Universidade do Minho
Escola de Engenharia
Departamento de Informática

Ana Marta Santos Ribeiro

Conceção e Implementação de Data Warehouses Baseados em Grafos

Maio 2022

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Ana Marta Santos Ribeiro

Conceção e Implementação de Data Warehouses Baseados em Grafos

Dissertação de mestrado
Mestrado Integrado em Engenharia Informática

Dissertação realizada sob orientação de
Orlando Manuel de Oliveira Belo

Maio 2022

AGRADECIMENTOS

Durante a realização desta dissertação pude contar com o apoio de diversas pessoas e às quais sou eternamente grata. Desta forma, quero deixar espresso um agradecimneto especial a cada uma delas.

Em primeiro lugar, gostaria de agradecer ao meu orientador, o professor Dr. Orlando Belo, da Universidade do Minho, pela disponibilidade demonstrada e por todo o acompanhamento que me fez durante este processo. Obrigada por todos os conselhos e sugestões que me deu e que me permitiram não só terminar este projeto da melhor forma possível, mas que também contribuíram para o meu crescimento.

Agradeço também aos meus pais que sempre me apoiaram e que durante todos estes anos se esforçaram para me dar todas as condições necessárias para ingressar no ensino superior.

Obrigada à minha irmã por todo o incentivo, pela paciência e por acreditar em mim, mesmo quando eu própria não o fiz.

Por fim, queria ainda agradecer aos meus amigos que sempre me apoiaram, em especial àqueles com quem tive a oportunidade de partilhar este percurso e com quem realizei alguns dos trabalhos ao longo do curso. Foi um prazer poder crescer e aprender ao vosso lado.

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.

RESUMO

Os *data warehouses* são concebidos para guardar uma grande quantidade de dados, usualmente provenientes de diversas fontes, de um modo consistente e integrado. Esta característica aliada à facilidade com que é possível aceder a todos os dados, faz com que estes surjam como uma ferramenta de eleição no que se refere a processos de tomada de decisão. Para as organizações é fundamental ter disponível a sua informação de um modo estruturado e coerente, para que seja possível sustentar adequadamente os seus processos de tomada de decisão, a qualquer momento. Contudo, atualmente a quantidade de dados produzida pelas empresas, bem como a sua complexidade, tem crescido exponencialmente, o que faz com que os *data warehouses* mais convencionais, apoiados em bases de dados relacionais e em modelos bem estruturados, apresentem algumas debilidades face a esta nova realidade. Tendo isto em consideração, nesta dissertação pretende-se explorar uma nova realidade na implementação de *data warehouses*: a modelação e construção de *data warehouses* baseados em grafos. Dadas as excelentes características que os grafos apresentam para lidar com situações que envolvam grandes volumes de dados fortemente relacionados, esperamos que a sua aplicação no domínio dos sistemas de *data warehousing* nos conduza a uma nova metodologia para suporte à modelação e desenvolvimento de *data warehouses* baseados em grafos.

PALAVRAS-CHAVE Sistemas de *Data Warehousing*, *Data Warehouses*, Modelos Dimensionais de Dados, Grafos, Modelação de *Data Warehouses* Baseados em Grafos, Metodologias de Desenvolvimento de *Data Warehouse*.

ABSTRACT

Data warehouses are designed to store a large amount of data, usually from a variety of sources, in a consistent and integrated way. This feature combined with the ease with which it is possible to access all data, makes these to appear as a tool of choice with regard to decision-making processes. For associations it is fundamental to have the information available in a structured and coherent way to make possible to adequately sustain your decision-making processes at any time. However, currently the amount of data produced by companies, as well as their complexity, has grown exponentially which makes data warehouses more conventionals, based on relational databases and well-structured models, present some weaknesses in this new reality. Taking this into account this dissertation will explore a new reality in the implementation of the data warehouses: the modeling and construction of graph-based data warehouses. Considering the excellent characteristics that graphs present to deal with situations involving large volumes of strongly related data, we hope that their application in the field of data warehousing systems leads us to a new methodology to support the modeling and development of graph-based data warehouses.

KEYWORDS Data Warehousing Systems, Data Warehouses, Dimensional Data Models, Graphs, Modeling of Data Warehouses Based on Graphs, Data Warehouse Development Methodologies.

CONTEÚDO

I MATERIAL INTRODUTÓRIO 1

1 INTRODUÇÃO 2
- 1.1 Contextualização 2
- 1.2 Motivação e Objetivos 3
- 1.3 Trabalho Realizado 3
- 1.4 Organização do Documento 4

II CONTEÚDO PRINCIPAL 6

2 SISTEMAS BASEADOS EM GRAFOS 7
- 2.1 Principais características e aplicações 8
- 2.2 Sistemas de Gestão de Base de Dados Suportadas por Grafos 10
  - 2.2.1 Neo4J 10
  - 2.2.2 OrientDB 11
  - 2.2.3 Microsoft Azure Cosmos DB 12
  - 2.2.4 JanusGraph 13
  - 2.2.5 Seleção do Sistema 14
- 2.3 Modelo de dados suportados por Grafos 15
  - 2.3.1 Implementação de Sistema de Dados com Grafos 16

3 SISTEMAS DE DADOS MULTIDIMENSIONAIS 18
- 3.1 Processamento Analítico (OLAP) 19
- 3.2 Estruturas de Dados Multidimensionais 21
- 3.3 Operações sobre Estruturas Multidimensionais 25
- 3.4 Estruturas Multidimensionais para Grafos 27
- 3.5 Implementação 32

4 HIPERCUBOS EM HIPERGRAFOS 36
- 4.1 Príncipios e Modelos 36
- 4.2 Representação de Hipercubos em Hipergrafos 37
- 4.3 Queries sobre Hipercubos em Hipergrafos 38
- 4.4 Modelação e implementação do Caso de Aplicação 40

4.5 Análise dos Resultados Obtidos 45

5 CONCLUSÕES E TRABALHO FUTURO 49

5.1 Conclusões 49

5.2 Trabalho Futuro 51

LISTA DE ACRÓNIMOS

ACID Atomicidade, Consistência, Isolamento, Durabilidade. 8, 10

GDBMS *Graph Database Management System*. 9

NoSQL *Not only SQL*. 2, 3, 7, 8, 27, 49, 50

OLAP *Online Analytical Processing*. 19, 20, 25, 31, 38, 43, 50, 52
OLTP *Online Transaction Processing*. 20

RDF *Resource Description Framework*. 3, 7

SQL *Structured Query Language*. 2, 9, 11

LISTA DE FIGURAS

Figura 1 Exemplo de um grafo 8
Figura 2 Ambiente de trabalho em Neo4J 11
Figura 3 Ambiente de trabalho em OrientDB 12
Figura 4 Ambiente de trabalho em Microsoft Azure Cosmos DB 13
Figura 5 Exemplo do ambiente de trabalho em JanusGraph 14
Figura 6 Exemplo de um grafo de propriedades 15
Figura 7 Representação dos nodos e relacionamentos 16
Figura 8 Modelo de Dados para uma aplicação de vendas 17
Figura 9 Visualização do grafo implementado em Neo4J 17
Figura 10 Arquitetura Data Warehouse adaptado de (Malinowski e Zimányi, 2008) 19
Figura 11 Exemplo de uma hierarquia total e de uma parcial 22
Figura 12 Dataset referente aos produtos 23
Figura 13 Dataset das Vendas 23
Figura 14 Dataset que define os produtos das vendas 23
Figura 15 Modelo multidimensional para uma aplicação de vendas 24
Figura 16 Operações Slice e Dice (Adaptado de (Bolt e der Aalst, 2015)) 25
Figura 17 Operações Roll-up e Drill-down (Adaptado de (Bolt e der Aalst, 2015)) 26
Figura 18 Operações Pivot (Adaptado de (Bolt e der Aalst, 2015)) 26
Figura 19 Representação do modelo multidimensional num grafo 29
Figura 20 Malha de cubóides que forma um cubo 4D para as dimensões ”Cliente”, ”Funcionario”, ”Dia”e ”Produto” 31
Figura 21 Exemplo de uma consulta ao modelo base 32
Figura 22 Grafo obtido com a criação dos nodos e relacionamentos referentes aos cuboides 33
Figura 23 Resultado de execução da query 4 35
Figura 24 Hiperaresta para representar os cubóides da dimensão Funcionário 38
Figura 25 Exemplo de hiperarestas que nos permitem responder às queries base definidas 40
Figura 26 Exemplo do hipergrafo para a dimensão cliente 42
Figura 27 Representação do modelo multidimensional num hipergrafo 43
Figura 28 Representação do modelo multidimensional num hipergrafo 45
Figura 29 Resultado da query quatro para o grafo de propriedades 47
Figura 30 Resultado da query quatro para o hipergrafo 48

LISTA DE TABELAS

Tabela 1 Bases de dados suportadas por grafos (DBEngine) 14
Tabela 2 Principais diferenças entre os sistemas OLAP e OLTP 20
Tabela 3 Tempos de execução das queries para o primeiro modelo 46
Tabela 4 Tempos de execução das queries para o segundo modelo 46
Tabela 5 Tempo médio de cada operação 47
Tabela 6 Quantidade de nodos envolvidos na construção dos modelos 48

Parte I

MATERIAL INTRODUTÓRIO

1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Atualmente, uma das principais preocupações que as organizações enfrentam é a dificuldade em extrair informação útil da grande quantidade de dados que produzem e que guardam nos seus diversos repositórios. Frequentemente, as empresas encontram-se estruturadas em departamentos em que cada um deles possui a sua própria base de dados, que foi construída e evoluindo de forma a poder dar resposta às suas tarefas operacionais, isto é, às operações diárias que permitem o funcionamento do seu negócio. Todavia, estas revelam-se pouco adequadas quando é necessário aceder à sua informação como um todo, para suportar, por exemplo, um processo de tomada de decisão que envolva toda a informação que está contida em cada uma delas. De modo a colmatar este problema, algumas organizações optam por implementar *data warehouses*, que são, basicamente, sistemas de dados centralizados e autónomos, organizados por assunto, integrados, temporais, e não voláteis (Inmon, 2005). O conteúdo de um *data warehouse* resulta da integração de um conjunto de dados heterogéneos provenientes de diversas fontes, que são analisados e tratados de forma consistente. Usualmente, os dados carregados para os *data warehouses* são históricos, pelo que representam transações que já aconteceram e, como tal, não sofrem alterações. São factos. Estes repositórios assentam normalmente em sistemas de bases de dados relacionais e em modelos de dados bem definidos, que foram desenvolvidos e implementados no sentido de permitirem consultas rápidas e eficientes para suportarem os processos de tomada de decisão.

Os sistemas baseados em grafos, representam um dos quatro tipos de base de dados NoSQL (*Not Only SQL*) mais predominantes, que se caracterizam tipicamente pela sua capacidade de armazenamento e processamento de elevadas quantidades de dados. Estes sistemas utilizam o grafo como modelo de dados (Liu e Vitolo, 2013). As bases de dados orientadas a grafos permitem tratar de uma das principais debilidades das bases de dados relacionais que se prende com a modelação e consulta de relacionamentos complexos (Tardivel). Estas base de dados suportadas por grafos lidam com os relacionamentos como se tratassem de objetos separados, o que lhes permite obter um bom desempenho e aumentar a legibilidade do modelo de dados. Além disso, também são bastante flexíveis, escaláveis e disponíveis. De salientar, que as bases de dados suportadas por grafos disponibilizam as suas próprias linguagens já que o SQL (*Structured Query Language*) não é a mais adequada para lidar com a informação mantida em grafos (Tardivel). A aplicação deste tipo de base de dados pode ser vista em diversos casos, como, por exemplo, nas redes sociais, em sistemas de deteção de

fraude, *profiling* de clientes, sistemas de recomendação, entre outras coisas mais. O aumento exponencial na quantidade de dados gerados que se verificou ao longo dos últimos anos, bem como a complexidade que têm revelado, criaram uma realidade na qual os sistemas baseados em grafos podem revelar-se muito vantajosos.

1.2 MOTIVAÇÃO E OBJETIVOS

Na sociedade atual é cada vez mais evidente o poder da informação, que representa uma fonte de obtenção de conhecimento fundamental. Hoje em dia, informação é poder. Nos processos de tomada de decisão, quanto mais informação fidedigna houver disponível, mais segura e fundamentada será a escolha tomada. Esta conjuntura faz com que, hoje em dia, cada vez mais as organizações optem por recorrer a *data warehouses* para armazenar todos os dados possíveis relativos ao seu negócio e, dessa forma, conseguirem adquirir o máximo de conhecimento possível quando necessário. Contudo, nos últimos anos tem-se observado um aumento da produção dos dados gerados diariamente, algo provocado pela emergência de novas tecnologias e desenvolvimentos de aplicações – e.g. redes sociais, sistemas de monitorização, disponibilização de conteúdos online, ou a proliferação generalizada de sítios de comércio eletrónico. Este crescimento exponencial tem exposto algumas das limitações dos sistemas de *data warehousing* convencionais, que, na sua grande maioria, estão implementados em sistemas de base de dados relacionais. Aliado a este crescimento surgiu também uma grande diversidade no tipo dos dados disponibilizados. De forma a atenuar problemas como os referidos, está-se a tentar mudar o rationale da sua implementação, recorrendo-se a novos modelos lógicos, mais flexíveis e capazes de gerir grandes quantidades de dados não estruturados, como é o caso, de um modo generalizado, das bases de dados NoSQL, entre as quais se incluem, os grafos (Lefons et al., 2014).

Nesta dissertação definiu-se um novo processo especialmente orientado para o desenvolvimento de *data warehouses* baseados em grafos. Após uma análise cuidadosa das metodologias de desenvolvimento existentes, idealizou-se um novo modelo de dados multidimensional suportado por grafos. Depois, seguindo a metodologia desenvolvida realizou-se a modelação conceptual e a implementação de um *data warehouse*. Posteriormente, efetuou-se uma análise detalhada à metodologia desenvolvida.

1.3 TRABALHO REALIZADO

O processo adotado para o desenvolvimento desta dissertação assentou essencialmente em três fases. Primeiro, foi necessário estudar o tema e o contexto dos *data warehouses*, de modo a aprofundar todos os princípios base desta dissertação. Posteriormente, foi preciso delinear a forma como um modelo multidimensional pode ser representado num modelo suportado por grafos. Por fim, foi implementado o modelo multidimensional num grafo e foram definidas um conjunto de operações para analisar o comportamento do mesmo.

A primeira fase de desenvolvimento, caracterizou-se pelo estudo e análise de todos os conceitos envolvidos neste projeto. Assim, foi indispensável uma análise cuidada aos sistemas baseados em grafos, de modo a conhecer os modelos existentes, como o de propriedades, os hipergrafos ou os RDF (*Resource Description Framework*). Tornou-se também essencial a investigação sobre os sistemas de gestão de base de dados existentes

no mercado atualmente, de forma a escolher o mais adequado para a implementação do *data warehouse*. Uma vez analisados os sistemas baseados em grafos, foi estudado o contexto de aplicação aos *data warehouses*. Foi averiguada a sua utilidade, o tipo de dados armazenados, os modelos tradicionalmente adotados, as operações realizadas sobre os mesmos, entre muitas outras coisas. Para podermos implementá-lo num modelo suportado por grafos, primeiro foi fundamental conhecer o seu comportamento habitual, para que o mesmo se refletisse no modelo desenvolvido. É ainda importante garantir que as suas principais características se mantenham independentemente do modelo que o suporta, seja relacional, grafos, documentos ou outros.

Terminada a fase de investigação de todo o contexto desta dissertação, começou-se a definir as principais etapas da metodologia a desenvolver, nomeadamente os seus fundamentos teóricos e práticos. De seguida, foi delineado um modelo multidimensional suportado por grafos. Foi ainda definido um conjunto de operações com base no modelo multidimensional estabelecido para se poder analisar o modo como se navega no grafo. Nesta fase foram idealizados dois modelos. O primeiro assentou-se inteiramente num grafo de propriedades e o segundo surgiu como uma derivação do primeiro, baseado num grafo de propriedades e num hipergrafo.

A fase final consistiu na implementação de ambos os modelos projetados e na análise dos resultados obtidos pelos mesmos. Desta forma, para cada um dos modelos foram desenvolvidas um conjunto de consultas típicas associadas a um sistema de *data warehousing*. Com base nos resultados obtidos estabeleceu-se uma comparação entre os modelos, quer com base no armazenamento necessário quer nos tempos de execução das *queries*.

1.4 ORGANIZAÇÃO DO DOCUMENTO

Para além do presente capítulo, esta dissertação encontra-se estruturada em mais quatro capítulos, nomeadamente:

- Capítulo 2 - Sistemas Baseados em Grafos. Neste capítulo são abordadas as principais características e aplicações destes sistemas orientados para grafos, bem como os sistemas de gestão de base de dados mais utilizados que suportam grafos. É ainda efetuado um estudo sobre os modelos de grafos, nomeadamente, sobre os grafos de propriedades. Por fim, a título de exemplo, é apresentada a implementação de um base de dados suportada por grafos.

- Capítulo 3 - Sistemas Multidimensionais. Após o capítulo dedicado a grafos, fazemos aqui uma introdução aos sistemas multidimensionais de dados e à sua arquitetura seguida de uma explicação detalhada acerca do modelo de dados. É ainda realizada uma abordagem à sua aplicação em grafos. Posteriormente, é analisado o modo como podemos representar um modelo multidimensional num modelo suportado por grafos. É apresentado o caso de estudo que será utilizado no âmbito desta dissertação, bem como a proposta de um modelo multidimensional para o mesmo e a sua representação num modelo suportado por grafos. No fim, é ainda abordada a implementação do mesmo.

- Capítulo 4 – Hipergrafos. Neste capítulo são abordadas as estruturas de hipergrafos e o modo como as podemos aproveitar para representar um cubo. É ainda delineada a forma como podemos construir

o modelo multidimensional através de um hipergrafo. Posteriormente, é demonstrada a implementação do hipergrafo para o caso de estudo definido. Por fim, é realizada uma análise dos resultados obtidos, estabelecendo uma comparação entre os modelos apresentados ao longo do documento. É avaliado o desempenho de cada um através de um conjunto de consultas padrão definidas.

- Capítulo 5 - Conclusões e Trabalho Futuro. Neste último capítulo fazemos uma breve síntese e análise crítica do trabalho que foi desenvolvido no âmbito desta dissertação, bem como uma reflexão sobre algumas linhas de trabalho para desenvolvimento futuro.

Parte II

CONTEÚDO PRINCIPAL

2

SISTEMAS BASEADOS EM GRAFOS

As bases de dados suportadas por grafos constituem um dos quatro tipos mais proeminentes de base de dados NoSQL (Studio3T), que surgiram para colmatar alguns dos problemas relacionados com a quantidade, variedade e complexidade da informação existente atualmente. Este tipo de base de dados é sustentada pela teoria dos grafos, que utiliza um conjunto de estruturas matemáticas para fundamentar os relacionamentos entre objetos (Das et al., 2020). Um grafo é uma estrutura representada por vértices ou nodos, e arestas ou arcos, que permitem relacionar os nodos. Estas base de dados têm como objetivo tirar partido das principais características destes componentes para o armazenamento de dados. É de salientar que os grafos representam a informação de um modo simples e intuitivo para o ser humano, o que facilita a sua interpretação e modelação. Os grafos são facilmente representados graficamente, pelo que se revelam preponderantes para a visualização e análise dos dados. Posto isto e face à capacidade de representar dados interligados, nos últimos anos, tem se verificado um crescimento deste tipo de base de dados, sendo que várias aplicações têm migrado os seus dados para modelos sustentados por grafos. Contudo, deve-se referir que existem vários modelos de grafos, como por exemplo, os grafos de propriedades, os hipergrafos ou os grafos de armazenamento triplo (RDF). Os grafos de propriedades, encontram-se entre os mais comuns, são considerados multigrafos direcionados, nos quais tanto os nodos como os relacionamentos podem possuir um conjunto de propriedades, que permitem adicionar informação extra (Das et al., 2020). Os hipergrafos caracterizam-se pelos seus relacionamentos que permitem conectar mais do que dois nodos (Das et al., 2020). Por sua vez, nos armazenamentos triplos ou RDF, os dados são armazenados num formato triplo, sujeito-predicado-objeto (Das et al., 2020). De referir que as características destes modelos podem convergir de modo a obter uma solução mais completa, por exemplo podemos ter um hipergrafo de propriedades, no qual os nodos e os relacionamentos são constituídos por propriedades e os relacionamentos podem envolver mais do que dois nodos. Na imagem apresentada de seguida, é possível observar um exemplo de um grafo, composto por quatro vértices e quatro arestas.

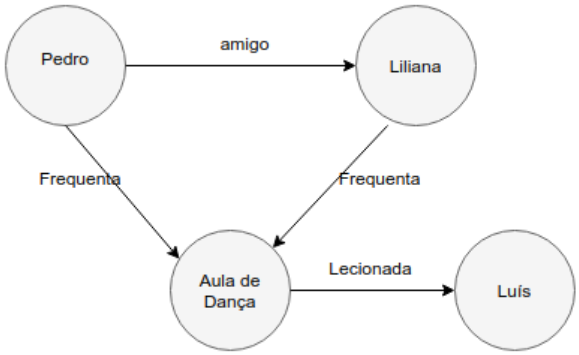

Figura 1: Exemplo de um grafo

2.1 PRINCIPAIS CARACTERÍSTICAS E APLICAÇÕES

Nos últimos anos, tem se verificado um crescimento exponencial das bases de dados orientadas a grafos. Estas caracterizam-se pela sua flexibilidade e performance. Como não necessitam de possuir um esquema de dados bem definido é possível alterá-lo de uma forma dinâmica (Das et al., 2020). Assim, são facilmente incluídos novos dados que não necessitam de possuir uma estrutura específica, como acontece com as bases de dados relacionais. Nos dias de hoje a informação é cada vez mais variada e provém de inúmeras fontes, pelo que esta particularidade se revela essencial para a garantir a integração de toda a informação. Por exemplo, um *post* do *facebook* tanto pode ser um texto, como uma imagem, um vídeo ou um link. Qualquer um destes formatos pode ser facilmente integrável num grafo na criação do nodo, ao contrário do que se verifica nas tabelas das bases de dados tradicionais, nas quais cada uma tem um conjunto de atributos com o tipo de dados bem definido. A simplicidade com que os relacionamentos se encontram definidos numa estrutura de um grafo faz com que estes sejam dos mais adequados para dados interligados (Rawat e Kashyap, 2017).

Ao nível do armazenamento, num grafo, todos os nodos mantêm uma referência direta para os seus nodos adjacentes, o que se revela menos custoso do que a implementação de índices globais (Pokorný, 2017). Isto permite simplificar as travessias, na medida em que apenas necessitamos de fazer a pesquisa para o nodo inicial visto que os restantes são obtidos através dos apontadores para os nodos adjacentes. Desta forma, os grafos permitem consultas extremamente eficazes, dado que apenas dependem das entidades e dos relacionamentos envolvidos na pesquisa, ao invés das bases de dados relacionais que dependem do número de *joins* a realizar e da quantidade de dados armazenados em cada tabela. Esta é, pois, uma das características mais importantes na perspetiva do utilizador final, uma vez que este espera respostas rápidas a todos os seus pedidos independentemente da complexidade ou do volume de dados a analisar. Assim, num grafo dois nodos podem partilhar inúmeros relacionamentos de diferentes tipos sem sacrificar a performance do sistema. Ao contrário da maior parte das bases de dados NoSQL, geralmente, as bases de dados suportadas por grafos respeitam as propriedades ACID (Atomicidade, Consistência, Isolamento Durabilidade), pelo que garantem a consistência e a confiabilidade dos dados. À medida que o grafo aumenta, os conflitos existentes entre transações simul-

tâneas vão diminuindo, uma vez que é possível distribuir a sobrecarga ao longo do grafo. Uma das principais características das bases de dados NoSQL é a sua capacidade para armazenar a grande quantidade de dados que é gerada atualmente e que já demonstra ser um problema para as bases de dados relacionais que apenas suportam escalabilidade vertical.

A linguagem utilizada nas bases de dados relacionais é a SQL (*Structured Query Language*). Esta não se adequa a operações de travessia de grafos, pelo que os GDBMS (*Graph Database Management System*) tiveram de criar a sua própria linguagem, como por exemplo a *Cypher* (Cypher) ou a *Gremlin* (Gremlin). A falta de uma linguagem padrão, como a SQL nas base de dados relacionais, dificulta o processo de migração entre produtos e obriga a uma constante aprendizagem por parte dos seus utilizadores. Nesta perspetiva, seria importante a existência de uma linguagem comum a todos os sistemas. Outro dos desafios deste tipo de base de dados prende-se com a complexidade de distribuir um grafo por várias máquinas tentando que os relacionamentos abranjam o menor número possível de máquinas para depois não terem um impacto negativo nas consultas.

O modelo de dados baseado em grafos tem ganho o seu espaço no mercado e cada vez mais tem sido uma opção. É um modelo versátil que, hoje em dia, já está a ser aplicado em inúmeras áreas, como por exemplo:

- Redes Sociais - Para modelar as relações entre utilizadores e as suas atividades (Das et al., 2020).
- Redes Biológicas - Armazenamento de informações relativas a proteínas, enzimas, genes e outros conceitos desta área que se revelam bastante complexos e interligados.
- Mecanismos de Recomendação - Atualmente a maior parte das empresas online recomendam serviços ou produtos que consideram ser relevantes para os seus clientes. Desta forma, é fundamental um mecanismo capaz de correlacionar inúmeros dados e de detetar rapidamente os novos interesses dos mesmos.
- Deteção de Fraudes - Tipicamente armazenam informações relativas aos clientes e às suas atividades, como por exemplo novas contas e pedidos de empréstimos. Para que a deteção seja efetuada em tempo real é necessária uma análise rápida aos dados e às suas associações, de modo a identificar comportamentos suspeitos ou referências entre utilizadores já designados como suspeitos e assim evitar invasões de conta, lavagem de dinheiro entre outros problemas.
- Análise de Redes de Tecnologia de Informação - Bastante utilizados nas telecomunicações, armazenam informações de configuração para alertar os operadores sobre possíveis pontos de falhas e para reduzir o tempo de resolução de problemas. Tipicamente as infraestruturas de tecnologia de informação possuem interdependências complexas e estão em constante crescimento.
- Gestão de Identidades e Acessos - Armazenam informações acerca dos diferentes tipos de utilizadores, recursos (arquivos, produtos, dispositivos de rede) juntamente com um conjunto de permissões que controlam o acesso a esses recursos. Basicamente define que utilizadores podem aceder aos diversos recursos e em que circunstâncias.

2.2 SISTEMAS DE GESTÃO DE BASE DE DADOS SUPORTADAS POR GRAFOS

Nesta secção é realizado um estudo sobre os Sistemas de Gestão de Bases de Dados (SGBD) suportados por grafos que mais são utilizados atualmente. Com este estudo, pretende-se, basicamente, conhecer as principais características destes novos SGBD. Segundo o portal *DBEngine*, em janeiro de 2021, os SGBD suportados por grafos mais utilizados foram (DBEngine):

1. *Neo4j* (Neo4J)
2. *Microsoft Azure Cosmos DB* (Markjbrown)
3. *OrientDB* (OrientDB)

Neste estudo, foi ainda incluído o JanusGraph (JanusGraph) que aparece em quinto lugar neste ranking, mas que depois do Neo4J é a primeira que não apresenta um multimodelo, ou seja, é exclusivamente para modelos orientados para grafos.

2.2.1 *Neo4J*

O *Neo4J* é um produto *open-source*, implementado em *Java*, que permite o armazenamento de dados em estruturas de grafos. No âmbito das bases de dados orientadas para grafos é a mais utilizada atualmente e das mais reconhecidas, pelo que se encontra em constante desenvolvimento. Para além disso, existe muita informação sobre o *Neo4J*, o que facilita a sua aprendizagem e aplicação (Das et al., 2020). Este suporta os modelos dos grafos de propriedades, em que tanto os nodos como os relacionamentos são constituídos por um conjunto de propriedades (Rawat e Kashyap, 2017). Ao nível do armazenamento de dados o *Neo4J* utiliza índices livres de adjacência que garantem a eficiência das consultas, mesmo com o aumento constante dos dados (Pokorný, 2017). No processamento de transações é garantida a consistência e confiabilidade dos dados, uma vez que respeitam as propriedades ACID guardando em memória os *logs* das transações e utilizando um mecanismo de replicação *master-slave*, em que todas as escritas têm de passar pelo diretor e, depois, são replicadas pelos restantes. Esta arquitetura, permite assegurar a redundância dos dados, a tolerância a falhas e a escalabilidade. Embora as operações de escritas tenham de passar pelo diretor, as de leitura podem ser efetuadas em cada um dos restantes servidores. Assim, é possível garantir que a performance das leituras aumenta linearmente com o número de servidores. O *Neo4J* utiliza a linguagem declarativa *Cypher*, cujos comandos são muito intuitivos e fáceis de aprender. Permite ainda a visualização interativa dos grafos, o que se revela muito útil na identificação de relacionamentos e possíveis falhas.

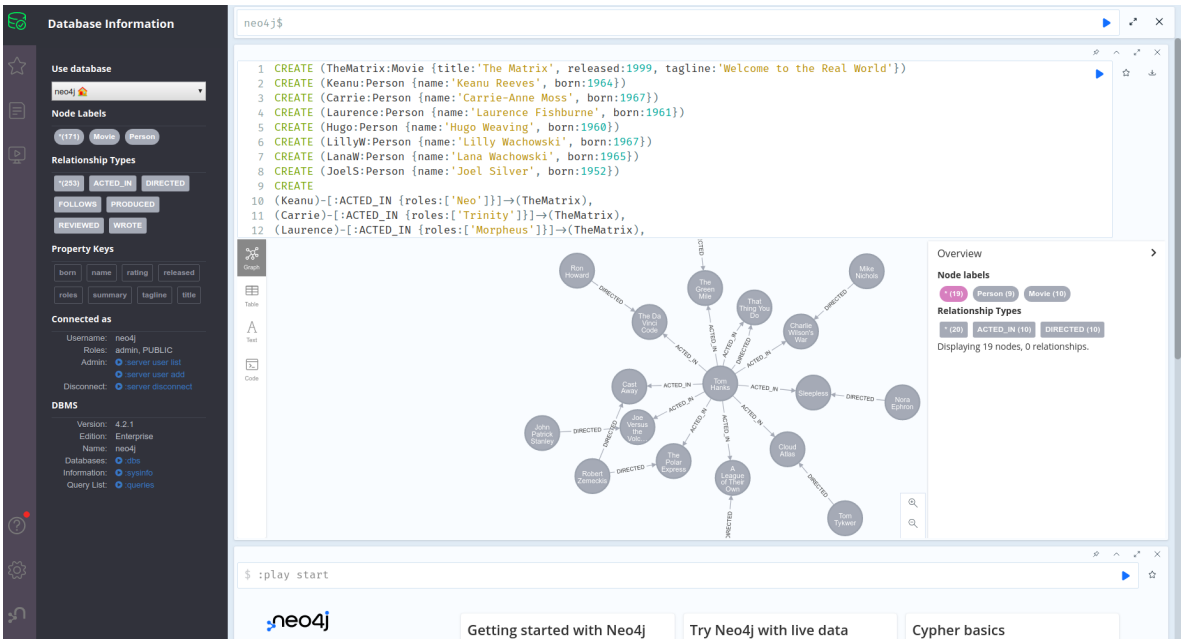

Figura 2: Ambiente de trabalho em *Neo4J*

2.2.2 *OrientDB*

O *OrientDB* (OrientDB) é um produto *open-source* desenvolvido em *Java* e multimodelo, isto é, suporta o armazenamento de dados em diversos modelos, como documentos, grafos, chave-valor e orientado a objetos. Este possui uma arquitetura distribuída, que utiliza os servidores de modo a alcançar o máximo de escalabilidade, desempenho e robustez. No armazenamento dos dados podem ser incorporados os diferentes modelos para melhorar a performance. Geralmente, *OrientDB* utiliza as listas de adjacência para manter a eficiência das consultas, mas pode recorrer a documentos, por exemplo, para armazenar os vértices físicos (Das et al., 2020). O sistema de replicação utilizado é o *multi-master*, em que todos os servidores podem efetuar operações de leituras e escritas que posteriormente são replicadas pelos restantes. A segurança dos dados confidenciais é garantida através de mecanismos de autenticação (Das et al., 2020). Este suporta diversas linguagens entre as quais o *Gremlin* e o SQL que através de extensões pode ser utilizado para manipular grafos.

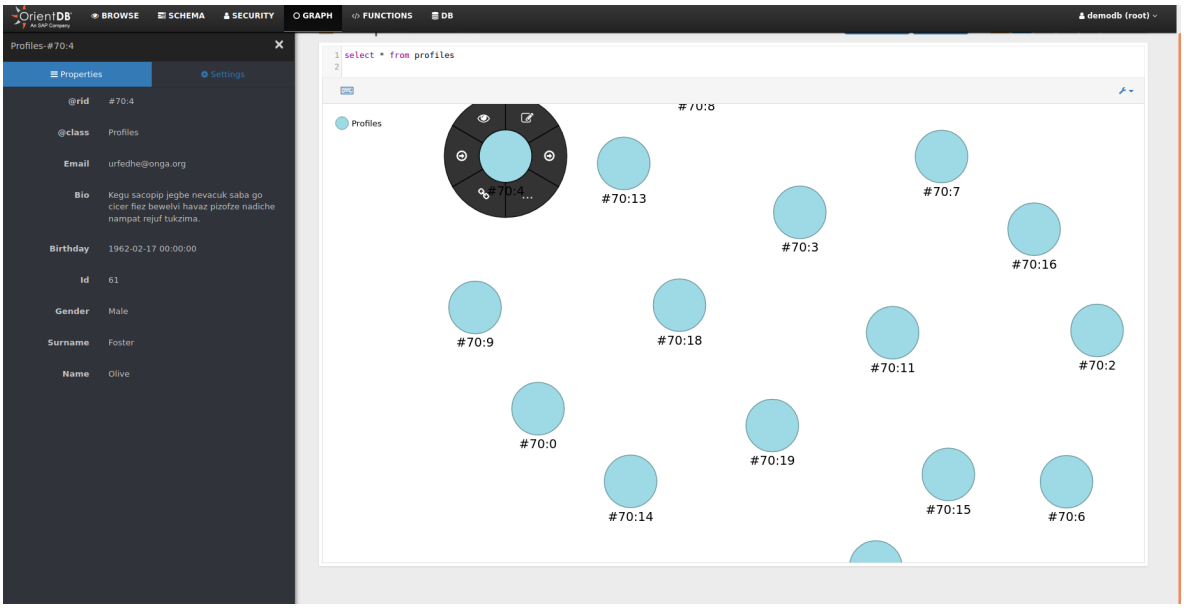

Figura 3: Ambiente de trabalho em *OrientDB*

2.2.3 *Microsoft Azure Cosmos DB*

À semelhança do *OrientDB* , o *Microsoft Azure Cosmos DB* (Markjbrown) também é considerado multimodelo, suportando modelos de dados orientados por grafos, documentos, chave-valor e colunas. Este funciona exclusivamente em *cloud*, pelo que permite *sharding*, isto é, os dados podem ser distribuídos por várias máquinas o que permite aumentar a sua capacidade para lidar com uma grande quantidade de dados e com taxas de transferências elevadas. Este sistema também possibilita a distribuição dos dados pelos centros de dados existentes nas diversas regiões, o que garante a eficiência tanto na escrita como nas leituras dos mesmos, independentemente do local onde se encontre (Markjbrown). Estas características tornam-no um sistema com uma alta performance. De salientar que os dados são replicados por todos os centros de dados disponíveis, o que lhe confere uma maior segurança, dado que em caso de existência de uma catástrofe na zona de um dos centros, os dados não se perdem, uma vez que se encontram também armazenados nos restantes centros. Ao nível da consistência, este fornece cinco níveis desde a forte que garante consistência total sacrificando a latência e a disponibilidade até à consistência eventual em que podem acontecer leituras inválidas, mas que apresentam uma maior disponibilidade. No caso dos grafos, a API (*Application Programming Interface*) de dados disponibilizada é o *Gremlin*

.

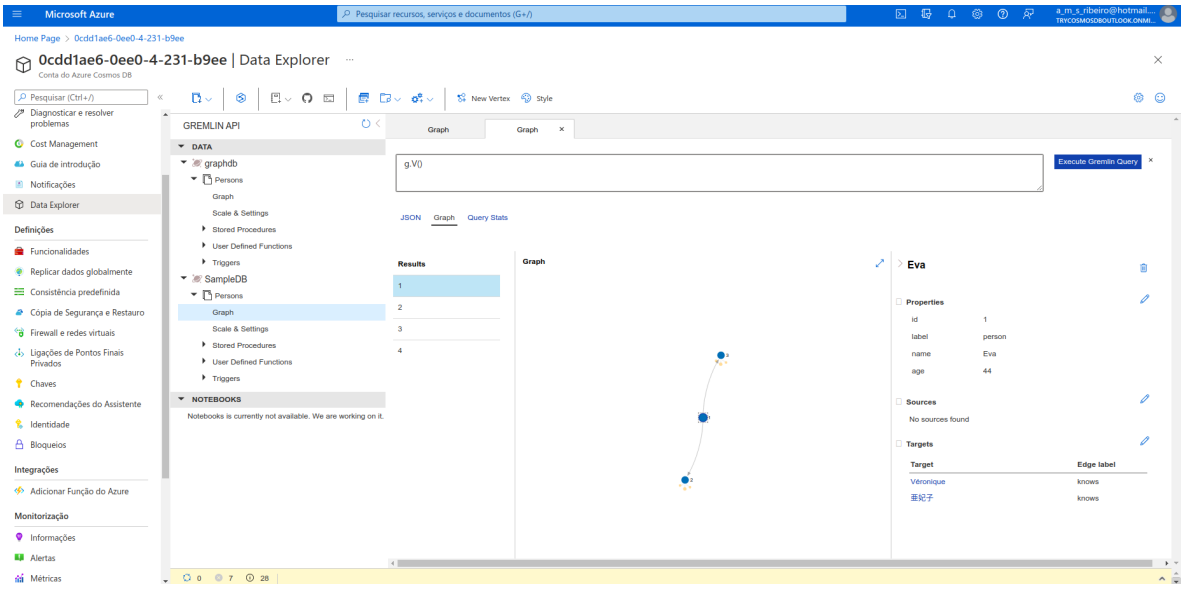

Figura 4: Ambiente de trabalho em *Microsoft Azure Cosmos DB*

2.2.4 *JanusGraph*

O *JanusGraph* é um produto *open-source* que permite o armazenamento de dados em estruturas de grafos. É escalável e foi otimizado para o armazenamento e a consulta de grafos compostos por uma grande quantidade de nodos e ramos. De forma, a elevar a sua performance, este sistema efetua a distribuição dos dados pelas várias máquinas no *cluster*, quanto maior for o número de máquinas maior é a sua capacidade transacional. Adicionalmente, é realizada a replicação dos dados, de modo a assegurar a tolerância a falhas, ou seja, permitir que o sistema continue operacional caso algum componente falhe. A disponibilidade dos dados é assegurada através do amazenamento dos mesmos em vários centros de dados. Para tal, este sistema possui um estratrégia de *backups* dinâmicos. Estas características permitem que o sistema suporte o processamento de transações simultâneas de inúmeros utilizadores. A linguagem disponibilizada para a travessia de grafos é o *Gremlin*. De realaçar que este sistema não permite nativamente a visualização dos grafos, no entanto, é facilmente integrável com várias ferramentas de visualização como o *Arcade Analytics* (Analytics) ou *Cytoscape* (Cytoscape), entre muitas outras.

```
==>graphtraversalsource[standardjanusgraph[inmemory:[127.0.0.1]], standard]
gremlin> g.V().count()
==>0
gremlin> g.addV('person').property('name', 'p1')
==>v[4208]
gremlin> g.addV('person').property('name', 'p2')
==>v[4272]
gremlin> g.V().count()
==>2
gremlin>
```

Figura 5: Exemplo do ambiente de trabalho em *JanusGraph*

2.2.5 *Seleção do Sistema*

De seguida, podemos observar uma breve súmula das principais características de cada um dos Sistema de Gestão de Base de Dados analisados, para que seja mais fácil compreender em que aspetos se assemelham e em que aspetos se distinguem.

Tabela 1: Bases de dados suportadas por grafos (DBEngine)

GDMS	Licença	Modelo	Linguagem de Consulta	Pontuação
Neo4J	open-source	grafos	cypher	50.54
Microsoft Azure Cosmos DB	commercial	multimodelo	gremlin, sql	30.35
OrientDB	open-source	multimodelo	gremlin, sql	4.93
JanusGraph	open-source	grafos	gremlin	2.48

Face às características apresentadas anteriormente, o sistema de gestão de base de dados que selecionámos para trabalho no âmbito desta dissertação foi o *Neo4J*. O facto do *Neo4J* possuir uma linguagem de consulta declarativa, o *Cypher*, que se revela muito fácil de utilizar e de ser atualmente a mais aplicada no mercado no segmento das bases de dados suportadas por grafos, o que representa um maior suporte, coloca-o em vantagem. Para além disso, suporta exclusivamente um modelo de grafos, não dependendo por isso de outros sistemas de base de dados, como por exemplo o *OrientDB* ou o *Microsoft Azure Cosmos DB*. Por fim, o *JanusGraph* quando comparado com os restantes sistemas possui uma pontuação bastante inferior, o que se reflete na falta de informação e suporte sobre o mesmo. O *Neo4J* possui ainda a vantagem de ser *open-source* o que facilita a sua utilização.

2.3 MODELO DE DADOS SUPORTADOS POR GRAFOS

Os modelos de dados suportados por grafos baseiam-se na teoria dos grafos e são definidos por um conjunto de nodos, que representam as entidades, e de ramos, que estabelecem os relacionamentos entre os mesmos (Castelltort e Laurent, 2019). Como o modelo a considerar é o de propriedades, a cada um destes componentes pode ser atribuído um conjunto de propriedades, constituídas por um par chave-valor (Patil et al., 2014). Assim, os nodos podem ser definidos por um identificador, um conjunto de propriedades que o descrevem e um conjunto de *labels* (Patil et al., 2014). As propriedades representam os seus atributos e metadados necessários como *timestamp* , número de versões entre outros. As *labels* permitem expressar a semântica de um nodo. Desta forma é possível agrupá-los.

Os relacionamentos possuem sempre uma direção e nunca são pendentes. Estes são definidos por um identificador, um nodo de entrada, um nodo de saída, um tipo de relação e um conjunto de propriedades que fornecem informação adicional sobre o relacionamento. Apesar de possuírem uma direção é possível considerar o relacionamento inverso sem ter de o representar. De seguida, na Figura 6 é possível observar a representação de cada um destes conceitos. Nessa figura estão representados três nodos, em que um corresponde a um "Funcionário" e os outros dois a "Venda", como podemos ver pelas *labels* atribuídas. Existem ainda dois relacionamentos, cujo tipo da relação é "Regista", o nodo de entrada é um funcionário e o de saída cada uma das vendas correspondentes. Também é possível verificar as propriedades que compõem cada um dos nodos, nomeadamente, a chave identificadora de cada um e o valor correspondente.

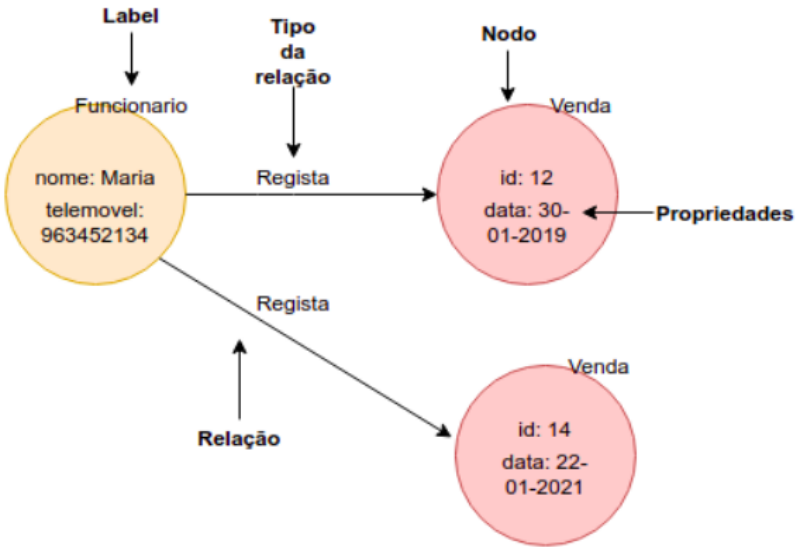

Figura 6: Exemplo de um grafo de propriedades

2.3.1 *Implementação de Sistema de Dados com Grafos*

Vejamos agora um exemplo de implementação de uma base de dados suportada por grafos. Para isso, consideramos uma aplicação de vendas que irá ser suportada por uma base de dados suportada por grafos em *Neo4J* . O primeiro passo consiste em definir o modelo de dados, pelo que é necessário começar por identificar todos os nodos e seus respetivos relacionamentos. O objetivo desta base de dados é armazenar a informação relativa às vendas, produtos, clientes e funcionários, as entidades do sistema. Ao nível dos relacionamentos serão estabelecidos três. Um entre o "Funcionário"e a "Venda", de modo a identificar as vendas registadas pelos diferentes funcionários. Outro entre a "Venda"e o "Produto", para conhecer todos os produtos envolvidos numa venda. Por fim, existe ainda um relacionamento entre a "Venda"e o "Cliente"para determinar as vendas efetuadas pelos clientes. Na Figura 7 podemos ver o modelo da base de dados que acabámos de descrever.

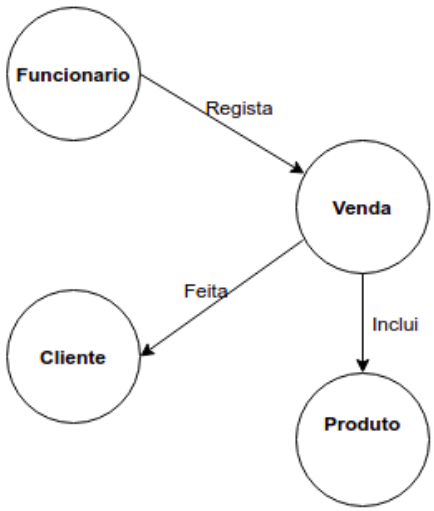

Figura 7: Representação dos nodos e relacionamentos

Uma vez definidos os nodos e os respetivos relacionamentos é necessário adicionar as *labels* e as propriedades necessárias. De salientar que o produto é constituído por duas *labels*, uma para identificar que se trata de um "Produto"e uma outra que permite reconhecer o tipo de produto, de forma a poder agrupá-los por categoria, como televisão, máquinas de lavar roupa, entre outros. O relacionamento entre a venda e o produto contém uma propriedade que estabelece a quantidade daquele produto incluído na venda. Na Figura 8 é possível observar o modelo final, onde se encontram representados todos os nodos e as respetivas propriedades, compostas pela chave que a identifica e o tipo de dados expectável, relacionamentos, propriedades, e ainda as *labels*.

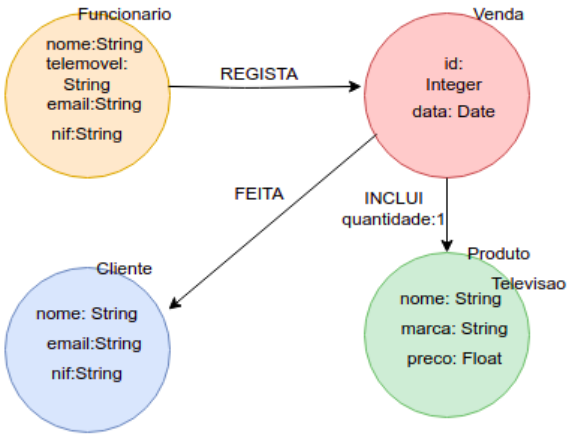

Figura 8: Modelo de Dados para uma aplicação de vendas

Uma vez finalizada a fase de modelação, podemos implementá-lo numa base de dados em *Neo4J* e através de diversas consultas podemos visualizar os dados representados em grafos. Na Figura 9 podemos observar o grafo total. Nesse grafo, podemos ver que existem cinco produtos, dos quais três televisões e duas impressoras, quatro clientes, dois funcionários e quatro vendas.

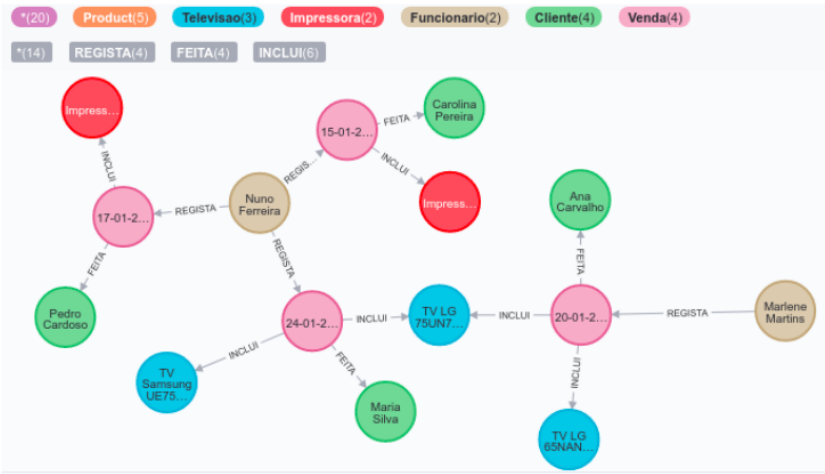

Figura 9: Visualização do grafo implementado em *Neo4J*

3

SISTEMAS DE DADOS MULTIDIMENSIONAIS

No ambiente competitivo que se vive hoje em dia, é essencial para as organizações possuírem o máximo de informação possível para sustentar os seus processos de tomada de decisão. Atualmente são produzidas grandes quantidades de dados que podem ser internos, proveniente das diversas bases de dados que suportam as tarefas operacionais, como externos por exemplo as redes sociais. De forma a extrair o conhecimento necessário para a tomada de decisão é importante que estes estejam integrados, para que possam ser consultados como um todo. Nesta perspetiva, as empresas têm recorrido a *data warehouses* para armazenar os seus dados. Tipicamente estes baseiam-se em base de dados relacionais. Um *data warehouse* surge como um sistema de dados centralizado e autónomo, organizado por assunto, integrado, temporal e não volátil (Inmon, 2005). Este resulta da integração dos dados de diferentes fontes, como os sistemas operacionais, que são analisados e tratados de um modo consistente. Tipicamente são relativos a transações, pelo que não é expectável que sofram alterações.

Na figura 10, podemos observar a arquitetura de um *data warehouse* constituída por duas camadas. Podemos identificar a camada de dados, que representa o conjunto de dados heterogéneos que podem ser internos, derivados das bases de dados operacionais que armazenam as transações relativas ao negócio, ou externos como por exemplo medidas provenientes das redes sociais que expressam a preferência do público alvo (Tria et al., 2018). Estes são responsáveis por alimentar o *data warehouse*. Para que estes dados possam ser incorporados num esquema comum é necessário tratá-los. Assim na área de preparação estes passam pelo processo ETL (*Extraction*, *Transformation*, *and Loading*), que os extrai, limpa, de forma a remover eventuais inconsistências e preencher lacunas, para que posteriormente possam ser integrados. De referir que este processo é repetido periodicamente de acordo com a estratégia de atualização definida (Tria et al., 2018). É importante que este período seja curto para que a informação analisada seja o mais fidedigna possível, caso contrário as decisões tomadas com base nesses dados podem ser desatualizadas e inadequadas face à situação mais atual. No entanto, é preciso também ter em atenção o custo associado a este processo de refrescamento dos dados, pelo que é fundamental adotar uma estratégia de equilíbrio. No nível seguinte, podemos identificar o *data warehouse*, repositório central responsável por guardar os dados agregados. Este tanto pode ser acedido diretamente como ser utilizado como uma fonte para sustentar os *data marts*, projetados para departamentos empresariais específicos (Malinowski e Zimányi, 2008). Os metadados armazenam informações sobre fontes, procedimentos de acesso, preparação de dados e esquema de *data marts*. Por fim, com o intuito de desenvolver

uma interface simples e intuitiva, estes dados são acedidos de um modo eficiente para gerar relatórios e gráficos que auxiliem os agentes no momento da decisão.

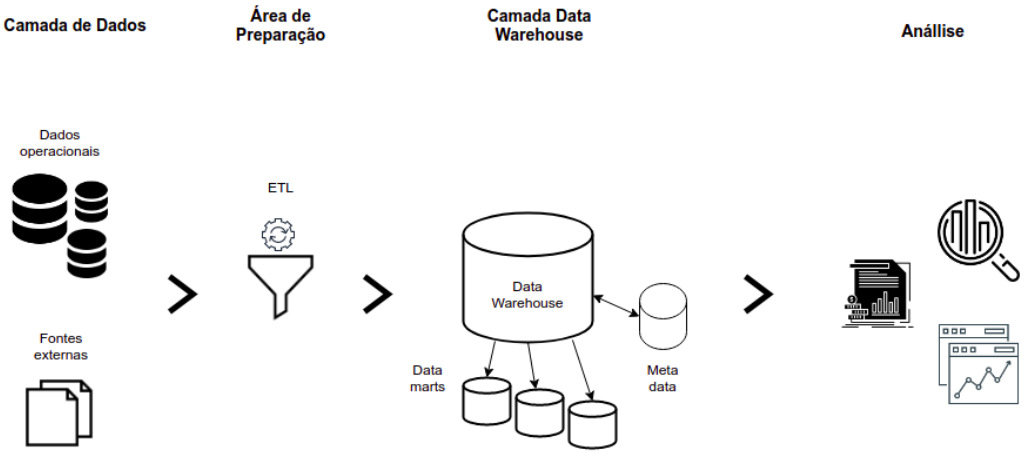

Figura 10: Arquitetura Data Warehouse adaptado de (Malinowski e Zimányi, 2008)

3.1 PROCESSAMENTO ANALÍTICO (OLAP)

O conceito de processamento analítico está diretamente relacionado com o de *Data Warehousing*, uma vez que estes se complementam. Com o intuito de facilitar a análise da informação necessária, o processamento analítico permite aos utilizadores finais visualizarem os dados detalhadamente e sobre diferentes perspetivas através de uma interface simples e intuitiva. Como já foi referido, um *Data Warehouse* é, basicamente, um repositório de dados que é responsável por armazenar uma grande quantidade de dados de forma eficiente, tendo em atenção as suas características temporais e áreas de definição. A combinação destes conceitos permite obter um sistema de análise muito eficiente e flexível que se revela indispensável, hoje em dia, para uma melhor gestão ao nível dos negócios. Desta forma é fundamental que, na construção do modelo multidimensional que suporta um *Data Warehouse*, se tenha em consideração aquilo que se pretende apresentar na interface respetiva.

O termo OLAP (*Online analytical processing*) representa a tecnologia que nos permite analisar e sintetizar todos os dados armazenados no sistema multidimensional do modo mais eficiente possível. Este tipo de tecnologia permite um acesso simples e rápido à informação, possibilitando à organização dar uma resposta mais adequada e célere face às exigências do mercado. Os ambientes de consulta facultados pelas aplicações desenvolvidas com esta técnica não exigem grande conhecimento tecnológico, pelo que podem ser utilizados de uma forma natural por qualquer tipo de utilizador, o que confere aos agentes de decisão uma menor dependência das equipas de desenvolvimento. Outra das principais características dos sistemas OLAP é a redução do tráfego sobre os dados armazenados no *data warehouse*, visto que a maior parte deles são carregados para o servidor OLAP. É de salientar ainda que a informação apresentada nestas interfaces são consideradas credíveis, uma vez que provêm de um sistema de *Data Warehousing*.

Tendo em conta o propósito de uma ferramenta OLAP é essencial que esta permita a gestão de uma elevada quantidade de dados, dado que o limite inferior para o volume de dados com que estes geralmente trabalham anda na ordem dos *terabytes* (TB). Assim, estes sistemas devem garantir que a performance permanece estável com o aumento do volume de dados, para que os seus utilizadores possam realizar sempre as consultas necessárias sem se preocuparem com o desempenho do mesmo. É importante lembrar que estes utilizadores geralmente não são experientes e portanto nem conhecem estes conceitos mais técnicos associados à performance, apenas pretendem obter os seus resultados. Para além disso, estas tecnologias devem permitir que sejam efetuadas consultas interativas, de modo a que o utilizador possa aceder a toda informação que considere relevante para a tomada de decisão. É também fundamental que estes forneçam mecanismos de segurança para a partilha de dados quer ao nível da confidencialidade, que representa um assunto muito sensível, quer na integridade dos dados fornecidos.

Em suma, um sistema OLAP é, essencialmente, utilizado em processos de análise de dados, pelo que, não é uma surpresa, saber que os seus principais utilizadores sejam gestores, analistas ou executivos. Normalmente, estes sistemas, lidam com uma elevada quantidade de dados históricos e têm como principal objetivo facilitar a síntese e a agregação dos mesmos. Sendo estes sistemas uma ferramenta de análise, apenas suportam operações de leituras que podem revelar-se muito complexas devido ao tipo de trabalho envolvido na mesma e em função do número de dimensões do modelo de dados que suportam.

Além dos sitemas OLAP, uma organização possui um conjunto de sistemas OLTP (*Online transaction processing*), que são responsáveis pelas operações diárias de processamento de dados da organização, como gestão de compras, de inventários, realização de pagamentos, entre outros muitas coisas. Posteriormente, os dados OLTP podem ser integrados no DW e utilizados nos sistemas OLAP. De forma a melhor compreender as principais diferenças entre estes dois tipos de sistemas, na Tabela 2 resumem-se as suas principais características.

Tabela 2: Principais diferenças entre os sistemas OLAP e OLTP

	OLAP	**OLTP**
Orientação	Analítica	Transacional
Utilizadores	Gestores, Executivos e Analistas	DBA, profissionais de BD
Dados	Históricos	Atuais
Unidades de trabalho	Consultas complexas	Transações simples
Acessos	Leitura	Leitura e Escrita
Foco	Extração de informação	Inserção de dados
Ordem de acesso aos dados	Milhões	Unidades
Tamanho	TB	GB
Número de Utilizadores	Centenas	Milhares

3.2 ESTRUTURAS DE DADOS MULTIDIMENSIONAIS

No processo de desenvolvimento a modelação dos dados é fundamental, de forma a assegurar que estes são guardados adequadamente e que os requisitos definidos são atingíveis. Tradicionalmente existem três tipos de modelos com conceitos e níveis de abstração diferentes. O modelo conceptual é independente da tecnologia utilizada, pelo que os dados se encontram organizados com um nível elevado de abstração. Nesta fase o *Data Warehouse* é representado através um modelo multidimensional, baseado em factos, medidas, dimensões e hierarquias. O modelo lógico é gerado a partir do modelo conceptual e já apresenta alguns detalhes de implementação, como a forma como os dados estão organizados. No entanto estes são independentes do *software* utilizado. As bases de dados tradicionais recorrem a um modelo relacional, sendo os mais populares o modelo em estrela, em floco de neve ou em constelação de factos (Pérez et al., 2008). O modelo mais comum é o modelo em estrela que é composto por uma tabela central contendo a maior parte dos dados, que corresponde à dos factos. Além disso, este modelo costuma envolver um conjunto de tabelas mais pequenas que retratam as dimensões, sendo uma tabela por cada dimensão. Por sua vez, o modelo em floco de neve deriva do modelo em estrela, sendo as tabelas relativas às dimensões normalizadas, resultando em tabelas adicionais. Por fim, o modelo da constelação de factos é utilizado quando é necessário que múltiplas tabelas de factos partilhem as mesmas tabelas referentes às dimensões, quer estas estejam organizadas individualmente em estrela ou em floco-de-neve.

No âmbito desta dissertação o modelo que escolhemos e usámos foi o de grafos, nomeadamente grafos de propriedades. O modelo físico é de baixo nível e requer uma implementação física do modelo lógico definido tendo em conta as propriedades individuais do sistema de base de dados a utilizar, neste caso o *Neo4J*.

Um modelo multidimensional é definido por um conjunto de factos, dimensões e por uma função que associa a cada facto um conjunto de dimensões. Assim, podemos representar um modelo multidimensional N por $(F^N, D^N, Func)$, sendo que: (Chevalier et al., 2015)

- F^N - representa um conjunto finito de factos.
- D^N - retrata um conjunto finito de dimensões.
- $Func$ - função que a cada facto associa um conjunto de dimensões.

Basicamente, um facto representa o assunto a ser analisado e é composto por um conjunto de medidas geralmente numéricas que retratam a informação que se pretende analisar (Yanguia et al., 2016). Deste modo, um facto pode ser definido por: (N^F, M^F), que representa, respetivamente, o seu nome e o conjunto finito de medidas que o constituem. De referir que normalmente cada medida encontra-se associada a uma função de agregação, como por exemplo a soma, a média ou máximo (Sellami et al., 2020). As dimensões retratam os eixos de análise e permitem restringir as consultas e limitar os tamanhos das respostas. Já uma dimensão possui um nome, um conjunto de atributos e um conjunto de hierarquias, sendo representada por (N^D, Att^D, H^D), em que os atributos se encontram organizados pelas hierarquias. Por sua vez, cada hierarquia contém um nome, um conjunto de atributos de dimensão e uma função que a cada atributo associa os seus atributos não

dimensionais (Sellami et al., 2020). Os atributos não dimensionais contêm informação adicional sobre o atributo de dimensão ao qual estão associados (Golfarelli et al., 1998). Na hierarquia os atributos encontram-se dispostos pelo nível de granularidade, isto é, do mais detalhado para o menos detalhado.

O conceito de hierarquia é fundamental num modelo multidimensional, uma vez que representa o modo como as dimensões e os respetivos atributos dimensionais se relacionam. Basicamente, uma hierarquia permite estabelecer uma sequência de mapeamentos desde um conceito de baixo nível até um conceito de alto nível (Han et al., 2017). Um exemplo simples e intuitivo que nos permite entender melhor este conceito é o das datas. Usualmente, uma data é constituída por dia, mês e ano e é facilmente identificado que o dia é o nível de maior detalhe e o ano o mais abstrato. Assim, a hierarquia da data é composta pelo dia que estabelece uma relação de precedência com o mês que por sua vez se relaciona com o ano. Contudo, os atributos dimensionais não são obrigados a constituir um hierarquia completa, isto é, nem todos os atributos da dimensão têm de estabelecer uma relação de precedência com os restantes. Por exemplo, considerando uma dimensão loja, constituída pelos atributos dimensionais cidade, distrito e país. É possível relacioná-los numa hierarquia sendo a sequência lógica "Loja", seguida da "Cidade", "Distrito"e "País- "Loja»>"Cidade»>"Distrito»>"País". No entanto, o objetivo pode passar por criar uma hierarquia parcial, em que, a loja se encontra seguida da cidade e do país ou se encontra seguida do distrito e do país, formando uma malha. Basicamente, uma hierarquia parcial permite que os atributos dimensionais não tenham de ser todos colocados ordenadamente face aos restantes formando apenas uma sequência lógica. Esta característica é fundamental, uma vez que confere flexibilidade ao modelo. Na Figura 11 podemos observar uma hierarquia total para o caso das datas e uma parcial para a loja.

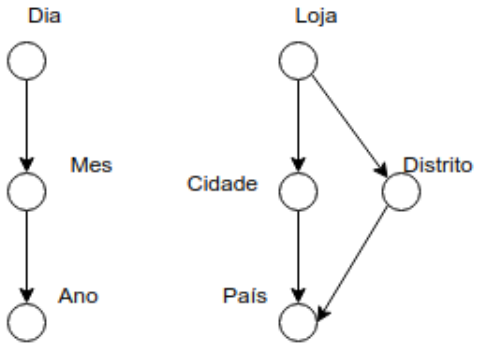

Figura 11: Exemplo de uma hierarquia total e de uma parcial

O caso que será apresentado nesta dissertação e que posteriormente será utilizado para a elaboração de todos os modelos de dados será relativo, mais uma vez, a uma aplicação de gestão de vendas, num ambiente comercial típico, tal como aconteceu previamente para a implementação da base de dados. Desta forma, no nosso caso foi considerado um cenário composto por "clientes", "funcionários", "vendas" e "produtos" a serem vendidos. O objetivo consiste em armazenar toda a informação necessária referente aos mesmos, bem como todos os elementos relativos às transações do negócio. Assim sendo, é possível identificar facilmente um conjunto de requisitos indispensável para o bom funcionamento.

O ponto fulcral deste cenário é a venda. Como tal, é essencial conseguir identificar os produtos incluídos em cada uma das vendas realizadas, tanto em termos do cliente que fez a compra, como do funcionário que a realizou. Neste ambiente é ainda fundamental rastrear as vendas efetuadas por datas, de modo a poder ser realizado um balanço do fluxo de vendas por ano, mês ou dia. Os dados que servem de apoio e que integram a base de dados provêm, neste caso, de três ficheiros que se encontram no formato *csv*. De seguida, apresentamos detalhadamente o conteúdo de cada um:

- *Produto.csv* - contém a informação relativa a todos os produtos designadamente, nome, preço, marca e categoria.

ProductID	ProductName	Category	Price	Mark
1	Piano	Strings	100	Abc
2	Violin	Strings	101	Bdf
3	Cello	Strings	102	Cfg

Figura 12: Dataset referente aos produtos

- *Vendas.csv* - contém os dados relativamente à venda, aos clientes e aos funcionários. Como por exemplo, a data e o total faturado na venda, o nome do cliente e do funcionário responsável pela mesma.

Day	Year	Month	SalesID	Sales	Units	customerID	contactName	phone	employeeID	name	birthDate	homePhone
1	2016	1	1	4251.00	693	ALFKI	Maria Anders	030-0074321	1	Nancy Davolio	1948-12-08 00:00:00.000	(206) 555-9857
1	2016	1	2	2789.00	571	ANATR	Ana Trujillo	(5) 555-4729	2	Andrew Fuller	Dr.	USA
1	2016	1	3	3908.00	311	ANTON	Antonio Moreno	(5) 555-3932	3	Janet Leverling	1963-08-30 00:00:00.000	(206) 555-3412

Figura 13: Dataset das Vendas

- *VendaProduto.csv* - permite definir os produtos envolvidos numa venda através dos identificadores dos mesmos.

SaleON	ProductID
1	1
1	2
2	3

Figura 14: Dataset que define os produtos das vendas

Uma vez definido o caso de estudo adotado foi necessário estabelecer o modelo multidimensional correspondente. O primeiro passo consistiu em definir o assunto a ser analisado que, no modelo, representasse o facto e que na nossa situação corresponde à venda. Associado ao facto temos as medidas que retratam o que se pretende analisar. Neste caso considerámos a quantidade de produtos envolvidos numa venda e o valor total faturado na mesma. A análise da venda poderia ser feita tendo em conta outras medidas e outros processos de cálculo, como a média ou a venda com o produto mais caro. Porém, no nosso caso, o processo foi simplificado a duas variáveis, altamente relacionadas a um processo de vendas, e que, de um modo geral, representam os fatores mais relevantes da mesma.

De seguida foi necessário estabelecer os eixos de análise, ou seja, as dimensões do modelo. Os eixos definidos foram "cliente", "produto", "funcionário" e data, mais concretamente o "dia" em que a venda foi realizada. No fundo estes representam as perspetivas sobre as quais se pretende analisar as vendas, de modo a sustentar as decisões que se possam tomar com o intuito de melhorar o negócio. Para esta escolha, tivémos em consideração os principais objetivos apresentados no caso de estudo. Cada dimensão possui um conjunto de atributos dimensionais que se encontram organizados pelas hierarquias. Cada hierarquia é composta pelos atributos dimensionais diretamente relacionados e sobre os quais é possível estabelecer ligações de precedência, de forma a que possam ser organizados por nível de detalhe da informação (de maior detalhe para o menor). O "produto" possui duas hierarquias, uma para a "marca"e outra para a "categoria"do produto em questão, por exemplo televisores, impressoras, entre outras. Cada um destes atributos possui como atributos não dimensionais a sua descrição. Todas as restantes dimensões possuem apenas uma hierarquia. O "cliente"e o "funcionário"não apresentam atributos dimensionais, apenas têm um conjunto de atributos não dimensionais que nos fornecem informações adicionais sobre os mesmos, como por exemplo, o "nome" e o "nif". A dimensão do "dia", é composta por dois atributos dimensionais, nomeadamente o "mês"e o "ano". De salientar que na hierarquia do dia é visível que os atributos se encontram organizados pelo nível de granularidade, ou seja, do mais detalhado, o dia, para o menos detalhado, o ano. Na Figura 15 ilustrada abaixo é possível identificar todos os componentes explicados e que compoêm o modelo multidimensional delineado que serve de base para todo o trabalho desenvolvido nesta dissertação.

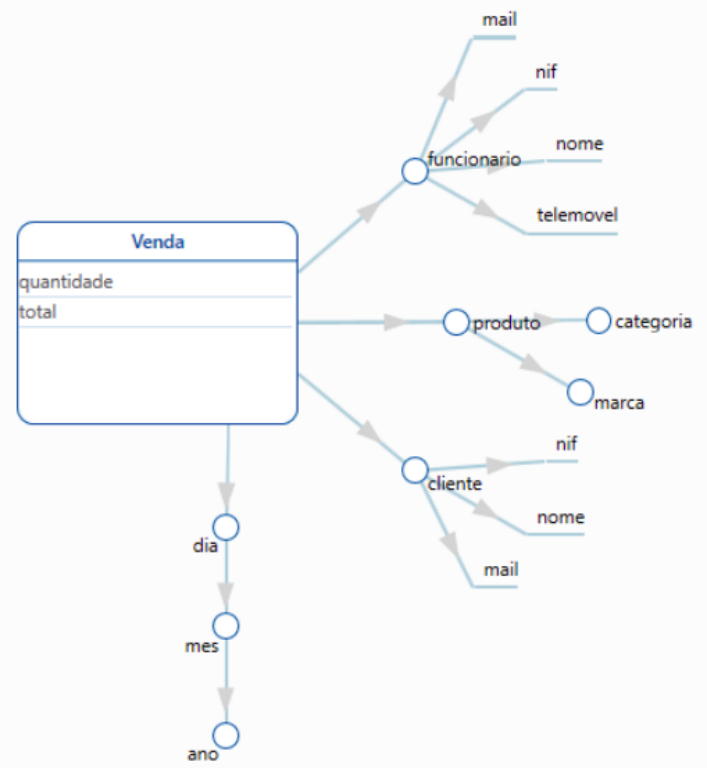

Figura 15: Modelo multidimensional para uma aplicação de vendas

3.3 OPERAÇÕES SOBRE ESTRUTURAS MULTIDIMENSIONAIS

De forma a permitir a observação dos dados através de diferentes perspetivas e com níveis de detalhe distintos, existem um conjunto de operações padrão OLAP especificamente orientadas para isso, que tiram partido da estrutura hierárquica do modelo multidimensional. Essas operações são suportadas por um conjunto especial de operadores, nomeadamente: *slice, dice, roll-up, drill-down* e *pivot*. O *slice* é o operador que nos permite selecionar os dados de acordo com uma condição baseada nos valores das dimensões do modelo. Este é essencial, uma vez que nos permite reduzir os dados de acordo com os eixos de análise. Por exemplo, no caso do modelo de vendas apresentado anteriormente, na Figura 15, podemos analisar as vendas efetuadas por um determinado cliente. O operador *dice* funciona de forma similar. No entanto, permite-nos filtrar por vários valores e por mais do que uma dimensão simultaneamente. Por exemplo, a seleção das vendas realizadas por um determinado cliente para um dado produto ou a seleção de vendas efetuadas por exemplo por um conjunto de clientes. Em última instância, podemos classificá-los como operadores de seleção, na medida em que ambos nos permitem obter um subconjunto do cubo através da aplicação de determinadas condições.

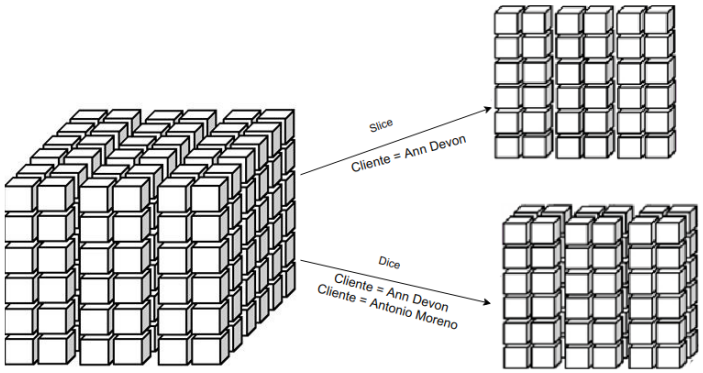

Figura 16: Operações Slice e Dice (Adaptado de (Bolt e der Aalst, 2015))

As operações de *roll up* consistem em aumentar o nível de granularidade da análise ao invés das operações de *drill down* que o diminuem. Estes operadores permitem navegar pela estrutura hierárquica dos eixos de análise de modo a explorar a informação em diferentes níveis de detalhe (Ravat et al., 2007). Por exemplo, no nosso cenário de aplicação podemos recorrer a operações *roll up* para através dos dados relativos às vendas ao nível da marca ou categoria obtermos a mesma análise mas a um nível mais abstrato, o produto. Como exemplo de uma operação *drill down* temos a obtenção dos dados relativos às vendas por dia através da síntese dos mesmos para o mês e o ano, mediante a soma dos valores.

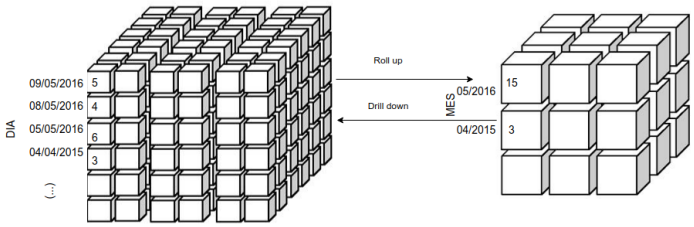

Figura 17: Operações *Roll-up* e *Drill-down* (Adaptado de (Bolt e der Aalst, 2015))

A operação *pivot*, também conhecida por rotação, basicamente, muda a posição do cubo de modo a apresentar um nova visão dos dados. Esta alteração é realizada através de uma rotação do cubo sobre as dimensões selecionadas, ou seja, modificando os eixos do mesmo. Existem ainda outro tipo de operações, como por exemplo, o *drill-across e drill-through*, que permitem, respetivamente, a junção de dois cubos e estabelecer a ligação entre os dados armazenados no cubo e os seus correspondentes nos sistemas de origem, ou seja, o *data warehouse*.

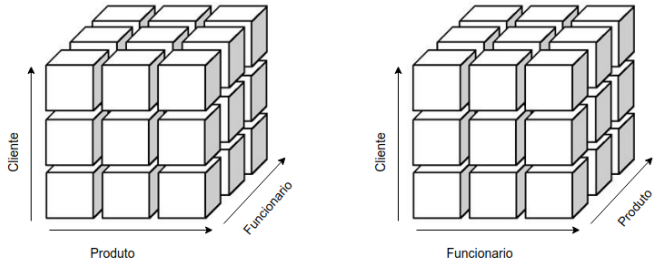

Figura 18: Operações *Pivot* (Adaptado de (Bolt e der Aalst, 2015))

Nesta situação pretende-se analisar como as consultas podem ser efetuadas nos modelos apresentados e como a sua estrutura pode ser utilizada para dar resposta às mais diversas análises, pelo que o foco são as operações de seleção, *slice* e *dice* respetivamente, que permitem a restrição dos dados de acordo com as dimensões. A estratégia adotada, quer para avaliar o modo como os dados são tratados no cubo na aplicação destas operações quer para avaliar a eficiência da mesma, consiste em começar com uma operação simples com apenas um filtro, referente a uma dimensão, e, posteriormente, ir adicionando complexidade através da junção de novas condições. Estas condições tanto podem ser referentes a novas dimensões, como à mesma estabelecendo assim mais do que um limite para uma mesma dimensão . De seguida pode-se observar um conjunto de consultas que foram definidas com base na estratégia apresentada e que posteriormente são executadas nas base de dados implementadas de modo a avaliar os modelos definidos:

1. Seleção dos resultados de vendas realizadas a um determinado cliente, por exemplo, com o nome Ann Devon;

2. Seleção dos resultados das vendas efetuadas para mais do que um cliente, por exemplo, para a Ann Devon e para o cliente Antonio Moreno;

3. Seleção dos resultados de vendas para a cliente Ann Devon referentes a uma determinada marca, por exemplo Blit. Nesta query já estão presentes filtros referentes a duas dimensões;

4. Seleção dos resultados de todas as vendas feitas pela cliente Ann Devon, para o produto Trumpet, no dia um de janeiro de 2016. Nesta consulta, às condições anteriores foi acrescentada mais uma dimensão, o dia, constituindo um total de três dimensões;

5. Seleção dos resultados de todas as vendas realizadas pelo cliente Ann Devon, para o produto Trumpet, no dia um de janeiro de 2016 registada pelo funcionário Andew Fulliew. Mais uma vez, nesta consulta foi adicionando mais uma dimensão tendo em conta a definida anteriormente. De referir que nesta consulta já temos nos critérios todas as dimensões.

3.4 ESTRUTURAS MULTIDIMENSIONAIS PARA GRAFOS

Nos *Data Warehouses* suportados por grafos, a partir do modelo multidimensional é gerado um modelo lógico no qual os dados se encontram organizados num grafo, ao invés das técnicas tradicionais que os colocam usualmente em tabelas. A utilização de grafos remete-nos logo para as vantagens que as bases de dados NoSQL geralmente apresentam relativamente às relacionais, por exemplo, maior capacidade de armazenamento e maior flexibilidade por possuírem um esquema adaptável. Como já foi referido previamente, nos *Data Warehouses* tradicionais a inclusão de novos dados é muito custosa, sendo o processo de extração, transformação e carregamento dos dados muito mais complexo. Uma vez que este tipo de base de dados apresenta um modelo fixo, e com os tipos de dados bem definidos, nesta fase de extração e carregamento, é necessário assegurar que os dados provenientes das fontes os respeitam. Caso tal não se verifique é preciso transformá-los de modo a garantir que se enquadram no modelo. Outra das complicações neste processo para este tipo de base de dados prende-se com os dados semi-estruturados. A adoção de um modelo de grafos torna este processo muito mais simples, uma vez que estes suportam todo o tipo de dados e não possuem uma estrutura definida. Assim, apenas necessitamos de criar o nodo, armazenar a informação necessária nos seus atributos e posteriormente estabelecer os relacionamentos respetivos. Esta simplicidade ao nível da inserção de dados permite também que os dados que constituem o *Data Warehouse* estejam sempre atualizados, pelo que a informação considerada no momento da tomada de decisão reflete o contexto atual.

Hoje em dia os dados gerados caracterizam-se pela sua variedade e velocidade, pelo que esta característica dos modelos suportados por grafos se revela essencial. Nos *Data Warehouse* tradicionais face à complexidade é necessário encontrar um equilíbrio entre o desempenho e o nível de atualização dos dados. Finalmente, sendo as análises de dados efetuadas baseadas usualmente em operações de consultas é possível tirar proveito da alta performance que este tipo de base de dados apresenta em operações de consultas, devido ao modo como armazena os seus dados.

Nesta secção é apresentada uma metodologia para a modelação de um modelo multidimensional num modelo orientado para grafos. Uma vez definido o modelo multidimensional que suporta o caso de estudo apresentado, é necessário averiguar o modo com podemos adaptá-lo num modelo suportado por grafos. De forma

a facilitar a transformação, foram definidas um conjunto de regras base que nos permitem mapear os conceitos associados a um modelo multidimensional, como sejam os factos, as dimensões ou as hierarquias, para um modelo de grafos. Estas regras foram definidas com base em todos os elementos que podem constituir um modelo multidimensional, de forma a que pudessem ser aplicadas a qualquer modelo multidimensional e não apenas àquele que demonstrámos anteriormente. Desta forma é indispensável definir como podemos representar factos, dimensões, hierarquias, medidas e atributos dimensionais e não dimensionais num modelo suportado por grafos de propriedades. De seguida apresentamos as cinco regras que foram definidas:

1. Cada facto é representado como um nodo cuja *label* é o nome do mesmo. As medidas associadas ao facto integram o conjunto de propriedades relativas ao mesmo.

2. Cada dimensão é também representada como um nodo cuja *label* é o seu nome. Todos atributos não dimensionais associados à dimensão, juntamente com o identificador, constituem o conjunto de propriedades referente ao nodo.

3. Os atributos dimensionais apresentados nas mais diversas hierarquias também são representados em nodos cuja *label* é o nome utilizado para representar o atributo. As propriedades associadas são o conjunto de atributos não dimensionais associados.

4. As ligações de precedência associadas às hierarquias são representadas através de relações em que o nodo de entrada corresponde ao atributo dimensional de menor detalhe e o nodo de saída a um atributo dimensional com um nível maior de detalhe. O relacionamento estabelecido entre estes nodos poderia ter um nome pré-definido, no entanto, considera-se que este deve refletir a relação. Portanto, o tipo da relação deve ser escolhido no momento da modelação com base nos nodos envolvidos.

5. Os factos encontram-se associados às dimensões através de uma relação em que o nodo de saída corresponde ao facto e o nodo de entrada à dimensão. Mais um vez, o tipo do relacionamento deve ser escolhido no momento da modelação tendo em conta os nodos incluídos no mesmo.

De referir que existem vários estudos que pré-estabelecem as relações entre o facto e a dimensão e entre os diferentes níveis hierárquicos. Por exemplo, alguns desses estudos definem que os relacionamentos entre o facto e a dimensão é FACTO e a na hierarquia PRECEDE (Sellami et al., 2020). No entanto, optou-se por não seguir esta abordagem, para que fosse possível garantir que o relacionamento fosse definido com base nos nodos envolvidos. Isto permite que o grafo possua uma leitura mais intuitiva, sendo facilmente identificados os relacionamentos estabelecidos entre os nodos. Assim sendo, no conjunto de regras apresentadas anteriormente foi estipulado entre que nodos se deve estabelecer os relacionamentos, mas não lhes foi atribuída uma identificação.

Tendo por base o conjunto de regras explicadas anteriormente é possível construir um modelo suportado por grafos para o modelo multidimensional definido previamente. O primeiro passo consiste em representar os factos como nodos e as medidas como suas propriedades. Assim, neste caso, vamos considerar o nodo "Venda"com as propriedades "total", referente ao valor faturado e "quantidade"relativo ao número total de artigos envolvidos

na mesma. Depois transformamos cada uma das quatro dimensões do modelo num nodo cujas propriedades correspondem aos atributos não dimensionais das mesmas. Assim, temos o nodo “Funcionário”, “Cliente”, “Dia” e “Produto”. O "Cliente"é constituído pelas propriedades "mail", "nif" e "nome". O nodo "Funcionario"possui informação relativa ao mesmo, nomeadamente, "mail", "nif", "nome" e "telemóvel". O produto contém o "nome" e o "preço". Por fim, o nodo "dia"é composto pelo "dia", "mês" e "ano". De seguida, todos atributos dimensionais foram representados num nodo, dando, neste caso, origem a mais quatro nodos, nomeadamente, "Categoria", "Marca", "Mês" e "Ano". Cada um destes nodos tem como propriedades os seus identificadores e atributos não dimensionais.

Uma vez estabelecidos todos os nodos do modelo é necessário definir os relacionamentos. O nodo "Venda"tem uma relação com cada uma das suas dimensões. Neste caso são quatro, com o "Funcionário"(REGISTADA_PELO), o "Produto"(INCLUI), o "Cliente"(REALIZADA_PELO) e com o "Dia"(DIA).Por fim, foram determinadas as relações estabelecidas nas hierarquias entre os atributos dimensionais de modo a definir os precedentes. O cliente e o funcionário não possuem atributos dimensionais, pelo que não foi necessário criar nenhum relacionamento. No caso do "Produto"que possui duas hierarquias foram gerados dois relacionamentos, um com a "Marca"(MARCA) e outro com a "Categoria"(CATEGORIA). Estamos perante uma hierarquia parcial, em que os atributos dimensionais não se relacionam. Para o dia que possui uma hierarquia com dois atributos dimensionais foi preciso criar dois relacionamentos. Um entre o "Dia"e o "Mês"(MES) e o outro entre o "Mês"e o "Ano"(ANO). Na Figura 19 é possível observar o modelo final obtido, depois de aplicadas todas as transformações necessárias.

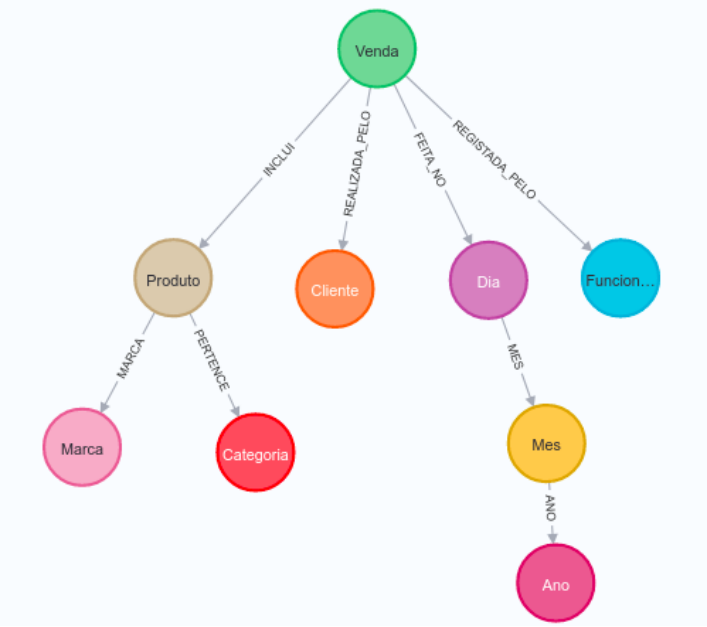

Figura 19: Representação do modelo multidimensional num grafo

“Cubo de dados” funciona como uma metáfora para o armazenamento de um modelo multidimensional (Han et al., 2017). Embora o associemos a uma estrutura 3D no contexto de *data warehouse* este pode ser N-dimensional, consoante o número de dimensões consideradas no modelo. Tipicamente as análises efetuadas

aos dados armazenados nestas estruturas envolvem a visualização dos mesmos tendo em conta diferentes graus de sumarização, ou seja, pesquisas através das diferentes dimensões e atributos dimensionais que compõem o modelo. Denominamos de cuboide a cada cubo gerado para os diferentes níveis de abstração e que nos permite observar os dados mediante as dimensões do modelo (Han et al., 2017). Basicamente, o número de cuboides corresponde ao número de *queries* com diferentes *group by's* que podemos aplicar ao modelo desenvolvido. Assim, dado um conjunto de dimensões podemos gerar um cuboide para cada subconjunto das mesmas que consigamos construir. Para um conjunto de N dimensões é possível obter um total de 2^N cuboides. Esta fórmula está correta apenas para um modelo multidimensional sem hierarquias definidas. No entanto, geralmente, estes modelos são compostos por várias hierarquias com diversos atributos dimensionais. Então, para um modelo N-dimensional considerando também os atributos dimensionais das hierarquias, o número total de cuboides pode ser obtido aplicando a seguinte fórmula:

$$Cuboides = \prod_{i=1}^{n}(Li + 1)$$

onde o *Li* corresponde ao número de níveis da dimensão *i*. O conjunto de todos os cuboides forma uma malha, que é denominada de cubo de dados (Han et al., 2017). O nível mais baixo de abstração é chamado cuboide base, que envolve todas as dimensões do modelo, e o nível mais alto 0-D é o ápice (*appex*), que corresponde ao cenário em que o *group by* está vazio, ou seja, não existe. Geralmente o ápice é denotado por *all* e representa o facto independentemente das dimensões. Por exemplo, no nosso caso o ápice contém as somas do total e da quantidade de produtos das vendas feitas por todos os funcionários, para todos os clientes, independentemente do dia e dos produtos envolvidos. Em suma, quanto mais baixo for o nível de abstração mais utilidade o cubo apresenta numa perspetiva de análise, pelo que para a realização de uma consulta deve ser escolhido o cuboide mais específico que permita realizá-la. Na imagem apresentada abaixo é possível observar os cuboides gerados, em concreto, para o nosso caso de estudo. Para facilitar a sua compreensão e visualização apenas foram considerados os cuboides referentes às dimensões, excluindo os atributos dimensionais das diversas hierarquias. Como temos quatro dimensões, são gerados dezasseis cuboides, ou seja, 2^4.

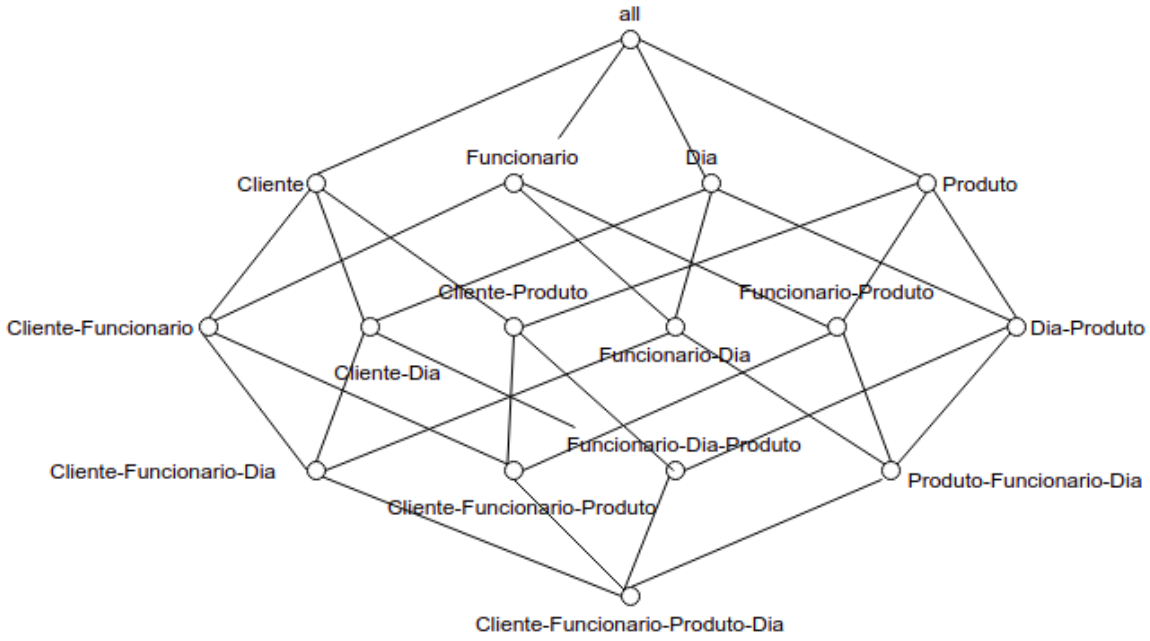

Figura 20: Malha de cubóides que forma um cubo 4D para as dimensões "Cliente", "Funcionario", "Dia"e "Produto"

Um dos aspetos que está associado a este conceito, e que é indispensável investigar, está relacionado com o nível de materialização do cubo. É necessário encontrar um equilíbrio entre a eficiência das operações OLAP padrão e os custos de armazenamento associados à pré-computação dos diversos cuboides. Atualmente, pode-se assumir que de um modo generalizado existem três estratégias:

- **Sem materialização**, consiste em não efetuar o pré-calculo de nenhum dos cubóides que não seja base, tendo apenas o essencial guardado. Esta opção pode tornar as consultas muito lentas dado que todos os cálculos têm de ser efetuado no momento da resolução das diferentes consultas.
- **Materialização parcial**, passa por calcular apenas um subconjunto de todos os cubóides possíveis para um determinado modelo multidimensional. Nesta situação a seleção dos cubóides a pré-calcular, pode ser feita com base nos requisitos necessários pelos agentes de decisão, de modo a torná-lo o mais eficiente possível para o contexto em que se encontra.
- **Materialização total**, requere que todos os cubóides possíveis sejam pré-calculados.

Tendo em conta que estes sistemas de *data warehousing* estão associados a uma quantidade de dados na ordem dos *TeraBytes* TB e a tendência é para aumentar, visto que atualmente são gerados cada vez mais dados a opção da materialização total é um risco. Os custos associados ao armazenamento completo do cubo são muito dispendiosos e exigem uma grande capacidade. Embora numa fase inicial até possa parecer aceitável é preciso ter em atenção que o *data warehouse* está em constante crescimento e a situação pode se tornar incomportável numa fase posterior. A escolha de não utilizar qualquer tipo de materialização, vai requerer um grande

esforço para a realização das operações de consulta, uma vez que todos os dados vão ter de ser sintetizados no momento, tornando-as extremamente lentas. Sendo o foco destes sistemas facilitar o acesso à informação por parte dos agentes é necessário garantir que tais operações sejam realizadas o mais rapidamente possível. Assim sendo, a materialização parcial, surge como um compromisso interessante entre os dois cenários anteriores. No entanto, é importante ter em atenção que cada caso é único e é fundamental definir uma estratégia adequada para o contexto a utilizar. Escolher um subconjunto de cuboides que posteriormente não acrescenta muito para a resolução das queries é um erro grave, que pode colocar o sistema num cenário idêntico àquele em que não foi feita qualquer tipo de materialização.

3.5 IMPLEMENTAÇÃO

Uma vez definido o modelo de dados que suporta o *data warehouse* e as fontes de dados a utilizar, que neste caso correspondem aos ficheiros *csv* enumerados anteriormente, é necessário implementá-lo. Para tal criou-se um script em *Cypher* (Cypher) responsável por criar os índices, os nodos estabelecidos e as respetivas relações, bem como por povoá-los com os dados contidos nos ficheiros. Na imagem apresentada de seguida, encontra-se o exemplo de uma consulta feita à base de dados de forma a verificar o modelo base.

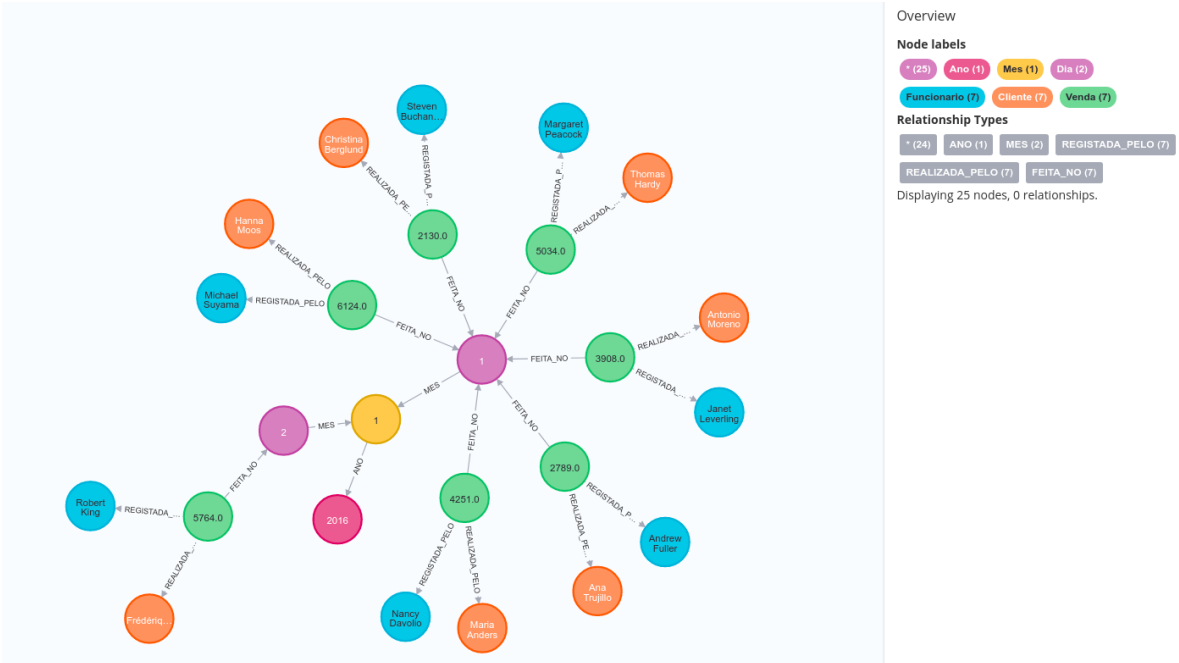

Figura 21: Exemplo de uma consulta ao modelo base

Depois de gerados os nodos e os relacionamentos foi necessário tratar dos respetivos cuboides, para facilitar a visualização dos dados por diferentes perspetivas. O conceito implementado baseou-se na agregação dos dados num nível mais alto de abstração, propagando a informação ao longo da hierarquia definida, sempre que necessário, até chegar à dimensão. Este processo foi efetuado de uma maneira generalizada, de modo

a poder ser reutilizado para diversos modelos com diferentes dimensões. Para isso ser concretizado, criou-se um ficheiro *Python* responsável por realizar estas alterações. Primeiro, dadas as dimensões do modelo, desenvolveu-se uma função para gerar todas as combinações possíveis. Desta forma ficamos a saber todos os cuboides que precisamos de formar. Depois, utilizando o driver *Neo4J Python*, estabelecemos a conexão à base de dados para numa fase posterior podermos executar as transações que nos permitem gerar os cuboides. Por fim, de uma forma dinâmica e com base em todas as combinações possíveis, criámos a operação que nos permitiu agregar as informações relativas às vendas pelas dimensões. Definimos que todos estes nodos de agregação têm como *label* "Ag_", seguido do nome das dimensões que representam. Por exemplo, o nodo que contém a informação da venda para um dado cliente, num determinado dia, vai ter como *label Ag_cliente_dia*. Estes nodos são agregados em função do nível hierárquico mais alto pelo que estabelecem uma relação AGREGADO_DE com as dimensões a que se referem. De referir ainda que as propriedades destes nodos retratam as informações relativas às vendas que pretendemos avaliar, ou seja, as medidas, no nosso caso o total e a quantidade que representam a soma de todas as vendas relativas às dimensões envolvidas. Uma vez executado este ficheiro para a base de dados gerada anteriormente podemos verificar que aos nodos já existentes acresceram todos os nodos agregados que correspondem aos cuboides como se pode observar na Figura 22:

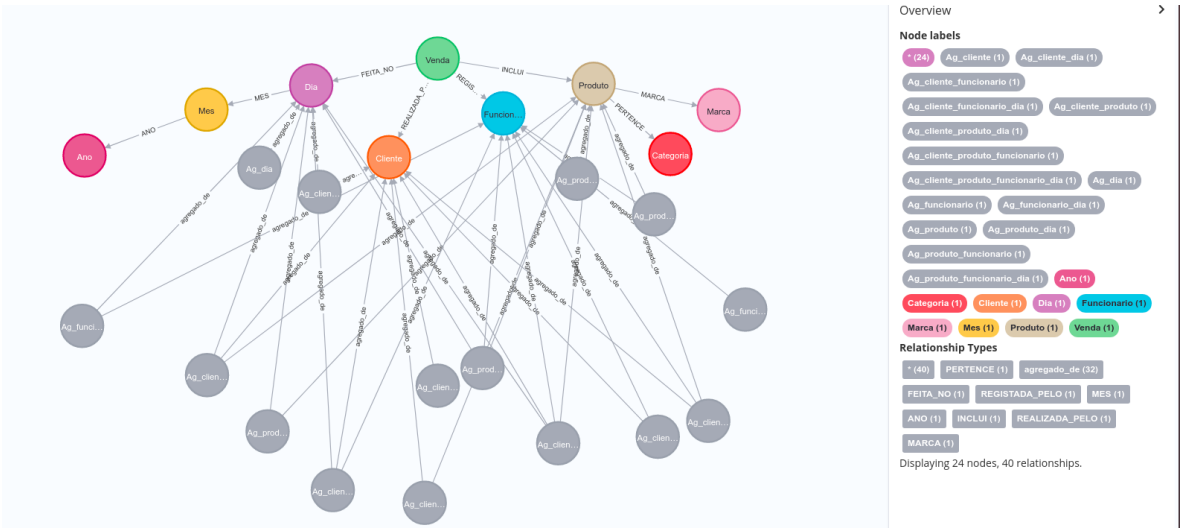

Figura 22: Grafo obtido com a criação dos nodos e relacionamentos referentes aos cuboides

Como podemos verificar na Figura 22 temos um total de vinte e quatro nodos, dos quais quinze correspondem aos nodos agregados, que correspondem a todos os cuboides possíveis excluindo o ápice. Tendo implementado o cubo estamos em condições de dar resposta às consultas OLAP definidas previamente. De uma forma generalizada, e atendendo ao modelo implementado, o processo para resolução das diferentes consultas será muito idêntico. Com base num dos filtros aplicados precisamos de encontrar o nodo da dimensão correspondente. Posteriormente e sabendo de que cuboide se trata é necessário encontrar o nodo de agregação que poderá relacionar todas as dimensões que compõem as condições. Supondo que temos uma query na qual os filtros são aplicados ao nível do produto e do cliente, o nodo que pode conter a informação relativa a essas

vendas seria Ag_produto_cliente. Na eventualidade deste nodo se encontrar relacionado com as duas dimensões, apenas necessitamos de aceder às suas propriedades para obter a quantidade dos produtos envolvidos e o total faturado relativo às vendas em que estes intervieram. Assim, podemos verificar se o cubo nos permite resolver a maior parte das operações padrão e a eficácia com que o faz. De seguida apresentam-se algumas das queries em *Cypher* que foram implementadas para as consultas que definidos anteriormente:

- Q1 - Seleção dos resultados das vendas efetudas pela cliente Ann Devon;

```
MATCH (a:Ag_cliente)-[r2:agregado_de]
       ->(c:Cliente {nome: 'Ann Devon'})
RETURN a;
```

- Q2 - Seleção dos resultados das vendas realizadas pelos clientes Ann Devon e Antonio Moreno;

```
MATCH (a:Ag_cliente)-[r2:agregado_de]
      ->(c:Cliente)
WHERE c.nome = 'Ann Devon' OR c.nome = 'Antonio Moreno'
RETURN SUM(a.quantidade), SUM(a.valor)
```

- Q3 - Seleção dos resultados das vendas realizadas pela cliente Ann Devon para o produto Trumpet;

```
MATCH (p:Produto {nome:'Trumpet'})<-[r1:agregado_de]
      -(a:Ag_cliente_produto)-[r2:agregado_de]
      ->(c:Cliente {nome: 'Ann Devon'})
RETURN a,p,r1,r2,c;
```

- Q4 - Selecção dos resultados das vendas cliente Ann Devon, produto Trumpet e dia 20/4/2016;

```
MATCH (p:Produto {nome:'Trumpet'})<-[r1:agregado_de]
      -(a:Ag_cliente_produto_dia)-[r2:agregado_de]
      ->(c:Cliente {nome: 'Ann Devon'})
MATCH (d:Dia {dia: 20, mes:4, ano:2016})
      <-[r3:agregado_de]-(a)
RETURN p,a,c,d,r1,r2,r3
```

- Q5 - Selecção dos resultados das vendas cliente Ann Devon, produto Trumpet, dia 20/4/2016 e funcionario Laura Callahan

```
MATCH (p:Produto {nome:'Trumpet'})<-[r1:agregado_de]
      -(a:Ag_cliente_produto_funcionario_dia)-[r2:agregado_de]
      ->(c:Cliente {nome: 'Ann Devon'})
```

```
MATCH (d:Dia {dia: 20, mes:4, ano:2016})<-[r3:agregado_de]
      -(a)-[r4:agregado_de]
      ->(f:Funcionario {nome: 'Laura Callahan'})
RETURN p,a,c,d,f,r1,r2,r3,r4
```

De seguida, a título de exemplo, encontra-se uma imagem, que reflete o resultado obtido através da execução da *query* 1. É possível identificar dois nodos e um relacionamento. O nodo de agregação indica-nos o valor faturado e o numero de produtos envolvidos nas vendas realizadas pela cliente *Ann Devon*.

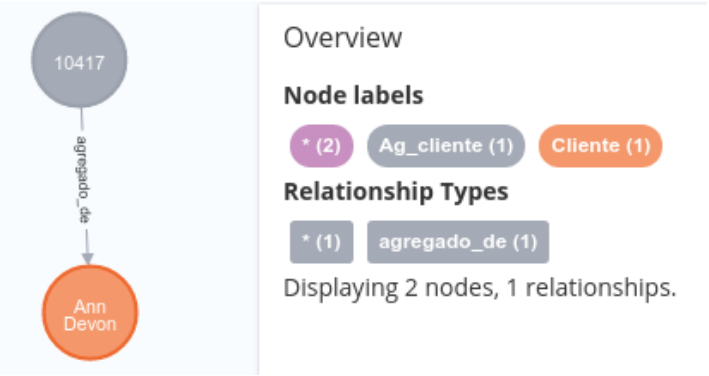

Figura 23: Resultado de execução da *query* 4

Através das consultas apresentadas podemos constatar que tendo o modelo definido é bastante simples analisar uma síntese das vendas, independentemente do filtro que pretendemos aplicar. Os nodos de agregação definidos, bem como a sua nomenclatura revelam-se essenciais para tal, na medida em que com base nas dimensões que pretendemos filtrar conseguimos identificar facilmente os nodos de agregação que se podem revelar bastante úteis. Uma vez reduzida a pesquisa pelos nodos através da sua *label*, apenas precisamos de aplicar os filtros com base nos relacionamentos estabelecidos.

4

HIPERCUBOS EM HIPERGRAFOS

4.1 PRÍNCIPIOS E MODELOS

O conceito de hipergrafos foi introduzido em 1973 pelo *Berge* (Ouvrard, 2020). Os hipergrafos surgem-nos como uma generalização dos princípios associados aos grafos. Num modelo de grafos as relações são estabelecidas aos pares, enquanto que os hipergrafos permitem-nos definir relacionamentos que envolvam mais do que dois nodos (Ouvrard, 2020). Esta característica torna-os mais apropriados para a modelação e representação de redes complexas, nas quais os relacionamentos abranjam vários nodos. Hoje em dia, existem vários sistemas de gestão de base de dados que suportam diretamente os hipergrafos, como é o caso do *HypergraphDB* (HyperGraphDB). No entanto, o *Neo4J*, o sistema que adotámos na realização deste trabalho, não o permite fazer nativamente. Contudo, já existe um conjunto de abordagens alternativas para a implementação deste conceito em *Neo4J*. Estas serão discutidas posteriormente na secção 4.4.

Em termos práticos um hipergrafo manifesta-se como um conjunto finito de vértices e hiperramos (1) (Bahmanian e Sajna, 2015). Os hiperramos retratam os relacionamentos (3) e podem envolver um ou mais nodos. Cada hiperramo é constituído por um conjunto finito de nodos (4), não vazio, que representa aqueles que pertencem ao relacionamento definido. Assim sendo, podemos definir um hiperramo como um sub-conjunto dos vértices que compõem o hipergrafo. O número total de vértices e de hiperramos constitutem respetivamente a ordem e o tamanho do hipergrafo (Bahmanian e Sajna, 2015). Isto significa que se considerássemos um hipergrafo constituído por trinta vértices e dez hiperramos, a ordem do mesmo seria trinta e o tamanho dez.

$$H = (V, E) \tag{1}$$

$$V = \{v_1, ..., v_m\} \tag{2}$$

$$E = \{e_1, ..., e_k\}, k > 0 \tag{3}$$

$$e = \{v_1, .., v_p\} \tag{4}$$

$$\{v1, .., vp\} \subset V \tag{5}$$

Esta característica dos hipergrafos, que lhes permite conectar mais do que dois nodos através de um relacionamento, torna-os uma boa opção para obter a semântica de dados complexos. Nestes, toda a lógica está expressa numa única hiperaresta, ao invés de ser desconstruída por múltiplas arestas, como acontece nos restantes modelos (Mutlu et al., 2019). Esta desconstrução constitui uma limitação, na medida em que pode modificar a ideia que se pretende transmitir, o que leva a que posteriormente a interpretação do modelo não reflita totalmente o planeado. Supondo que temos três destinos (D1, D2, D3) e uma linha aérea (L1), vamos considerar que a linha aérea permite ligar o destino D1 ao D2, bem como o destino D2 ao D3. Num grafo de propriedades teríamos então os quatro nodos representados sendo que todos os destinos estabeleceriam um relação com o nodo representativo da linha aérea, dado que todos fazem parte da mesma. Esta abordagem faz com que não seja claro quais os destinos que são conectados por aquela linha. Este problema de semântica pode ser facilmente colmatado num hipergrafo, no qual poderíamos ter duas hiperarestas, uma composta por D1, D2 e L1 e outra pelo D2, D3 e L1. Desta forma, ficariam evidentes quais os destinos se encontram conectados por aquela linha aérea.

Em última instância o conceito de hipergrafos constitui uma mais valia para a modelação de modelos com um elevado número de relacionamentos. As suas principais características permitem a construção de um modelo que privilegia a interpretação do mesmo. Contudo, é preciso ter em atenção que nem tudo são vantagens e que, dependendo dos dados a modelar e dos objetivos a atingir, é importante avaliar se a utilização de um hipergrafo é mesmo necessária. Geralmente os hipergrafos têm associados um custo de armazenamento mais elevado, na medida em que possuem mais nodos e relacionamentos, quando comparados com o modelo de grafos de propriedades (Mutlu et al., 2019).

4.2 REPRESENTAÇÃO DE HIPERCUBOS EM HIPERGRAFOS

Com o intuito de melhorar a performance e a facilidade com que são realizadas as análises necessárias ao nosso modelo multidimensional foi estudada uma alternativa ao modelo de propriedades apresentado anteriormente, na secção 3.4. A ideia passa por manter o modelo base representado num grafo de propriedades, isto é, não alterar a forma como os factos, dimensões e hierarquias estão modelados. No entanto, ao invés de construir todo o cubo num grafo de propriedades, vamos experimentar uma abordagem diferente, assente nos conceitos de um hipergrafo.

Basicamente, o objetivo consiste em separar estas estruturas de análise por dimensões e atributos dimensionais, de modo a criar um subgrafo que facilite a sua análise. Na realidade, o subgrafo, corresponde a um hipergrafo que agrupa em cada uma das dimensões todos os nodos agregados referentes à mesma. Os nodos agregados, neste caso, correspondem a uma síntese dos factos e, como tal, são compostos pelas mesmas propriedades, ou seja, as mesmas medidas às quais é necessário aplicar uma função de agregação. Tipicamente a função utilizada é a soma. Portanto, em termos práticos, o conjunto de cada dimensão ou atributo dimensional e dos seus nodos agregados constitui uma hiperaresta. O conjunto destas hiperarestas, que nos permitem

obter uma síntese dos factos sobre as diferentes perspectivas fornecidas pelas dimensões, constituem o hipergrafo. Desta forma, é expectável que seja mais simples realizar uma análise por dimensão, uma vez que a informação se encontra toda concentrada e relacionada através do hipergrafo. Nesta situação, cada hiperaresta é desenhada para conter a informação referente aos cuboides que compõem o cubo. De referir que estas se encontram organizadas por dimensão ou atributo dimensional. Pretende-se, pois, que estes cuboides, que têm por base a mesma dimensão ou atributo dimensional, sejam representados numa hiperaresta. A título de exemplo, foram escolhidos três dos cuboides que envolvem a dimensão referente ao "Funcionario"para se representar a hiperaresta que é possível obter (Figura 24). Os cubóides escolhidos para o efeito foram: “Funcionario-Dia”, “Funcionario-Cliente” e “Funcionaio-Dia-Cliente”.

Assim, a hiperaresta é composta pelos nodos do "Funcionario"e ainda três nodos de agregação, representados a verde na Figura 24, que se encontram identificados pela *label* "Valor". Adicionalmente, foram utilizadas *labels* para identificar o cuboide a que se referem. Por exemplo, o nodo com as *labels* "Valor"e "Cliente"representam o cuboide “Funcionario-Dia”. Por sua vez, os nodos com a labels "Valor", "Dia"e "Cliente"retratam o cuboide "Funcionario-Dia-Cliente". O nodo com a *label* "Valor"e "Cliente"refere-se ao cuboide "Funcionario-Cliente". Para o nodo funcionário poderíamos construir mais quatro cuboides na hiperaresta, mas, para não sobrecarregar a imagem, optou-se por apresentar apenas estes três como exemplo.

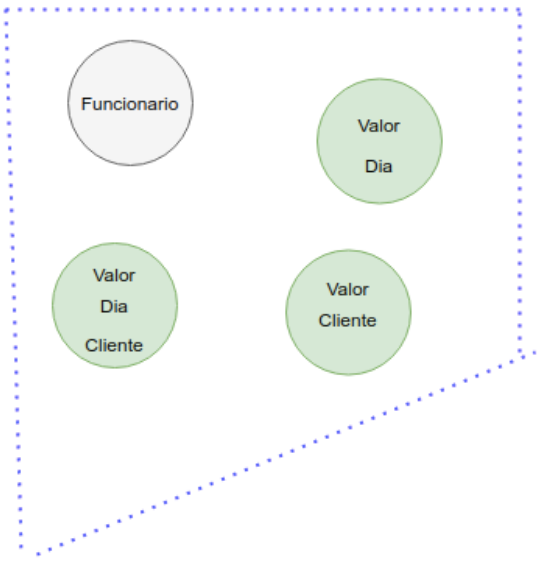

Figura 24: Hiperaresta para representar os cubóides da dimensão Funcionário

4.3 QUERIES SOBRE HIPERCUBOS EM HIPERGRAFOS

Os sistemas OLAP focam-se fundamentalmente em operações de análise sobre dados sintetizados, de forma a fornecer aos agentes de decisão uma visão geral sobre os mesmos, para que estes possam tomar decisões

melhor fundamentadas. Desta forma, todas as operações padrão OLAP implementadas concentram-se na parte do modelo de dados constituído pelo hipergrafo, uma vez que é este o responsável por todos os cuboides que contêm a visão sobre as vendas, através dos mais diversos eixos de análise. Nesta perspetiva é essencial fazer um estudo sobre o modo como podemos realizar as queries sobre os hipercubos num hipergrafo.

Para tal, vamos começar por considerar uma simples operação *Dice* com a qual pretendemos obter a informação sobre as vendas aplicando apenas um filtro relativo a uma dimensão. Com a intenção de avaliar como podemos navegar pelo cubo para obter uma resposta, vamos recorrer à consulta já apresentada anteriormente, na qual se pretende obter a perspetiva das vendas para um determinado cliente, neste caso, a 'Ann Devon'. O primeiro aspeto passa por verificar que dimensões ou atributos dimensionais estão a ser aplicados como filtro. Neste caso concreto verificamos que é apenas o cliente, no entanto, numa operação *slice* podemos envolver muitos mais. Com base no filtro, é possível identificar a hipearesta que contém os dados que precisamos para a resposta. Nesta *query*, a hipearesta necessária seria composta apenas pelo cliente e por todos os nodos de agregação referente ao mesmo. Sendo que, não é aplicado nenhum filtro adicional, apenas precisamos de encontrar o nodo de agregação correspondente na hiperaresta que contém as métricas relativas às vendas para o cliente. Nesta situação, o nodo de agregação teria como propriedades o número total de produtos envolvidos em todas as vendas efetuadas pelo cliente, bem como o total gasto pelo mesmo em vendas até ao momento.

Se considerarmos um exemplo mais complexo, no qual são aplicados mais filtros, como por exemplo, uma consulta sobre as vendas para a cliente 'Ann Devon', efetuada pela funcionária 'Laura Callahan', no dia vinte de abril de 2016, verificamos que, embora as condições aumentem, o processo de resolução mantém-se. Neste caso temos três filtros, referentes a três dimensões, e, como tal, a hipearesta será composta, mais uma vez, por um dos nodos de dimensão do filtro, e todos os nodos de agregação respetivos.

Na Figura 25 é possível observar a hipearesta que é responsável por fornecer a resposta a cada uma das consultas definidas anteriormente. Como todas as consultas têm a cliente 'Ann Devon' como filtro vamos encontrar a solução sempre na mesma hipearesta. Apenas necessitamos de encontrar o nodo de agregação correto, tendo em conta, os restantes filtros. O nodo apresentado a verde na figura, representa o nodo de agregação que nos permite dar resposta à *query* um, na qual os resultados apresentados referem-se apenas à cliente 'Ann Devon'. Assinalado a vermelho está o nodo de agregação referente à terceira consulta sendo os resultado obtidos relativos à cliente 'Ann Devon' para o produto 'Trumpet'. O nodo a laranja representa os dados relativos à consulta número quatro, ou seja, ao cubóide "Cliente-Produto-Dia", pelo que a consulta envolve uma condição para cada uma das dimensões. Por fim, o roxo, é referente à consulta número cinco que envolve todas as dimensões. Tirando o filtro referente ao cliente, que se encontra representado através do nodo "Cliente", os restantes vão ser aplicados ao nível do relacionamento pelo que ainda não se encontram visíveis na figura as condições adicionais aplicadas a cada um dos nodos de agregação.

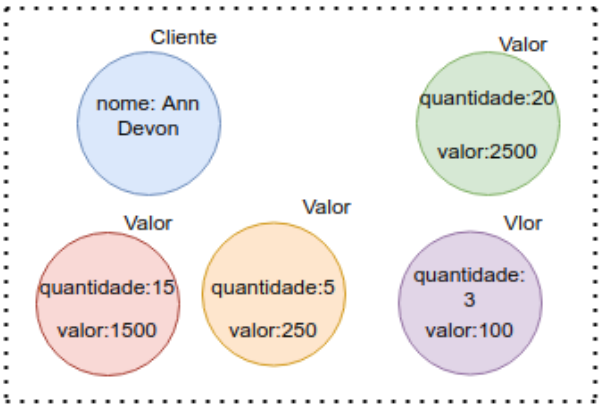

Figura 25: Exemplo de hiperarestas que nos permitem responder às *queries* base definidas

4.4 MODELAÇÃO E IMPLEMENTAÇÃO DO CASO DE APLICAÇÃO

Uma vez que estamos a utilizar como sistema de gestão de base de dados o *Neo4J* e este não suporta nativamente a implementação de um modelo de hipergrafos foi necessário arranjar uma forma de aplicar os mesmos conceitos (Mutlu et al., 2019). A solução encontrada passou por criar um nodo adicional para servir de ligação entre todos os nodos constituintes da hiperaresta. Podemos então dizer que este nodo adicional representa a hiperaresta. No nosso caso este nodo serve para efetuar a ligação entre uma dimensão e todos os seus nodos agregados. Em termos práticos, isto significa que, para além das dimensões que constituem a agregação e do nodo que contém essa informação, temos um nodo adicional, que não é composto por nenhuma propriedade e cujo objetivo apenas consiste em servir de ponte para que todos esses nodos possam estar ligados de algum modo.

De forma a implementar o hipergrafo em *Neo4J*, vamos então considerar três tipos de nodos que, juntos, formam o conjunto de nodos de cada uma das hiperarestas, que constituem o modelo e consequentemente o hipergrafo 7. Tal separação é necessária para ser mais fácil identificar o nodo que representa a dimensão, que suporta a hiperaresta, e os nodos que contêm as sínteses das vendas. Assim, em termos práticos, o hipergrafo será constituído por três tipos de nodos, nomeadamente:

- **Nodos entidade** - nodo base da hiperaresta e com o qual todos os restantes nodos se relacionam, este representa a entidade real do modelo multidimensinal, neste caso concreto, as dimensões do modelo. De referir que as *labels* e as propriedades destes nodos vão ser as atribuídas no modelo base do grafo de propriedades já apresentado. Posto isto, as propriedades destes nodos podem ser consideradas de certa forma imutáveis(Zimanyi et al., 2013).

- **Nodos atributo** - são diretamente relacionáveis com os nodos de entidade e funcionam como uma ponte entre os nodos entidades e os nodos literais de forma a se poder criar a hipearesta. Este nodo não

terá qualquer tipo de propriedades e neste caso possui como *label* o nome da dimensão representado no nodo entidade seguida de agregado. No nosso caso, esta *label* é suficiente, uma vez que apenas possuímos um facto, mas caso seja necessário identificar o facto a que se referem os nodos literais é importante que seja incluído o mesmo à *label* deste nodo.

- **Nodos literais** - representam os nodos que armazenam os diferentes valores para os factos tendo em conta um conjunto de fatores, ou seja, dependendo do cubóide que representam. No nosso contexto estes nodos têm como propriedades as medidas referentes às vendas, ou seja, número total de artigos e o valor faturado. A *label* atribuída a estes nodos foi "Valor". De salientar, que os valores destes nodos são atribuídos de acordo com o contexto (Zimanyi et al., 2013).

$$H = (V, E) \tag{6}$$

$$V = (Ve \cup Va \cup Vl) \tag{7}$$

$$E = (Ee \cup El) \tag{8}$$

Para adicionar mais semântica ao grafo, é necessário utilizar as arestas. Recorrendo aos relacionamentos estabelecidos entre os diferentes nodos e das propriedades das mesmas é possível acrescentar informação extra (Ghrab et al., 2015). Para representar a hiperaresta são necessários dois tipos de relacionamentos. Um que permita estabelecer uma ligação entre os nodos entidade e os nodos atributo e outro que represente o relacionamento que os nodos atributos formam, por sua vez, com os diversos nodos literais 8. O tipo destes relacionamentos entre os nodos atributos e literais será definido posteriormente. A estes foram adicionados um conjunto de propriedades com informação relativa às restantes dimensões do modelo. Supondo que no nodo entidade temos representada a dimensão cliente. O nodo atributo tem o nome de "cliente_agregado"e as propriedades no relacionamento deste nodo com os literais são referentes ao "Funcionario", "Dia"e "Produto". De referir que estas propriedades contêm o identificador de cada uma. No total, para cada hiperaresta, teremos tantos nodos literais como cuboides em que a dimensão representada no nodo entidade pertence. De seguida é possível observar a título de exemplo o modelo para a hipearesta referente ao cliente que será implementada em Neo4J. De referir que não se encontram todos os cuboides representados. Assim, na Figura 26 é possível identificar, um nodo entidade, um nodo atributo e três nodos literais. Estes referem-se ao cuboide "cliente-dia", "cliente-produto"e "cliente-funcionario", respetivamente, como se pode verificar na figura.

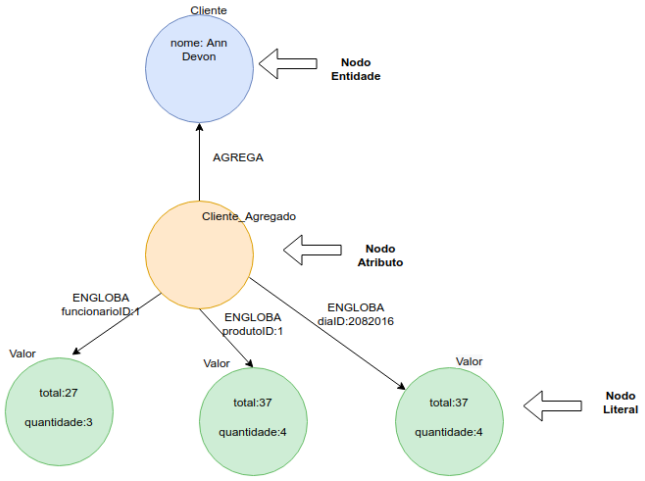

Figura 26: Exemplo do hipergrafo para a dimensão cliente

O modelo base composto por todos os factos, dimensões e atributos dimensionais encontra-se representado num grafo de propriedades como o apresentado na secção 3.4, pelo que o processo de implementação foi exatamente o mesmo. O procedimento para a criação dos cuboides é que foi modificado, de modo a que os cuboides pudessem ser representados num hipergrafo. Esta metodologia foi também idealizada de uma forma generalizada, tal como, a apresentada anteriormente na secção 3.5 para o grafo de propriedades. Assim sendo, foram reaproveitadas as funções *Python* desenvolvidas para gerar todas as combinações possíveis dadas as dimensões e atributos dimensionais do modelo e de conexão à base de dados. Partindo desta base, apenas foi necessário implementar a função que, sustentando-se em todas combinações possíveis entre as dimensões do modelo, cria todos os hipergrafos necessários.

O algoritmo utilizado consiste em para cada dimensão gerar todos os seus cubóides. A função criada para gerar os cuboides guarda num *array* todas as combinações possíveis, pelo que, posteriormente, só precisamos de iterá-lo para construção de todos os cuboides. Para cada dimensão, os cuboides vão ser gerados sequencialmente, isto é, para cada uma delas geramos todos os nodos de agregação, atendendo às condições necessárias e, depois, passamos à seguinte. Assim sendo, o primeiro passo consiste em encontrar todos os nodos da dimensão selecionada. Ou seja, supondo que começamos pela dimensão do cliente para cada um dos clientes armazenados na BD, vamos gerar a hipearesta com todos os nodos agregados e assim sucessivamente.

Além disto, foi ainda necessário estabelecer um conjunto de nomenclaturas específico para que todos os elementos pudessem ser facilmente identificáveis. Assim, definiu-se que os nodos adicionais que foram criados, que representam uma hiperaresta, iriam ter como *label* o nome da dimensão seguido da terminação agregado. Por exemplo, no caso do produto, será "produto_agregado". Estes nodos possuem uma relação AGREGA com a dimensão em questão. No caso do produto, temos, então, o nodo "produto_agregado"que estabelece relações do tipo AGREGA, com o produto em questão. Para além destas relações o nodo que representa a hiperaresta também tem relacionamentos com todos os nodos agregados associados à dimensão. Tal como na situação anterior, os nodos agregados possuem como *label* "Valor"e têm como propriedades a soma do valor e

da quantidade de todas as vendas. O relacionamento definido entre estes nodos e os nodos das hiperarestas possuem como propriedades valores das outras dimensões, que também definem o cuboide representado no nodo "Valor"correspondente. Na imagem ilustrada posteriormente pode-se verificar que a base do modelo continua a ser um grafo de propriedades e que os cuboides foram gerados com base em hipergrafos através dos nodos adicionais.

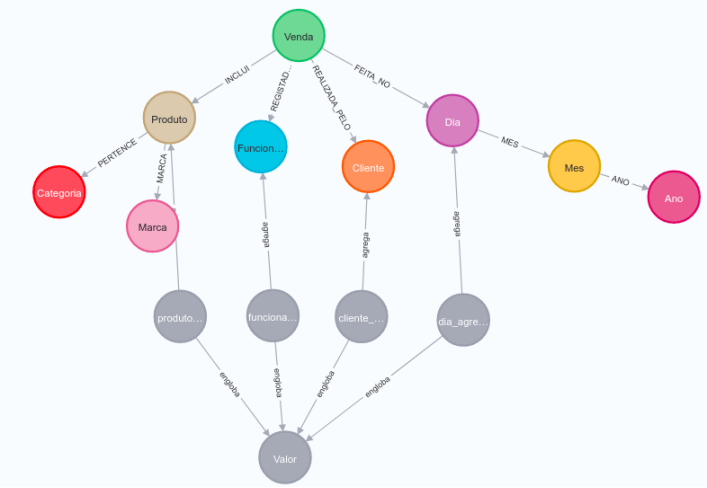

Figura 27: Representação do modelo multidimensional num hipergrafo

Dado que o modelo de dados que suporta o cubo já se encontra devidamente estabelecido, bem como implementado, estão reunidas as condições necessárias para se poder efetuar as mais variadas operações padrão OLAP. Mais uma vez, de forma a garantir a coerência e a possibilitar a realização de uma análise comparativa com o primeiro modelo apresentado, as consultas efetuadas são exatamente as mesmas.

De um modo generalizado o processo que será implementado para cada uma das *queries* será o mesmo, dado que, com maior ou menor complexidade o método de resolução assenta nos mesmos princípios. Considerando o conjunto de filtros que compõem a consulta precisamos de selecionar um dos envolvidos e, com base na dimensão filtrada, encontrar o nodo que armazena a informação pretendida. Uma vez identificado o nodo, temos acesso a todos os nodos agregados que o envolvem e, portanto, independentemente dos restantes filtros, vamos conseguir obter os resultados mais facilmente. Com base nas restantes condições da consulta podemos alcançar o nodo agregado correspondente, através dos atributos do relacionamento que o mesmo realiza com o nodo criado para suportar a hiperaresta. De forma a entender mais facilmente o processo e a poder visualizá-lo, de seguida, encontram-se as *queries* que foram desenvolvidas, na linguagem *Cypher* do *Neo4J*, para responder às consultas definidas na secção 3.3. De salientar que o resultado de cada uma, é um nodo de agregação constituído pelas métricas sintetizadas das vendas, que corresponde, em termos práticos, a uma célula do cubo. Nos relacionamentos efetuados entre os nodos de agregação e aqueles que suportam a hipearesta, foram utilizados os identificadores das dimensões. Assim, caso o filtro aplicado seja relativo a outro dos seus atributos não dimensionais, é necessário obter o identificador correspondente, como se verifica com o produto na terceira consulta. Quando existe mais do que um filtro aplicado à mesma dimensão, como é o caso

da *query* Q2, que aplica duas condições à dimensão cliente, é necessário apresentar a soma dos dois nodos de agregação obtidos.

- Q1 - Seleção dos resultados das vendas efetudas pela cliente Ann Devon;

```
MATCH (v:Valor)<-[r:engloba]-(a:cliente_agregado)
      -[r1:agrega]->(c:Cliente {nome: 'Ann Devon'})
WHERE r.funcionario is null AND r.dia is null
  AND r.produto is null
RETURN v,a,r,r1,c;
```

- Q2 - Selecção dos resultados das vendas realizadas pelos clientes Ann Devon e Antonio Moreno;

```
MATCH (v:Valor)<-[r:engloba]-(a:cliente_agregado)
      -[r1:agrega]->(c:Cliente)
WHERE c.nome = 'Ann Devon' OR c.nome = 'Antonio Moreno'
  AND r.funcionario is null AND r.dia is null
  AND r.produto is null
RETURN SUM(v.quantidade), SUM(v.valor);
```

- Q3 - Selecção dos resultados das vendas realizadas pela cliente Ann Devon para o produto Trumpet;

```
MATCH (p:Produto {nome: 'Trumpet'})
MATCH (v:Valor)<-[r:engloba]-(a:cliente_agregado)-[r1:agrega]
      ->(c:Cliente {nome: 'Ann Devon'})
WHERE r.produto = p.produtoID AND r.dia is null
  AND r.funcionario is null
RETURN v,r,a,r1,c;
```

- Q4 - Selecção dos resultados das vendas cliente Ann Devon, produto Trumpet e dia 20/4/2016;

```
MATCH (p:Produto {nome: 'Trumpet'}),
      (d:Dia {dia:20,mes:4,ano:2016})
MATCH (v:Valor)<-[r:engloba]-(a:cliente_agregado)
      -[r1:agrega]->(c:Cliente {nome: 'Ann Devon'})
WHERE r.produto = p.produtoID AND r.funcionario is null
  AND r.dia = d.diaID
RETURN v,c,a,r,r1;
```

- Q5 - Selecção dos resultados das vendas cliente Ann Devon, produto Trumpet, dia 20/4/2016 e funcionario Laura Callahan

```
MATCH (p:Produto {nome: 'Trumpet'}),
      (d:Dia {dia:20,mes:4,ano:2016}),
      (f:Funcionario {nome:'Laura Callahan'})
MATCH (v:Valor)<-[r:engloba]-(a:cliente_agregado)
      -[r1:agrega]->(c:Cliente {nome: 'Ann Devon'})
WHERE r.produto = p.produtoID AND r.dia = d.diaID
  AND r.funcionario = f.funcionarioID
RETURN v,c,a,r,r1;
```

De seguida, a título de exemplo é possível observar o resultado de execução da *query* 1, onde é possível identificar os três nodos, entidade (laranja), literal (azul) e atributo (cinzento).

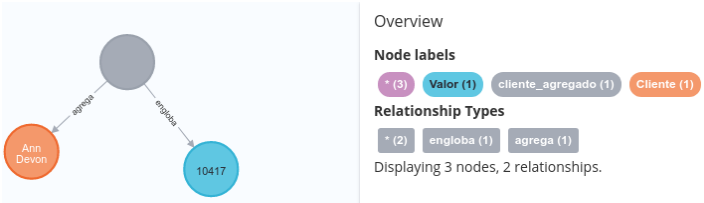

Figura 28: Representação do modelo multidimensional num hipergrafo

4.5 ANÁLISE DOS RESULTADOS OBTIDOS

Tendo implementado ambas as propostas dos modelos apresentados previamente, é necessário avaliá-los. Sendo este tipo de sistemas definidos pela eficácia com que permitem o acesso aos dados, independentemente da complexidade da consulta envolvida, uma das métricas que iremos utilizar no processo de análise será o tempo de execução de cada uma das queries implementadas.

Com o intuito de reduzir o impacto que a cache pode ter nos resultados dos tempos de execução de cada consulta, optou-se por executar cada uma mais do que uma vez. Assim sendo, cada *query* foi executada dez vezes. Na Tabela 3 podemos verificar quanto tempo o primeiro modelo, ou seja, aquele que está assente apenas num grafo de propriedades, demorou a executar cada uma das cinco consultas definidas. Os tempos de execução apresentados nas tabelas estão em milissegundos.

Tabela 3: Tempos de execução das *queries* para o primeiro modelo

Modelo 1				
Q1	Q2	Q3	Q4	Q5
298	339	418	586	750
12	10	17	13	12
7	8	10	13	13
5	10	8	7	8
5	7	9	7	4
4	5	7	5	6
3	6	6	7	6
6	5	7	4	4
5	6	8	5	7
4	5	4	5	6

De forma a tornar possível a comparação entre ambos os modelos, foi necessário efetuar o mesmo processo para o segundo modelo. De salientar, que o segundo refere-se ao modelo apresentado na secção 4.4 em que os cuboides estão assentes num hipergrafo. De seguida, na Tabela 4, é possível observar os tempos de execução, em milissegundos, para cada uma das cinco consultas nas dez tentativas realizadas.

Tabela 4: Tempos de execução das *queries* para o segundo modelo

Modelo 2				
Q1	Q2	Q3	Q4	Q5
394	479	625	713	743
10	15	16	16	17
9	15	16	14	10
8	8	10	9	9
10	7	8	8	9
10	7	8	9	9
4	9	6	8	6
6	6	7	7	6
11	6	4	7	5
11	4	5	6	6

Como podemos observar pelos resultados apresentados em ambas as tabelas, a primeira execução de qualquer uma das *queries* possui um tempo muito díspar de todos os restantes. Este comportamento deve-se ao facto de na primeira tentativa ainda não existir nada em *cache*. De modo a evitar que estes valores tenham um impacto muito negativo nos tempos de execução, vamos excluí-los dos tempos finais. Para os valores obtidos nas restantes tentativas vai ser calculada a média, de forma, a obter o valor mais fidedigno possível para o tempo de execução de cada *query*. Na Tabela 5 podemos verificar os tempos médios obtidos por cada modelo para cada uma das cinco consultas realizadas.

Tabela 5: Tempo médio de cada operação

Query	Modelo_1	Modelo_2
1	5.6	8.7
2	6.8	8.6
3	8.4	8.9
4	7.3	9.3
5	7.3	8.6

A quantidade dos dados com que o *data warehouse* foi povoado não é suficientemente elevada para a análise dos tempos de execução refletir os números reais no contexto de uma organização, que geralmente é na ordem dos *TeraBytes*. Contudo, no âmbito deste trabalho, é suficiente para termos uma noção, em termos comparativos, do modo como cada um dos modelos se comporta perante os diferentes tipos de *queries*. Como podemos observar pelos tempos apresentados na tabela 5, embora a diferença não seja substancial, o modelo suportado por um grafo de propriedades tem tendência a ser mais rápido, o que é, em parte, muito influenciado pela pouca quantidade de dados associada. Esta diferença entre os modelos é muito reduzida, uma vez que estamos a falar em valores na ordem dos milissegundos. À medida que o número de dimensões aumenta na query, verificamos que os tempos de execução associado a ambos os modelos se mantêm bastantes constantes. No modelo do grafo de propriedades, cada condição envolve um nodo e uma relação que necessitam de ser encontradas, ao invés do modelo de hipergrafos que contém esta informação mais condensada. Isto implica que no caso do primeiro modelo, quantas mais dimensões estiverem no filtro, mais nodos e relacionamentos são necessários para resolver a query. No modelo de hipergrafos são sempre três, nomeadamente, os nodos entidade, atributo e valor. De salientar ainda que nas queries executadas raramente foram utilizados os identificadores das dimensões para filtrar. Isto implica um esforço extra no caso do hipergrafo para o obter.

Com o intuito de melhorar a análise desta questão, nas Figura 29 encontra-se um exemplo da resposta obtida pela execução da query número quatro com base no primeiro modelo apresentado na secção 3.5, enquanto que a Figura 30 está representa a resposta à mesma *query* mas assente no modelo de hipergrafos. De relembrar que esta consulta consiste na seleção dos resultados referentes às vendas realizada pela cliente, 'Ann Devon', no dia 'vinte de abril de 2016', para o produto 'Trumpet'.

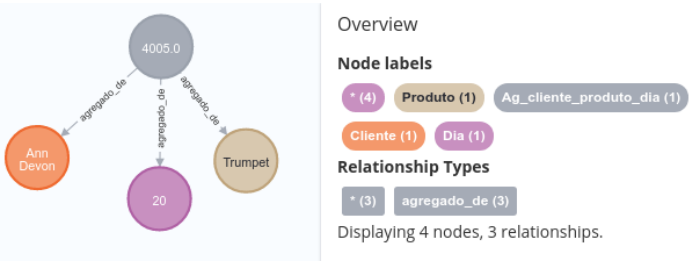

Figura 29: Resultado da query quatro para o grafo de propriedades

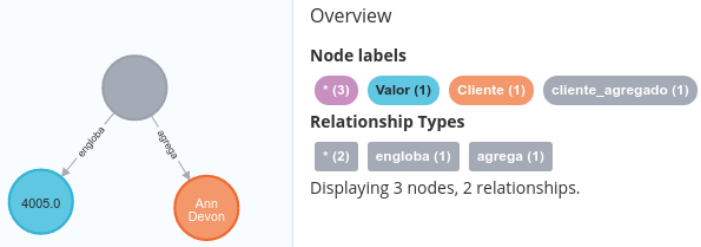

Figura 30: Resultado da query quatro para o hipergrafo

Nestas Figuras 29 e 30 é facilmente observável que, com apenas três filtros, o grafo de propriedades necessita de recorrer a mais nodos e relacionamentos para responder à query do que o hipergrafo. Sempre que um filtro é adicionado à consulta, o grafo de propriedades obriga à utilização de pelo menos mais um nodo e um relacionamento. Este comportamento já não se verifica no caso do hipergrafo, dado que os filtros estão aplicados ao nível do relacionamento ENGLOBA, que, neste caso, está estabelecido entre o nodo valor e o nodo "cliente_agregado". Podemos também constatar que o valor obtido por ambos os modelo foi o mesmo, de '4005'.

Por fim, é fundamental analisar o armazenamento utilizado. Como já foi referido, a materialização dos cuboides tem um elevado custo ao nível do armazenamento. Assim, na Tabela 6 podemos observar o número de nodos necessários para construir o modelo base num grafo de propriedades, sem a materialização de qualquer cuboide, para o primeiro modelo que implementámos (na Figura 22), que retrata os cuboides num grafo de propriedades, e para o segundo modelo (Figura 28), no qual os cubóides estão representados num hipergrafo.

Tabela 6: Quantidade de nodos envolvidos na construção dos modelos

Modelo	Número de nodos
Modelo Base	6 025
Modelo 1	30 058
Modelo 2	32 625

Nesta Tabela 6 é possível constatar que o número de nodos aumentou aproximadamente em cinco vezes com a implementação dos cubóides face ao modelo base que não tinha nenhum cuboide materializado. Este comportamento evidencia o custo de armazenamento associado à sua implementação. De salientar, que apenas implementamos os cuboides ao nível das dimensões. Contudo, se tivéssemos considerado também todos os níveis hierárquicos do modelo multidimensional a quantidade de nodos envolvidos na contrução dos modelos seria consideravelmente superior. Portanto, a questão da materialização dos cuboides é muito importante e delicada, pelo que merece um estudo mais aprofundado num futuro próximo. Também podemos verificar que o número de nodos associados ao hipergrafo é ligeiramente superior ao do grafo de propriedades, uma vez que tivemos de criar nodos adicionais para sustentar as hiperarestas, uma vez que, como já referido, o *NEO4J* não os suporta nativamente.

5

CONCLUSÕES E TRABALHO FUTURO

5.1 CONCLUSÕES

A conceção e implementação de um *data warehouse* numa base de dados NoSQL é um processo complexo e subjetivo que ainda está em investigação. Ao longo dos últimos anos têm surgido várias propostas na literatura, mas ainda não existe um método consensual, que tenha sido aceite pela generalidade dos investigadores e profissionais do domínio. Geralmente os *data warehouses* assentam em base de dados relacionais cujos modelos se encontram bem definidos e são conhecidos por todos. Isso deve-se à existência de uma metodologia devidamente cimentada. No entanto, o aumento exponencial dos dados gerados atualmente, provenientes de redes sociais ou de sistemas de sensorização, por exemplo, têm exposto algumas das debilidades deste tipo de base de dados. Este contexto fez com que o estudo sobre a implementação dos mesmos numa base de dados NoSQL crescesse. Hoje, as base de dados NoSQL são vistas como uma alternativa viável às bases de dados relacionais, na medida em que permitem colmatar algumas das suas debilidades, como a reduzida capacidade de armazenamento e a falta de flexibilidade.

Ao longo desta dissertação foi a apresentado um processo para a implementação de um *data warehouse* usando um modelo suportado por grafos, que constitui um dos quatro tipos de base de dados NoSQL existentes e que se caracterizam pela sua performance e flexibilidade.

Os *data warehouses* surgem como um elemento essencial nos processos de tomada de decisão. Usualmente, estes são responsáveis por armazenar informação relativa ao negócio de uma determinada organização, de modo a que esta possa ser utilizada para fundamentar as decisões a tomar. Os dados que o integram tanto podem ser provenientes das base de dados operacionais, que mantêm o negócio em funcionamento, como podem resultar de informação externa ao sistema, por exemplo, de redes sociais, para indicar a tendência do mercado e a preferência dos utilizadores. O conhecimento guardado num data warehouse pode ser utilizado, posteriormente, por sistemas de análise, que permitem apresentar ao utilizador final, uma análise sintetizada dos mesmos sobre diferentes perspetivas. Normalmente quem tira mais beneficio destes sistemas são gestores, executivos e analistas, que os utilizam para os auxiliar nas escolhas e decisões que precisam de efetuar.

Com o intuito de identificar qual o sistema de gestão de base de dados mais adequado para implementar os modelos desenvolvidos foi necessário realizar, numa primeira fase deste trabalho de dissertação, um estudo sobre sistemas de *data warehousing* e bases de dados NoSQL, nomeadamente, as suportados por grafos.

Nesta fase foram analisados apenas quatro sistemas de gestão de base de dados suportadas por grafos o que constitui apenas uma ínfima parte dos existentes atualmente no mercado. Apesar de terem sido vistas as principais características de cada um, seria interessante verificar, também, como seria possível implementar uma base de dados em cada um deles e não apenas no sistema que foi selecionado para a implementação do trabalho desenvolvido. Para além disso, foi também realizado um estudo sobre alguns dos diferentes modelos de grafos, em particular, sobre grafos de propriedades e hipergrafos. No entanto, de forma a melhorar a modelação de um *data warehouse* num modelo orientado por grafos, seria importante examinar outros tipos de modelos que pudessem contribuir para a representação do modelo multidimensional num grafo, de forma mais completa e apropriada.

Ao nível do *data warehouse* foi abordada a sua arquitetura, de um modo generalizado, ou seja, desde a fase de recolha dos dados até à apresentação dos mesmo numa interface intuitiva e fácil de utilizar. No entanto, o foco deste estudo foi a modelação, a implementação e utilização de um *data warehouse*. Desta forma, o processo de extração, transformação e carregamento dos dados, algo que constitui um elemento fundamental no desenvolvimento de um *data warehouse*, não foi analisado detalhadamente. Este processo serve para tratar os dados provenientes das diferentes fontes para que possam ser integrados num sistema comum. Nas bases de dados NoSQL é expectável que este processo seja muito mais simples, uma vez que estas não possuem um esquema tão inflexível quanto as base de dados relacionais. Todavia, este aspeto não foi alvo desta dissertação. Basicamente, o estudo efetuado centrou-se na modelação, nomeadamente, nos conceitos associados ao modelo multidimensional, e no processamento analítico de dados – OLAP.

A segunda fase de desenvolvimento centrou-se na conceção de um processo para a representação de um modelo multidimensional num modelo suportado por grafos. O modelo multidimensional é essencialmente composto por factos, dimensões e hierarquias. Nesta fase estabeleceu-se um conjunto de regras para ajudar a representar os conceitos associados a um modelo multidimendonal num modelo baseado em grafos. Num modelo relacional, os factos e as dimensões são armazenados em tabelas e existem vários modelos definidos, como por exemplo, em estrela, floco de neve ou constelação de factos. Em grafos, este mapeamento ainda não está definido. Como tal, apresentámos uma proposta para a sua representação num grafo de propriedades. Um grafo de propriedades é constituído por um conjunto de nodos e de ramos que os permitem relacionar. Cada um destes elementos pode possuir um conjunto de propriedades bastante diversas, que contêm informação adicional acerca da entidade respetiva. Posteriormente, foi abordada a questão do processamento de um cubo e a sua consequente materialização. Um cubo é uma estrutura de dados que é constituída por um conjunto de cuboides, no qual cada um deles representa uma síntese de factos tendo em conta as diversas dimensões que integram o modelo de dados. A materialização de um cubo é considerada uma mais valia para a execução das consultas, uma vez que reduzem os cálculos necessários a efetuar no momento de execução. Contudo, é um processo que também tem custos de armazenamentos elevados, o que pode constituir um risco. Esta é uma questão sensível, que merece atenção e um estudo mais aprofundado. Nesta dissertação optamos por materializar apenas os cuboides que envolvem as dimensões do modelo.

Tendo bem definido e implementado o cubo referente a um modelo multidimensional associado a um contexto de vendas num modelo de grafos de propriedades, estudámos uma possível alternativa para este primeiro

processo de implementação de um conjunto orientado por grafos. A parte mais complexa do modelo de dados em mãos consistiu na implementação dos cuboides que revelou um crescimento elevado quer ao nível dos nodos quer dos relacionamentos no modelo, pelo que constituiu o foco do estudo. A representação de um cuboide pode envolver o relacionamento de inúmeras dimensões. Por exemplo, o cuboide base envolve todas as dimensões do modelo, o que num modelo N-dimensional implica a síntese de N dimensões. Como os grafos de propriedades apenas permitem estabelecer relacionamentos binários estas representações exigem vários nodos e ligações entre os mesmos. Nesta perspetiva, surgiu a opção de utilizar hipergrafos para modelar os cuboides. Os hipergrafos são grafos que ao contrário de todos os outros permitem estabelecer relacionamentos entre mais do que dois nodos, aos quais denominam de hiperramos. Assim, foi possível representar os cuboides referentes a uma dimensão numa hipearesta. Isto significa que, a cada dimensão está associado um conjunto de nodos que contêm uma síntese dos factos, tendo em conta as diferentes vistas proporcionadas pelas dimensões. O conjunto formado pela dimensão e pelos nodos de agregação formam a hipearesta. Este novo modelo baseado em hipergrafos permitiu que o trabalho necessário para resolver as diferentes consultas permanecesse constante, independentemente do número de dimensões a filtrar. Apesar dos esforços desenvolvidos na fase de modelação ficaram ainda várias outras alternativas que precisariam de ser examinadas.

Uma vez implementado também o cubo num hipergrafo foi possível realizar uma análise comparativa das duas abordagens. O objetivo passou por analisar o modo como os grafos poderiam ser percorridos, a eficiência com que permitiriam executar as mais variadas queries e ainda o armazenamento que requereriam para a sua materialização. Para tal, foi definido um conjunto de operações padrão OLAP, aplicadas através de um conjunto de consultas normalmente realizadas sobre o *data warehouse* pelos sistemas de análise. As operações definidas foram executadas em ambos os modelos implementados e o tempo de execução da sua execução, bem como da sua evolução consoante o aumento da complexidade da *query*, foi comparado. No geral, constatou-se que o modelo assente no grafo de propriedades é ligeiramente mais rápido do que o hipergrafo. No entanto, os tempos obtidos nem a quantidade e variedade de operações padrão definidas para o efeito foram suficientes para retirar conclusões. Em última instância, e considerando que a quantidade de dados utilizada nesta situação foi muito reduzida, os resultados apresentados pelo hipergrafo parecem mais prometedores devido à estabilidade que apresentam ao nível da representação dos cuboides. Isto é, independentemente do número de filtros a aplicar, a quantidade de nodos necessários para responder à consulta é sempre a mesma.

5.2 TRABALHO FUTURO

Os sistemas de *data warehousing* são bastantes complexos e envolvem um leque muito diversificado de áreas de trabalho. Como tal, é possível identificar diferentes linhas para dar seguimento ao trabalho desenvolvido até ao momento. De um modo geral, podemos considerar que estes sistemas se encontram divididos em quatro fases fulcrais, nas quais existe espaço para melhorar e investigar cada uma. Entre elas destacam-se a preparação dos dados, a modelação e implementação do *data warehouse* e o processamento analítico.

No desenvolvimento destes sistemas é fundamental realizar um pré processamento dos dados, de modo a garantir a integridade e consistência dos mesmos. Este foi um tema não muito abordado ao longo desta disser-

tação, mas no qual existe muito espaço para estudo e que tornaria mais completa e interessante o trabalho que foi realizado. Os dados que alimentam o *data warehouse* são provenientes de diversas fontes, pelo que não é possível garantir a consistência dos mesmos. Assim sendo, é importante efetuar um pré processamento para assegurar a qualidade dos mesmos. As principais etapas a realizar nesta fase consistem na limpeza, integração, redução e transformação dos dados. A limpeza dos dados está associada à identificação e remoção de outliers, preenchimento dos valores em falta e resolução de inconsistências. A integração consiste na inclusão dos dados das diversas bases de dados, ficheiros ou cubos, que podem ter nomes diferentes para os mesmos atributos, o que pode causar inconsistências e redundância. A redução dos dados envolve a seleção de um subconjunto dos mesmos para a realização de eventuais análises estatísticas. A transformação dos dados envolve vários processos, por exemplo, a normalização dos dados. Ainda ao nível do processamento dos dados seria interessante analisar com mais cuidado o processo ETL, extração, transformação e carregamento dos dados. Neste caso, os dados que alimentam o *data warehouse* provêm de ficheiros *csv*, uma vez que não é o foco da dissertação. Contudo, como trabalho futuro seria importante construir um processo automatizado que, periodicamente, extraísse a informação necessária de um conjunto de fontes definidas, procedia ao seu processamento e posteriormente carregava-os para os modelos de dados definidos. Ainda nesta ótica é fundamental realizar um estudo mais cuidado sobre a frequência com que se deve efetuar o processo de refrescamento do sistema. É, assim, necessário garantir um equilíbrio entre o esforço despendido e a exigência de dados atualizados, para que as decisões tomadas sejam o mais corretas possíveis.

Apesar do estudo da modelação e implementação do *data warehouse* ter sido o foco de desenvolvimento desta dissertação, ainda existe muito trabalho para ser realizado. Ao nível da modelação ainda existem várias alternativas para serem abordadas no que diz respeito aos modelos de grafos apresentados. No entanto, consideramos que isso não é o mais importante, mas sim o tratamento de pequenos detalhes que nos permitam tornar o trabalho realizado mais consistente e completo para os modelos desenvolvidos e que (ainda) não foram alvo de grande estudo. Assim, a questão da materialização dos cubos deve ser repensada e analisada. Esta é uma parte fundamental, uma vez que tem impacto direto na eficiência com que as consultas podem ser feitas e nos custos de armazenamento associados aos dados. Então, seria interessante desenvolver novas estratégias para o caso de estudo apresentado. Posteriormente, poder-se-ia analisar os resultados obtidos pelas diferentes estratégias, de forma a encontrar a opção mais adequada. Outro assunto a tratar nesta fase remete-nos para a quantidade e qualidade dos dados utilizados para alimentar um *data warehouse*, sendo importante testar o modelo para uma quantidade de dados mais aproximada da realidade destes modelos.

Ainda no contexto de análise do modelo desenvolvido e implementado, seria fundamental abordar mais detalhadamente as operações OLAP padrão. Definir mais operações para o caso de estudo apresentado e analisar o comportamento e os tempos de execução das diferentes queries. Desta forma, os resultados obtidos refletiriam melhor a realidade e talvez fosse possível tirar mais conclusões sobre os aspetos a melhorar em ambos os modelos estabelecidos.

Ao nível do processamento analítico, existe muito trabalho que pode ser realizado, uma vez que não foi efetuado nenhum desenvolvimento centrado nesta questão. Associado ao processamento analítico estão os sistemas OLAP que permitem efetuar consultas sobre os dados armazenados no *data warehouse* como base em

diferentes vistas disponibilizadas pelas dimensões. Posteriormente, estas consultas são utilizadas para apresentar numa interface simples e intuitiva ao utilizador final – agentes de decisão, que normalmente correspondem a gestores, analistas e executivos. Estes agentes, por sua vez, utilizam estas APIs para sustentarem as decisões que tomam para o negócio. Desta forma, seria interessante analisar as diferentes ferramentas que permitem implementar a interface e os gráficos mais utilizados, isto é, os grafismos típicos destes sistemas e preferenciais dos utilizadores.

REFERÊNCIAS BILIOGRÁFICAS

Arcade Analytics. Arcade analytics. URL https://arcadedb.com/analytics#arcade-analytics.

Mohammad A. Bahmanian e Mateja Sajna. "connection and separation in hypergraphs". *Theory and Applications of Graphs*, 2015.

Alfredo Bolt e Wil Van der Aalst. "multidimensional process mining using process cubes". *Lecture Notes in Business Information Processing*, páginas 4–5, 2015. doi: http://dx.doi.org/10.1007/978-3-319-19237-6_7.

Arnaud Castelltort e Anne Laurent. "nosql graph-based olap analysis". páginas 217–224, 2019.

Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, e Ronan Tournier. "implementing multidimensional data warehouses into nosql". *17th International Conference on Enterprise Information Systems, Proceedings*, 2015.

Cypher. Cypher. URL https://neo4j.com/developer/cypher/.

Cytoscape. Cytoscape. URL https://cytoscape.org/.

Anupam Das, Anirban Mitra, Surendra Nath Bhagat, e Subrata Paul. "issues and concepts of graph database and a comparative analysis on list of graph database tools". *International Conference on Computer Communication and Informatics*, páginas 20–25, 2020. doi: https://doi.org/10.1109/ICCCI48352.2020.9104202.

DBEngine. Db engines ranking. URL https://db-engines.com/en/ranking/graph+dbms.

Amine Ghrab, Oscar Romero, Sabri Skhiri1, Alejandro Vaisman, e Esteban Zimanyi. A framework for building olap cubes on graphs. *Computer Science*, 2015.

Matteo Golfarelli, Dario Maio, e Stefano Rizzi. The dimensional fact model: A conceptual model for data warehouses. *International Journal for Innovative Research in Science Technology*, 1998.

Gremlin. Gremlin query language. URL https://tinkerpop.apache.org/gremlin.html.

Jiawei Han, Micheline Kamber, e Jian Pei. *DATA MINING Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2017.

HyperGraphDB. Hypergraphdb. URL http://www.hypergraphdb.org/.

William H. Inmon. *Building the Data Warehouse*. Wiley Publishing Inc, 4 edition, 2005.

JanusGraph. URL https://janusgraph.org/.

Ezio Lefons, Francesco Di Tria, e Filippo Tangorra. "design process for big data warehouses". *DSAA 2014 - Proceedings of the 2014 IEEE International Conference on Data Science and Advanced Analytics*, páginas 512–518, 2014. doi: https://doi.org/10.1109/DSAA.2014.7058120.

Yunkai Liu e Theresa M. Vitolo. "graph data warehouse: Steps to integrating graph databases into the traditional conceptual structure of a data warehouse". *Proceedings - 2013 IEEE International Congress on Big Data*, páginas 255–264, 2013. doi: https://doi.org/10.1109/BigData.Congress.2013.72.

Elzbieta Malinowski e Esteban Zimányi. Advanced data warehouse design from conventional to spatial and temporal applications. *Data Vault 2.0*, 2008.

Markjbrown. "introdução ao azure cosmos db.". URL https://docs.microsoft.com/pt-pt/azure/cosmos-db/introduction.

Alev Mutlu, Furkan Goz, Mert Erdemir, e Pinar Karagoz. Comparison of querying performance of neo4j on graph and hyper-graph data model. *Conference: KDIR 2019 (11th International Conference on Knowledge Discovery and Information Retrieval)*, 2019.

Neo4J. Neo4j. URL https://neo4j.com/.

OrientDB. Orientdb. URL https://orientdb.org/.

Xavier Ouvrard. Hypergraphs: an introduction and review. *CERN*, 2020.

Shefali Patil, Gaurav Vaswani, e Anuradha Bhatia. "graph databases- an overview". *International Journal of Computer Science and Information Technologies*, páginas 2657–660, 2014.

Jaroslav Pokorný. "graph databases: Their power and limitations". 2017.

Juan Manuel Pérez, Rafael Berlanga, María José Aramburu, e Torben Bach Pedersen. "integrating data warehouses with web data: A survey". *EEE Transactions on Knowledge and Data Engineering*, páginas 940–955, 2008.

Franck Ravat, Olivier Teste, Ronan Tourniera, e Gilles Zurfluh. "graphical querying of multidimensional databases". *Conference: Advances in Databases and Information Systems*, 2007.

Deepak Singh Rawat e Navneet Kumar Kashyap. "graph database: A complete gdbms survey.". *IJIRST –International Journal for Innovative Research in Science & Technology 3*, páginas 217–226, 2017. doi: https://doi.org/10.1109/ICCCI48352.2020.9104202.

Amal Sellami, Ahlem Nabli, e Faiez Gargouri. Transformation of data warehouse schema to nosql graph data base. *Advances in Intelligent System and Computing*, 2020.

Studio3T. Nosql. URL https://studio3t.com/knowledge-base/articles/nosql-database-types/.

Kevin Tardivel. What are the benefits of graph databases in data warehousing? - sonra. URL https://sonra.io/2017/06/12/benefits-graph-databases-data-warehousing/.

Francesco Di Tria, Ezio Lefons, e Filippo Tangorra. A proposal of methodology for designing big data warehouses. 2018.

Rania Yanguia, Ahlem Nablib, e Faiez Gargouria. Automatic transformation of data warehouse schema to nosql data base: Comparative study. *Procedia Computer Science*, páginas 255–264, 2016.

Esteban Zimanyi, Salim Jouili, Sabri Skhiri, e Amine Ghrab. An analytics-aware conceptual model for evolving graphs. *Conference: International Conference on Data Warehousing and Knowledge Discovery)*, 2013.