles contendo o par chave-valor com variação “*classificacao*”, veremos que o processo de atualização da dimensão vai exigir tantas operações, quanto o número de clientes que alugaram esse mesmo filme. No exemplo que utilizamos, em concreto, vai ser necessário atualizar a informação de 10 documentos diferentes com a mesma informação. Torna-se então perceptível que quanto maior for o número de documentos contendo um par chave-valor com variação, maior será o número de operações que vão ser necessárias fazer para manter todos os documentos atualizados. Este problema surge da redundância de dados que existe entre os documentos presentes no *data warehouse* quando este tipo de variação (tipo 3) é aplicado.

- **Tipo 4 - Adicionar uma Mini-Dimensão**

Dimensões com variação do tipo 4 são uma outra forma de dar resposta a um aumento significativo do número de alterações de um dado atributo numa tabela de dimensão. Neste tipo de situação, no qual o número de alterações poderá ser bastante grande, uma abordagem do tipo 2 poderá não conseguir responder de forma eficiente às necessidades do sistema, uma vez que o número elevado de alterações a registar fará com que o número de linhas, no caso de um sistema relacional, ou o número de documentos, no caso de um sistema orientado a documentos, aumente drasticamente, causando perdas de *performance* consideráveis.

Uma possível solução para esta situação é a criação de uma mini-dimensão. A mini-dimensão pode ser vista como uma fração de uma dimensão com variação, na qual vão estar definidos os atributos da dimensão que estão sujeitos a alterações com grande frequência. Assim, na mini-dimensão vão ser definidas todas as combinações possíveis de valores desses atributos, o que evitará o crescimento de registos tal com referido anteriormente. Para exemplificar esta situação, de seguida, na Figura 17, apresenta-se um exemplo prático da aplicação de uma mini-dimensão num modelo de dados relacional.

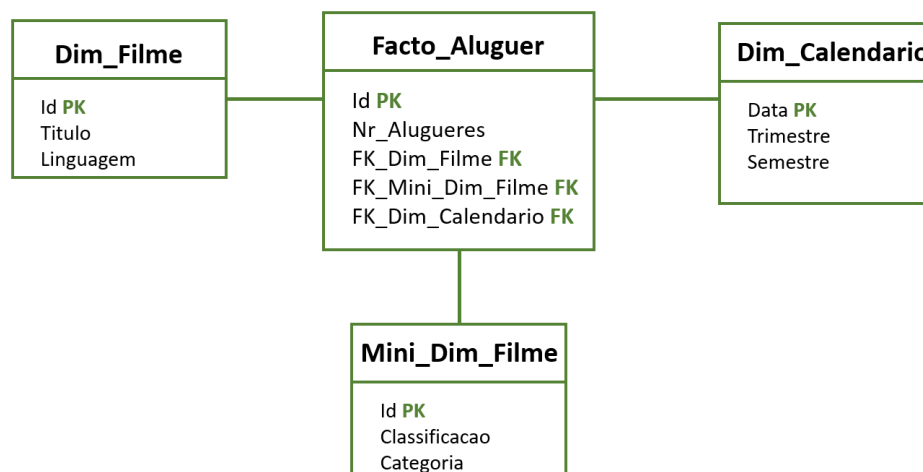


Figura 17: Modelo exemplificativo contendo uma dimensão com variação tipo 4.

Neste exemplo foi considerado que os atributos *"Classificacao"* e *"Categoria"* são atributos cujos valores variam ao longo do tempo. Na tabela de factos foi adicionada uma chave estrangeira que permite estabelecer o relacionamento entre a tabela de factos e a tabela referente à mini-dimensão. Desta forma é possível identificar, para uma determinada data, qual o valor dos atributos *"Classificacao"* e *"Categoria"* de um determinado filme.

A implementação de uma mini-dimensão num modelo orientado a documentos, requer a utilização de uma nova coleção na qual será implementada a sua estrutura, na qual os seus documentos conteriam os pares chave-valor classificados com variação do tipo 4.



Figura 18: *Data Mart* exemplificativo relativo a alugueres de filmes.

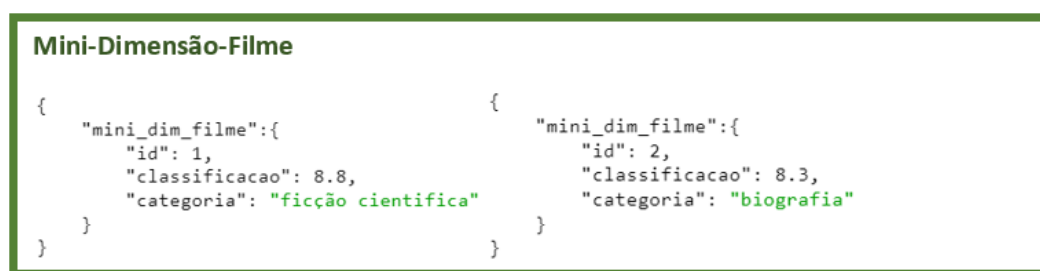


Figura 19: Mini Dimensão relativa à dimensão filmes do *data mart* alugueres.

Na Figura 18 está apresentado um *data mart* relativo a alugueres de filmes que contem dois documentos distintos. Nesses documentos é possível observar um par chave-valor identificado por *"mini_dim_filme"*, no qual é armazenado o valor corresponde ao identificador do documento presente na coleção *"Mini-Dimensão-Filme"*. Apesar deste tipo de abordagem implicar uma menor redundância de dados, poderá não ser uma boa opção quando se modela um *data warehouse*

orientado a documentos. As bases de dados não relacionais têm um fraco desempenho quando a execução de uma dada *query* exige um grande volume de agregações. Isto significa que, com o crescimento do volume de dados do *data warehouse* a *performance* do sistema ir-se-á deteriorar significativamente à medida que os dados disponíveis vão aumentando, o que sugere que este tipo de abordagem possa não ser uma alternativa muito viável. Em contrapartida, as bases de dados não relacionais conseguem lidar com grandes volumes de dados, em especial quando não existem relacionamentos entre as estruturas de dados que compõem o sistema. Assim, pensamos que existem alternativas melhores do que a apresentada.

- **Tipo 5 - Adição de uma Mini-Dimensão e um Atributo (chave) Tipo 1**

As dimensões com variação do tipo 5 podem ser vistas como uma junção das aplicações das dimensões com variação do tipo 4 e do tipo 1. Da mesma forma que as dimensões com variação do tipo 4 possuem uma mini-dimensão, o tipo 5 também a inclui. A diferença que existe é que nas dimensões com variação do tipo 4 a mini-dimensão encontra-se diretamente relacionada com a tabela de factos do sistema, enquanto que em dimensões com variação tipo 5 esta encontra-se relacionada com a dimensão primária, a partir da qual foi criada a mini-dimensão. Na dimensão primária terá que estar presente um atributo que corresponde a um identificador único do registo da mini-dimensão. Este atributo deverá ser tipo 1 (Secção 3.5.1), e o seu valor deverá corresponder ao valor atual do atributo que lhe corresponde na mini-dimensão. Na Figura 20 está representada esta modelação quando aplicada a sistemas relacionais.

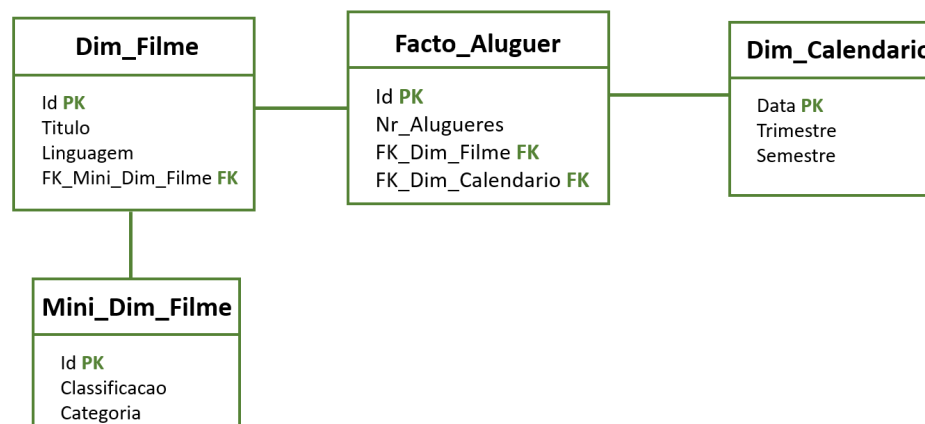


Figura 20: Um modelo contendo uma dimensão com variação tipo 5.

Tendo em conta um modelo orientado a documentos, a implementação de uma dimensão com variação do tipo 5, em termos práticos, não implica uma grande alteração quando fazemos a sua comparação com a aplicação de uma dimensão com variação tipo 4 (Secção 3.5.1). Apesar de estruturalmente o par chave-valor mudar de posição, o comportamento em termos de *performance* e *queries* entre os dois tipos vai ser idêntico. Na Figura 21 está representada uma possível implementação de dimensões com variação tipo 5 em modelos orientados a documentos. Como se pode ver quando comparado com o tipo 4, anteriormente apresentado, apenas houve uma

deslocação do par chave-valor *"mini_dim_filme"* para dentro do subdocumento relativo à dimensão filme.



Figura 21: Data Mart exemplificativo relativo a alugueres de filmes.

- **Tipo 6 - Adição de Atributos com variação do Tipo 1 a uma dimensão com variação do Tipo 2**

As dimensões com variação tipo 6 podem ser vistas como uma combinação de três dos tipos de dimensão com variação já abordados, nomeadamente as dimensões com variação do tipo 1 (Secção 3.5.1), do tipo 2 (Secção 3.5.1) e do tipo 3 (Secção 3.5.1). Tendo com base um exemplo de um modelo relacional, vejamos como é que este tipo de variação pode ser abordado e implementado.

Assim, se tomarmos em consideração o atributo com variação *"Classificação"* e à semelhança do que acontece em dimensões com variação tipo 2 (Secção 3.5.1), três novos atributos serão adicionados à dimensão cujo propósito será a gestão do estado do registo sujeito a variação. O primeiro atributo será um identificador do estado do registo, variando entre 'Ativo' ou 'Inativo', o segundo será responsável por identificar a data de inserção desse registo e, por último, um atributo que identifica a sua data de vencimento. Desta forma, sempre que uma atualização do atributo for efetuada, um novo registo é adicionado à Tabela 6, tornando possível manter um histórico das alterações que foram realizadas nos valores dos atributos com variação da dimensão. Para além destes três novos atributos, o atributo *"Classificação"* terá que ser "dividido" em dois atributos. Um desses atributos será responsável pelo armazenamento do último valor inserido no atributo com variação *"Classificação Atual"* e o outro que acolherá o valor do atributo na altura em que o registo foi atualizado, sendo este armazenado no atributo *"Classificação História"*. À medida que vai sofrendo alterações, o atributo *"Classificação Atual"* vai sendo atualizado para todos os registos que contêm o

valor obsoleto desse atributo, o que justifica a sua classificação como um atributo com variação do tipo 1 (Secção 3.5.1).

ID	Título	Classificação Atual	Classificação História	...	Data Inserção	Data Ex-pira	Estado
2	Interstellar	8.8	8.8	...	2020-06-01	2120-08-01	Ativo

Tabela 5: Tabela com o registo inicial.

Na Tabela 5 podemos ver o estado do registo antes das atualizações relativas à classificação do dado filme ocorrerem. Veja-se que, inicialmente, os valores dos atributos "*Classificação Atual*" e "*Classificação História*" contêm o mesmo valor. Supondo-se que ocorre uma atualização do valor da classificação do filme para '9.0', então é esperado que ocorram atualizações do registo inicial, em que o estado do mesmo passa a 'Inativo', e será efetuada uma nova inserção na tabela.

ID	Título	Classificação Atual	Classificação História	...	Data Inserção	Data Ex-pira	Estado
2	Interstellar	9.0	8.8	...	2020-06-01	2020-08-01	Inativo
3	Interstellar	9.0	9.0	...	2020-08-02	2120-08-01	Ativo

Tabela 6: Tabela após atualização do atributo classificação.

A realização da atualização do valor relativo à classificação do filme "*Interstellar*" provocou uma série de modificações que são importantes notar. Como era espetável, nesse processo foi adicionada uma nova linha à tabela que continha o atributo sujeito a variação (Tabela 6), o que, por sua vez, provocou uma atualização da data de vencimento e do estado do registo inicial. Neste caso, este último assumiu o valor de 'Inativo'. Para além disto, o valor do atributo "*Classificação Atual*" foi atualizado para o valor atual relativo à classificação do filme "*Interstellar*", passando este de '8.8' para '9.0'. Esta manutenção do valor relativo ao atributo "*Classificação Atual*" permite que para qualquer consulta à Tabela 6, o valor mais recente do atributo sujeito a variação tipo 6, seja obtida de uma forma imediata. Por fim, é também importante realçar que o atributo "*Classificação História*" preserva sempre o valor do atributo naquele intervalo de tempo, o que permite ter conhecimento acerca do valor do mesmo em qualquer altura do tempo.

Num sistema orientado a documentos, a implementação deste tipo de dimensão com variação segue os mesmos passos que a implementação em sistemas relacionais, e a estrutura dos documentos é semelhante àquelas que foram adotadas anteriormente para outros tipos de dimensão com variação – e.g. tipo 2 (Secção 3.5.1) e tipo 3 (Secção 3.5.1). Apesar disso, nestes sistemas esta abordagem não é, de todo, a melhor. Isto porque, tal como aconteceu anteriormente para a dimensão com variação do tipo 3 (Secção 3.5.1), também neste caso é necessário atualizar um grande número de documentos sempre que o par chave-valor com variação é alterado. De forma a podermos entender

um pouco melhor a aplicação duma dimensão com variação do tipo 6 criámos um documento que permite ver a forma como este processo de atualização ocorre. Esse documento está apresentado na Figura 22.

```
{
  "_id": 2,
  "numero_alugueres": 1,
  "dimensao_filme": [
    {
      "titulo": "Interstellar",
      "classificacao_atual": 8.8,
      "classificacao_historia": 8.8,
      "data_insercao": "13-12-2019",
      "data_expira": "18-10-2100",
      "estado": "ativo",
      "linguagem": "ingles",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ],
  "dimensao_cliente": {
    "nome": "Ricardo Pascoal",
    "morada": "Rua dos Capelistas, Braga"
  },
  "dimensao_calendario": {
    "data": "13-12-2019",
    "dia_da_semana": "sexta-feira",
    "semestre": 2
  }
}
```

Figura 22: Documento contendo um atributo com variação tipo 6.

No documento apresentado na Figura 22 é possível observar os pares chave-valor que foram adicionados à dimensão. Estes pares chave-valor controlam o estado do documento, bem como os pares chave-valor referentes ao valor atual e histórico da classificação. Os documentos apresentados na Figura 23a permite ver como a implementação deste tipo de variação poderá ser efetuada para sistemas orientados a documentos. Para tal, é inserido um novo documento com o valor mais recente para a chave *"classificacao_atual"* e um par chave-valor que contem o valor da classificação no intervalo de tempo identificado no documento (Figura 23a). Na Figura 23b, está representado o documento que antecedeu a inserção do documento classificado como 'Activo' (Figura 23a), no qual é possível verificar que o seu estado foi atualizado para 'Inativo' e o valor identificado pela chave *"classificacao_atual"* foi atualizado para o valor mais recente, sendo este o mesmo que o valor do documento considerado 'Ativo'. Assim sendo, sempre que houver uma atualização da classificação, todos os documentos considerados inativos vão sofrer atualização do valor identificado pela chave *"classificacao_atual"*.

```

{
  "_id": 12,
  "numero_alugueres": 1,
  "dimensao_filme": [
    {
      "titulo": "Interstellar"
      "classificacao_atual": 9.0,
      "classificacao_historia": 9.0,
      "data_insercao": "23-12-2019",
      "data_expira": "18-10-2100",
      "estado": "ativo",
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ],
  "dimensao_cliente": {
    "nome": "Maria Ferreira",
    "morada": "Rua dos Chaos, Braga"
  },
  "dimensao_calendario": {
    "data": "23-12-2019",
    "dia_da_semana": "segunda-feira",
    "semestre": 2
  }
}

```

```

{
  "_id": 2,
  "numero_alugueres": 1,
  "dimensao_filme": [
    {
      "titulo": "Interstellar"
      "classificacao_atual": 9.0,
      "classificacao_historia": 8.8,
      "data_insercao": "13-12-2019",
      "data_expira": "22-10-2019",
      "estado": "inativo",
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ],
  "dimensao_cliente": {
    "nome": "Ricardo Pascoal",
    "morada": "Rua dos Capelistas, Braga"
  },
  "dimensao_calendario": {
    "data": "13-12-2019",
    "dia_da_semana": "sexta-feira",
    "semestre": 2
  }
}

```

(a) Novo documento inserido.

(b) Documento com a informação obsoleta.

Figura 23: Documentos contendo atributo com variação tipo 6.

Verde: Pares chave-valor que identificam o estado do documento*Cinza:* Pares chave-valor que armazenam o valor atual e histórico da classificação

A abordagem que realizámos para demonstrar o processo de atualização de uma dimensão com variação do tipo 6 num sistema orientado a documentos não nos parece muito promissora, uma vez que a sua implementação impõe a necessidade de atualizar um grande número de documentos sempre que o par chave-valor alvo é atualizado. Poderíamos melhorar um pouco esta situação utilizando, por exemplo, uma vista materializada, um pouco à semelhança daquilo que foi feito para o caso de uma dimensão com variação do tipo 7 (Secção 3.5.1). Para o caso que apresentámos, esta vista teria a finalidade de guardar um documento por cada filme numa coleção externa ao *data mart*. Esse documento conteria apenas o valor da classificação atual do filme. Desta forma, sempre que um documento seja inserido no *data mart*, basta atualizar um único registo em vez dos múltiplos documentos que contenham referências ao filme em questão.

• Tipo 7 - Dimensões Tipo 1 e Tipo 2

As dimensões com variação do tipo 7 podem ser vistas como uma alternativa a dimensões com variação tipo 6, quando o número de atualizações é bastante frequente, isto é, quando o número de registos escala rapidamente. Numa dimensão do tipo 6, quando as alterações de dados começam a ser muito frequentes, o número de atualizações que afetam o atributo responsável por armazenar o valor atual do atributo com variação também aumenta. Para implementarmos uma dimensão com variação do tipo 7 precisamos de fazer a criação de uma vista contendo o estado atual da dimensão, ou seja, que seja capaz de acolher o valor mais recente do atributo com variação. Para que isto seja

possível é necessário adicionar um identificador único que caracteriza o objeto de dados presente na dimensão.

Este tipo de dimensão pode ser vista como uma combinação de dois outros tipos de dimensão com variação, nomeadamente as dimensões com variação do tipo 2 (Secção 3.5.1) e do tipo 1 (Secção 3.5.1). Assim sendo, é mantida uma dimensão do tipo 2 representada na Tabela 7, e é ainda mantida uma vista desta dimensão, que pode ser vista como uma dimensão do tipo 1, pois é responsável por manter sempre a informação mais atual do atributo.

Num sistema de dados relacional, a implementação de dimensões com variação tipo 7, passa pela adição das chaves estrangeiras que referenciam o registo armazenado na dimensão que contém a informação do atributo na altura temporal identificada na tabela de factos (Tabela 7), e uma outra que referencia o valor mais atual do atributo com variação (Tabela 8).

ID	Filme_ID	Título	Classificação	...	Data Inserção	Data Ex-pira	Estado
2	Foo1	Interstellar	8.8	...	2020-06-01	2020-08-01	Inativo
3	Foo1	Interstellar	9.0	...	2020-08-02	2120-08-01	Ativo

Tabela 7: Dimensão responsável por guardar informação relativa a um dado filme.

Filme_ID	Título	Classificação Atual	...
Foo1	Interstellar	9.0	...

Tabela 8: Dimensão onde é armazenada a informação atual de um dado filme.

Filme_ID	Nr_Alugueres	Dim_Filme_ID	Dim_Date_ID	...
Foo1	1	2	123	...
Foo1	1	3	126	...

Tabela 9: Tabela de factos, responsável por relacionar ambas as dimensões ao longo do tempo.

Como é possível observar no exemplo apresentado na Tabela 9, ao adicionarmos o atributo chave que identifica, neste caso, o filme *"Interstellar"*, é possível obter a informação do valor atual da classificação do filme para qualquer registo a partir da tabela de factos, bem como conhecer o seu valor em qualquer altura do tempo. Vejamos agora como este tipo de dimensões pode ser aplicado a modelos orientados a documentos.

```

{
  "_id": 1,
  "numero_alugueres": 1,
  "dimensao_filme": [
    {
      "filme_id": "F001",
      "titulo": "Interstellar",
      "classificacao": 8.8,
      "data_insercao": "10-09-2019",
      "data_expira": "12-12-2019",
      "estado": "inativo",
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ],
  "dimensao_cliente": {
    "nome": "Ricardo Pascoal",
    "morada": "Rua dos Capelistas, Braga"
  },
  "dimensao_calendario": {
    "data": "10-09-2019",
    "dia_da_semana": "terça-feira",
    "semestre": 2
  }
}

```

```

{
  "_id": 2,
  "numero_alugueres": 1,
  "dimensao_filme": [
    {
      "filme_id": "F001",
      "titulo": "Interstellar",
      "classificacao": 9.0,
      "data_insercao": "13-12-2019",
      "data_expira": "22-10-2100",
      "estado": "ativo",
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ],
  "dimensao_cliente": {
    "nome": "Ricardo Pascoal",
    "morada": "Rua dos Capelistas, Braga"
  },
  "dimensao_calendario": {
    "data": "13-12-2019",
    "dia_da_semana": "sexta-feira",
    "semestre": 2
  }
}

```

(a) Documento com valor do par chave-valor obsoleto. (b) Documento com valor do par chave-valor mais recente.

Figura 24: Documentos que representam o processo aquando a atualização de um valor.

```

{
  "_id": 1,
  "filme_id": "F001",
  "classificacao_atual": "8.8"
}

```

```

{
  "_id": 1,
  "filme_id": "F001",
  "classificacao_atual": "9.0"
}

```

Figura 25: Dimensão onde é armazenado o valor mais recente corresponde ao filme com "filme_id" 'F001'.

No caso de utilizarmos um modelo orientado a documentos para implementarmos um caso de uma dimensão com variação do tipo 7, este pode ser visto como uma otimização do modelo que utilizámos para uma dimensão com variação do tipo 6 (Secção 3.5.1). Porém, no caso de uma dimensão com variação do tipo 7, sempre que um documento novo é inserido é necessário atualizar o par chave-valor "classificacao_atual", caso este seja alterado. O custo desta inserção extra é mínimo, quando comparado com o custo das atualizações que teriam de ser feitas em dimensões com variação do tipo 6 (Secção 3.5.1).

3.5.2 Dimensões Conforme ou Partilhadas (*Conformed Dimensions*)

As dimensões conforme ou partilhas são dimensões que, como o próprio nome indica, podem ser partilhadas por diferentes tabelas de factos ou *data marts*. Uma dimensão conforme desempenha os seus papéis (*roles*) independentemente da tabela de factos ou do *data mart* com que esteja relacionada,

mantendo as suas chaves, atributos e valores comuns [Kimball e Ross,2013]. Este tipo de dimensão permite reutilizar dados evitando ter informação repetida em diferentes *data marts*.

O esquema que está apresentado na Figura 26 revela, de uma forma bastante simples, um exemplo de aplicação de uma dimensão conforme num esquema dimensional relacional.

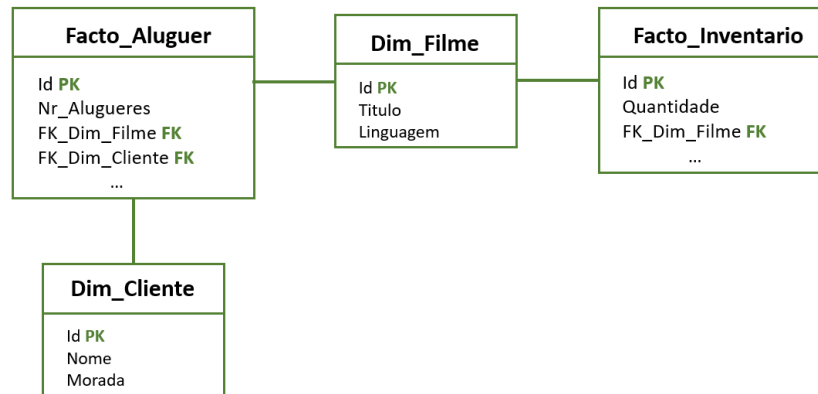


Figura 26: Um esquema dimensional utilizando uma dimensão conforme.

Nessa figura podemos ver a tabela de dimensão “*Dim_Filme*” a ser partilhada por duas tabelas de factos, nomeadamente “*Facto_Aluguer*” e “*Facto_Inventario*”. Tal como referimos anteriormente, esta dimensão tem o mesmo papel, independentemente dos factos envolvidos, que partilham a informação guardada na dimensão referida. Desta forma, torna-se possível relacionar a informação de ambas as tabelas de factos e obter diferentes análises a partir desta dimensão partilhada.

De forma a exemplificar a integração de dimensões conforme em modelos orientados a documentos, elaborámos dois pequenos exemplos de documentos representativos referentes a dois *data marts* diferentes (“*DM-Aluguer*”) e (“*DM-Inventário*”). Esses exemplos estão apresentados nas Figuras 27 e 28.

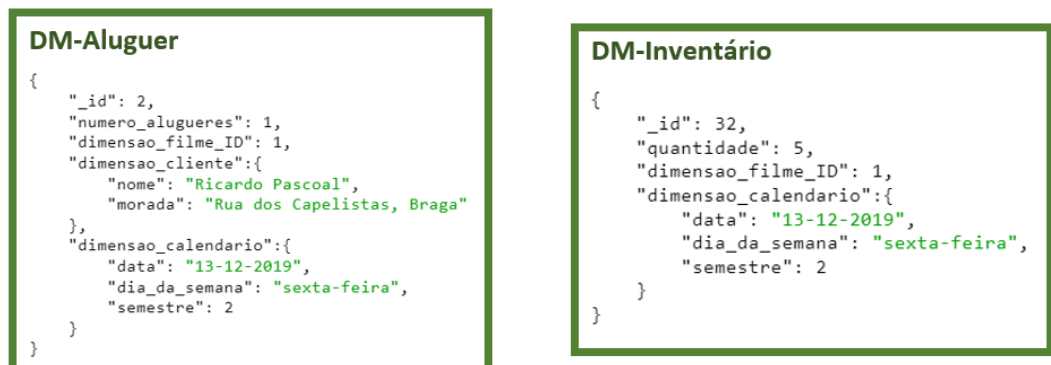


Figura 27: *Data marts* do sistema.

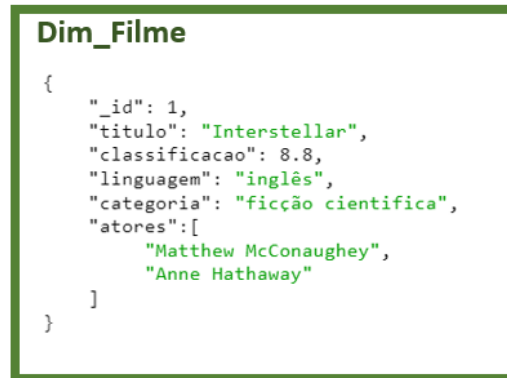


Figura 28: Dimensão conforme "Dim_Filme".

Os dois elementos dos *data marts* referidos (Figura 27) partilham informação de um determinado filme. Essa informação é exatamente a mesma nos dois casos, o que implicou a utilização de uma nova dimensão partilhada (Figura 28). Desta forma, conseguimos reduzir o nível de redundância no sistema. Porém, aquilo que apresentamos nas Figuras 27 e 28 corresponde a uma representação num sistema não relacional e, como tal, a implicação da necessidade de efetuar *queries* mais complexas, que advêm de relacionar a informação dos *data marts* com a dimensão conforme, pode não compensar a eliminação da redundância em termos de *performance*. No caso de o nível de redundância se manter, a retenção de toda a informação relativa ao filme em questão no próprio documento da dimensão tornaria a *query* relativamente simples e rápida. Porém, se optarmos pela implementação apresentada nas Figuras 27 e 28, será necessário utilizar uma função de agregação, que vai prejudicar consideravelmente a *performance* do sistema à medida que o volume de dados aumenta.

3.5.3 Dimensões Degeneradas (*Degenerate Dimensions*)

Uma dimensão é considerada degenerada quando a sua chave ou identificador está presente na estrutura de factos do modelo, mas que não tem uma estrutura física implementada que corresponda a essa dimensão, isto é, que não tem uma tabela específica associada. Para efeito de demonstração, imagine-se um sistema com três dimensões sem variação, nomeadamente "Dim_Filme", "Dim_Cliente" e "Dim_Calendario", uma tabela de factos "Facto_Aluguer" e uma dimensão degenerada "Nr_Fatura", na qual apenas é guardado o número da fatura gerada pelo aluguer. Neste caso, achámos que não seria necessário criar uma dimensão para as faturas, uma vez que toda a informação detalhada da fatura pode ser acedida em outras dimensões como "Dim_Filme", na qual se encontra o filme alugado, ou em "Dim_Cliente", na qual pode ser consultado o cliente que alugou o filme e para qual a fatura se destina. Apesar disso, o número da fatura pode ser importante para responder a algumas questões, como, por exemplo, saber qual o número médio de filmes alugados de uma só vez, ou até mesmo,

para gerar relatórios de faturação [Kimball e Ross,2013]. Este modelo pode ser observado na Figura 29.

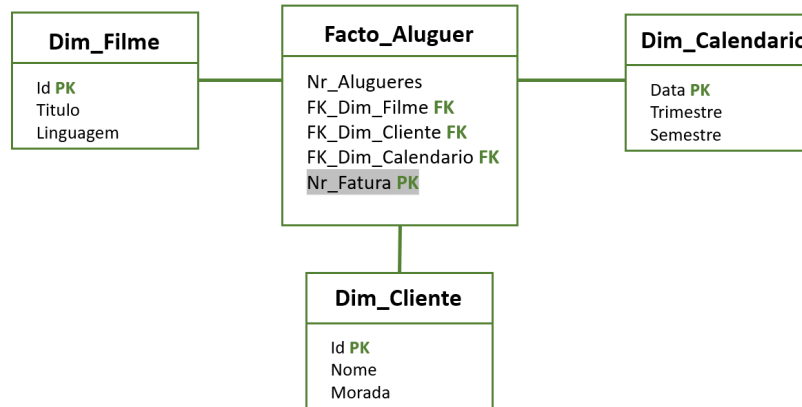


Figura 29: Modelo exemplificativo contendo uma dimensão degenerada.

Cinza: Dimensão Degenerada

No caso de um modelo de dados orientado a documentos, as dimensões degeneradas são adicionadas ao documento que engloba todas as dimensões e medidas do registo. Veja-se o exemplo apresentado na Figura 30.

```
{
  "_id": 2,
  "numero_alugueres": 1,
  "nr_fatura": "222333",
  "dimensao_filme": [
    {
      "titulo": "Interstellar",
      "classificacao": 8.8,
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ],
  "dimensao_cliente": {
    "nome": "Ricardo Pascoal",
    "morada": "Rua dos Capelistas, Braga"
  },
  "dimensao_calendario": {
    "data": "13-12-2019",
    "dia_da_semana": "sexta-feira",
    "semestre": 2
  }
}
```

Figura 30: Um modelo de um documento contendo uma dimensão degenerada.

Dada a estrutura desnormalizada do documento, na qual cada documento contém toda a informação relativa a um registo do *data warehouse*, os dados que possam ter interesse em analisar, por exemplo, em conjunto com o número da fatura, como sejam os casos do nome do filme ou do cliente que efetuou o aluguer, podem ser acedidos de uma forma bastante eficiente, tendo em conta que a

query para essa análise será de conceção bastante simples, uma vez que não implica a realização de qualquer operação de agregação para obter os dados presentes em outras dimensões

3.5.4 Mini-dimensões

As Mini-dimensões são definidas a partir de um subconjunto de atributos que pertencem a uma dimensão na qual os valores de alguns dos seus atributos sofrem frequentemente alterações. Usualmente, este tipo de situação faz com que o número de registos nessa dimensão cresça bastante, quando se pretende manter o seu histórico usando dimensões com variação tipo 2 (Secção 3.5.1). Através da criação de uma mini-dimensão conseguimos “extrair” esses atributos com grande variação, permitindo armazenar as diferentes combinações de valores que estes atributos podem tomar. Na tabela de factos é armazenada a chave estrangeira que identifica o relacionamento com a mini-dimensão [Thornthwaite, 2010]. As mini-dimensões são utilizadas quando optamos por implementar dimensões com variação do tipo 4 (Secção 3.5.1). Na Secção 3.5.1, em particular na Figura 17, podemos ver, com detalhe, a utilidade e a forma como uma mini-dimensão atua nesse tipo de aplicação.

3.5.5 Dimensões Gigantescas (*Huge Dimensions*)

Tal como o próprio nome indica, as dimensões gigantescas são objetos de dados que contêm um elevado número de registos. De acordo com Kimball e Ross, uma dimensão pode ser considerada gigantesca quando possuir mais de 1 milhão de linhas/registos. Porém, este indicador deve ter em consideração o tipo de sistema com que estamos a lidar.

Vejamos, agora, o esquema apresentado na Figura 31. No modelo apresentado, considere-se que os atributos “Classificacao” e “Stock” da dimensão “Dim.Filme” sofrem uma grande variação. Considerando-se o hipotético cenário onde o valor dos mesmos são alterados em média cerca de 100 vezes por dia e onde é necessário preservar o histórico dos registos anteriores, torna-se de fácil percepção que a dimensão “Dim.Filme” vai crescer rapidamente.

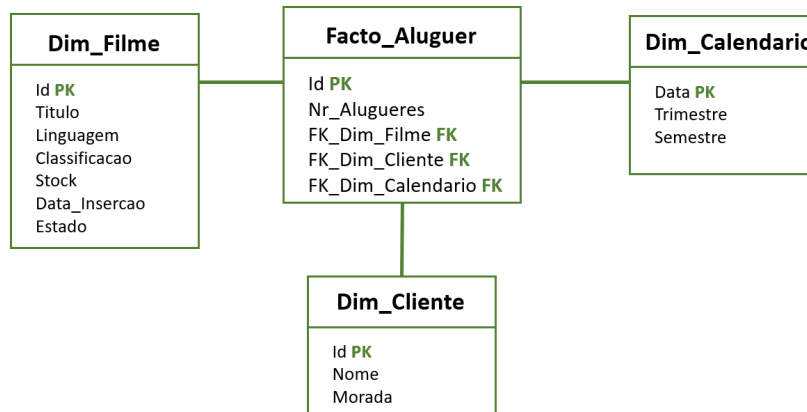


Figura 31: Modelo de dados contendo uma dimensão gigantesca.

Supondo agora que a dimensão “*Dim_Filme*” possui 5 milhões de registos, como é que poderíamos obter uma lista contendo toda a informação relativa aos filmes com classificação superior a um dado valor, bem como os dados dos clientes que efetuaram o aluguer dos mesmos. Num sistema de base de dados relacional, para se obter essa informação, tendo em conta o esquema definido na Figura 31, seria necessário filtrar todos os registos com classificação superior ao valor indicado e, ainda, efetuar duas operações de junção com as tabelas “*Facto_Aluguer*” e “*Dim_Cliente*”.

Para exemplificar de forma prática e quantitativa o exemplo apresentado foi criada uma base dados utilizando o sistema de gestão de bases de dados *MySQL*, utilizando o modelo de dados apresentado na Figura 31. Neste exemplo foram inseridos 5 milhões de registos nas tabelas “*Dim_Filme*”, “*Dim_Cliente*” e na tabela de factos (“*Facto_Aluguer*”). Destes 5 milhões de registos, 3 milhões têm uma classificação com um valor superior a 8 e contêm, pelo menos, um aluguer relativo a um cliente específico. Este exemplo foi também inserido num sistema de gestão de base de dados orientado a documentos, o *MongoDB*. Os documentos inseridos neste sistema contêm o número de alugueres (“*Facto_Aluguer*”), bem como os pares chave-valor relativos às dimensões filme (“*dimensao_filme*”) e cliente (“*dimensao_cliente*”). A estrutura dos documentos criados pode ser observada na Figura 32. Neste exemplo foram adicionados 5 milhões de documentos, contendo a mesma informação que foi inserida no sistema relacional referido anteriormente.

```

{
  "_id": 2,
  "numero_alugueres": 1,
  "dimensao_filme": {
    "titulo": "Interstellar",
    "classificacao": 8.8,
    "linguagem": "inglês",
    "categoria": "ficção científica",
    "data_insercao": "19-12-2019",
    "estado": "ativo"
  },
  "dimensao_cliente": {
    "nome": "Ricardo Pascoal",
    "morada": "Rua dos Capelistas, Braga"
  }
}

```

Figura 32: Documento contendo uma dimensão gigantesca.

Em ambos os casos de demonstração foi avaliado o tempo de execução das *queries* implementadas para obter a informação relativa aos alugueres efetuados de filmes com classificação superior a '8', tendo em consideração que esta informação deveria conter toda a informação existente dos filmes e dos clientes que fizeram esses alugueres. Os resultados obtidos no processo de avaliação estão apresentados na Tabela 10.

	MySQL	MongoDB
Tempo de execução (segundos)	44.34	2.3

Tabela 10: Tabela relativa aos tempos de execução da *query* apresentada.

Como podemos ver, o tempo para obter a mesma informação quando se está a lidar com dimensões gigantescas é bastante maior no sistema de gestão de bases de dados relacional do que no sistema de gestão de bases de dados orientada por documentos. Isto deve-se, essencialmente, à influência dos relacionamentos estabelecidos no modelo relacional nas operações sobre os dados. No sistema não relacional, tais relacionamentos não existem.

3.5.6 Subdimensões (*Outriggers*)

Uma subdimensão ou *outrigger* surge quando uma dimensão é particionada em duas tabelas distintas, em que uma delas prevalece como a tabela de dimensão enquanto que a segunda (subdimensão) torna-se numa nova tabela, relacionando-se ambas através de, pelo menos, uma chave estrangeira. Apesar da criação de subdimensões ser uma possibilidade que permite poupar algum espaço de armazenamento, através da redução da redundância de dados, por exemplo, deverá acontecer com moderação, uma vez que a sua criação adiciona complexidade ao sistema, implicando o estabelecimento (e manutenção) de relacionamentos entre diferentes estruturas de dados, o que

poderá ser custoso em termos de *performance*. Como [Kimball e Ross, 2013] referiram, as subdimensões ou *outriggers* devem ser a exceção e não a regra.



Figura 33: Modelo exemplificativo de uma dimensão contendo informação pessoal.

Consideremos, agora, o exemplo apresentado na Figura 33, que inclui uma tabela de dimensão “*Dim_Cliente*” com sete atributos. Como é possível verificar, a dimensão apresentada contém informação pessoal relativa a um dado cliente. Atributos como “*Nome*” e “*Rua*” são atributos que podem conter um grande leque de valores, podendo ser bastante específicos a cada cliente. Em contrapartida, atributos como “*Distrito*”, “*Código-Postal*” e “*Localidade*”, são atributos em que se espera um grande número de repetições dos mesmos valores, pois são atributos com maior granularidade e como tal é espectável que um grande número de instâncias de ruas tenham os mesmos valores nestes atributos. Assim sendo, tais atributos podem ser vistos como fortes candidatos a incorporar uma subdimensão, sendo removidos da dimensão base, neste caso “*Dim_Cliente*”. Veja-se o caso apresentado na Figura 34.

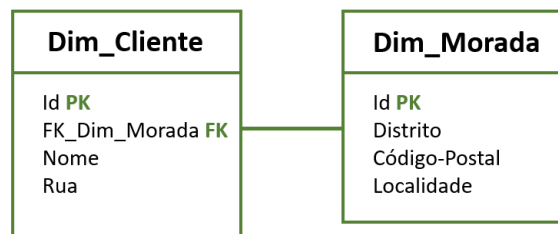


Figura 34: Modelo exemplificativo de uma subdimensão.

3.5.7 Dimensões com Diferentes Papéis (*Role-Playing Dimensions*)

As dimensões com diferentes papéis, ou *role-playing dimensions*, surgem quando uma dimensão pode ter diferentes significados para um mesmo facto, isto é, um mesmo elemento da dimensão providencia-nos uma diferente perspetiva de um dado facto. Para vermos a aplicação de uma dimensão deste tipo, vejamos um exemplo concreto, no qual figura uma dimensão temporal, “*Dim_Calendario*”, relacionada com uma tabela de factos, “*Facto_Aluguer*” (Figura 35). Nesse exemplo, para além da data de aluguer de um dado filme foi adicionada uma segunda data relativa à data na qual deverá ser efetuada a devolução de o dado filme. Assumindo que os dois atri-

butos, "FK_Data_Aluguer" e "FK_Data_Devolução", presentes na tabela de factos, correspondem à mesma dimensão "Dim_Calendarario", pode-se ver que este tipo de associação pode-nos trazer alguns problemas. Por exemplo, quando for efetuada uma junção entre as duas tabelas do esquema apresentado, utilizando-se os dois atributos da dimensão "Dim_Calendarario", "FK_Data_Aluguer" e "FK_Data_Devolução", isto vai ser interpretado como se estas datas fossem iguais, o que não é obviamente, aquilo que se pretende.

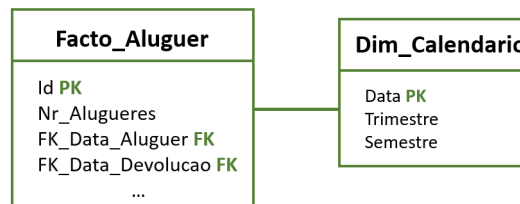


Figura 35: Modelo de uma *role playing dimension*.

Uma solução para a implementação de uma *role-playing dimension* passa pela criação de vistas independentes para os diferentes significados possíveis, a partir da dimensão sujeita a diferentes papéis. O exemplo apresentado na Figura 36 ilustra este tipo de abordagem. Desta forma, as chaves estrangeiras presentes da tabela de factos, vão corresponder a diferentes valores, definidos pelas vistas criadas para cada categorização de data. Num modelo de dados orientado a documentos, este tipo de dimensão é visto como um novo conjunto de pares chave-valor no próprio documento, sendo feita, neste caso, uma distinção literal dos diferentes tipos de significados da dimensão. O exemplo apresentado na Figura 37 ilustra este tipo de modelo.

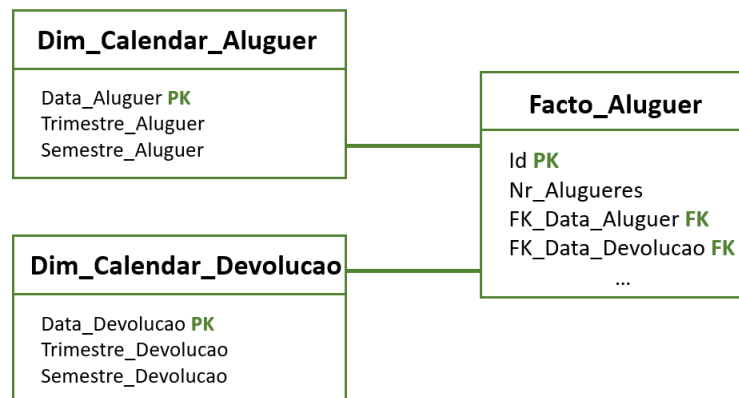


Figura 36: Modelo contendo as vistas relativas aos diferentes significados da dimensão *calendario*.¹

¹ Imagem adaptada de [Kimball e Ross, 2013], capítulo 6 página 170.

```

{
  "_id": 2,
  "numero_alugueres": 1,
  "dimensao_filme": [
    { "titulo": "Interstellar",
      "classificacao": 8.8,
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ],
  "dimensao_cliente": {
    "nome": "Ricardo Pascoal",
    "morada": "Rua dos Capelistas, Braga"
  },
  "dimensao_calendar_aluguer": {
    "data": "13-12-2019",
    "dia_da_semana": "sexta-feira",
    "semestre": 2
  },
  "dimensao_calendar_devolucao": {
    "data": "20-12-2019",
    "dia_da_semana": "sexta-feira",
    "semestre": 2
  }
}

```

Figura 37: Documento contendo os pares chave-valor referentes aos diferentes significados da dimensão calendário.

3.5.8 Dimensões de Controlo (*Junk Dimensions*)

Por vezes, durante a modelação do sistema de *data warehousing* que se pretende implementar, surgem alguns atributos que apenas podem variar dentro de um pequeno intervalo de valores, que devem ser guardados por motivos de criação de relatórios, por exemplo. Para tratar da sua preservação, podemos adotar vários tipos de abordagens, como, por exemplo, adicionar esses atributos a uma tabela de factos ou criar dimensões especificamente definidas para isso, que contenham esses atributos e fazer a sua associação com as correspondentes tabelas de factos.

Vejamos uma dessas abordagens. Consideremos o exemplo apresentado na Figura 38. Nesse exemplo, para além das dimensões já mencionadas anteriormente "*Dim_Filme*", "*Dim_Cliente*" e "*Dim_Calendario*", foram adicionadas outras duas dimensões, correspondentes a dois domínios diferentes. Uma delas é utilizada para guardar informação relativa ao tipo de pagamento. Enquanto que a outra é utilizada para guardar informação relativa a como foi efetuado o aluguer. Considerando que o tipo de pagamento apenas pode conter 3 opções, nomeadamente 'dinheiro', 'cartão de crédito' ou 'paypal', e o tipo de aluguer apenas pode ser 'online' ou 'físico', pode-se assumir que o tamanho das dimensões vai ser bastante reduzido: a primeira dimensão apenas irá conter 3 registos e a segunda dimensão 2 registos. Tendo isto em consideração, este tipo de implementação aumentará a complexidade do modelo de dados do sistema (aumentando o número de relacionamentos necessários) sem grande necessidade.

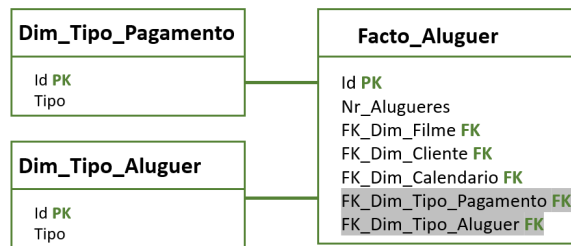


Figura 38: Modelo contendo duas novas dimensões.

Cinza: Novas chaves adicionadas relativas às duas novas dimensões

As dimensões de controlo, ou *junk dimensions*, são uma proposta de solução para este tipo de problema. Segundo [Kimball e Ross,2013], uma *junk dimension* consiste em agrupar indicadores de baixa cardinalidade numa única dimensão. Aplicando esta definição ao exemplo que apresentámos anteriormente (Figura 38), as duas dimensões criadas vão ser “transformadas” numa única dimensão, que conterá os dois tipos de atributos (Figura 39).

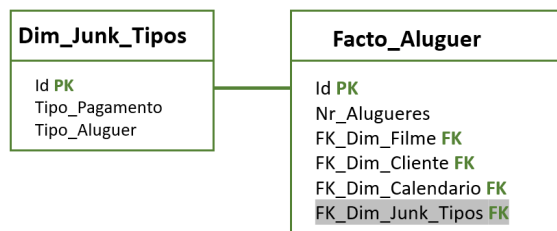


Figura 39: Modelo contendo *junk dimension*.

Cinza: Chave referente à *junk dimension*

Como podemos observar, na Figura 39 temos uma *junk dimension* contendo os dois indicadores de tipos diferentes. Desta forma, “transformou-se” duas dimensões de baixa cardinalidade (3 e 2 registos, respetivamente) numa única dimensão de cardinalidade 6. De notar que este tipo de dimensões apenas deverá ser aplicada quando a sua cardinalidade é relativamente baixa. Caso contrário, será melhor optar por uma abordagem como a que está apresentada na Figura 38, que mantém os registos em diferentes dimensões, reduzindo assim o número total de registos do *data warehouse*.

Se optássemos por um modelo de dados orientado a documentos e se fosse seguida a abordagem que aplicámos sobre um modelo relacional, a implementação de uma dimensão de controlo poderia ser realizada tal como aquela que aplicámos anteriormente para as mini-dimensões (Seção 3.5.1). Porém essa abordagem, quando aplicada num modelo que envolva um grande número de registos, não seria a melhor, pois o número de agregações que seria necessário realizar iria ser muito grande, o que deterioraria a *performance* do sistema. Uma abordagem mais sensata seria a adição dos dois atributos, “*tipo_pagamento*” e “*tipo_aluguer*”, ao próprio documento. Isso não implicaria um aumento da complexidade do sistema, nem maiores custos em termos de *performance*. O modelo de dados orientado a documentos relativo a esta última abordagem pode ser observado na Figura 40.

```

{
  "_id": 2,
  "numero_alugueres": 1,
  "tipo_pagamento": "dinheiro",
  "tipo_aluguer": "físico",
  "dimensao_filme": [
    {
      "titulo": "Interstellar",
      "classificacao": 8.8,
      "linguagem": "inglês",
      "categoria": "ficção científica",
      "atores": [
        "Matthew McConaughey",
        "Anne Hathaway"
      ]
    }
  ],
  "dimensao_cliente": {
    "nome": "Ricardo Pascoal",
    "morada": "Rua dos Capelistas, Braga"
  },
  "dimensao_calendario": {
    "data": "13-12-2019",
    "dia_da_semana": "sexta-feira",
    "semestre": 2
  }
}

```

Figura 40: Modelo exemplificativo de um documento contendo atributos “indicadores”.

3.5.9 Dimensões de Origem Externa

As dimensões de origem externa [Belo, 2013] são dimensões que contêm informação relevante para o domínio do sistema, proveniente de fontes de informação externas. Vejamos o caso da dimensão “*Dim.Filme*”. Esta dimensão poderia ser vista como uma dimensão de origem externa, se os seus dados fossem provenientes de uma base de dados externa ao sistema, como, por exemplo, a *IMDb*, [IMDb, n.d]. Esta abordagem seria perfeitamente exequível. Além disso, acrescentaria valor ao *data warehouse*, pois, permitiria manter a classificação dos filmes sempre atualizada, com base nesta fonte de informação externa.

3.5.10 Chaves de substituição (*surrogate keys*)

As chaves de substituição são atributos chave que não têm significado ao nível do domínio do sistema. Vulgarmente, as chaves de substituição são números inteiros, que à medida que novos registos são adicionados ao sistema vão sendo gerados sequencialmente. Em sistemas relacionais, estas chaves são usadas em muitas aplicações, em detrimento das chaves naturais, para que seja facilitada, por exemplo, a manutenção do histórico de uma dimensão, ou quando se necessita de conciliar informação proveniente de diferentes fontes. De seguida, apresentaremos um exemplo prático da aplicação de chaves de substituição numa dimensão com variação do tipo 2 (Secção 3.5.1).

Consideremos a dimensão filme (“*Dim.Filme*”), contendo os atributos “*titulo*”, “*linguagem*”, “*classificacao*” e “*categoria*”, bem como, uma chave primária que identifica os registos dessa dimensão de forma

única. Tenha-se em conta, também, que esta é uma dimensão com variação do tipo 2, na qual o atributo com variação é "*classificacao*". Vamos assumir que, inicialmente, o sistema contém os registos apresentados na Tabela 11.

ID Nat	Título	Classificação	Linguagem	Categoria	Data Inserção	Data Expira	Estado
Foo1	Interstellar	8.8	Inglês	Ficção Científica	2020-06-01	2100-08-01	Ativo
Foo2	Ford v Ferrari	8.3	Inglês	Biografia	2020-07-01	2100-09-01	Ativo

Tabela 11: Estado inicial da tabela correspondente à dimensão "*Dim.Filme*".

Suponhamos que ocorre nessa tabela de dimensão uma alteração do valor da classificação do filme 'Foo1'. Uma vez que esta é uma dimensão com variação, do tipo 2, o sistema insere na tabela um novo registo contendo o novo valor do atributo "*Classificação*", bem como o valor relativo ao estado do registo e as respetivas datas de inserção e de vencimento. Para além desta inserção, o valor estado do registo considerado obsoleto é atualizado para 'Inativo'.

ID Nat	Título	Classificação	Linguagem	Categoria	Data Inserção	Data Expira	Estado
Foo1	Interstellar	8.8	Inglês	Ficção Científica	2020-06-01	2020-08-01	Inativo
Foo2	Ford v Ferrari	8.3	Inglês	Biografia	2020-07-01	2020-09-01	Ativo
Foo1	Interstellar	9.8	Inglês	Ficção Científica	2020-08-01	2100-08-01	Ativo

Tabela 12: Inserção hipotética de um novo registo para um filme já existente.

Como é possível observar na Tabela 12, com a tentativa de adição do novo registo surge um problema que faz com que a operação seja inviabilizada. Quando a atualização do valor do atributo "*Classificação*" acontece, é necessário adicionar um novo registo à Tabela 11 com o novo valor deste atributo. Porém, dado que a chave primária corresponde ao identificador do filme, neste caso 'Foo1', a inserção do novo registo não é possível, pois já existe um registo identificado com esse valor.

Com a introdução do conceito de chave de substituição o problema é mitigado. Como é possível observar na Tabela 13 foi adicionado um novo atributo à tabela denominado "*SK*" sendo este considerado uma chave de substituição. Desta forma, os registos desta tabela passam a ser identificados por uma chave genérica gerada de forma sequencial, possibilitando a inserção de filmes com o mesmo identificador natural, porém com valores de "*Classificação*" diferentes.

SK	ID Nat	Título	Classificação	Linguagem	Categoria	Data Inserção	Data Expira	Estado
1	F001	Interstellar	8.8	Inglês	Ficção Científica	2020-06-01	2020-08-01	Inativo
2	F002	Ford v Ferrari	8.3	Inglês	Biografia	2020-07-01	2020-09-01	Ativo
3	F001	Interstellar	9.8	Inglês	Ficção Científica	2020-08-02	2100-08-01	Ativo

Tabela 13: Tabela que representa a tentativa de inserção de um novo registo para um filme já existente.

Num modelo de dados orientado a documentos podemos ter o mesmo problema. Se o valor referente à chave primária que identifica o documento for o mesmo que o identificador do filme, quando for necessário inserir um novo documento com o novo valor referente à chave "*Classificação*", não vai ser possível processar tal operação. A solução parte também por adicionar uma chave de substituição. Desta forma, os documentos passam a ser identificados por uma chave alheia ao domínio do sistema e um valor distinto da mesma é gerado à medida que novos documentos são adicionados.

Vejamos, agora, um outro tipo de aplicação das chaves de substituição. Anteriormente referimos que, por vezes, um *data warehouse* é povoado com dados provenientes de diferentes fontes de dados, que, não raras vezes, são heterogêneas. Isso significa que os dados dessas fontes podem estar representados com formatos distintos ou identificarem os mesmos elementos de dados através de valores diferentes. Situações como estas implicam fazer a conciliação desses dados antes de serem colocados no *data warehouse*.

ID	Título	Classificação	Linguagem	Categoria
F001	Interstellar	8.8	Inglês	Ficção Científica

Tabela 14: Tabela relativa à fonte de dados número 1.

ID	Título	Classificação	Linguagem	Categoria
1001	Interstellar	8.8	Inglês	Ficção Científica

Tabela 15: Tabela relativa à fonte de dados número 2.

Tomemos em consideração as duas fontes de dados apresentadas nas Tabelas 14 e 15. Nestas duas tabelas é possível observar que para o mesmo registo cada uma das fontes de dados utiliza identificadores diferentes. Supondo que o objetivo é alimentar um *data warehouse*, nomeadamente a dimensão "*Dim.Filme*" do sistema, é necessário proceder à conciliação dos dados das duas fontes antes da inserção dos mesmos no *data warehouse*. Uma opção é a utilização de uma chave de

substituição. Na Tabela 16 é possível verificar que foi criada uma chave única para identificar o registo presente em ambas as fontes.

ID	Título	Classificação	Linguagem	Categoria
1	Interstellar	8.8	Inglês	Ficção Científica

Tabela 16: Tabela referente à dimensão "Dim_Filme".

DATA WAREHOUSE ORIENTADO A DOCUMENTOS - CASO DE ESTUDO

Durante os últimos três anos, na Universidade do Minho, tem estado a ser desenvolvido um sistema de aferição de conhecimento assistido por computador: o Leonardo [BELO et al., 2019]. Este sistema foi inicialmente projetado para ser utilizado por alunos e professores, com o propósito de complementar o estudo do aluno, disponibilizando exercícios sujeitos a avaliação de um tutor artificial cujo o intuito é adaptar o conteúdo disponibilizado pela plataforma, de acordo com o desempenho do estudante. Por outro lado, o professor será o responsável pela inserção de conhecimento no sistema, tratando, em particular, da gestão dos diversos elementos utilizados nos processos de avaliação e da forma como estes se podem desenvolver. Posteriormente, o professor terá acesso a um conjunto de métricas de avaliação, desempenho e utilização da plataforma pelos seus alunos, podendo, assim, acompanhar a evolução do seu conhecimento num determinado domínio ao longo do tempo. Para suportar as suas atividades, tem incorporado na sua estrutura vários módulos, que incluem serviços de avaliação, de *profiling* ou de gamificação, entre outros.

Neste capítulo iremos abordar a implementação do seu modulo de análise de dados. Este módulo analítico terá como utilizador principal o professor. Porém, não está excluída a opção da inclusão do aluno como utilizador, para que ele possa, a cada momento, acompanhar a evolução do seu processo de aprendizagem. O modulo de análise de dados incorporará na sua estrutura mecanismos especificamente orientados para o armazenamento da informação num *data warehouse* específico, bem como um conjunto de *dashboards* especialmente concebidos para rever, ao longo do tempo, informação relativa ao desempenho global e específico dos alunos e, ainda, informação relativamente à utilização do próprio sistema de aferição, disponibilizando meios de *tracking* das interações realizadas com o sistema.

4.1 Implementação de um Sistema Analítico

A implementação de um sistema analítico para o Leonardo surgiu como resposta a uma necessidade de retirar informação selecionada a partir dos dados gerados e armazenados pelos mecanismos de aferição de conhecimento do sistema, apresentando-a posteriormente de forma organizada e estruturada, num conjunto diversificado de *dashboards*, para que possa servir como suporte a processos de tomada de decisão despoletados pelos utilizadores do sistema. Os *dashboards* foram idealizados para serem utilizados pelos dois principais atores do sistema: o professor e o aluno. Ao professor será permitido aceder a uma vista global dos resultados dos seus alunos, dos domínios de estudo pelos quais é responsável ou pelas questões que foram lançadas pelo sistema e respondidas pelos alunos, entre outras coisas mais. A implementação do sistema analítico não servirá somente para fornecer informação acerca da evolução dos alunos ao professor, mas também manter o aluno a par dos seus resultados, dando-lhe elementos para ele saber como dirigir o seu processo de aprendizagem, o seu estudo.

Apesar de já ter algum tempo de desenvolvimento, o sistema Leonardo é, ainda, um sistema em construção. Como tal, de acordo com os módulos que lhe vão sendo integrados, vai evoluindo gradualmente, não tendo ainda as suas estruturas e serviços num estado considerado estável. Assim, apesar da existência de um esquema de dados definido anteriormente à implementação do sistema analítico, devido à constante atualização do sistema, a informação que está a ser gerada no seu ambiente está também ela sujeita a alterações.

Os dados gerados pelo sistema Leonardo estão a ser armazenados num sistema de gestão de base de dados *MongoDB* [MongoDB, n.d.]. Essencialmente, o *MongoDB* foi escolhido como base de suporte ao sistema de dados do Leonardo pela sua flexibilidade em termos de esquema de dados (*schema less*). Essa característica permite que possamos, a qualquer altura, com a devida justificação, alterar a estrutura dos nossos modelos de dados sem grandes custos em termos de tempo e mesmo de implementação. Em *MongoDB* a informação é armazenada em formato *JSON* que caracteriza os documentos. É nos documentos que esta estrutura contendo a informação do sistema vai ser armazenado no formato chave-valor. Devido à flexibilidade do esquema de dados, é possível adicionar novos pares chave-valor ao sistema em qualquer altura sem grandes custos em termos de reestruturação do mesmo. Isto é, sem ser necessário haver alterações na informação que foi armazenada até ao momento.

Todas estas opções de implementação, em particular as relacionadas com o sistema de dados, foram decididas para que fosse possível criar um sistema analítico tão flexível quanto o sistema de aferição de conhecimento e que tivesse, também, a capacidade de se adaptar, de forma rápida e eficiente, a eventuais mutações do esquema geral da base de dados do sistema. Tendo isso em consideração, dividimos a estrutura do sistema em três componentes. De referir:

- O *Data Warehouse*, que foi implementado em *MongoDB* [MongoDB, n.d] e que é responsável pelo acolhimento e preservação de toda a informação de suporte ao sistema de análise, ao sistema de apoio à decisão.

- O Sistema de *Backend*, que está implementado em *Python*, é o componente do sistema responsável pelo processamento e gestão da informação da base de dados do Leonardo.
- Os *dashboards*, desenvolvidos em *Plotly Dash*, para se poder visualizar e analisar os dados armazenados no *data warehouse* de acordo com as várias perspetivas de utilização do sistema.

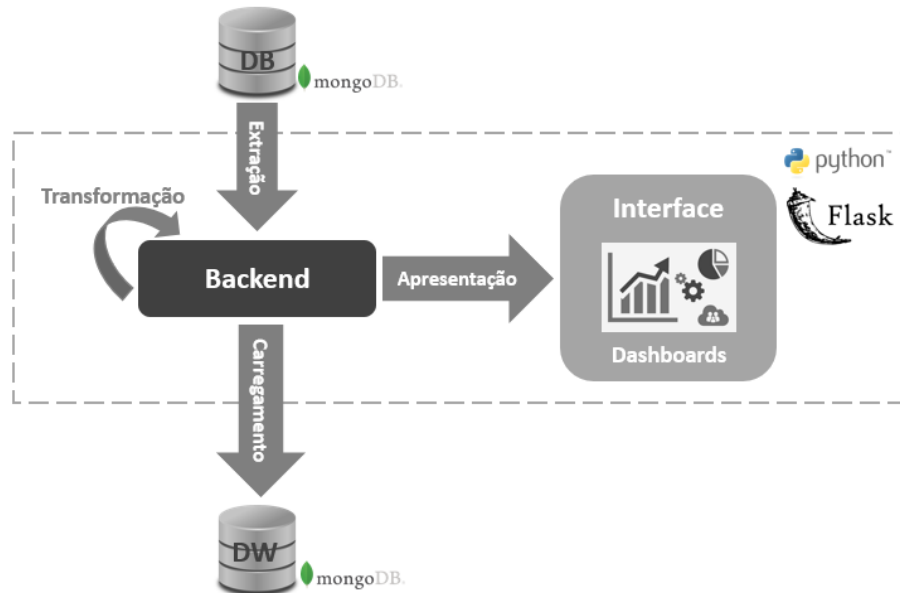


Figura 41: Arquitetura da implementação do sistema.

Na Figura 41 podemos ver uma ilustração da arquitetura geral do sistema de análise de dados. No sistema Leonardo o processo *ETL*, nomeadamente o processo de limpeza e conciliação de dados é todo ele tratado a nível do *backend*, antes da inserção dos dados no *data warehouse*. Desta forma, o sistema que suporta a área de retenção, normalmente implementado em sistemas relacionais para suportar a transformação dos dados, não foi necessário na nossa implementação, sendo a responsabilidade da sua implementação do próprio *backend* do sistema base.

4.2 Estrutura Base do *Data Warehouse* do Sistema

O principal objetivo do sistema analítico é disponibilizar aos dois principais tipos de utilizador do sistema, o professor e o aluno, uma plataforma de análise da informação gerada pela Leonardo, em particular relacionada diretamente com os processos de aprendizagem e de avaliação do aluno, que contemple a temporalidade dos dados adquiridos nesses processos e a revele aos utilizadores de forma simples e clara através de um conjunto de *dashboards* específicos.

Antes de partirmos para a implementação do sistema analítico foi necessário realizar uma análise da informação gerada pela plataforma. A partir desta análise foram identificadas duas áreas de análise que foram modeladas e materializadas num *data warehouse* específico, que ficou organizada em dois *data marts*, nomeadamente:

- *Data mart* de respostas, que é responsável por armazenar toda a informação relativa às respostas dos utilizadores às questões que lhes foram lançadas pelo sistema de aferição.
- *Data mart* do sistema, que tratará de acolher a informação relativa às interações dos utilizadores com o sistema, em particular a informação relativa às sessões realizadas, às páginas visitadas, etc.

A partir de um estudo prévio dos dados relevantes para a implementação do sistema analítico, estruturaram-se os esquemas dos dois *data marts*, que estão apresentados respetivamente nas Figuras 42 e 45. Tal como a base de dados geral do sistema, o *data warehouse* foi implementado em *MongoDB*.

4.2.1 Data Mart de Respostas

O *data mart* de respostas foi criado com o objetivo de armazenar os diversos elementos de dados relativos às respostas dadas às questões geradas pelo sistema. Cada resposta foi fornecida por um aluno a uma determinada questão. A estrutura do documento de acolhimento deste dados pode ser observada na Figura 42.

```
{
  "_id" : ObjectId("5f70661038b9030b9372d261"),
  "Answer" : {
    "Answer_Start_Time" : ISODate("2020-02-17T21:18:10.341Z"),
    "Answer_End_Time" : ISODate("2020-02-17T21:20:10.341Z"),
    "Answer_Time" : 2,
    "Correction" : 1,
    "Answer_Points" : 1
  },
  "Dim_Calendar" : {
    "Date" : ISODate("2020-01-20T00:00:00Z"),
    "Month" : "January",
    "Quarter" : 1,
    "Semester" : 1,
    "Year" : 2020,
    "Week" : 4,
    "Weekday" : "Monday"
  },
  "Dim_Time" : 21,
  "Dim_Period" : "N",
  "Dim_Language" : "pt",
  "Dim_User" : {
    "UserId" : "u001",
    "Name" : "Ana Castro e Cunha",
    "Gender" : "F",
    "Degree" : "BSc IE"
  },
  "Dim_Question" : {
    "QuestionId" : "PTEINSBD0264",
    "Study_Cycle" : "Curso Universitário",
    "Scholarity" : "Engenharia Informática",
    "Domain" : "Sistemas de Bases de Dados",
    "Subdomain" : "Dados, Informação e Conhecimento",
    "Subsubdomain" : "",
    "Difficulty_Level" : 2,
    "Answering_Time" : 60,
    "Type" : "Conceito"
  }
}
```

Figura 42: Exemplo de um documento do *Data mart* de Respostas.

Na estrutura base deste *data mart* podemos encontrar definidas as seguintes dimensões.

- *Dim_Calendar* (temporal), esta dimensão é de extrema importância, uma vez que é ela que vai fornecer os dados necessários para que seja possível fazer a análise temporal da informação acolhida no *data mart*. Nesta dimensão são armazenados apenas atributos “temporais”, relativos à data da inserção da pergunta, e daí derivados todos os outros elementos de forma a que, posteriormente, seja possível fazer análises temporais.
- *Dim_Time* (degenerada), que acolhe as diversas horas que temos ao longo de um dia.
- *Dim_Period* (degenerada), que armazena as diversas instâncias dos vários períodos do dia.
- *Dim_Language* (degenerada), que mantém as diversas instâncias das linguagens nas quais as perguntas foram realizadas.
- *Dim_User* (regular), que armazena a informação acerca dos utilizadores.
- *Dim_Question* (regular), que acolhe a informação relativa às questões que foram lançadas pelo sistema e que tiveram uma resposta.

Para além da relação das dimensões apresentadas, a estrutura base dos documentos deste *data mart* também inclui um conjunto de medidas, todas elas relacionadas com as respostas que foram dadas pelos alunos às questões que lhes foram lançadas. De referir:

- *Correction* (medida), que armazena um inteiro que pode tomar os valores 1 e 0, caso a resposta esteja correta ou errada, respetivamente.
- *Answer_Points* (medida), que acolhe os pontos obtidos com a resposta dada.

4.2.2 Data Mart do Sistema

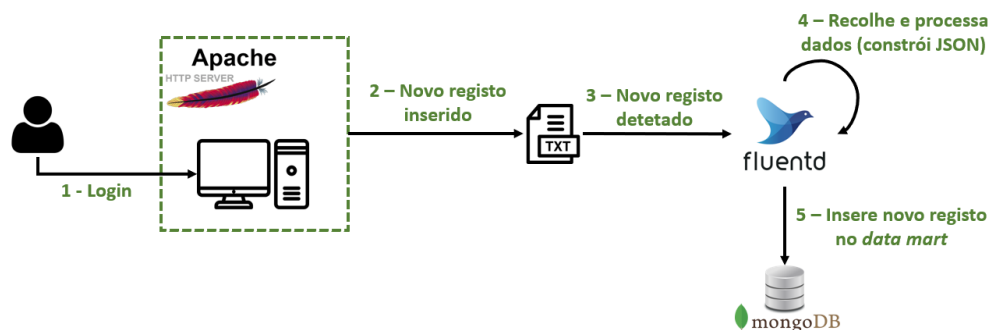


Figura 43: Ilustração do processo de captura, tratamento e registo da informação de acesso.

No primeiro passo está representado uma interação do utilizador com o sistema, nomeadamente o seu *login* na aplicação. Com esta interação foi gerado um registo (*log*) pelo servidor *Apache* tendo sido escrito num ficheiro de texto. Um exemplo deste *log* pode ser visto na Figura 44. A ferramenta *Fluentd* [Fluentd, n.d] vai estar à "escuta" de entradas neste ficheiro, e assim que um novo registo é inserido, é executado um script *Python* que estrutura o *JSON* a partir do *log* gerado e o insere no *data warehouse*.

```
193.136.19.202 - - [19/Jul/2021:23:02:19 +0100] 302 217 POST / HTTP/1.1 56
Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36
```

Figura 44: Exemplo de um log gerado pelo servidor *Apache*.

Por fim é apresentada a estrutura da informação que resulta deste processo..

```
{
  "_id" : ObjectId("5f5e42863e90161987a7623a"),
  "Dim_User" : {
    "Ip_address" : "193.136.19.202",
    "Contry" : "Portugal",
    "City" : "Braga",
    "Latitude" : 41.55179977416992,
    "Longitude" : -8.422900199890137
  },
  "Dim_Request" : {
    "Method" : "GET",
    "URL" : "/",
    "Code" : "302",
    "Protocol" : "HTTP/1.1",
    "User_Agent" : "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                  AppleWebKit/537.36 (KHTML, like Gecko)
                  Chrome/85.0.4183.83 Safari/537.36"
  },
  "Dim_Response" : {
    "Nr_Bytes" : "217",
    "ResponseTime" : "8150"
  },
  "Dim_Session" : {
    "Session_Token" : "13/Sep/2020:17:01:59 +0100193.136.19.202",
    "Start_Time" : ISODate("2020-09-13T16:01:59Z"),
    "End_Time" : ISODate("2020-09-13T16:01:59Z"),
    "Session_Time" : 98
  },
  "Dim_Calendar" : {
    "Date" : ISODate("2020-09-13T00:00:00Z"),
    "Month" : "September",
    "Quarter" : 3,
    "Semester" : 2,
    "Week" : "37",
    "Weekday" : "Sunday",
    "Year" : 2020
  },
  "Dim_Time" : "17:30"
}
```

Figura 45: Exemplo de um documento do *Data mart* do sistema.

A informação após ter passado pelo processo descrito na Figura 43, é guardada no *data mart* do sistema em documentos com a estrutura que podemos ver na Figura 45. Nessa estrutura de dados foram incluídas as seguintes dimensões:

- *Dim_User* (regular), que armazena a informação relativa à localização do utilizador, inclusive o seu endereço IP.
- *Dim_Request* (regular), que acolhe a informação referente ao tipo de pedido efetuado ao sistema.
- *Dim_Response* (regular), que recebe o número de bytes da resposta, bem como o tempo que demorou a ser obtida.
- *Dim_Session* (regular), que guarda a informação relativa à sessão do utilizador, que permitirá determinar o tempo que o utilizador esteve autenticado no sistema.
- *Dim_Calendar* (temporal), a dimensão temporal do sistema, que fornece os dados necessários para realizar análises temporais sobre a informação guardada no *data mart*.
- *Dim_Time* (temporal), que mantém as diversas instâncias hora:minuto, relativas a um dia.

A estrutura dos elementos base deste *data mart* apenas contém elementos de dados do tipo dimensão, não contendo qualquer elemento do tipo medida. Assim, estamos perante uma tabela de factos do tipo *fact-less*.

4.3 Dimensões com variação

A abordagem que seguimos para fazer a criação do modelo dimensional do nosso sistema de análise pode ser vista como uma adaptação do processo de modelação dimensional proposto por Kimball [Kimball e Ross, 2013]. Este processo foi originalmente idealizado para sistemas relacionais e é hoje em dia uma referência quando se abordam sistemas de *data warehousing*. Assim sendo, ao longo desta dissertação, Kimball foi sempre uma referência, onde se abordou os seus conhecimentos e aplicou a sistemas orientados a documentos, em que foi tido sempre em consideração possíveis adaptações que podiam melhorar o desempenho deste tipo de sistemas.

Na modelação dimensional, aplicada a sistemas relacionais, evitar a redundância de informação nos vários elementos de dados que integram o esquema do *data warehouse* é geralmente uma preocupação, em particular ao nível das suas dimensões. Porém, na modelação dimensional de um sistema suportado por uma base de dados orientada a documentos o processo muda significativamente, em particular na forma como os relacionamentos entre os diversos elementos de dados são estabelecidos. Nestes sistemas, o foco principal passa por criar modelos de dados que permitam uma boa *performance*, em que a redundância de dados pode ser considerada como um pequeno custo para alcançar essa *performance* e não como algo que é prejudicial ao sistema de dados. Assim, devemos ter em consideração a construção de estruturas dimensionais bastante desnormalizadas, de modo a evitar, tanto quanto possível, operações de relacionamento de dados. Vejamos, então, como é que os vários elementos de dados dos *data mart* implementados foram influenciados por este processo de adaptação.

No nosso processo de implementação do sistema de análise foram identificados em algumas dimensões um conjunto de pares chave-valor, cujo valor poderia sofrer alterações ao longo do tempo. Após alguma análise da sua importância para o sistema, considerámos que seria relevante fazer a preservação do histórico desses valores ao longo do tempo. Isso implicou a criação de estruturas de dados adequadas no *data warehouse*, em particular nos *data marts* que incorporassem as dimensões com variação. Nesse sentido, para determinar qual a estrutura mais apropriada para a implementação de dimensões com variação e história foram contempladas duas possíveis estruturas e realizadas análises do desempenho do sistema perante a execução de algumas *queries* que trabalhassem sobre essas dimensões. Para proceder à análise referida utilizámos como base de trabalho o *data mart* de respostas apresentado anteriormente (Secção 4.2.1). Nesse *data mart* modificámos a dimensão "*Dim_User*", acrescentando-lhe um novo par chave-valor, identificado por "*Study_Plan*", cujos valores, ao longo do tempo, podem ser modificados. O par chave-valor identificado por "*Study_Plan*" permitirá armazenar o conjunto de valores relativos às disciplinas frequentadas pelo utilizador durante o seu percurso académico. Na sua estrutura está incluído o par chave-valor identificado por "*Subjects*", que varia semestralmente durante o período de frequência de um dado curso, sendo esperado que a sua influência não seja muito grande no par chave-valor identificado por "*Study_Plan*", no qual esse histórico é armazenado, não crescendo demasiado.

Com o intuito de determinar qual a melhor opção para albergar estes valores, foram criadas duas estruturas distintas. Na primeira estrutura incluímos, num único documento, o par chave-valor

identificado por *"Study_Plan"*, cujo valor é uma lista de valores, sendo eles subdocumentos, contendo os valores identificados pela chave *"Subjects"*, para um determinado intervalo de tempo. De notar que o primeiro elemento dessa lista corresponde ao valor atual desse par chave-valor. Desta forma, evita-se algum processamento adicional como, por exemplo, analisar a lista completa para encontrar o valor mais atual. Na Figura 46 podemos ver um exemplo representativo da dimensão *"Dim_User"*, com este novo par chave-valor na sua estrutura e com dois registos. De seguida, na Tabela 17 apresentamos, de forma sucinta, algumas das vantagens e desvantagens desta opção de implementação de uma dimensão com variação.

```
"Dim_User" : {
  "UserId" : "u001",
  "Name" : "Ana Castro e Cunha",
  "Gender" : "F",
  "Degree" : "BSc IE",
  "Study_Plan": [
    {
      "Subjects": [
        "Contabilidade Financeira",
        "Ferramentas e Princípios de Gestão",
        "Matemática Empresarial Aplicada",
        "Fundamentos de Marketing",
        "Construção de Relacionamentos",
        "Humanidades 1"
      ],
      "Year":1,
      "Semester": 2,
      "Insertion_Date": ISODate("2020-03-20T00:00:00Z"),
      "Expiration_Date": ISODate("2100-03-20T00:00:00Z")
    },
    {
      "Subjects": [
        "Contabilidade de Custos",
        "Matemática para Gestão",
        "Introdução à Programação",
        "Finanças Corporativas",
        "Micro-economia",
        "Pesquisa e Habilidades de Escrita Académica",
        "Humanidades 2"
      ],
      "Year":1,
      "Semester": 1,
      "Insertion_Date": ISODate("2018-09-20T00:00:00Z"),
      "Expiration_Date": ISODate("2020-03-19T23:59:59Z")
    }
  ],
}
```

Figura 46: Exemplo de um documento da dimensão *"Dim_User"* contendo um novo par chave-valor com variação.

Vantagens	Desvantagens
<ul style="list-style-type: none"> • Maior facilidade em obter informação externa ao histórico do utilizador, relativamente às outras dimensões do sistema 	<ul style="list-style-type: none"> • Armazenamento da informação relativa ao histórico do utilizador numa lista, o que requer processamento adicional. • Nível de redundância elevado, devido ao facto de cada documento referente a uma resposta, efetuada por um utilizador, conter a sua informação histórica. • Mais processamento na manutenção do histórico do par chave-valor com variação. Devido à redundância da informação contida no histórico do par chave-valor com variação, todos os documentos referentes às diversas respostas dadas pelos alunos contendo este par, vão ter que ser atualizados.

Tabela 17: Vantagens e desvantagens de usar a estrutura apresentada no documento da Figura 46.

Na segunda estrutura optámos por uma abordagem diferente, criando, especificamente, uma nova coleção de dados. Esta coleção será responsável por armazenar o histórico do novo par chave-valor, bem como o valor atual do mesmo. Desta forma, por cada novo valor identificado por "*Study Plan*", será adicionado um novo documento na referida coleção. Nos documentos do *data mart* "*Respostas*", apenas irá estar o registo atual referente a essa chave. De seguida, nas Figuras 47 e 48, podemos ver um exemplo de uma possível materialização dessas estruturas de dados.

```

"Dim_User" : {
  "UserId" : "u001",
  "Name" : "Ana Castro e Cunha",
  "Gender" : "F",
  "Degree" : "BSc IE",
  "Study_Plan" : {
    "Subjects" : [
      "Estratégias para a Corporação Multinacional",
      "Estratégias num Mundo Disruptivo",
      "Projeto Final",
      "Cursos Eletivos (IE/Estagio)"
    ],
    "Year" : 4,
    "Semester" : 2,
    "Insertion_Date" : ISODate("2020-01-02T12:00:00Z"),
    "Expiration_Date" : ISODate("2100-01-02T12:00:00Z"),
  }
}

```

Figura 47: Exemplo de um documento da dimensão "Dim_User" contendo o novo par chave-valor "Study_Plan" com variação.

```

{
  "_id" : ObjectId("60f2b5d0544ad486b6896ed2"),
  "UserId" : "u001",
  "Subjects" : [
    "Estratégias para a Corporação Multinacional",
    "Estratégias num Mundo Disruptivo",
    "Projeto Final",
    "Cursos Eletivos (IE/Estagio)"
  ],
  "Year" : 4,
  "Semester" : 2,
  "Insertion_Date" : ISODate("2020-01-02T12:00:00Z"),
  "Expiration_Date" : ISODate("2020-06-02T12:00:00Z")
}

```

Figura 48: Exemplo de um documento contendo valor histórico do chave-valor "Study_Plan".

Contudo, devemos realçar que o par chave-valor que suporta o relacionamento da dimensão com o seu histórico, corresponde ao identificador do utilizador, visto que este histórico é específico de cada utilizador do sistema.

Vantagens	Desvantagens
<ul style="list-style-type: none"> • Baixa redundância da informação relativa ao histórico do utilizador. • Rápida e simples manutenção da informação relativa ao par chave-valor com variação, sendo que implica apenas a inserção de um novo documento na nova coleção quando é atualizada. 	<ul style="list-style-type: none"> • Baixo desempenho quando é necessário obter informação extra do utilizador, como o nome por exemplo.

Tabela 18: Vantagens e desvantagens de usar a estrutura base de uma nova coleção de documentos.

4.4 *Queries* sobre Dimensões com Variação

Após a definição das estruturas que acabámos de apresentar, procedemos à realização de alguns testes para avaliação do desempenho do sistema na sua manipulação, com alguns exemplos concretos de aplicação. Para tal foram definidas em *MongoDB* três *queries* específicas que, depois, foram executadas sobre as estruturas concebidas, contendo diferentes volumes de dados. Estes testes foram executados numa máquina virtual na qual foram alocados dois *cores* de um processador Intel Xeon Gold 6130, contendo 4GB de RAM.

Estes resultados são apresentados em formato tabular contendo o número de registos analisados e o tempo de execução da *query* respetiva. Para efeitos identificativos consideremos que o modelo 1 corresponde à primeira estrutura apresentada na Figura 46 e o modelo 2 à segunda estrutura apresentada na Figura 47. Cada uma das três *queries* serão executadas para diferentes níveis de registos no *data warehouse*, bem como um diferente número de utilizadores presente no sistema, com o objetivo de avaliar o impacto da dimensão com variação definida. É importante referir que cada utilizador identificado na dimensão "*Dim_User*" irá conter um histórico com 8 valores diferentes, isto é, na prática, cada utilizador terá associado 8 registos referentes ao "*Study_Plan*" cuja "*Expiration_Date*" já foi atingida.

De seguida, apresentamos as várias *queries* que foram desenvolvidas e executadas para cada um dos modelos apresentados, de forma a conseguir responder às questões propostas. Posteriormente, serão apresentados os tempos de execução destas *queries*, tendo em consideração os diferentes volumes de dados envolvidos.

• **Questão 1 - Quais os "Subjects" de um dado utilizador numa dada data?**

Para respondermos a esta questão tendo como base de trabalho as duas estruturas referidas anteriormente, desenvolvemos as seguintes *queries* em *MongoDB*:

```
modelo1.aggregate([
  {"$match":{"Dim_User.UserId":"u001"}},
  {"$limit":1},
  {"$unwind": "$Dim_User.Study_Plan"},
  {"$match":{"Dim_User.Study_Plan.Insertion_Date":{"$lte": "2020-01-01"},
    "Dim_User.Study_Plan.Expiration_Date":{"$gte": "2020-01-01"}}}
])
```

Figura 49: Query executada para a estrutura 1.

```
modelo2.aggregate([
  {"$match":{"Dim_User.UserId":"u001"}},
  {"$group":{"_id":"$Dim_User.UserId", "Name": {"$first":"$Dim_User.Name"}}},
  {"$lookup":{"
    "from": "history_study_plan_new",
    "pipeline":[{"$match": {
      "UserId":"u001",
      "Insertion_Date":{"$lte": "2020-01-01"},
      "Expiration_Date":{"$gte": "2020-01-01"}}
    ]},
    "as": "user_info"
  }},
  {"$unwind": '$user_info' }
])
```

Figura 50: Query executada para a estrutura 2.

Número de Registos	Número de Utilizadores	Tempo de execução Modelo 1 (s)	Tempo de execução Modelo 2 (s)
10^6	20	0.01	1.68
10^6	5×10^5	0.02	5.12
10^6	10^6	0.02	10.20
3×10^6	10^6	0.02	9.61
3×10^6	3×10^6	0.02	27.83

Tabela 19: Resultados da execução das *queries* relativas à questão 1.

Através de uma breve análise dos dados contidos na Tabela 19, podemos verificar que os tempos de execução da *query* realizada para o primeiro modelo implementado mantêm-se praticamente os mesmos para os vários conjuntos de registos e de utilizadores que considerámos. Tal circunstância deve-se ao facto da *query* executada sobre este modelo devolver apenas o primeiro documento encontrado para o utilizador indicado numa clausula de filtragem da *query*. É possível efetuar este filtro ("*limit:1*"), pois

todos os documentos inseridos para um dado utilizador vão conter a mesma informação referente à dimensão *"Dim_User"*, assim sendo, apenas precisamos de processar a informação de um único documento por utilizador para obter a resposta à questão proposta.

Como vimos, no segundo modelo, a dimensão com variação está armazenada numa outra coleção. Ao executarmos a *query* sobre este modelo verificámos um aumento gradual do tempo de execução à medida que íamos aumentando o número de registos, bem como o número utilizadores. Este aumento aconteceu devido ao facto de a dimensão com variação e o seu histórico se encontrarem em coleções separadas. Desta forma, para ser possível obter a informação de um dado utilizador, como por exemplo, o nome, é necessário consultar a dimensão *"Dim_User"* e para obter os *"Subjects"* do mesmo numa dada data, é necessário consultar o histórico dessa dimensão.

- **Questão 2 - Quais os nomes dos utilizadores que tiveram aulas ao *"Subject"* 'Lei Empresarial' numa dada data?**

Para respondermos a esta questão mantendo como base de trabalho as duas estruturas referidas anteriormente, desenvolvemos as seguintes *queries* em *MongoDB*:

```
modelo1.aggregate([
  {"$unwind": "$Dim_User.Study_Plan"},
  {"$match": {"Dim_User.Study_Plan.Subjects": "Lei Empresarial",
    "Dim_User.Study_Plan.Insertion_Date":{"$lte": "2020-01-01"},
    "Dim_User.Study_Plan.Expiration_Date":{"$gte": "2020-01-01"}}}
])
```

Figura 51: Query executada para a estrutura 1.

```
history_modelo2.aggregate([
  {"$unwind": '$Subjects' },
  {"$match": {"Subjects": "Lei Empresarial",
    "Insertion_Date":{"$lte": "2020-01-01"},
    "Expiration_Date":{"$gte": "2020-01-01"}}},
  {"$group":{"_id":"$UserId"}}
])

modelo2.find({"$or":lista_json_users_id})
```

Figura 52: Query executada para a estrutura 2.

Número de Registos	Número de Utilizadores	Tempo de execução Modelo 1 (s)	Tempo de execução Modelo 2 (s)
10^6	20	0.02	0.01
10^6	5×10^5	0.02	5.34
10^6	10^6	0.01	9.63
3×10^6	10^6	0.04	9.87
3×10^6	3×10^6	0.04	43.06

Tabela 20: Resultados da execução das queries relativas à questão 2.

Tendo por base os resultados apresentados na Tabela 20, obtidos na execução das queries para ambos os modelos, é possível verificar que para o primeiro modelo, em que a sua estrutura é definida num único documento, os tempos de execução foram muito semelhantes, mesmo quando o número de registos aumentou para 3 milhões, é possível notar que o aumento do tempo de execução da query é mínimo.

Para conseguir obter resposta para a questão proposta a partir da estrutura apresentada nas Figuras 47 e 48, tiveram que ser executadas duas queries sequencialmente. Numa primeira fase foi executada uma query para obter os "UserId" referentes aos utilizadores que tiveram aulas ao "Subject" 'Lei Empresarial' na data proposta. Posteriormente, dado que o nome dos utilizadores se encontram numa outra coleção, foi necessário consultar essa coleção passando os "UserId" obtidos a partir da primeira query.

Em relação aos resultados obtidos para o segundo modelo, é possível verificar na Tabela 20 que diferem bastante entre si. O aumento do número de utilizadores no sistema tem bastante impacto em termos de performance. Este aumento implica um maior número de dados que é necessário filtrar e agregar e, para além disso, quanto maior o número de utilizadores maior o número de consultas à dimensão "Dim_User" de forma a obter todos os dados necessários.

- **Questão 3 - Qual o número de alunos presente em cada "Subject" ao longo do tempo?**

Esta questão vai implicar queries com um nível de complexidade mais elevado do que as que foram apresentadas até agora. As queries desenvolvidas foram as seguintes:

```

modelo1.aggregate([
  {"$unwind": "$Dim_User.Study_Plan"},
  {"$unwind": "$Dim_User.Study_Plan.Subjects"},
  {'$group': {'_id': {'subject': '$Dim_User.Study_Plan.Subjects',
                      'startDate': '$Dim_User.Study_Plan.Subjects.Insertion_Date',
                      'endDate': '$Dim_User.Study_Plan.Subjects.Expiration_Date'},
              "total": {"$sum": 1}}},
  {'$group': {'_id': '',
              'dates': {'$push': "$_id"},
              'students_number': {'$push': '$total'}}}
])

```

Figura 53: Query executada para a estrutura 1.

```

historia_modelo2.aggregate([
  {"$unwind": '$Subjects' },
  {"$group":{
    "_id":{"subjects":"$Subjects",
           "startDate":"$Insertion_Date",
           "endDate":"$Expiration_Date"},
    "total":{"$sum":1}}
  },
  {"$group":{
    "_id":'',
    "dates":{"$push": "$_id"},
    "studentsNumber":{"$push":"$total"}}
  }
])

```

Figura 54: Query executada para a estrutura 2.

Número de Registos	Número de Utilizadores	Tempo de execução Modelo 1 (s)	Tempo de execução Modelo 2 (s)
10 ⁶	20	44.74	1.67
10 ⁶	5x10 ⁵	43.23	14.26
10 ⁶	10 ⁶	42.06	25.92
3x10 ⁶	10 ⁶	131.33	26.32
3x10 ⁶	3x10 ⁶	130.17	135.63

Tabela 21: Resultados da execução das queries relativas à questão 3.

A partir de uma breve análise dos dados apresentados na Tabela 21, é possível verificar que para o primeiro modelo o tempo de execução apenas aumenta quando o número de registos é aumentado, mantendo-se praticamente constante quando apenas o número de utilizadores varia. Tal circunstância deve-se ao facto de o processamento que é feito na query presente na Figura 53 apenas depender do número de registos e ser independente do número de utilizadores. Se tomarmos em consideração a situação em que a query foi executada para um conjunto de 1 milhão de registos e onde variou o número de utilizadores de 500 mil para 1 milhão, podemos verificar pela query apresentada, que o

processamento vai depender apenas do número de registos, pois não é efetuado nenhum filtro ou agrupamento de dados por utilizador.

No segundo modelo é possível verificar que acontece o oposto, o aumento do número de registos no sistema não tem praticamente impacto no tempo de execução da *query*. Porém, o aumento do número de utilizadores tem um impacto significativo. Este resultado deve-se ao facto da *query* apenas depender dos valores contidos na coleção responsável por guardar o histórico de cada utilizador. Se o número de documentos na coleção que contém o *data mart* aumentar mas o número de utilizadores se mantiver contante, isto não vai implicar que o número de registos relativos ao histórico por utilizador aumente, porém se o número de utilizadores aumentar, este aumento vai implicar que o número de documentos na coleção responsável por armazenar o histórico de cada utilizador aumente. Este aumento vai implicar que o processamento da *query* apresentada na Figura 54 seja maior, resultando num deterioramento da *performance* da mesma.

4.5 Utilização de Vistas Materializadas

As vistas materializadas são muitas vezes criadas com o propósito de melhorar a *performance* de um sistema na execução de determinadas *queries*. Neste tipo de estruturas pode ser armazenado o resultado de uma dada *query*, com um grande custo de execução, ou que seja executada de forma recorrente, o que permitirá reduzir os custos computacionais do sistema e melhorar o seu desempenho. Ao guardar-se este resultado numa estrutura física, não é necessário executar a *query*, a não ser quando os dados base sejam modificados, reduzindo o seu custo de execução. Porém, este custo é substituído, em parte, pela manutenção do resultado da *query*.

Em bases de dados orientadas a documentos, a criação de vistas materializadas pode ser um processo fundamental quando se tem *queries* que envolvam operações de agregação exigentes, que causem um grande impacto na *performance* do sistema.

Na implementação do nosso sistema foi necessário fazer a criação de uma vista materializada. Nesta vista foi guardado o resultado de uma *query* envolvendo uma agregação de alto custo. Esta *query* permite obter uma lista ordenada das datas relativas a todos os registos de um determinado domínio de estudo. Para conseguir demonstrar o impacto no sistema desta *query*, em particular em termos do tempo da apresentação dos resultados ao utilizador, na Figura 55 apresentamos um *dashboard* cujos valores são obtidos a partir dessa *query*.

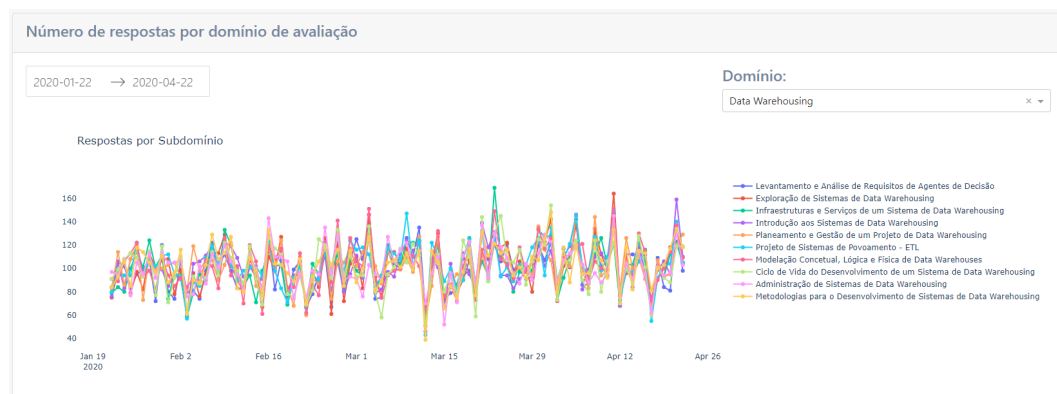


Figura 55: Número de respostas dadas a questões de um dado domínio de estudo.

No *dashboard* apresentado é possível observar a existência de um filtro temporal, que, dependendo do domínio de estudo, permite definir o período (um intervalo de datas) para o qual queremos obter os valores das respostas realizadas – a data de início deste período será a data do primeiro registo efetuado para aquele domínio e a data de fim será a data da inserção do valor do último registo de respostas. A vista que criámos suportará o cálculo deste intervalo, para cada domínio de estudo. Inicialmente, para se calcular este intervalo de datas e obter as datas de início e do último registo (data de fim), executávamos uma *query* que, para um dado domínio de estudo, consultava todos os registos de respostas, ordenando-os por data. Nesta implementação, cada vez que um utilizador seleccionava um novo domínio de estudo no *dashboard*, o esforço da execução desta *query* causava um

enorme *delay* na apresentação de resultados no *dashboard*. Para além disso, notou-se que com esta *query* haviam resultados que estavam a ser calculados inúmeras vezes sem necessidade, como, por exemplo, a data de inserção do primeiro registo no sistema.

```
dm_answers.find({"Dim_Question.Domain":"Data Warehousing"})
    .sort({'Dim_Calendar.Date':1})

dates = {"Domain": "Data Warehousing",
        "Start_Date": start,
        "End_Date": end
}
materializedView.update_one({"Domain":"Data Warehousing"},
    {"$set":dates},
    {upsert:true})
```

Figura 56: Processo de criação da vista.

Para solucionar este problema foi então criada uma vista materializada que envolveu uma nova coleção, contendo um documento para cada domínio presente no sistema, no qual eram armazenadas a data de início e data de fim relativas ao intervalo temporal dos registos de cada domínio de estudos presentes no *data warehouse*. Esta vista, foi criada através do processo apresentado na Figura 56. Este processo inicia-se pelo cálculo da primeira e última data de inserção de um registo relativas a um dado domínio. Num segundo passo foi construído um *JSON* que caracteriza a estrutura da vista para esse domínio. Na última parte deste processo, foi então inserido este documento no *data warehouse* caso ainda não existisse nenhum para o dado domínio, caso contrário apenas atualizava os valores. É importante ter em consideração que a primeira parte deste processo, o cálculo das datas, apenas ocorre uma única vez, sendo que, a partir deste momento o valor da "*Start_Date*" mantém-se inalterado e apenas o valor da "*End_Date*" é mantido à medida que novos registos são inseridos no sistema. Na Figura 57 está apresentado um exemplo de um dos documentos que resultam da execução deste processo.

```
{
  "_id" : ObjectId("5f26abfd20c38abf9df3a176"),
  "Domain" : "Data Warehousing",
  "Start_Date" : ISODate("2019-04-24T00:00:00Z"),
  "End_Date" : ISODate("2020-04-22T00:00:00Z")
}
```

Figura 57: Exemplo de documento com informação relativa ao cálculo do intervalo temporal dos registos do sistema para um dado domínio.

Assim, com a implementação desta vista, a data de início foi calculada uma única vez, já que esta data é relativa ao primeiro registo inserido no sistema para o domínio de estudo em causa, não sendo necessário recalculá-la pois o seu resultado será sempre o mesmo. O único valor que necessita de ser mantido é a data de fim, uma vez que esta é relativa ao último registo inserido no sistema, circunstância que varia com bastante frequência. Dado que o sistema relativo ao caso de estudo em questão é atualizado em tempo real, para se manter este valor atualizado é necessário efetuar uma atualização do mesmo sempre que um novo registo for inserido no *data warehouse*. Desta forma, para além do custo da inserção do registo no *data warehouse*, acresce o custo de atualização do valor identificado por "*End_Date*" na vista. Porém este custo é bastante pequeno quando comparado com o

tempo que demorava a executar a anterior *query* de agregação, que requeria ordenar inúmeras vezes todos os registos envolvidos. Para concluir, o tempo necessário para atualizar a vista e consultar a informação do *dashboard* é muito menor do que o tempo de execução da *query* de agregação feita inicialmente, implementada para servir exatamente o mesmo propósito.

Para demonstrar esta situação realizámos uma análise do tempo de execução da *query* de agregação envolvendo 377560 registos. O tempo de resposta foi 1.58 segundos. Este tempo pode não parecer muito grande. Porém, no contexto da aplicação em questão, esta porção de tempo causa grande impacto para o utilizador, pois de cada vez que o filtro relativo ao domínio era atualizado, o *delay* referido era aplicado. Com a adição da vista materializada, a consulta do documento contendo este resultado é praticamente instantânea, não causando qualquer atraso na apresentação dos resultados ao utilizador.

4.6 Visualização e Análise de Dados

A partir do sistema de *data warehousing* desenvolvido para o caso de estudo em análise foram implementadas quatro *dashboards* distintos, que permitem efetuar diversas análises sobre os dados armazenados, com propósitos distintos. A saber: o *dashboard* geral, o *dashboard* de respostas, o *dashboard* de utilizador e o *dashboard* de controlo de acessos.

• Dashboard Geral

No *dashboard* presente na Figura 58, está apresentada uma análise global dos dados presentes no *data mart* de respostas do sistema. No painel superior são apresentadas informações quantitativas referentes a alguns elementos do sistema, nomeadamente o número de domínios, o número de respostas certas e erradas, o tempo médio de resposta dos utilizadores e o número de avaliações efetuadas até ao momento. O primeiro gráfico apresentado, permite fazer uma avaliação do número de repostas efetuadas por subdomínio ao longo de um dado intervalo de tempo. Para um determinado domínio selecionado pelo utilizador, são apresentados os respetivos subdomínios e o número de respostas a cada um deles. No último conjunto de gráficos presente na Figura 58, são apresentadas mais três novas perspetivas de análise do sistema. No primeiro gráfico está representada a pontuação média, obtida a partir das respostas fornecidas pelos utilizadores, por semana. No segundo gráfico, está representada a distribuição das questões presentes nos subdomínios de um dado domínio. E, por fim, no terceiro painel, são apresentados os três utilizadores com melhor pontuação ao subdomínio selecionado.

Este painel de análise de dados (Figura 58) trabalha sobre os dados contidos no *data mart* de respostas, tendo como opções de análise os seguintes elementos:

- o número total de domínios de estudo integrados no sistema;
- o número de respostas certas e erradas;

- o tempo médio de resposta;
- o número de avaliações presentes no sistema;
- o número de respostas por domínio, num determinado período;
- a pontuação média semanal para um determinado mês e ano;
- a distribuição do número de questões por subdomínio;
- o top de classificações entre os utilizadores.

O propósito principal deste *dashboard*, é fornecer ao professor e administrador do sistema, uma vista global da evolução do sistema, bem como, da interação dos alunos com a plataforma. A partir da informação disponibilizada neste *dashboard*, o professor consegue saber de forma imediata quais os melhores alunos a qualquer subdomínio do sistema. Para além da informação explícita no *dashboard*, o professor consegue inferir potenciais dificuldades dos seus alunos, a partir, por exemplo, da pontuação média obtida ou através do aumento ou descida do número de respostas a um determinado subdomínio. Caso a pontuação média de um dado subdomínio comece a descer, ou o número de respostas de um determinado subdomínio comece a aumentar, pode indicar alguma falta de conhecimento nesse subdomínio

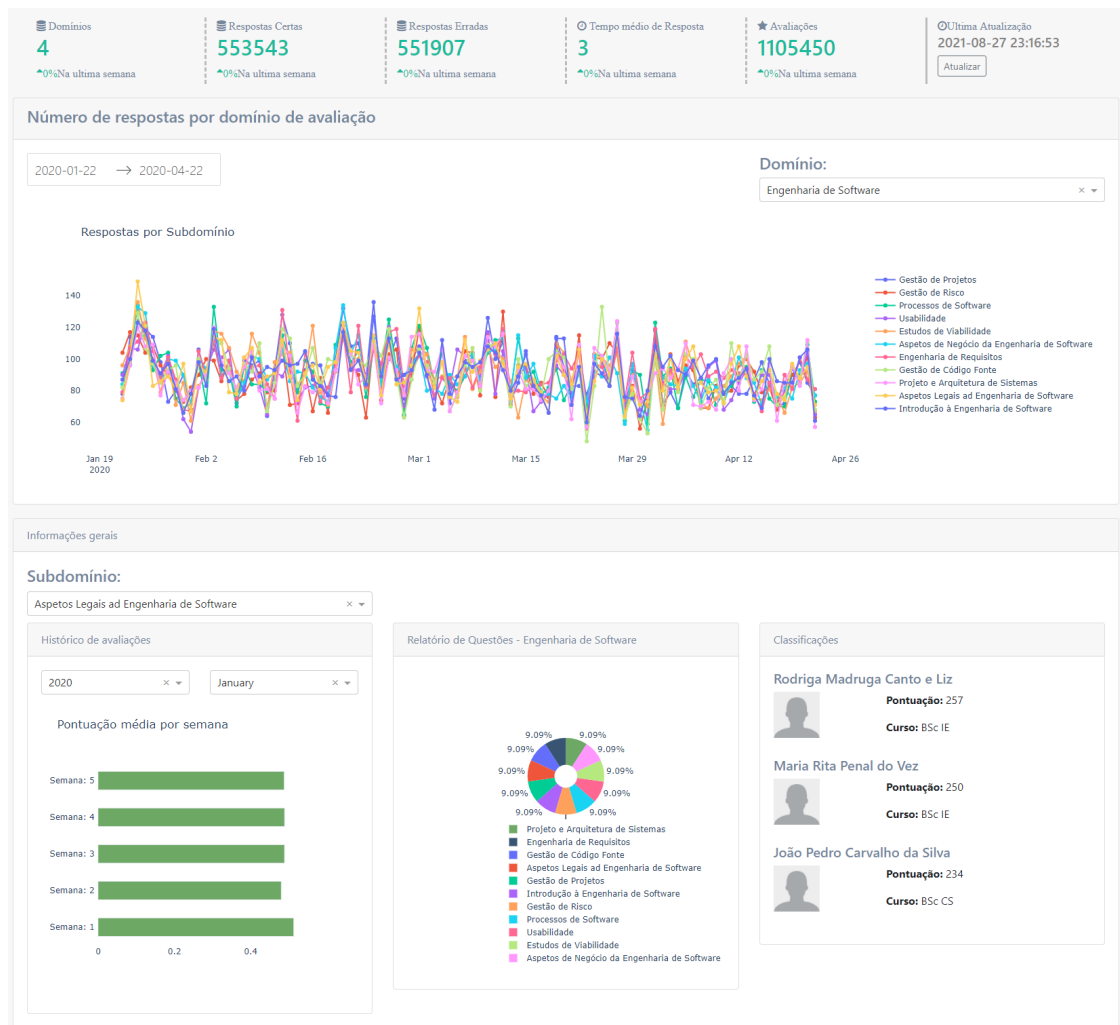


Figura 58: *Dashboard* que contém a vista mais geral do sistema.

• *Dashboard de Respostas*

O *dashboard* de respostas pretende organizar os dados armazenados no *data mart* de respostas em diferentes subtemas, permitindo a realização de análises por domínio, género, nível ou curso de utilizador.

Na Figura 59, estão apresentadas algumas vertentes de análise referentes ao *data mart* respostas do sistema. No primeiro gráfico presente no primeiro painel desta figura, está representada a análise do desempenho dos alunos por domínio e subdomínio de avaliação. Neste gráfico, é possível obter uma comparação entre o número de respostas certas e erradas dentro de um dado intervalo de tempo. De seguida, estão apresentados três gráficos cujo o objetivo é fornecer uma análise quantitativa de vários elementos de dados por domínio de avaliação, tais como, o número de respostas, a pontuação obtida por género, e o tempo médio de resposta. Por fim, no último gráfico deste painel, está representada uma análise do desempenho dos alunos por curso ao longo de um

dado intervalo do tempo, sendo que, este desempenho é avaliado com base no número de pontos obtidos. Em acréscimo, é também apresentado o top dos três alunos com melhor pontuação. No segundo painel deste *dashboard*, está representada uma análise do desempenho dos alunos por nível de dificuldade. A partir de cada um dos gráficos, é possível saber o número de respostas por nível, o tempo médio de resposta por nível, e ainda, a pontuação total por nível para ambos os géneros presentes no sistema.

Através deste *dashboard* podemos realizar diversos tipos de análise relacionadas com as respostas que os utilizadores foram dando ao sistema ao longo das suas sessões de avaliação, nomeadamente, saber:

- o desempenho dos utilizadores por domínio e subdomínio de avaliação;
- o número de respostas por subdomínio;
- a pontuação por género do utilizador, por subdomínio;
- o tempo médio de resposta por subdomínio;
- o desempenho por curso por subdomínio;
- o top de alunos por subdomínio;
- o número de respostas por nível de dificuldade;
- o tempo médio de resposta por nível de dificuldade;
- a pontuação por género de utilizador e nível de dificuldade da questão colocada.

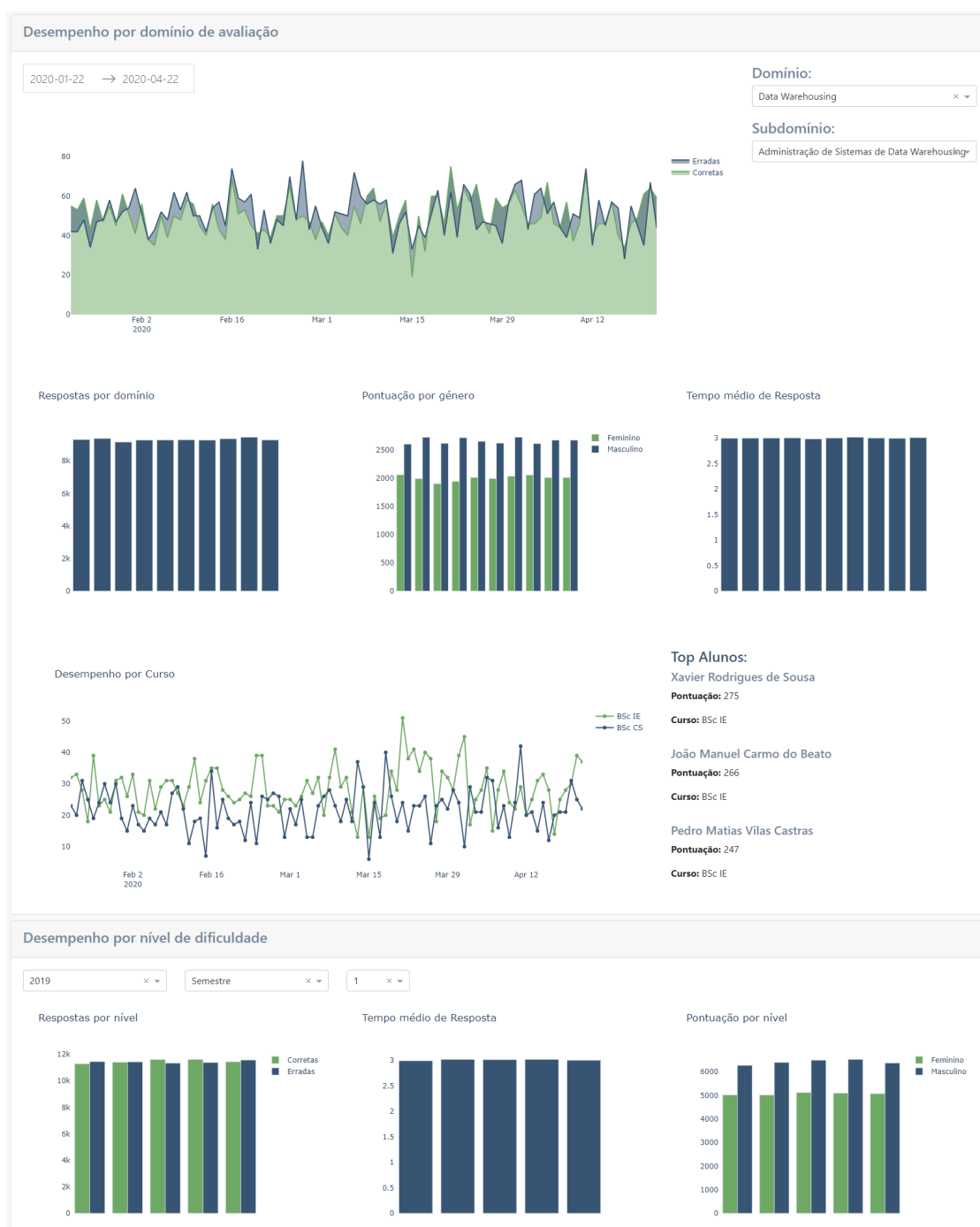


Figura 59: *Dashboard* que contém vistas direcionadas às respostas armazenadas no sistema por nível de desempenho.

A partir deste *dashboard* o professor consegue averiguar a evolução do conhecimento adquirido pelos seus alunos. Para além de conseguir saber de forma imediata o número de respostas corretas e erradas para determinados domínios e subdomínios, consegue ainda efetuar uma análise comparativa entre os diferentes cursos que efetuaram respostas a estes domínios ou

subdomínios. Desta forma, caso haja uma discrepância acentuada entre as pontuações obtidas para os diferentes cursos, o professor pode, por exemplo, ajustar a forma como leciona a matéria para o curso com pior pontuação.

- **Dashboard de Utilizador**

Nas Figuras 60 e 61 está representado um *dashboard* direcionado a um utilizador específico. No painel superior da Figura 60 está representada uma análise quantitativa do número de respostas certas e erradas, do tempo médio de resposta e do número de avaliações de um determinado utilizador do sistema. De seguida, é apresentada alguma informação pessoal do aluno e um *pie chart* com a percentagem de respostas certas e erradas efetuadas pelo utilizador. No segundo gráfico, está representada a transição de nível do utilizador ao longo do tempo.

No primeiro gráfico da Figura 61 está representado o desempenho do aluno por domínio e subdomínio de avaliação ao longo de um dado intervalo de tempo, sendo este desempenho avaliado com base no número de respostas certas e erradas presentes no sistema para esse utilizador. Por fim, nos últimos dois gráficos, são apresentadas duas análises quantitativas por nível de dificuldade do número de respostas certas e erradas fornecidas, e, do respetivo tempo médio de resposta.

Este *dashboard* pretende disponibilizar um conjunto de vistas direcionadas ao desempenho de um utilizador específico. Em particular, permite visualizar e analisar os seguintes elementos de dados:

- o valor percentual do número de respostas corretas e erradas às quais o aluno respondeu;
- o desempenho do aluno por domínio e subdomínio de avaliação;
- o número de respostas corretas e erradas de um utilizador, por nível de avaliação;
- o tempo médio de resposta por nível de desempenho.

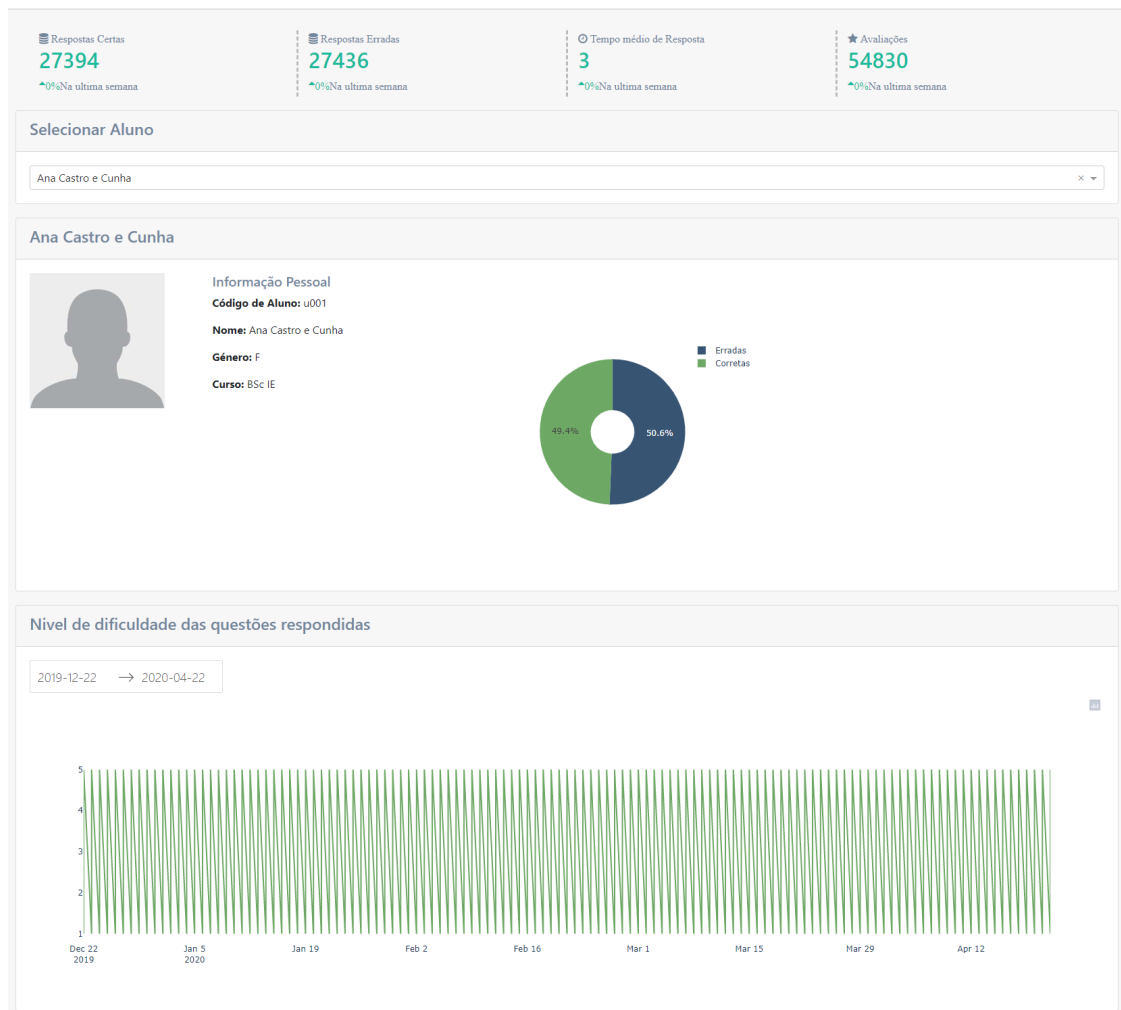


Figura 60: *Dashboard* que contém vistas direcionadas às respostas individuais de um determinado utilizador armazenadas no sistema.

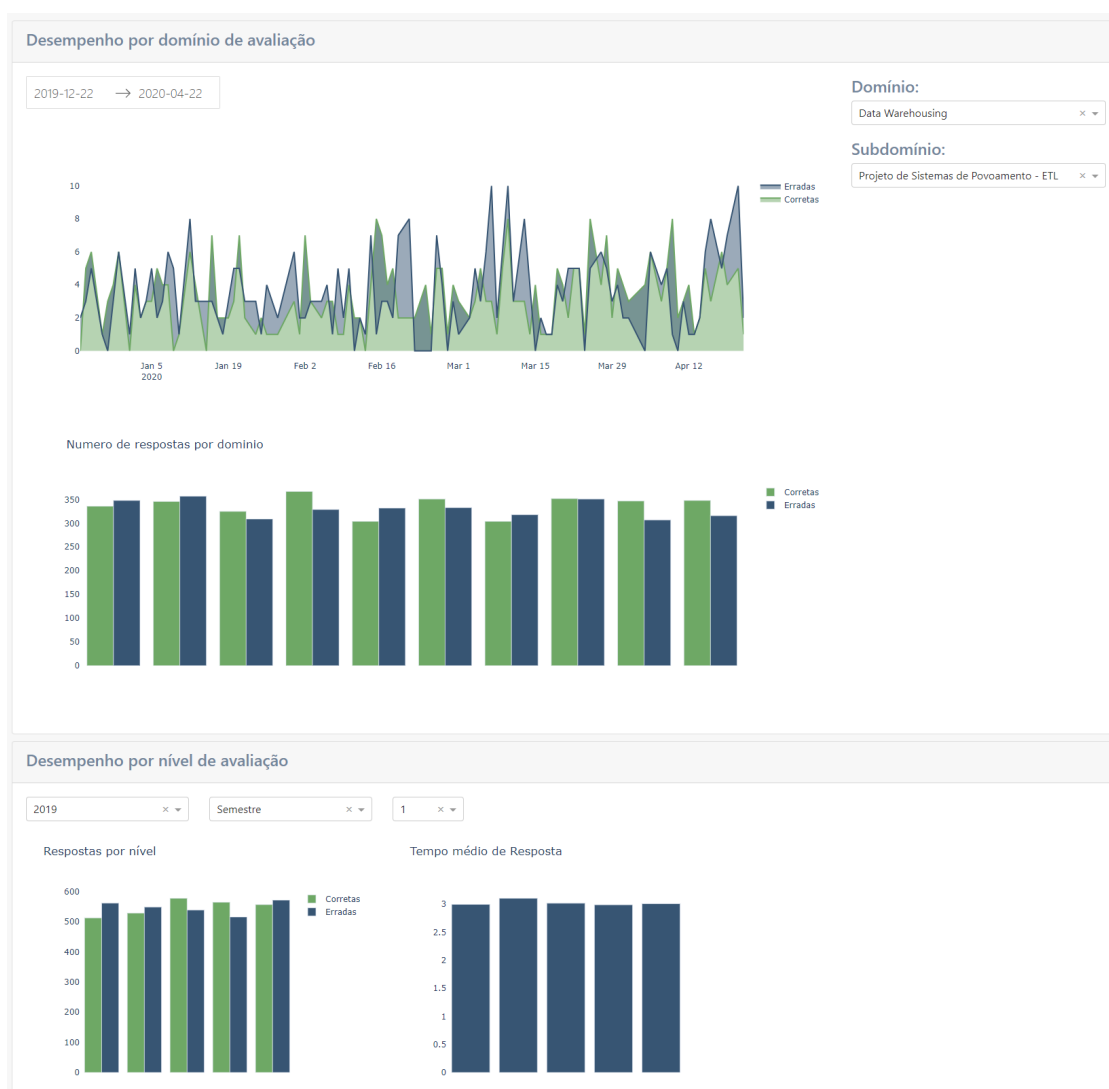


Figura 61: *Dashboard* que contém vistas direcionadas às respostas individuais de um determinado utilizador armazenadas no sistema.

Os principais objetivos deste *dashboard* passam por fornecer ao professor uma análise do progresso do aluno ao longo do tempo e fornecer ao aluno uma visão global da sua evolução, permitindo identificar quais os aspetos que requerem mais estudo de forma a melhorar o seu desempenho. Com este *dashboard*, o aluno e o professor, conseguem identificar quais os domínios e subdomínios em que o aluno demonstra mais dificuldades ou falta de conhecimento, e ainda, através da análise do número de avaliações, por exemplo, o professor consegue inferir se o utilizador é um utilizador que interage muito ou pouco com a plataforma.

- **Dashboard de Controlo de Acessos**

Na Figura 62 está representado um *dashboard* gerado a partir do *data mart* de logs do sistema. No primeiro painel presente neste *dashboard* está representado um gráfico de barras, contendo o número de pedidos para cada página da aplicação para um dado intervalo de tempo. No último painel, é apresentado um mapa contendo o número de alunos que acedeu a plataforma em cada localização.

O *dashboard* de controlo de acessos permite ao administrador analisar os acessos ao sistema. Neste *dashboard* é possível efetuar análise como as seguintes:

- Número de *requests* por página da plataforma dentro de um dado intervalo temporal;
- Número de utilizadores registados na plataforma por localização.

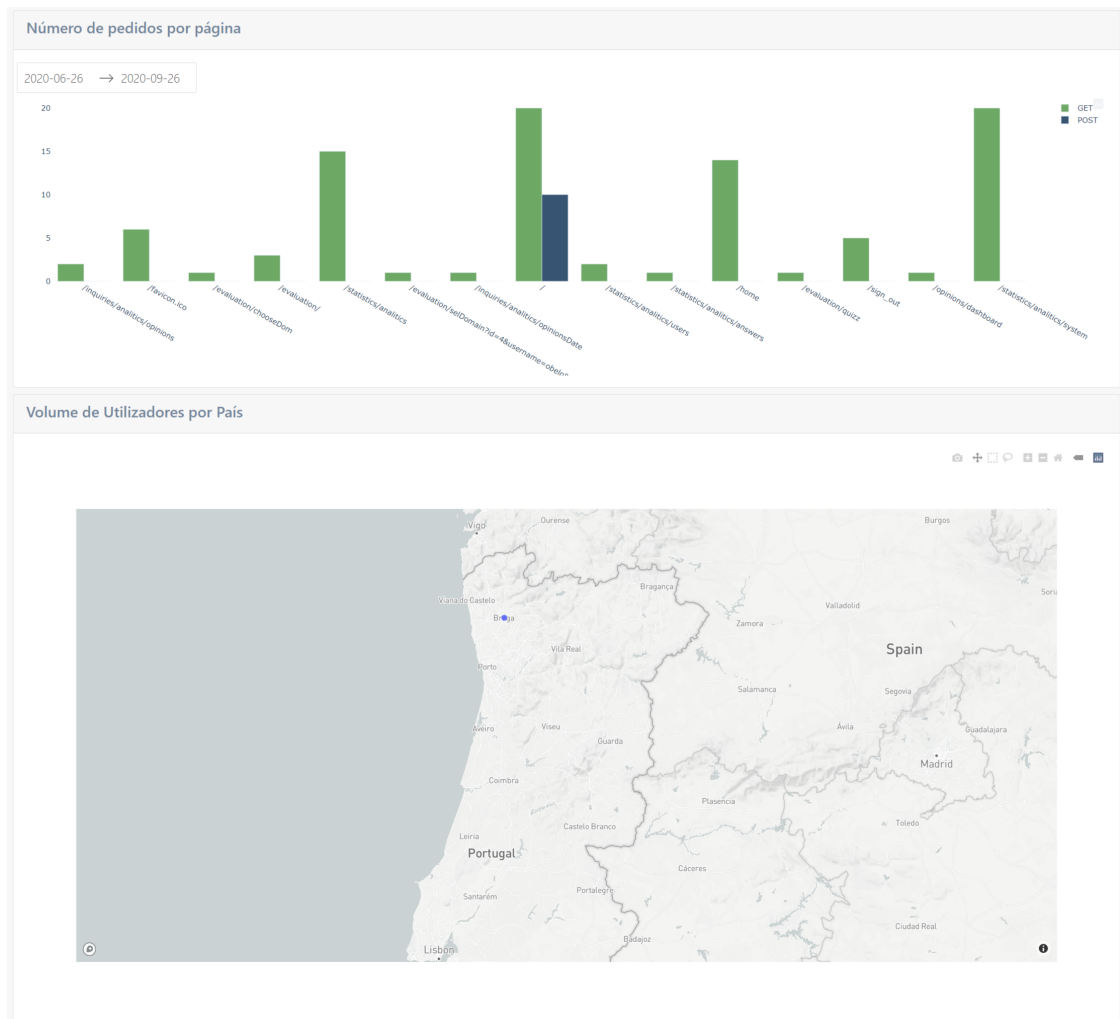


Figura 62: *Dashboard* que contém vistas direcionadas aos acessos da plataforma.

A partir deste *dashboard*, o professor consegue determinar quais as páginas mais acedidas pelos seus alunos, bem como, averiguar o volume de alunos por localização que utilizou a plataforma, e desta forma, determinar em qual a localização a adesão à plataforma foi maior, por exemplo.

CONCLUSÕES E TRABALHO FUTURO

5.1 Conclusões

A integração de uma ferramenta analítica de dados nas plataformas de suporte à decisão que são implementadas é vista como uma ação fundamental, tanto para a sua administração como para os seus utilizadores ou clientes. De facto, a existência de meios de análise de dados nestas plataformas permite aos seus utilizadores avaliar as suas interações, detetar falhas operacionais e de negócio e inferir acerca de formas que permitam alcançar melhores resultados. Usualmente, a implementação deste tipo de meios passa pela implementação de um sistema de *data warehousing*. Este tipo de sistema permite estruturar e armazenar a informação de forma organizada e orientada às diversas áreas que se pretende analisar. Muitas das vezes este tipo de sistemas requiere análise em tempo real, sendo que, para que isto seja possível, a *performance* torna-se um requisito bastante importante num sistema analítico. Desta forma, toda a estrutura de *data warehouse* deve ser pensada de forma a que os acessos que serão necessários efetuar, sejam obtidos rapidamente para serem mostrados ao utilizador o mais rápido possível.

Neste trabalho de dissertação foram apresentadas algumas abordagens para a implementação de um *data warehouse* orientado a documentos, avaliando-se as suas vantagens e desvantagens, de modo a se poder concluir qual das abordagens seria aquela que melhor se adequava a um dado problema. A seleção destes modelos dependerá sempre de variadíssimos fatores, como o volume de dados, o número de dados relacionados, o volume de informação com história e variação, entre outros. Por estes motivos, é importante que a escolha do modelo seja a mais adequada, de forma a que a redundância dos dados não seja um fator muito pesado em termos de volume e que a *performance* das *queries* não seja muito prejudicada, tanto em termos de redundância, como em termos das agregações que sejam necessárias realizar para a obtenção da informação imposta.

No capítulo 3 desta dissertação foram apresentados vários tipos de dimensões com variação, que, de uma forma geral foram apresentadas e discutidas para sistemas relacionais em [Kimball e Ross, 2013]. Para cada um dos tipos apresentados e discutidos foi efetuada uma análise comparativa entre a sua aplicação em sistemas de bases de dados relacionais e sistemas de bases de dados orientados a documentos.

Em dimensões com variação e sem história, nomeadamente dimensões com variação do tipo 1 (Secção 3.5.1), vimos que a aplicação do modelo desnormalizado de dados em que cada documento contém os dados das diferentes dimensões e factos, pode ser problemático em termos de *performance*. Isto porque este tipo de modelo não evita a redundância da informação, permitindo que existam inúmeros documentos referentes ao mesmo valor da chave sujeita a variação, o que conduz a um deterioramento da *performance* à medida que o sistema vai escalando.

Em dimensões com variação do tipo 2 (Secção 3.5.1) vimos que a aplicação em sistemas orientados a documentos foi, também ela, efetuada tendo por base um modelo desnormalizado de dados. Dado que este tipo de dimensão com variação tem história, a abordagem para efetuar o seu registo passou pela adição de um novo documento quando uma variação do par chave-valor ocorre. Este tipo de abordagem implica um grande crescimento de dados no *data warehouse*, sendo que para cada variação do par chave-valor sujeito a variação, um novo documento é inserido.

Na Secção 3.5.1 foi abordado outro tipo de dimensões com variação e história: as dimensões com variação do tipo 3. Contudo a noção de história neste tipo de dimensões difere das dimensões com variação do tipo 2. Em dimensões com variação do tipo 3, a história apenas contém o valor referente ao par chave-valor sujeito a variação que precedeu a sua atualização. Desta forma, o que se obteve para uma modelação orientada a documentos foi a adição de uma nova chave para armazenar o valor antigo. A sua manutenção é bastante semelhante a dimensões com variação do tipo 1, dado que, para cada atualização efetuada ao par chave-valor sujeito a variação, poderão ter que ser atualizados múltiplos documentos.

Uma abordagem um pouco diferente das apresentadas foi analisada na Secção 3.5.1, quando exploramos dimensões com variação do tipo 4. Até essa altura apenas foram abordadas estruturas utilizando modelos desnormalizados de dados, nos quais toda a informação era armazenada num único documento, ao mesmo nível. Nas dimensões com variação do tipo 4 isto já não se verifica. Para este tipo de dimensões foi criada uma mini-dimensão de forma a reduzir a redundância dos dados. A informação presente nesta mini-dimensão foi relacionada com a dimensão, da qual esta surgiu, a partir do identificador presente nos documentos que a compunham. Desta forma, podemos referir que um documento contém uma chave identificadora de outro documento. A redundância fica de facto reduzida quando se opta por esta abordagem, porém os acessos à informação de um sistema deste tipo vão ser mais complexos do que de um sistema que use um formato desnormalizado de dados, dado que as *queries* de consulta vão ser mais complexas e implicam, usualmente, a realização de operações de agregação dos dados. Em relação às dimensões com variação do tipo 5, abordado na Secção 3.5.1, foi possível verificar que quando aplicada a modelos orientados a documentos, a abordagem era semelhante àquela apresentada para dimensões com variação do tipo 4, mantendo os mesmos problemas de implementação.

Na Secção 3.5.1 foram apresentadas as dimensões com variação do tipo 6, onde este tipo foi definido como uma junção das dimensões com variação do tipo 1, 2 e 3. Sendo este tipo de dimensão vista como uma conjugação dos diferentes tipos, seria espectável que os problemas que foram identificados para todos estes tipos fossem, também eles, encontrados em dimensões com variação do tipo 6. Na Secção 3.5.1 é comprovado que os problemas identificados para os diferentes tipos eram mantidos,

quando aplicado a sistemas orientados a documentos. A adição de um novo par chave-valor aos documentos, cujo objetivo foi manter o valor mais recente do par chave-valor sujeito a variação, levantou algumas preocupações, também elas já identificadas para outros tipos. Apesar de nas *queries* de consulta a presença deste par chave-valor ser uma vantagem, pois garantia o acesso ao valor mais atual de uma forma praticamente gratuita, a manutenção deste acarreta um esforço bastante grande no caso de se ter um elevado número de documentos. À semelhança das dimensões com variação do tipo 7 (Secção 3.5.1) e com o intuito de mitigar este esforço, foi sugerida a utilização de uma vista materializada, em que este par chave-valor foi extraído para uma outra coleção e armazenado num outro documento. Desta forma apenas seria necessário atualizar um documento para manter este valor. Porém esta atualização poderia prejudicar a *performance* das *queries* de consulta, pois para obter toda a informação, nomeadamente a do valor mais recente, seria necessário consultar também esta vista materializada. Tudo isto dependeria do propósito para o qual o sistema de *data warehouse* estaria a ser concebido, e quais as *queries* que seriam necessárias executar para responder aos seus requisitos.

Na Secção 3.5.5 foram abordadas as dimensões gigantescas onde se apresentou dois modelos de dados distintos, um deles aplicado a um sistema de base de dados relacional e um outro a um sistema de base de dados orientado a documentos. Para além desta modelação foi efetuada uma análise quantitativa do tempo de execução de uma *query* que consultava uma dimensão que continha 5 milhões de registos. Foi interessante ver que a discrepância em termos de tempo de execução foi bastante acentuada, tendo sido 22 vezes mais rápida quando executada no sistema orientado a documentos. Esta análise permite inferir que a questão da redundância da informação em sistemas orientados a documentos, pode não ser assim tão prejudicial em termos de desempenho das *queries* de consulta, e que, pode ser preferível manter esta redundância em muitas situações em vez de se tentar normalizar os dados e aumentar a complexidade das *queries*.

No Capítulo 4 foram apresentados dois modelos distintos aplicados a sistemas orientados a documentos. Estes modelos foram criados com o objetivo de apurar qual a melhor solução para a introdução dimensões com variação e história no sistema, quando aplicado ao caso de estudo em análise. O primeiro modelo consistiu em dispor toda a informação num único documento, onde o par chave-valor sujeito a variação era uma lista que continha todos os seus valores ao longo do tempo. No segundo modelo foi apresentada uma alternativa em que o histórico do par chave-valor foi deslocado para uma outra coleção. Essa coleção conteria um documento por cada atualização que o par chave-valor com variação sofresse. Numa fase seguinte foi efetuada uma análise de *performance* de algumas consultas aos modelos referidos. Estas consultas surgiram como resposta a três questões colocadas ao sistema, cujo objetivo era simular possíveis requisitos, aos quais, o sistema, teria que conseguir dar resposta. Em relação ao primeiro modelo, em que um único documento continha toda a informação do *data mart*, verificou-se que os resultados foram, no geral, melhores do que o segundo modelo abordado, onde os dados estavam divididos em duas coleções distintas. Na primeira questão apresentada (Secção 4.4), foi possível verificar que o segundo modelo foi-se deteriorando à medida que o volume de dados aumentava, tendo chegado a ser quase 4 vezes pior do que o primeiro modelo apresentado. Isto deve-se ao facto, de no segundo modelo ter sido necessário consultar uma coleção extra para obter toda a informação necessária para responder à questão proposta, enquanto que no primeiro modelo, como a informação estava toda centrada no mesmo documento, a obtenção dessa informação foi imediata.

Na segunda questão proposta verificou-se que à semelhança da questão abordada anteriormente, o primeiro modelo manteve um desempenho constante ao longo dos diversos volumes de dados. Em relação ao segundo modelo, o tempo de execução foi aumentando à medida que o volume de dados aumentava, sendo que, o tempo de execução da *query* executada para este modelo, chegou a ser 40 vezes superior ao tempo de execução da *query* executada para o primeiro modelo. Mais uma vez o problema deveu-se à separação existente entre o documento que contem a dimensão sujeita a variação e o seu histórico. Os resultados obtidos para estas duas questões, levam a crer que o problema de *performance* não estaria diretamente associado ao aumento volume de dados, nem aos filtros existentes sobre os mesmos, mas sim à necessidade de agregação dos dados entre diferentes documentos.

Na última questão apresentada, podemos concluir que o tempo de execução de ambas as *queries* para cada um dos modelos não foi aceitável. Em relação à *query* executada para o primeiro modelo o tempo de execução foi aumentando com o aumento do número de registos no sistema, sendo este independente do aumento do número de utilizadores. Para a *query* executada para o segundo modelo, contrariamente ao que aconteceu para as *queries* abordadas anteriormente, o desempenho foi melhor do que aquele obtido para o primeiro modelo. Tal circunstância deveu-se à menor redundância de dados presente no segundo modelo. No primeiro modelo cada registo presente no *data mart* referente a um determinado utilizador continha sempre o seu histórico, porém, no segundo modelo isto não se verificava, pois todos os documentos inseridos na coleção que armazenava o histórico eram únicos por utilizador, assim sendo, o número de documentos por utilizador nesta coleção era igual ao número de variações associado ao par chave-valor sujeito a variação.

Após esta análise, o caso de estudo que escolhemos para analisar os diferentes modelos permitiu-nos concluir, que, qualquer que seja o modelo escolhido, a sua utilização é condicionada pelo tipo de sistema que pretendemos implementar. Como tal, é importante fazer uma análise da dimensão do sistema, dos vários elementos de dados (tabelas de factos, dimensões, *outriggers*, etc.), determinar quais os dados que vão sofrer variação e quais aqueles que será necessário guardar histórico e, desta forma, concetualizar o modelo tendo em consideração as *queries* que vão ser posteriormente executadas. Como foi possível comprovar na análise feita na Secção 4.4, a modelação da estrutura de dados está sempre condicionada pelo tipo de *queries* que irão ser necessárias para satisfazer os requisitos propostos. Como vimos, *queries* que impliquem agregação de dados de diferentes coleções têm um grande impacto negativo no desempenho do sistema, porém a redundância de dados quando associada a *queries* que impliquem processar e agregar dados, também têm um fraco desempenho. Em contrapartida quando tivemos um grande volume de dados associados a *queries* mais simples, em que o processamento que faziam eram apenas filtros da informação, obtivemos desempenhos positivos, levando a crer que a redundância poderá não ser um problema, tudo depende do propósito do sistema a ser implementado, nomeadamente as necessidades que deve dar resposta.

5.2 Trabalho Futuro

Durante esta dissertação foi feita uma análise detalhada a vários tipos de modelos de dados aplicados a sistemas orientados a documentos. Porém considero que ainda há bastante trabalho a ser feito. Como trabalho futuro seria importante avaliar outros modelos de dados, que sejam adequados para lidar com dimensões com variação e história.

Uma das grandes dificuldades encontradas ao longo desta dissertação, foi a identificação de estruturas de dados que permitissem lidar esse tipo de dimensão, sem degradar a *performance* do sistema, ao ponto de tornar a solução inviável. Porém, não foi encontrada uma solução ideal para lidar com este tipo de estruturas, sendo que, se concluiu que esta estrutura estaria sempre dependente dos requisitos aos quais o sistema teria que responder. Outro ponto que carece de mais exploração, é a definição e implementação de hierarquias em sistemas orientados a documentos. Considero que será benéfico analisar diferentes formas de implementar este tipo de estruturas, nomeadamente a definição das relações hierárquicas entre os diferentes valores armazenados no *data warehouse*, quando aplicado a sistemas orientados a documentos.

REFERÊNCIAS

- [1] A. Aggoune e M. S. Namoune. A method for transforming object-relational to document-oriented databases. *2020 2nd International Conference on Mathematics and Information Technology (ICMIT)*, páginas 154–158, 2020. doi: 10.1109/ICMIT47780.2020.9047011.
- [2] A. C. Alvarado, E. V. Santacruz, e M. G. Zúñiga. Construction of a basic intelligent agent. *2017 Intelligent Systems Conference (IntelliSys)*, páginas 341–345, 2017. doi: 10.1109/IntelliSys.2017.8324315.
- [3] Michael Beatty. Ibm cloud object storage, 2017. URL <https://www.slideshare.net/MichaelBeatty/ibm-cloud-storage-cleversafe>. Acedido em [07 Novembro 2019].
- [4] O. Belo. Lição de síntese, provas de agregação. “*Modelação Dimensional de Dados*”, Universidade do Minho, Março 2013.
- [5] O. BELO, J. COELHO, e L. FERNANDES. An evolutionary software tool for evaluating students on undergraduate courses. “*In Proceedings of 13th Annual International Conference of Education, Research and Innovation (ICERI'2019)*”, Seville, Spain. 11-13, November 2019.
- [6] R. Britto, W. G. de Oliveira Filho, C. G. Barros, e E. C. Lopes. Intelligent tutor system model applied to basic electronics. *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–5, Junho 2017. doi: 10.23919/CISTI.2017.7975832.
- [7] Inc. Carnegie Learning, 1998. URL <https://www.carnegielearning.com/products/software-platform/mathia-learning-software/>. Acedido em [13 Novembro 2021].
- [8] M. Chavalier, M. El Malki, A. Kopliku, O. Teste, e R. Tournier. Document-oriented data warehouses: Models and extended cuboids, extended cuboids in oriented document. *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, páginas 1–11, Junho 2016. doi: 10.1109/RCIS.2016.7549351. URL <https://ieeexplore.ieee.org/document/7549351>. Acedido em [05 Dezembro 2019].
- [9] S. Chickerur, A. Goudar, e A. Kinnerkar. Comparison of relational database with document-oriented database (mongodb) for big data applications. *2015 8th International Conference on Advanced Software Engineering Its Applications (ASEA)*, páginas 41–47, 2015. doi: 10.1109/ASEA.2015.19.
- [10] Commission Of The European Communities. Communication from the commission to the council and the european parliament, the elearning action plan, designing tomorrow’s education. 2001. URL <https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2001:0172:FIN:EN:PDF>. Acedido em [19 Dezembro 2019].

- [11] Fluentd. What is fluentd?, n.d. URL <https://www.fluentd.org/architecture>. Acedido em [13 Novembro 2021].
- [12] Matteo Golfarelli, Dario Maio, e Stefano Rizzi. The dimensional fact model: A conceptual model for data warehouses. *Int. J. Cooperative Inf. Syst.*, 7:215–247, 06 1998. doi: 10.1142/S0218843098000118.
- [13] S. Hamidullah e P. Samiullah. Intelligent tutoring system: Approaches, researches and e-learning solution. *2015 IEEE 8th International Workshop on Computational Intelligence and Applications (IWCIA)*, páginas 53–58, Nov 2015. doi: 10.1109/IWCIA.2015.7449462.
- [14] Dongming Xu Wayne W. Huang Huaqing Wang Jon Heales. Enhancing e-learning effectiveness using an intelligent agent-supported personalized virtual learning environment: An empirical investigation, 2014. URL <http://dx.doi.org/10.1016/j.im.2014.02.009>. Acedido em [14 Fevereiro 2021].
- [15] IDC. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, 2012. URL <https://www.emc.com/leadership/digital-universe/2012iview/call-to-action.html>. Acedido em [07 Novembro 2019].
- [16] IMDb, 1990. URL <https://www.imdb.com/>. Acedido em [13 Novembro 2021].
- [17] J. Kapounova e K. Kostolanyova. Specific tool for evaluation of elearning. *2011 14th International Conference on Interactive Collaborative Learning*, páginas 423–427, Sep. 2011. doi: 10.1109/ICL.2011.6059619.
- [18] Aytürk Keleş, Rahim Ocak, Ali Keleş, e Aslan Gülcü. Zosmat: Web-based intelligent tutoring system for teaching–learning process, 2009. URL <https://www.sciencedirect.com/science/article/pii/S0957417407005532>. Acedido em [15 Novembro 2019].
- [19] Lorna Earl Kim Schildkamp, Meu Kuin Lai. Data-based decision making in education: Challenges and opportunities, 2013. ISBN 978-94-007-4816-3.
- [20] R. Kimball e J. Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Wiley, 2011. ISBN 9781118079683. URL <https://books.google.pt/books?id=TCLfzU2ilVkC>.
- [21] R. Kimball e M. Ross. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley, 2013. ISBN 9781118530801. URL <https://books.google.pt/books?id=WMEqTf2lK84C>.
- [22] Alex Koohang e Keith Harman. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, 2005. URL <http://www.inform.nu/Articles/Vol8/v8p075-086Kooh.pdf>. Acedido em [07 Novembro 2019].
- [23] Bernard Marr, 2020. URL <https://www.forbes.com/sites/bernardmarr/2020/10/16/the-amazing-ways-duolingo-is-using-artificial-intelligence-to-deliver-free-language-learning/>.
- [24] MongoDB. URL <https://www.mongodb.com/>. Acedido em 19-07-2021.

- [25] Inc. MongoDB, 2009. URL <https://www.mongodb.com/database-management-system>. Acedido em [23 Agosto 2020].
- [26] C. Pahl. *Architecture Solutions for E-Learning Systems*. Gale virtual reference library. Information Science Reference, 2007. ISBN 9781599046358. URL https://books.google.pt/books?id=e_uKwQpX9XUC.
- [27] D. Pereira, P. Oliveira, e F. Rodrigues. Data warehouses in mongodb vs sql server: A comparative analysis of the querie performance. *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–7, Junho 2015. doi: 10.1109/CISTI.2015.7170400.
- [28] P. Ponniah. *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*. Wiley, 2004. ISBN 9780471463894. URL <https://books.google.pt/books?id=n2nIM011TQ0C>.
- [29] P. D. Reddy e M. Sasikumar. Student model for an intelligent language tutoring system. *2014 IEEE 14th International Conference on Advanced Learning Technologies*, páginas 441–443, July 2014. doi: 10.1109/ICALT.2014.132.
- [30] G. S. Santos e J. Jorge. Interoperable intelligent tutoring systems as open educational resources. *IEEE Transactions on Learning Technologies*, 6(3):271–282, July 2013. doi: 10.1109/TLT.2013.17.
- [31] Lauren Schaefer. What is nosql?, n.d. URL <https://www.mongodb.com/nosql-explained>. Acedido em [13 Novembro 2021].
- [32] R. Sharma e S. Kaushik. *DATABASE MANAGEMENT SYSTEM*. Horizon Books (A Division of Ignited Minds Edutech P Ltd), 2015. ISBN 9789384044800. URL <https://books.google.pt/books?id=QQVKDwAAQBAJ>.
- [33] Warren Thornthwaite. Design tip #127 creating and managing mini-dimensions. 2010. URL <https://www.kimballgroup.com/2010/09/design-tip-127-creating-and-managing-mini-dimensions/>. Acedido em [18 Agosto 2020].
- [34] Luis von Ahn. Duolingo, 2011. URL <https://www.duolingo.com/>. Acedido em [13 Novembro 2021].
- [35] Paweł Wancerz, Marek e Wancerz. History management of data – slowly changing dimensions. *INFORMATYKA, AUTOMATYKA, POMIARY W GOSPODARCE I OCHRONIE ŚRODOWISKA*, 3, 09 2013. doi: 10.35784/iapgos.1464.
- [36] Business Wire. Mathia by carnegie learning named “best artificial intelligence solution” in 2019 edtech awards, 2019.