Universidade do Minho
Escola de Engenharia

Maria Inês Peixoto da Costa

Etiquetagem e Rastreio de Fontes de Dados num Big Data Warehouse

outubro de 2019

Etiquetagem e Rastreio de Fontes de Dados num Big Data Warehouse

Maria Inês Peixoto da Costa

UMinho | 2019

Universidade do Minho
Escola de Engenharia

Maria Inês Peixoto da Costa

Etiquetagem e Rastreio de Fontes de Dados num *Big Data Warehouse*

Dissertação de Mestrado

Mestrado Integrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação da

Professora Doutora Maribel Yasmina Santos

Outubro de 2019

DESPACHO RT - 31 /2019 - ANEXO 3

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

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

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

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

Licença concedida aos utilizadores deste trabalho

Atribuição-NãoComercial
CC BY-NC

https://creativecommons.org/licenses/by-nc/4.0/

AGRADECIMENTOS

Olhando para trás o sentimento de saudade intensifica-se, o percurso na universidade passou a correr, é um misto de sentimentos, por um lado a tal nostalgia por outro a felicidade de ter sido capaz de seguir em frente, apesar dos obstáculos, e de finalmente chegar a esta última fase. Estes anos contribuíram para a construção da minha personalidade e fizeram-me crescer, pois permitiram-me conhecer e trabalhar com várias pessoas cujas ideias e métodos de trabalho diferem dos meus. Passo agora aos agradecimentos a algumas das pessoas que marcaram este percurso de uma forma especial.

Destaco o primeiro agradecimento à professora Maribel, por estar sempre disponível para ajudar e pelas suas sugestões, que fazem sempre muita diferença. Considero-a uma referência de profissional, pelo gosto que demonstra em tudo que faz e principalmente pela sua a exigência em ensinar, pela sua disponibilidade e pacificidade e ao mesmo tempo boa disposição ao lidar com as pessoas.

Ao João que esteve sempre bastante presente ao longo deste trabalho e que contribui de forma fundamental para a conclusão dele. Por estar sempre disponível por reunir quase sempre semanalmente e por toda a paciência investida nessas reuniões. Para além das revisões dos documentos que efetuou com todo o pormenor e por todas as dicas que foram sempre vantajosas para a melhoria do trabalho.

Ao restante pessoal do lid4, nomeadamente à Carina e ao Carlos que desde o início me receberam da melhor forma, por me terem posto à vontade e por estarem sempre dispostos ajudar com as suas opiniões e ideias.

Ao Fernando, a pessoa mais importante que a universidade me deu oportunidade de conhecer, por todo o apoio que me deste, por sempre tentares fazer-me ver que sou capaz de tudo e mais alguma coisa com trabalho e esforço. Pela tua enorme paciência para tolerar o meu mau feitio e por me acalmares sempre que necessário. Por estares sempre disponível para ajudar e por partilhares o teu empenho e força de vontade. E claro, por me acompanhares nas longas noites de estudo.

À minha prima Joana que sempre se preocupou em perguntar pelo estado trabalho e por todas revisões à dissertação que realizou.

Por último, mas não menos importante, agradeço a toda a minha família, principalmente à minha mãe e ao meu pai por sempre acreditarem em mim, por todo o apoio moral, por nunca me deixarem desistir, por reconhecerem o esforço que investi ao longo deste percurso e por me proporcionarem sempre tudo. Foi verdadeiramente graças a vocês que consegui alcançar o sucesso neste trajeto!

DESPACHO RT - 31 /2019 - ANEXO 4

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 avanços nas Tecnologias de Informação levam as organizações a procurar valor comercial e vantagem competitiva por meio da recolha, armazenamento, processamento e análise de dados. Os *Data Warehouses* surgem como uma peça fundamental no armazenamento dos dados, facilitando a sua análise sob diversas perspetivas e permitindo a extração de informação que poderá ser utilizada na tomada de decisão. A elevada disponibilidade de novas fontes de dados e os avanços que surgiram para a recolha e armazenamento dos mesmos, fazem com que seja produzida uma imensa quantidade de dados heterogéneos, gerados a taxas cada vez maiores. Adjacente a este facto surgiu o conceito de *Big Data*, associado ao volume, velocidade e variedade dos dados, ou seja, grandes volumes de dados com diferentes graus de complexidade, muitas vezes sem estrutura nem organização, caraterísticas estas que impossibilitam o uso de ferramentas tradicionais. Como tal, surge a necessidade de adotar o contexto de *Big Data Warehouses*, que naturalmente acarreta outros desafios, pois implica a adoção de novas tecnologias, assim como a adoção de novos modelos lógicos que permitem uma maior flexibilidade na gestão de dados não estruturados e desnormalizados. Por conseguinte, quando o volume de dados e a sua heterogeneidade começam a aumentar, uma vez que derivam de várias fontes que apresentam caraterísticas muito diferentes, emergem novos desafios associados ao *Big Data*, nomeadamente a Governança de Dados. A área de Governança de Dados abrange um grupo de subáreas, tais como Qualidade dos Dados e Gestão de Metadados, as quais oferecem um conjunto de processos para suportar a elevada complexidade inerente nos dados. À medida que o volume de dados num *Big Data Warehouse* começa a aumentar, os processos de negócio também aumentam, pelo que se torna necessário ter informação adicional sobre esses dados, por exemplo, que tabelas e atributos foram armazenados, quando e por quem foram criados e as diversas atualizações que sofreram. O objetivo desta dissertação é propor um sistema para a governança de um *Big Data Warehouse*, de modo a dar a conhecer o conteúdo do mesmo e a forma como este está a evoluir ao longo do tempo. Para tal, é proposto um sistema de catalogação de dados do *Big Data Warehouse*, baseado num grafo, através da etiquetagem e do rastreio de fontes de dados e posterior armazenamento dos metadados recolhidos numa base de dados. Para além de reunir as caraterísticas mais básicas dos dados, regista informações sobre políticas de acesso, *profiling*, a similaridade, *key performance indicators* e processos de negócio.

Palavras-Chave

Big Data Warehouse, Etiquetagem, Governança de Dados, Grafos, Metadados, Rastreio.

Abstract

Advances in Information Technologies lead organizations to search for commercial value and competitive advantage through collecting, storing, processing and analyzing data. Data Warehouses appear as a fundamental piece in data storage, facilitating data analysis from different perspectives and allowing the extraction of information that can be used in decision making. The high availability of new data sources and the advances that have been made for their collection and storage lead to the production of an enormous amount of heterogeneous data generated at increasing rates. Adjacent to this fact, the concept of Big Data appeared, associated to the volume, velocity and variety of data, that is, large volumes of data with different degrees of complexity, often without structure or organization, which makes it impossible to use traditional tools. Thus, the need arises to adopt the Big Data Warehouses context, which naturally brings other challenges, because it implies the adoption of new technologies, as well as the adoption of new logical models that allow greater flexibility in the management of unstructured and denormalized data. Therefore, when the volume of data and its heterogeneity start to increase, once they derive from several sources with very different characteristics, new challenges associated with Big Data emerge, namely Data Governance. The Data Governance domain covers a group of subdomains, such as Data Quality and Metadata Management, which provide a set of processes to support the high complexity inherent in the data. As the volume of data in a Big Data Warehouse starts to increase, the business processes also increase, meaning that it becomes important and necessary to know some additional information about these data, for example, which tables and attributes were stored, when and by whom were created and the several updates they suffered. The aim of this dissertation is to propose a governance system for the governance of a Big Data Warehouse, in order to make its content available, as well as how it is evolving over time. To this end, a graph-based Big Data Warehouse data cataloging system is proposed, by tagging and lineage of data sources and storing metadata in a database. In addition to gathering the basic characteristics of data, it records information about access policies, profiling, similarity, key performance indicators and business processes.

Keywords

Big Data Warehouse, Data Governance, Graphs, Lineage, Metadata, Tagging.

ÍNDICE

ÍNDICE DE FIGURAS

Índice de Tabelas

Lista de Abreviaturas, Siglas e Acrónimos

Neste documento é utilizada uma lista de siglas e acrónimos, tal como enumerado de seguida:

ACID – *Atomicity, Consistency, Isolation, Durability*

API – *Application Programming Interface*

BASE – *Basic Availability, Soft state, Eventual consistency*

BDW – *Big Data Warehouse*

BI – *Business Intelligence*

BPMN – *Business Process Model and Notation*

CAP – *Consistency, Availability, Partition tolerance*

CI – *Competitive Intelligence*

CSF – *Critical Success Factor*

DDL – *Data Definition Language*

DL – *Data Lake*

DSRM – *Design Science Research Methodology*

DW – *Data Warehouse*

ETL – *Extraction, Transformation and Loading*

GFS – *Google File System*

HDFS – *Hadoop Distributed File System*

HiveQL – *Hive Query Language*

HOLAP – *Hybrid-OLAP*

HTTP – *Hypertext Transfer Protocol*

IGC – *Information Governance Catalog*

IO – *Input/Output*

JSON – *JavaScript Object Notation*

JVM – *Java Virtual Machine*

KPI – *Key Performance Indicator*

MOLAP – *Multidimensional-OLAP*

NoSQL – *Not only SQL*

OLAP – *On-line Analytical Processing*

OLTP – *On-line Transaction Processing*

OWL – *Web Ontology Language*

RDBMS – *Relational Database Management System*

REST – *REpresentational State Transfer*

ROLAP – *Relational-OLAP*

SCD – *Slowly Changing Dimension*

SQL – *Structured Query Language*

XML – *eXtensible Markup Language*

YAML – *Yet Another Markup Language*

YARN – *Yet Another Resource Negotiator*

1. INTRODUÇÃO

Neste capítulo são apresentados o enquadramento e a motivação do tema da presente dissertação, bem como os seus principais objetivos e resultados esperados. Para além disso, encontra-se a descrição da abordagem metodológica adotada, o processo realizado para a revisão da literatura e, por último, a organização e estrutura deste documento.

1.1 Enquadramento e Motivação

Os *Data Warehouses* surgem como uma peça central no armazenamento de dados históricos, sendo suportados por modelos de dados que facilitam a pesquisa e a análise sob várias perspetivas (M. Y. Santos & Costa, 2016b). No entanto, estes repositórios tradicionais/relacionais são ineficientes em alguns contextos como, por exemplo, no armazenamento de grandes volumes de dados semiestruturados ou não estruturados, tais como texto, vídeo, imagens, estruturas *nested*, entre outros (C. Costa & Santos, 2017b).

A elevada disponibilidade de fontes de dados, devido ao uso crescente e frequente de *smartphones*, *tablets* e até das próprias redes socias, gera um fluxo gigantesco de dados, produzidos a diferentes velocidades, que contêm dados bastante heterogéneos com formatos muito distintos, surgindo assim, o conceito de *Big Data* (Cassavia, Dicosta, Masciari, & Saccà, 2014). *Big Data* expressa uma nova era na utilização e exploração de dados, pois estes dados não podem ser processados ou analisados recorrendo a processos e ferramentas tradicionais, uma vez que as suas principais características são o volume, a variedade e a velocidade, o que destaca as limitações dos *Data Warehouses* tradicionais (C. Costa & Santos, 2018; Zikopoulos & Eaton, 2011).

Surge então a necessidade de se adotar o contexto de *Big Data Warehouses*, que pode ser visto como um repositório flexível, escalável e de alto desempenho. Este utiliza técnicas e tecnologias de *Big Data* para dar suporte a cargas de trabalho analíticas e complexas (C. Costa & Santos, 2018). Desta forma, os *Big Data Warehouses* trazem novos desafios, pois implicam a adoção de novos modelos lógicos, utilizados, por exemplo, nas bases de dados NoSQL que garantem escalabilidade, flexibilidade e melhor desempenho na gestão de dados não estruturados e desnormalizados (Tria, Lefons, & Tangorra, 2014). Outro dos desafios dos *Big Data Warehouses* é a adoção de novas tecnologias que suportem grandes quantidades de dados, tais como as ferramentas existentes no ecossistema Hadoop, sendo esta uma tecnologia *open-source* que permite o processamento paralelo (Sathi, 2012; Thusoo et al., 2010).

A governança surge como um desafio associado ao conceito de *Big Data*, pelo que nestes ambientes é muito complexo fazer-se a governança de massivos volumes de dados, provenientes de distintas fontes, com diferentes propósitos, diferentes públicos-alvo e consequentemente diferentes restrições (Hashem et al., 2015). Dessa forma, sem as ferramentas adequadas, gerir um ambiente tão heterogéneo e definir políticas de acesso para todos os cenários pode tornar-se quase impossível (C. Costa & Santos, 2017a). Devido às exigências provocadas pela heterogeneidade dos dados será fundamental a capacidade de gerar de forma automática os metadados sobre os dados (Chandarana & Vijayalakshmi, 2014; Katal, Wazid, & Goudar, 2013; Labrinidis et al., 2012). Assim, surge o Atlas, uma ferramenta presente no Hadoop e que inclui capacidades de governança e Gestão de Metadados (Atlas, 2018). Quando o número de dados começa a aumentar, consequentemente aumenta o número de processos de negócio num *Big Data Warehouse*, pelo que é necessário saber que tabelas e atributos foram armazenados, quando e por quem foram criados e as diversas atualizações que sofreram (quando e quantas linhas foram incluídas). Portanto, é necessário saber qual o conteúdo do *Big Data Warehouse* e como este está a evoluir ao longo do tempo. Assim, será fundamental a existência de um sistema para a governança de um *Big Data Warehouse*, através da etiquetagem e do rastreio de fontes de dados e posterior armazenamento dos metadados recolhidos numa base de dados.

1.2 Objetivos e Resultados Esperados

Esta dissertação tem como finalidade propor um sistema de catalogação e governança de dados, cujo componente principal será um grafo de suporte ao sistema, bem como a sua implementação e validação. Este sistema irá contribuir para a manutenção eficiente de um *Big Data Warehouse*, de modo a permitir que se lide melhor com o crescente aumento dos dados e dos processos de negócio. Portanto, com esta dissertação pretende-se que os seguintes objetivos e resultados sejam atingidos:

- Definir os componentes de etiquetagem e rastreio (*tagging* e *lineage*) do sistema de catalogação e governança, e como estes devem ser representados e armazenados numa base de dados de grafos[1];
- Utilizar e estender a API (*Application Programming Interface*) do Apache Atlas[2], que fornece a gestão e governança de metadados, para as organizações criarem um catálogo dos seus dados.

[1] Conceito abordado com maior detalhe nos capítulos 2 e 3.

[2] Conceitos abordados com maior detalhe no capítulo 3.

Esta API deverá permitir a etiquetagem e o rastreio tendo em consideração informação que pode ser obtida de *scripts* DDL (*Data Definition Language*), código ETL (Extração, Transformação e Carregamento), *Metastore* do Hive e outras fontes possíveis[3];

- Derivar, utilizando a API, uma base de dados de grafos, JanusGraph[4], utilizada pelo Atlas, contendo o catálogo do *Big Data Warehouse*, incluindo os metadados sobre as tabelas armazenadas, atributos, refrescamentos, entre outros.

1.3 Abordagem Metodológica

A metodologia selecionada para a realização desta dissertação, enquadrada na área dos sistemas de informação, é a "*Design Science Research Methodology* (DSRM) *for Information Systems*" de Peffers, Tuunanen, Rothenberger e Chatterjee (2007), sendo adequada à elaboração de investigação científica. Assim, ao mesmo tempo que permite ser coerente com os princípios e objetivos estabelecidos, possibilita também a produção, apresentação e avaliação da investigação científica, o que proporcionará uma disciplina de trabalho orientada ao sucesso (Peffers et al., 2007). A metodologia é constituída por seis fases que se adequam aos objetivos e resultados propostos para esta dissertação. Apesar de ser uma abordagem flexível, pois permite que o projeto de investigação se inicie em qualquer uma das quatro primeiras fases presentes no fluxo do processo, este trabalho inicia-se pela identificação do problema e motivação. Posteriormente, percorrem-se as próximas fases, conforme ilustrado na Figura 1.

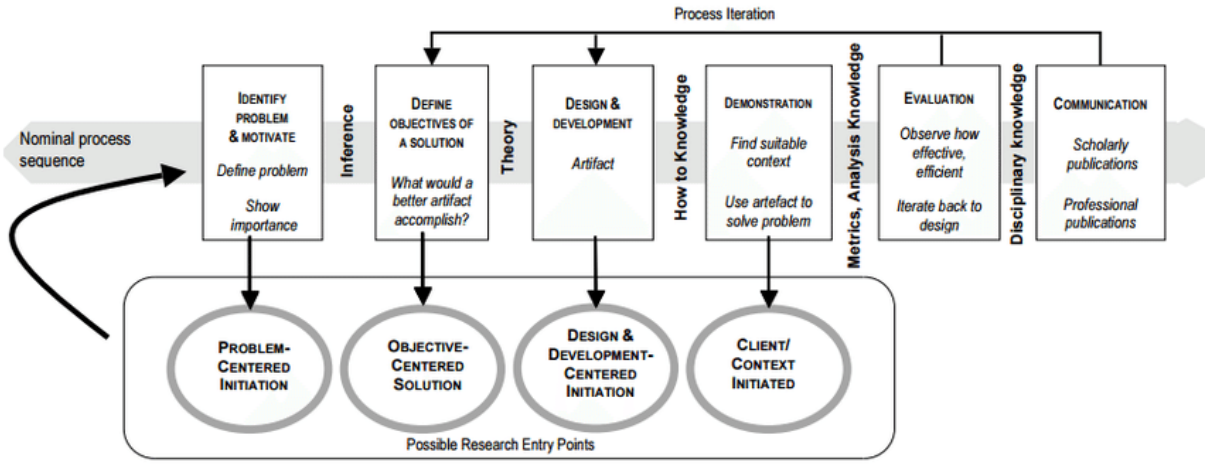

Figura 1. Design Science Research Methodology for Information Systems. Retirado de Peffers et al. (2007).

[3] As fontes indicadas serão exploradas com maior detalhe nos capítulos 2 e 3.

[4] https://janusgraph.org/

1.3.1 Fases da Abordagem Metodológica

Assumindo a abordagem metodológica de investigação selecionada e as datas de entregas intermédias exigidas pela instituição de ensino, surgem várias tarefas e fases a desenvolver:

1. De forma a iniciar-se a abordagem à metodologia DSRM, primeiramente **identifica-se o problema e a motivação** (*Identify Problem & Motivate*), que consiste em identificar e definir os aspetos que diferenciam a solução e o valor que poderão trazer. Esta fase inclui a definição de um **plano de trabalhos** para a dissertação, que consiste num enquadramento inicial ao tema, onde se encontram a apresentação da motivação, objetivos e resultados esperados e identificação da metodologia a adotar;
2. Na fase seguinte, na **definição dos objetivos da solução** (*Define Objectives of a Solution*) são identificados os objetivos a atingir que permitam solucionar o problema e que forneçam o conhecimento necessário para pôr a solução em prática. Procede-se assim, à revisão da literatura que consiste no **enquadramento conceptual** que se baseia na leitura e análise de documentos científicos com o intuito de saber qual o estado da arte, assim como conhecer conceitos e técnicas da área em que se enquadra o tema em questão, nomeadamente *Big Data Warehouses* [5] e *Data Governance* [5]. De seguida, no **enquadramento tecnológico** são apresentadas as tecnologias a utilizar ao longo da dissertação, sendo desenhada uma arquitetura tecnológica e conceptual. Esta fase estará centrada nos recursos presentes no Hadoop[6], nomeadamente o Apache Atlas[6] que fornece a gestão e governança de metadados para as organizações criarem um catálogo dos seus dados;
3. Posteriormente, a **conceção e desenvolvimento** (*Design & Development*) visa a criação do artefacto com modelos, métodos e instanciações necessários para a solução. Esta fase baseia-se na definição dos requisitos do sistema e na modelação do grafo com as regras que orientam os processos de etiquetagem e rastreio de fontes de dados;
4. A **demonstração** (*Demonstration*) consiste na validação do artefacto desenvolvido anteriormente, ou seja, na implementação do grafo proposto;
5. Na **avaliação** (*Evaluation*) é analisado em que medida é que a solução responde ao problema, tendo em conta os objetivos e requisitos definidos, assim como os resultados obtidos da

[5] Conceitos abordados com maior detalhe no capítulo 2.

[6] Conceitos abordados com maior detalhe no capítulo 3.

demonstração, sendo nesta fase tiradas as devidas conclusões. Possivelmente, podem ser realizados ajustes ao que foi sendo desenvolvido como forma de melhorar os resultados, sendo necessário retomar à fase de conceção e desenvolvimento;

6. A última fase, a comunicação (*Communication*) tem por base a documentação e apresentação da dissertação que consiste na divulgação dos resultados que irão ser obtidos, de modo a destacar a utilidade do projeto desenvolvido.

1.3.2 Processo de Revisão de Literatura

Para o processo de revisão do estado da arte é necessária a pesquisa pelos projetos e conceitos chave nos quais se baseia a presente dissertação. Inicialmente, são definidas as bases de dados de referência, as palavras chave a utilizar na pesquisa e os métodos para selecionar os documentos.

Para a pesquisa de documentos, sejam eles livros, revistas, artigos, entre tantos outros, inclui-se bases de dados de referência, nomeadamente, "ACM Digital Library", "IEEE Xplore", "ScienceDirect" e "ResearchGate", a partir de onde as pesquisas são feitas utilizando-se certas palavras chave. As palavras chave definidas são "data warehouse", "big data" "big data warehouse", "data governance", "data lake", "tagging", "lineage", "data profiling", "data quality", "semantic" e "automatic".

Tendo em conta que a área de *Big Data Warehouse* é relativamente recente, não se consideraram filtros temporais. Para além disso, as áreas de *governance*, *tagging* e *lineage*, aplicadas e exploradas, explicitamente e diretamente ao *Big Data Warehouse* são inexistentes, pelo que é necessário pesquisar e analisar estes conceitos no contexto mais tradicional (*Data Warehouse*) e no contexto de *Data Lakes* [7], para posteriormente se estender o estudo ao contexto de *Big Data Warehouse*.

Os documentos que são obtidos nas pesquisas são filtrados através do título e da leitura do *abstract*, sendo posteriormente classificados de leitura prioritária, ou apenas de leitura potencialmente importante ou complementar.

Assim, à medida que se executa a leitura e análise dos documentos recolhidos, quando estes revelam ser pertinentes, procede-se à citação dos mesmos. Para além disso, alguns dos artigos a analisar são disponibilizados pela orientadora da dissertação, podendo estes também ser utilizados e referenciados ao longo do processo de revisão de literatura.

[7] Conceito abordado com maior detalhe no capítulo 2.

1.4 Organização do Documento

De acordo com a metodologia adotada este documento está organizado em sete capítulos que são de seguida sucintamente apresentados.

No primeiro capítulo, encontra-se a introdução que consiste na identificação do problema e a sua motivação que viabilizam o desenvolvimento da dissertação nesta área. Assim, este capítulo encontra-se dividido na apresentação do enquadramento e motivação, dos objetivos e resultados esperados, da abordagem metodológica adotada, sendo aqui explicado o método utilizado no processo de revisão de literatura e, por fim, definida a organização geral do documento.

O capítulo dois diz respeito ao enquadramento conceptual, isto é, à revisão de literatura de estudos e de conceitos chave relacionados ao tema da dissertação. São eles, conceitos associados aos *Big Data Warehouses*, desdobrando este conceito e explorando cada componente, ou seja, *Data Warehouses* e *Big Data*, analisando o seu conceito, caraterísticas e desafios e, por último, *Big Data Warehouses*. Existe ainda outra secção relativa ao armazenamento dos dados, onde serão descritos os conceitos fundamentais dos vários tipos de bases de dados, assim como os modelos de dados. Para terminar, são expostos o conceito de *Data Governance* e as áreas relacionadas para as quais esta dissertação se irá direcionar, assim como são referidos alguns trabalhos relacionados com este assunto.

No terceiro capítulo é apresentado o enquadramento tecnológico, focando-se essencialmente em algumas das ferramentas presentes no ecossistema do Hadoop, nomeadamente, o Hive e o Atlas. É também realizado um breve enquadramento à base de dados que o Atlas incorpora.

O quarto capítulo inclui a fase de *design* e desenvolvimento onde os requisitos do sistema de catalogação e governança são definidos, sendo, posteriormente, modelados os componentes de etiquetagem e rastreio, através da representação do grafo.

No quinto capítulo encontra-se exposto o ambiente de implementação e os dados utilizados, assim como a instanciação e o modo de implementação do grafo. De seguida, é, ainda, efetuada a validação do sistema tendo em conta os requisitos anteriormente indicados.

O sexto e último capítulo apresenta as conclusões a retirar do trabalho realizado e propõe o trabalho futuro associado ao tema em questão.

Por fim, o documento termina com as referências bibliográficas e os apêndices que suportam a dissertação.

2. ENQUADRAMENTO CONCEPTUAL

Como forma de enquadrar a dissertação neste capítulo irão ser apresentados os conceitos associados ao presente tema. Para começar irá ser introduzida uma secção sobre *Data Warehouses*, onde é apresentado o seu conceito. Adiante, encontra-se outra secção onde se descreve o conceito de *Big Data*, assim como as suas caraterísticas, oportunidades, desafios e processamento. Em seguida, é apresentado o conceito de *Big Data Warehouses*. Posteriormente, é abordado o armazenamento dos dados, onde são descritos os conceitos fundamentais e as evoluções dos vários tipos de bases de dados existentes atualmente (SQL, NoSQL e NewSQL), para além de que são expostos os vários modelos de dados definidos. Para fechar este capítulo, existe ainda outra secção relativa ao *Data Governance*. No início, é abordado o conceito de um modo mais individual e depois são aprofundadas e exploradas algumas das áreas que abrange, consideradas relevantes para este trabalho, tais como *Data Quality*, *Data Profiling* e *Metadata*. Por último, são apresentados alguns estudos e trabalhos relacionados com Gestão de *Metadata*, *Tagging* e *Lineage*.

2.1 *Data Warehouses*

Anteriormente as bases de dados das organizações armazenavam apenas dados operacionais, por outras palavras, dados criados pelas operações de negócio envolvidas no processo diário de gestão, tais como gestão de compras, gestão de vendas e faturação. Todavia, a totalidade da organização deve ter acesso rápido e atempado à informação exigida pelos processos de tomada de decisão, para conseguir fazer frente às mudanças dos negócios e à crescente competitividade do mercado (Golfarelli & Rizzi, 2009; Vaisman & Zimányi, 2012).

Desde os últimos anos, as organizações recorrerem cada vez mais ao *Data Warehousing* para melhorar o fluxo da informação e para suportar a tomada de decisão. *Data Warehousing* é um conjunto de métodos, técnicas e ferramentas utilizados para auxiliar os decisores a realizar análises aos dados que ajudem na execução de processos de tomada de decisão e na melhoria dos recursos informacionais (Golfarelli & Rizzi, 2009; Türkmen, 2007).

Um *Data Warehouse* (DW) pode ser um recurso valioso para o acesso fácil aos dados, para execução de análises e relatórios. Porém, implementar e gerir um DW traz vários desafios (Türkmen, 2007). Os dados de diferentes operações de uma empresa são reconciliados e armazenados num repositório central (DW) a partir de onde os analistas extraem informação que permite uma melhor

tomada de decisão (Cho & Ngai, 2003). O DW é um componente fundamental da infraestrutura de *Business Intelligence* (BI) que, para além de permitir consolidar dados heterogéneos a partir de bases de dados, ainda permite transformá-los em indicadores estratégicos para a tomada de decisão (Vaisman & Zimányi, 2012). Um DW reúne e armazena dados provenientes de várias fontes de dados heterogéneas, pelo que aquando da execução de uma consulta permite que esta seja feita localmente e, assim, pode ser obtido um elevado desempenho em consultas de agregação complexas que são necessárias para a análise detalhada, suporte à decisão e *data mining*, pois não há necessidade de se aceder às fontes de informação originais (Theodoratos & Sellis, 1999).

Com base nos seus estudos, Krishnan (2013) afirma que um DW permite recolher, armazenar e gerir os dados, suportando a tomada de decisão da gestão. De acordo com Inmon (2005) a definição de DW, que é aceite como padrão pela indústria, é que o DW se trata do armazenamento de um conjunto de dados:

- Com **variações de tempo**, pois apresenta os dados ao longo do tempo, isto é, sob uma perspetiva histórica, pelo que os dados têm que possuir uma dimensão temporal;
- Para além disso é **orientado ao assunto**, ou seja, DW foca-se nas necessidades analíticas que estão relacionadas com a área em que a empresa em questão se insere;
- É **integrado**, já que os dados que armazena provêm de múltiplas fontes de dados heterogéneas que são agrupadas para dar uma visão unificada dos dados;
- E é **não volátil**, ou seja, os dados são estáveis, não se realizando continuamente operações de inserção, atualização e eliminação, apenas se realiza o acesso aos dados para processar consultas, assim como refrescamentos para acrescentar dados entretanto acumulados na base de dados, pelo que quando a informação é armazenada, não se perde, pois existe um histórico de dados. Logo, após carregamento dos dados estes não podem ser alterados ou eliminados, contudo, podem surgir algumas exceções através das SCD (*Slowly Changing Dimension*) (M. Y. Santos & Ramos, 2017).

Para Kimball e Ross (2013) existem quatro componentes distintos a ser considerados num DW/BI, tal como ilustrado na Figura 2, são eles: os sistemas operacionais de origem, o sistema ETL (com a área de estágio), área de apresentação para suporte de BI e as aplicações de BI.

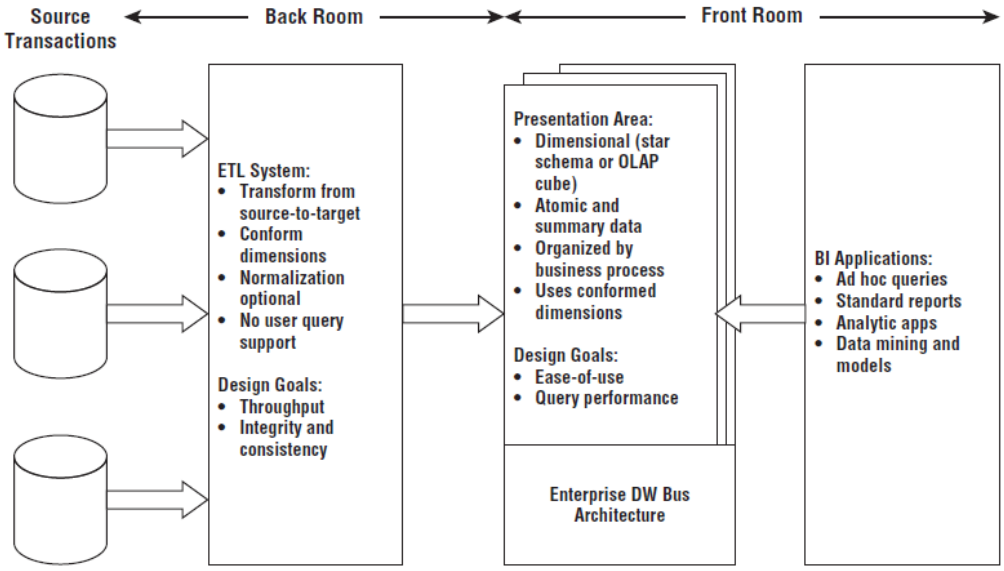

Figura 2. Elementos principais da arquitetura de DW/BI. Retirado de Kimball e Ross (2013).

1. Os **sistemas operacionais** (*Source Transactions*) armazenam as transações operacionais da empresa, por conseguinte, as suas principais prioridades são a disponibilidade e o desempenho no processamento dos dados.
2. O **sistema ETL** (*ETL System*) que faz a ligação entre os sistemas operacionais e a área de apresentação. A fase de extração é o primeiro passo no processo de obtenção de dados e consiste em ler e entender os dados provenientes das fontes e posterior cópia dos que são necessários para manipulação, sendo assim exportados para o DW. Após extração dos dados, existem várias transformações de limpeza que são necessárias (tais como, tratar os elementos ausentes/inválidos, analisar formatos, corrigir erros ortográficos, entre outros), para além disso, destaca-se também a combinação e integração de dados provenientes de múltiplas fontes, de modo a melhorar a informação disponibilizada, assim como a eliminação da duplicação de dados. A última fase do ETL consiste no carregamento e estruturação física dos dados em modelos dimensionais que definem a estrutura do DW.
3. A **área de apresentação** (*Presentation Area*) que corresponde ao local onde os dados são organizados, armazenados e disponibilizados para consultas e *reports*. A área de apresentação deve conter dados atómicos, isto é, detalhes, para possibilitar respostas a consultas *ad hoc* dos utilizadores.
4. As **aplicações de *Business Intelligence*** (*BI Applications*) que se referem ao conjunto de recursos analíticos fornecidos aos utilizadores para a tomada de decisão. Todas estas aplicações solicitam os dados na área de apresentação do DW. Estas aplicações podem incluir *dashboards*,

ferramentas e algoritmos de consultas *ad hoc* dos dados e ferramentas e algoritmos de *data mining*.

Porém existem outras propostas de arquitetura, pelo que as componentes apresentadas anteriormente podem sofrer diferentes nomeações, assim como também podem ser acrescentadas outras componentes. Por exemplo Vaisman & Zimányi (2014), referem-se às componentes da arquitetura por camadas. A **camada *Back-End***, engloba a área de estágio e o sistema ETL. A **camada *Data Warehouse*** considera o DW e os *Data Marts*, que são mais fáceis de implementar do que o DW, contudo, a arquitetura de Kimball e Ross (2013) não menciona os *Data Marts*. Apesar de não os referirem na sua arquitetura, Kimball e Ross (2013) descrevem *Data Marts* como um subconjunto do DW. O DW é a soma de todos os *Data Marts*, cada um representando um processo de negócio ou um subconjunto de dados da organização. A **camada OLAP** (*On-Line Analytical Processing*) é constituída pelo servidor OLAP, que fornece uma visão multidimensional dos dados, contudo, também não é referida na primeira arquitetura, anteriormente apresentada. A última, é a **camada *Front-End*** e basicamente corresponde às aplicações de BI. É possível constatar que em algumas situações, as fontes de dados acedem diretamente à camada *Front-End*, pelo que deixa de existir o servidor OLAP e o DW. Para além disso, existem outros casos em que a camada *Front-End* acede diretamente ao DW, não existindo neste caso somente o servidor OLAP, tal como acontece na arquitetura de Kimball e Ross (2013).

Normalmente, é possível identificar sistemas operacionais e sistemas analíticos dentro de uma organização. Os sistemas operacionais que suportam OLTP (*On-line Transaction Processing*), tratam das operações diárias da organização, refletindo os dados mais atualizados, enquanto que os DWs, que suportam OLAP (*On-Line Analytical Processing*), vão mantendo um histórico para avaliar com precisão o desempenho dos processos operacionais da organização ao longo do tempo (Kimball & Ross, 2013).

Devido a problemas de escalabilidade, os dados transacionais, armazenados em sistemas OLTP, são frequentemente modelados de tal forma que nem todos os dados dos sistemas de origem são utilizados na análise (Krishnan, 2013). Os dados armazenados nestes sistemas OLTP, tradicionalmente, são carregados no DW e periodicamente exportados para sistemas OLAP, através do processo ETL.

Os modelos dimensionais implementados em bases de dados multidimensionais são referidos como cubos OLAP. Sempre que os dados são carregados num cubo OLAP, estes são armazenados e indexados usando formatos e técnicas definidas para dados dimensionais. Consequentemente, os cubos OLAP oferecem um desempenho de consulta superior devido aos cálculos prévios, às estratégias de indexação e a outras otimizações (Kimball & Ross, 2013). O OLAP abarca um conjunto de ferramentas

e algoritmos que permitem a consulta eficiente de bases de dados multidimensionais contendo grandes quantidades de dados, os DWs (Kimball & Ross, 2013; Vaisman & Zimányi, 2012).

No OLAP, os dados são organizados como um conjunto de dimensões e tabelas de factos. No modelo multidimensional, os dados podem ser entendidos como um cubo de dados, onde cada célula contém medidas (Kimball & Ross, 2013; Vaisman & Zimányi, 2012). As dimensões do OLAP são organizadas em hierarquias o que favorece o processo de agregação de dados (Cabibbo & Torlone, 1998).

De forma a sintetizar as diferenças existentes entre os sistemas operacionais e os analíticos é apresentada a Tabela 1.

Tabela 1. Sistemas Operacionais vs Sistemas Analíticos. Baseado em Sá (2010) e M. Santos e Ramos (2006).

Sistemas Operacionais	Sistemas Analíticos (DW)
Destina-se a administradores de sistemas e aplicações informáticas operacionais;	Destina-se a analistas de negócio, gestores e executivos;
Operações diárias, registo de transações diárias (OLTP);	Suporte à tomada de decisão, transações analíticas (OLAP);
Orientado às aplicações;	Orientado aos assuntos;
Dados correntes, atualizados, atómicos, relacionais (normalizados) e isolados;	Dados históricos, sumarizados, multidimensionais e integrados;
Acesso por transações simples predefinidas, repetitivas e rotineiras;	Acesso por questões *ad hoc*, *queries* complexas;
Otimizado para acesso a poucos registos de cada vez (1 a 3 tabelas envolvidas);	Otimizado para acesso a muitos registos de cada vez (várias tabelas envolvidas);
Acessos de leitura e escrita;	Acessos só de leitura;
Dados atualizados em tempo real;	Carregamento periódico de mais dados, permite analisar o desempenho do negócio;
Estrutura otimizada para atualizações.	Estrutura otimizada para processamento de questões.

Um DW geralmente possui um esquema multidimensional, sendo constituído por dois tipos de componentes: as dimensões e os factos. As **dimensões** são tabelas que armazenam registos descritivos referentes aos factos, ou seja, na tabela dimensão encontram-se registos ocorridos na tabela de factos, sendo utilizadas para analisar as medidas dos factos através de operações de agregação. A tabela de **factos** é a entidade que interliga, por chaves estrageiras, as várias tabelas de dimensão, o facto representa um elemento, ou uma transação, ou um evento associado ao tema da modelação em questão. A tabela de factos poderá incluir medidas numéricas que podem ser aditivas, semiaditivas e não aditivas (Inuwa & N. D. Oye, 2015; Schneider, 2008).

A modelação multidimensional de um DW é amplamente aceite como a técnica de eleição para a estruturação de dados analíticos, uma vez que fornece aos utilizadores dados compreensíveis e proporciona um rápido desempenho nas consultas, tornando o processo de criação de bases de dados

mais simples. Este esquema multidimensional pode ser: um esquema **estrela**, um esquema em **floco de neve** ou um esquema em **constelação**. (Inuwa & N. D. Oye, 2015; Kimball & Caserta, 2011; Kimball & Ross, 2013).

Segundo Tria et al. (2014) os DWs tradicionais são projetados de acordo com duas abordagens diferentes:

- A abordagem *data-driven*, orientada aos dados, que consiste numa reengenharia de fontes de dados, onde é criado um esquema multidimensional com base nas fontes disponíveis, no entanto, que pode não satisfazer os requisitos dos decisores de negócio.
- Opostamente encontra-se a abordagem *requirement-driven*, isto é, orientada aos requisitos, nesta define-se esquemas multidimensionais usando metas de negócio resultantes das necessidades dos decisores, sendo que as fontes de dados são apenas consideradas mais tarde, quando a fase ETL é abordada. No entanto, esta abordagem poderá trazer alguns problemas, uma vez que informação importante poderá não estar a ser utilizada, ou seja, os dados necessários podem não se encontrar nas fontes disponíveis.

Contudo, ambas as abordagens têm as suas vantagens, pelo que conjugadas originarão uma metodologia **híbrida** que tenha em conta as melhores caraterísticas de cada uma. Por outro lado, estas abordagens híbridas poderão tornar-se mais complexas pelo facto de ser necessário integrar a abordagem orientada aos dados e a abordagem orientada aos requisitos, ou seja, integrar os dados e os requisitos do negócio.

Assim, a disponibilidade de enormes quantidades de dados provenientes de diferentes domínios exige uma mudança na forma como as práticas de DW e BI se realizam. A abordagem tradicional, na qual os dados produzidos no dia a dia de uma organização são armazenados num repositório comum para posterior análise, requer reavaliação, pois é necessário responder eficientemente à manipulação de dados em grande escala (Vaisman & Zimányi, 2012).

2.2 *Big Data*

Uma das razões fundamentais para a notoriedade do fenómeno de *Big Data* é a atual dimensão em que a informação pode ser gerada e disponibilizada, quer isto dizer que os dados estão a ser gerados a taxas cada vez mais crescentes (De Mauro, Greco, & Grimaldi, 2014). Grande parte destes dados resultam do aumento nas evoluções das tecnologias de *Cloud Computing*, da *Internet* e da disponibilidade de dispositivos localizados na rede, nomeadamente, sensores, *smartphones* e *tablets*.

Todos estes dados criam novas oportunidades para extrair valor nas diversas áreas onde estão inseridos, tais como saúde, finanças, entre tantas outras (Dumbill, 2013; Villars, Olofson, & Eastwood, 2011). Esses abundantes fluxos de dados contêm informação heterogénea, gerada a taxas diferentes e com formatos muito distintos (Cassavia et al., 2014).

Existem várias ferramentas que auxiliam na gestão desse tipo de dados e com isso facilitam e aceleram os processos de tomada de decisão. No entanto, as ferramentas tradicionais acabam por perder capacidades com dados semiestruturados ou não estruturados, porém estes também precisam de ser processados, visto que as organizações dependem cada vez mais do conhecimento extraído dos mesmos (Kubina, Varmus, & Kubinova, 2015; Oussous, Benjelloun, Ait Lahcen, & Belfkih, 2018). Uma vez que as técnicas e plataformas tradicionais são menos eficientes nesses contextos, já que apresentam elevados tempos de resposta, baixa escalabilidade, desempenho e precisão, vários tipos de tecnologias têm vindo a ser desenvolvidas até então para enfrentar os desafios decorrentes (Oussous et al., 2018).

Consequentemente surgiu o conceito de *Big Data*, conceito dúbio a partir de onde é difícil de quantificar o nível em que os dados se tornam "grandes", pelo que não existe uma única definição aceite por todos, assim sendo, alguns autores acabam por dar as suas próprias perspetivas sobre o termo (Ward & Barker, 2013).

Nos pontos seguintes é apresentado o conceito de *Big Data*, bem como as suas caraterísticas e desafios.

2.2.1 Conceito

A origem do conceito de *Big Data* é uma incógnita, todavia a sua definição evoluí rapidamente o que aumenta a incerteza (C. Costa & Santos, 2017a). Conforme analisado por Costa e Santos (2017a) principalmente a partir de 2011, houve um aumento do interesse pelo *Big Data*, sendo até destacado como fundamental para o crescimento da produtividade, inovação e relacionamento com clientes, nas áreas da saúde, setor público, comércio, manufatura e cidades modernas.

De acordo com os seus estudos acerca das várias perspetivas sobre o conceito, Ward e Barker (2013) concluem que cada uma faz referência a pelo menos a um destes aspetos: volume, complexidade e tecnologias. Assim, os autores acabam por dar a sua própria definição para o termo, mencionando que *Big Data* diz respeito ao armazenamento e análise de um conjunto de dados amplo e complexo, usando uma série de técnicas.

Para Krishnan (2013), *Big Data* pode ser definido como o volume de dados disponíveis em distintos graus de complexidade, gerados em diferentes velocidades pelo que não é possível processá-los em tecnologias tradicionais.

Normalmente, *Big Data* inclui grandes quantidades de dados não estruturados que necessitam de análises em tempo real, como tal, cada vez mais organizações estão a enfrentar os desafios de *Big Data* (Chen, Mao, & Liu, 2014; Zikopoulos & Eaton, 2011).

O conceito de *Big Data* pode aplicar-se a dados que excedem a capacidade de processamento das bases de dados convencionais, portanto, volumes de dados que se movem muito rapidamente, não se encaixam nas restrições das arquiteturas das bases de dados tradicionais (Dumbill, 2013). Em suma, para obter valor a partir desses conjuntos de dados de elevada extensão, são necessárias alternativas para o seu processamento, armazenamento e análise, tendo em conta a complexidade das suas operações (Cassavia et al., 2014; De Mauro et al., 2014; Dumbill, 2013).

Tal como é referido por Ward e Barker (2013) "Big" tem impacto e implica complexidade e desafio, contudo, se as organizações encontrarem uma forma de extrair valor comercial desses dados, provavelmente alcançarão vantagens competitivas consideráveis (Villars et al., 2011).

Big Data é frequentemente visto como uma palavra de referência para análises de dados mais inteligentes e perspicazes, mas pode-se alegar que é mais do que isso, uma vez que está associado a novas fontes de dados complexas e granulares (Davenport, Barth, & Bean, 2012). Possibilitará, inclusive, novas maneiras de interagir com os clientes ou até o desenvolvimento de novos produtos, serviços e estratégias, bem como formas de responder mais rapidamente às mudanças nos negócios, levando ao aumento da lucratividade das organizações (Davenport et al., 2012; M. Y. Santos, Oliveira e Sá, et al., 2017).

Além disso, para Chen et al. (2014), *Big Data* traz também novas oportunidades, desafios e a possibilidade de conhecer com maior detalhe os padrões e tendências, o que permite uma gestão mais eficiente dos dados.

Depois da análise realizada anteriormente, é possível concluir que *Big Data* se trata de uma temática complexa que envolve várias perspetivas, mas que pode e deve ser introduzida em vários setores e áreas. Os dados das organizações são imensos e com os avanços tecnológicos que têm ocorrido, surge a necessidade de os gerir de uma forma mais eficiente, uma vez que estes podem provir de várias fontes distintas e apresentarem caraterísticas bastante diferentes e complexas. Portanto, a adoção de *Big Data* permitirá às organizações tirar o melhor proveito das tecnologias emergentes e com isso aumentar a sua vantagem competitiva.

2.2.2 Caraterísticas

As principais caraterísticas de *Big Data* foram identificadas pela primeira vez por Laney (2001) e estão associadas ao modelo dos *3 V's*: volume, variedade e velocidade, tal como se encontra representado na Figura 3.

Gartner (2017) define *Big Data*, como "ativos da informação de elevado volume, elevada velocidade e/ou elevada variedade que exigem formas inovadoras e económicas de processamento e que permitem uma melhor perceção, tomada de decisão e automatização de processos".

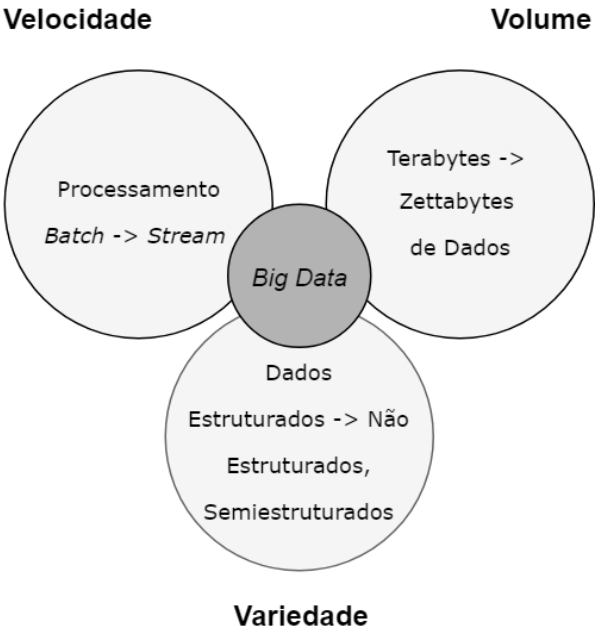

Figura 3. Modelo dos 3 V's. Adaptado de Zikopoulos e Eaton (2011).

O **volume** é a caraterística mais associada ao *Big Data* e indica a amplitude dos dados, isto é, refere-se às quantidades em massa de dados continuamente gerados e armazenados, os quais estão a ser aproveitados para melhorar a tomada de decisão das organizações. Normalmente, o tamanho de *Big Data* é associado entre *terabytes* e *pentabytes* (Gandomi & Haider, 2015; Michael Schroeck, Shockley, Smart, Romero-Morales, & Tufano, 2012). No entanto, é considerado que o tamanho dos dados é relativo e varia de acordo com a periodicidade e o tipo de dados (Gandomi & Haider, 2015; Krishnan, 2013). Para os autores não faz sentido definir um limite específico para o volume de *Big Data*, uma vez que diferentes tipos de dados exigem tecnologias díspares, por exemplo dados tabulares *versus* dados de vídeo. O volume dos dados continua a crescer extraordinariamente, pelo que o que se considera hoje como grande poderá ser ainda maior amanhã, quer isto dizer que possivelmente no futuro a capacidade de armazenamento irá aumentar (Michael Schroeck et al., 2012). A principal causa para o crescente

volume é o facto de que atualmente se armazena todas as interações com a maioria dos serviços disponíveis no mundo (Zikopoulos & Eaton, 2011).

A **variedade** refere-se à estrutura heterogénea dos conjuntos de dados, isto é, inúmeras fontes de dados, sendo que as evoluções tecnológicas permitem às organizações gerar e utilizar vários tipos de dados (Gandomi & Haider, 2015; Michael Schroeck et al., 2012). O *Big Data* pode ser classificado como **estruturado**, que diz respeito, por exemplo, a dados tabulares encontrados em bases de dados relacionais; **semiestruturado**, por exemplo, *logs* de servidores *web*, documentos XML (*eXtensible Markup Language*: linguagem textual para troca de dados na *web*); e **não estruturado**, tais como, textos, imagens, áudio, vídeo, dados de sensores e *tweets* (Gandomi & Haider, 2015; Krishnan, 2013; Zikopoulos & Eaton, 2011). A variedade presente nos dados representa diferentes formatos e inconsistências na semântica dos dados, portanto esta variedade afeta diretamente a integridade dos dados (N. Khan et al., 2014; Laney, 2001). Estas inconsistências tornam necessária a junção e integração dos dados, contudo, isto pode até se revelar um desafio, tendo em conta a possibilidade dos dados não estarem documentados (Liu, Yang, & Zhang, 2013).

A última caraterística do modelo dos *3 V's* é a **velocidade** e refere-se à taxa na qual os dados são gerados, processados, analisados e utilizados para o apoio à decisão (Gandomi & Haider, 2015; Kubina et al., 2015). A difusão dos dispositivos digitais, levou a uma progressiva taxa de criação de dados que está a impulsionar a necessidade crescente de processamento e análise em tempo real (Gandomi & Haider, 2015; Michael Schroeck et al., 2012; Oussous et al., 2018). Os dados podem ser gerados em taxas diferentes, variando de ***batch*** a ***streaming***. Os fluxos de dados contínuos só serão vantajosos se o tempo entre a recolha e o seu processamento (latência) for relativamente curto, pelo que esses fluxos podem criar vantagens competitivas em contextos onde a identificação de tendências deve ocorrer em curtos períodos de tempo (Krishnan, 2013; Michael Schroeck et al., 2012; Zikopoulos & Eaton, 2011). Além da elevada velocidade, os dados geralmente não são estáveis e apresentam uma distribuição variável ao longo do tempo (Oussous et al., 2018). Para Zikopoulos e Eaton (2011) o volume de dados é uma consequência direta da elevada velocidade com que os dados são produzidos. Ao invés de restringir a ideia de velocidade à taxa na qual os dados são recolhidos, armazenados e recuperados do sistema de armazenamento, Zikopoulos e Eaton (2011) sugerem que se aplique essa definição aos dados em movimento, ou seja, à velocidade na qual os dados estão a fluir.

Além de todas as caraterísticas de *Big Data* anteriormente exploradas, Krishnan (2013) identificou mais três: ambiguidade, viscosidade e viralidade, tal como ilustrado na Figura 4.

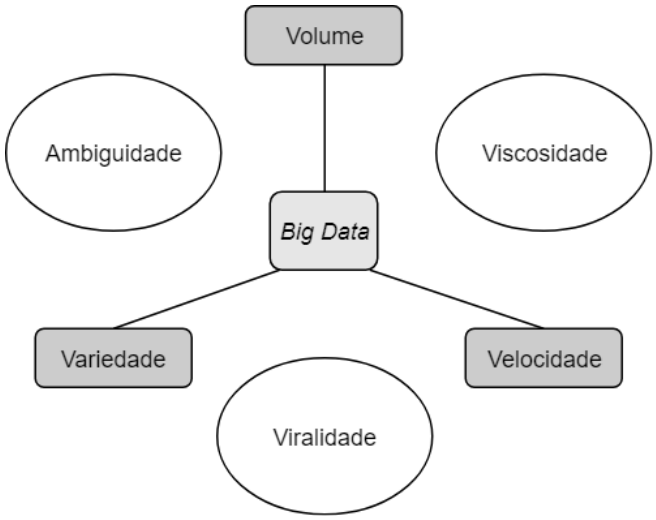

Figura 4. Modelo dos 3 V's com características adicionais. Adaptado de Krishnan (2013).

A **ambiguidade** é associada à falta de metadados adequados e resulta da combinação do **volume** e **variedade**. Passando a exemplificar, numa representação dos dados, o "M" e o "F", podem significar coisas distintas, pode dizer respeito ao sexo "Male" e "Female", respetivamente, ou, por outro lado, podem representar os dias da semana em inglês, "Monday" e "Friday" (Krishnan, 2013).

Viscosidade mede a resistência para "fluir" no volume dos dados. Ocorre quando o **volume** e a **velocidade** causam arraste nos fluxos de dados. A resistência pode-se manifestar nos fluxos de dados, nas regras de negócio ou até mesmo ser uma limitação da tecnologia. Por exemplo, a monitorização de redes sociais, pelo facto de diversas empresas não entenderem como os dados daí provenientes afetam o seu negócio, acabando por resistir à sua utilização (Krishnan, 2013).

Para terminar, identifica-se ainda a **viralidade** que mede o tempo de propagação dos dados entre os nós de uma rede. Portanto, abrange a **velocidade** e a **variedade**. Por exemplo, os *tweets* que são partilhados a partir de um *tweet* principal são uma forma de seguir um tópico ou uma tendência (Krishnan, 2013).

Muitos outros autores estendem o modelo dos *3 V's* e como resultado outras caraterísticas (outros *V's*) de *Big Data* surgiram, são elas: o valor e a veracidade.

O **valor** representa os resultados do processamento do grande volume, velocidade e variedade para análise de *Big Data* (Chandarana & Vijayalakshmi, 2014). Geralmente é caraterizado por uma "baixa densidade de valor", quer isto dizer que os dados, na sua forma original, têm um baixo valor em relação ao volume, no entanto, o valor elevado pode ser obtido com uma análise adequada de tais dados (Chandarana & Vijayalakshmi, 2014; Gandomi & Haider, 2015; Oussous et al., 2018). O valor pode ser alcançado através da integração de diferentes tipos de dados para extrair conhecimento oculto, melhorar

os negócios e obter vantagens competitivas (Chandarana & Vijayalakshmi, 2014). Para Dijcks (2012) o desafio está em identificar o que é valioso e depois transformar e extrair esses dados para análise.

De outro lado, tem-se a **veracidade** que se refere ao nível de confiabilidade associado a certos tipos de dados, pelo que chama atenção para possíveis dados imprecisos, uma vez que as análises são baseadas em conjuntos com vários graus de precisão, autenticidade e confiabilidade (Chandarana & Vijayalakshmi, 2014; Michael Schroeck et al., 2012). Não se deve despender tempo na limpeza dos dados antes de utilizá-los, pois embora a qualidade dos dados seja um importante desafio e requisito de *Big Data*, até mesmo os melhores métodos de limpeza de dados não conseguem remover a imprevisibilidade inerente a alguns deles (Michael Schroeck et al., 2012). A falta de confiabilidade em algumas fontes de dados, por exemplo, os sentimentos de cliente retirados das redes sociais são incertos por natureza, já que envolvem o julgamento humano, contudo, contêm informação valiosa (Gandomi & Haider, 2015). Assim, a necessidade de lidar com dados incertos requer mecanismos, ferramentas e análises especificamente desenvolvidos para a gestão e o *mining* desses dados (Chandarana & Vijayalakshmi, 2014; Gandomi & Haider, 2015).

M. A. Khan, Uddin e Gupta (2014) consideram mais dois *V's*: a validade e a volatilidade.

A **validade** é similar ao conceito de veracidade e representa a exatidão e a precisão dos dados, de acordo com a utilização pretendida. Os dados têm que ser devidamente compreendidos de forma a perceber se fazem ou não sentido no contexto em que se inserem. Estes dados podem ser autênticos, isto é, podem não ter problemas de veracidade, mas ao mesmo tempo são inválidos se não forem devidamente compreendidos. Para além disso, o mesmo conjunto de dados pode ser válido para uma aplicação e inválido para outra aplicação (M. A. Khan et al., 2014).

A **volatilidade** está relacionada com a política de retenção de dados de uma organização, referindo-se ao tempo no qual os dados armazenados são válidos e devem ser mantidos, até que expiram e podem ser destruídos, pois deixam de ser relevantes. É necessário ter em consideração o armazenamento disponível e a segurança dos dados, pois o período de retenção pode ser excedido (M. A. Khan et al., 2014).

Surgiram ainda outras características de *Big Data*, não tão conhecidas, são elas a variabilidade e a complexidade.

A **variabilidade** refere-se às variações das taxas nas quais os dados fluem, de acordo com diferentes velocidades, que não são consistentes, tendo altos e baixos (Gandomi & Haider, 2015).

A **complexidade** está relacionada à gestão integrada de várias fontes de dados, o que se torna um desafio, nomeadamente, a necessidade de conectar, combinar, limpar e transformar os dados provenientes dessas fontes (Gandomi & Haider, 2015; Oussous et al., 2018; Suthaharan, 2014).

A Figura 5 apresenta um resumo de todas as características de *Big Data* identificadas anteriormente.

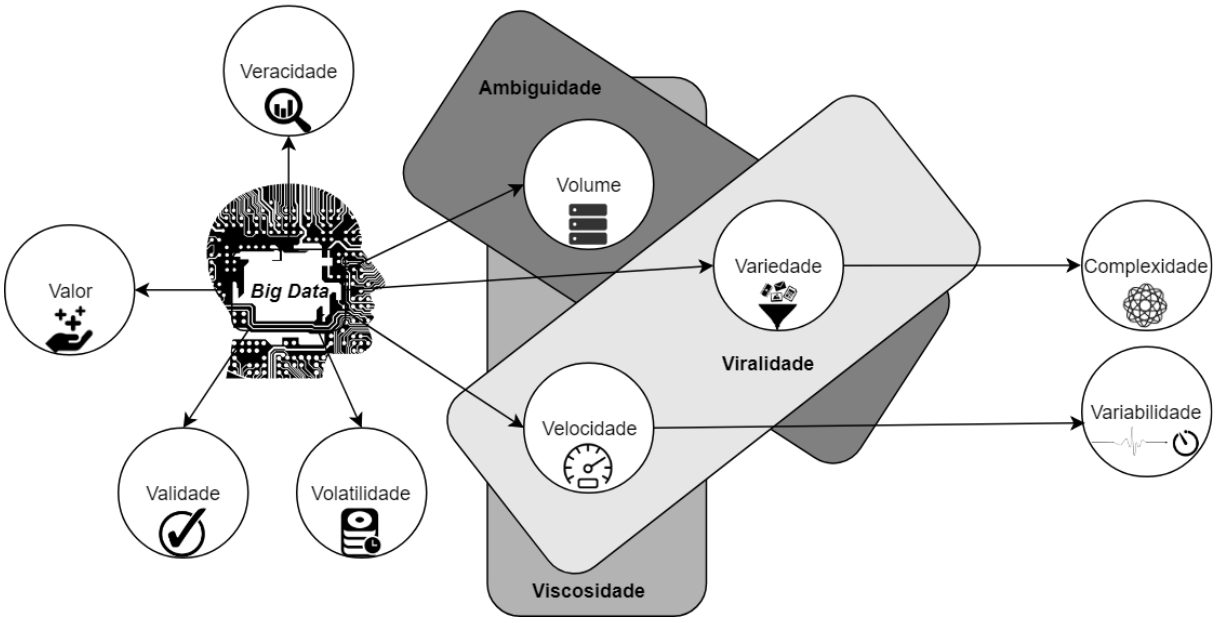

Figura 5. Principais características de Big Data identificadas na literatura. Baseado em Costa e Santos (2017a).

2.2.3 Oportunidades, Problemas e Desafios

Desde o aparecimento da *Internet* que se tem mudado constantemente a forma de comunicação, ou seja, anteriormente era baseada apenas em texto e atualmente consiste na troca de informação mais rica que inclui imagens, vídeos, mapas interativos, entre outros (Michael & Miller, 2013).

Em diversas áreas os dados estão a ser recolhidos e armazenados numa escala sem precedentes, sendo que as decisões que anteriormente eram baseadas em suposições podem ser agora feitas com base na análise de dados (Labrinidis et al., 2012).

O potencial valor de *Big Data* é amplamente associado às vantagens nos processos de tomada de decisão. Para possibilitar uma tomada de decisão baseada em evidências, as organizações necessitam de processos eficientes para transformar grandes volumes de dados dinâmicos em conhecimento significativo. Tirar proveito de conhecimentos valiosos provenientes de grandes quantidades de dados é a competição básica para as empresas de hoje (Gandomi & Haider, 2015).

A utilização de *Big Data* introduz uma nova onda de crescimento produtivo nas organizações, que obtêm muitas vantagens, tais como orientações estratégicas mais informadas, que implicam o

aumento da eficiência operacional, melhor atendimento aos clientes, bem como a identificação e desenvolvimento de novos produtos, novos clientes e mercados (N. Khan et al., 2014; Manyika & Chui, 2011; Philip Chen & Zhang, 2014).

O valor de *Big Data* para as organizações é impulsionado pela capacidade de analisar os dados, de forma a desenvolver informação relevante (Kaisler, Armour, Espinosa, & Money, 2013). Kaisler et al. (2013) sugerem cinco formas de criar valor a partir de *Big Data*:

1. Criar transparência, através da disponibilização para análise de grandes quantidades de dados;
2. Apoiar a análise experimental que pode testar decisões ou abordagens;
3. Auxiliar na segmentação do mercado, com base na informação do cliente;
4. Suportar análises e decisões em tempo real;
5. E ajudar na inovação de produtos tendo em conta as necessidades dos clientes e as tecnologias emergentes.

Para além das oportunidades que surgem a partir da adoção de *Big Data*, com ela também advêm problemas e desafios, pelo que as organizações devem encontrar formas de ultrapassá-los.

Para a exposição dos vários desafios associados ao *Big Data* irá ser seguida a caraterização utilizada por Costa e Santos (2017a), que os divide em várias categorias, são elas: "Desafios Gerais de *Big Data*", "Desafios no Ciclo de Vida de *Big Data*", "*Big Data* em Ambientes Seguros, Privados e Monitorizados" e "Mudança Organizacional".

Desafios Gerais de *Big Data*

Os estímulos gerais de *Big Data* incluem desafios relacionados com a falta de rigor e consenso na sua definição, nos seus modelos e arquiteturas (C. Costa & Santos, 2017a).

- **Definição de *Big Data***: Um dos principais problemas concentra-se na necessidade de uma definição rigorosa e holística, isto é, um modelo estrutural e uma descrição formal de *Big Data* e ainda um sistema teórico relativo ao *Data Science*, pelo que os modelos existentes não são rigorosos nem verificados corretamente. Atualmente, as discussões sobre o termo causam muita especulação que, maioritariamente, tratam-se apenas de especulações comerciais ao invés de investigação científica (Chen et al., 2014).
- **Padronização de *Big Data***: A falta de *standards* em *Big Data* é outro dos seus problemas. Há necessidade de se desenvolver um sistema de avaliação da qualidade dos dados e um padrão de avaliação ou *benchmark* para comparar a eficiência das soluções, através de métodos rigorosos, sendo assim possível melhorar o processamento e a análise de dados. Atualmente,

apenas se consegue avaliar o desempenho após o sistema ser implementado e implantado, o que não permite comparar horizontalmente as vantagens e desvantagens entre as várias soluções alternativas (Chen et al., 2014). Tal como mencionado por Baru, Bhandarkar, Nambiar, Pöss e Rabl (2013), a falta de padrões também se deve à constante evolução das tecnologias de *Big Data*.

- **Quantidade *versus* Qualidade dos Dados**: Os utilizadores têm a ideia de que mais dados são sinónimo de qualidade, pelo que estão sempre a adquirir mais, pois acreditam que assim conseguirão ter resposta para qualquer fenómeno em estudo. Ter grandes quantidades de dados armazenados, mas com qualidade insuficiente, normalmente, só permite tirar conclusões irrelevantes, mais vale estar concentrado na qualidade, não ter muitos dados disponíveis, mas os que se tem serem de qualidade, isso possibilitará tirar conclusões mais precisas e valiosas (Kaisler et al., 2013). Ao lidar com grandes quantidades de dados é um desafio decidir quais fontes de dados são mais apropriadas, quais dados têm qualidade suficiente, quais são relevantes, quais se deve efetivamente selecionar e ainda como se deve estimar o valor que irão acrescentar (Chandarana & Vijayalakshmi, 2014). Contudo, tudo depende do contexto, mas os vários autores consideram que não se deve assumir que mais dados é a melhor abordagem (C. Costa & Santos, 2017a).

Desafios no Ciclo de Vida de *Big Data*

Estes desafios estão relacionados às dificuldades técnicas de *Big Data*, tais como, a recolha, integração, limpeza, transformação, armazenamento, processamento, análise ou governança de dados (C. Costa & Santos, 2017a).

- **Armazenamento e Processamento de Dados**: As tecnologias tradicionais de recolha e armazenamento não conseguem processar os grandes volumes de dados provenientes de diferentes fontes (Chandarana & Vijayalakshmi, 2014; Chen et al., 2014; Kaisler et al., 2013; Philip Chen & Zhang, 2014). O grande desafio é repensar os sistemas de armazenamento, arquiteturas, mecanismos e redes, a fim de serem capazes de processar e analisar cada vez mais conjuntos de dados complexos e em expansão (Acharjya & Ahmed, 2016; Chen et al., 2014; C. Costa & Santos, 2017a). Assim, a acessibilidade, disponibilidade e a transmissão dos dados, para além da escalabilidade de armazenamento são as principais prioridades do processo de descoberta de conhecimento (Liu et al., 2013; Philip Chen & Zhang, 2014). Uma vez que processar grandes quantidades de dados implica despender tempo significativo, são necessários novos mecanismos que consigam reduzir consideravelmente o tempo de processamento,

portanto, esses dados requerem um processamento paralelo que garanta que estejam disponíveis a tempo para suportar as decisões (Katal et al., 2013).

- **Assegurar a Qualidade dos Dados**: Garantir a qualidade dos dados e agregar valor através da sua preparação torna-se também um desafio (C. Costa & Santos, 2017a; Philip Chen & Zhang, 2014). A partir das diferentes fontes de dados podem surgir distintos problemas de qualidade dos dados, pelo que se deve definir estratégias e métodos para os tratar, no sentido de se detetarem e corrigirem automaticamente alguns dos seus problemas (Chen et al., 2014; N. Khan et al., 2014).
- **Redundância e Heterogeneidade dos Dados**: A heterogeneidade e a redundância podem trazer problemas na qualidade dos dados, assim como desafios que podem dificultar a integração de um grande número de fontes de dados (Chen et al., 2014; C. Costa & Santos, 2017a). Chen et al. (2014) enumeram alguns efeitos negativos da redundância, nomeadamente, os defeitos que podem causar nos sistemas de armazenamento, por exemplo, a inconsistência e a redução da fiabilidade dos dados. As técnicas tradicionais estão adaptadas a lidar com dados homogéneos, assim sendo, a heterogeneidade proveniente de múltiplas fontes torna-se também um desafio (Labrinidis et al., 2012). Os dados atualmente produzidos encontram-se em diferentes formatos, sem qualquer tipo de estrutura (Katal et al., 2013). A natureza não estruturada das fontes de dados representa vários desafios em relação às transformações a realizar para suportar as tarefas analíticas (C. Costa & Santos, 2017a). Por exemplo, os dados recolhidos em *stream*, podem alterar o seu formato e mudar a necessidade de serem analisados ou a forma como são analisados (Kaisler et al., 2013).
- **Visualização de *Big Data***: A visualização de *Big Data* é outro desafio e também uma necessidade (Fang et al., 2015). As tecnologias emergentes estão a evoluir no sentido de lidar com o desafio de manipular o *Big Data* interactivamente e para reduzir o custo da conceção de visualizações, permitindo que se tornem parte integrante do processo científico (Fang et al., 2015; Krishnan, 2013). O principal objetivo da visualização de dados é representar o conhecimento de forma mais intuitiva, usando vários tipos de gráficos distintos (Keim, Panse, Sips, & North, 2004; Philip Chen & Zhang, 2014; Simoff, Böhlen, & Mazeika, 2008). Tanto a aparência como as funcionalidades são fundamentais para transmitir o conhecimento que se encontra implícito nos conjuntos de dados complexos e de grande escala, assim como transmitir a informação mais facilmente (Philip Chen & Zhang, 2014). São necessárias visualizações que permitam extrair valor de *Big Data*, com a capacidade de representar dados provenientes de diversas fontes, lidar

com vários tipos de dados e ser de fácil utilização para satisfazer os utilizadores, aquando da sua utilização (C. Costa & Santos, 2017a; Russom, 2011). Assim, oss resultados analíticos poderão, efetivamente, ser usados pelos utilizadores quando forem exibidos de forma amigável e intuitiva (Chen et al., 2014).

- **Governança de *Big Data***: A Governança de Dados implica o controlo e a autoridade sobre as regras relacionadas às grandes quantidades de dados provenientes de diferentes fontes, com diferentes propósitos e restrições, bem como a transparência e a responsabilidade dos indivíduos e dos sistemas de informação para atingir os objetivos de negócio (Hashem et al., 2015). Tal como Costa e Santos (2017a) indicam, sem as ferramentas de governança adequadas, gerir esse ambiente heterogéneo, definir as políticas de acesso e garantir a rastreabilidade pode ser uma tarefa quase impossível.

Big Data em Ambientes Seguros, Privados e Monitorizados

Nos dias de hoje, uma das tarefas que mais preocupa as organizações é manter os dados seguros e privados (Chen et al., 2014). Os utilizadores necessitam que a sua informação não escape para o público indevido (Chandarana & Vijayalakshmi, 2014). Nesta categoria estão inseridos os desafios relacionados à privacidade e segurança dos dados, assim como os riscos, as políticas e a propriedade.

- **Privacidade e Riscos**: Devido às características de *Big Data*, surgem mais riscos de segurança pelo que os métodos tradicionais de proteção de dados devem ser repensados (C. Costa & Santos, 2017a). Os dados recolhidos possuem informação acerca dos utilizadores, informação essa que é importante para se conhecer tendências e opiniões, pelo que a disponibilidade inadequada de dados permite até, por vezes, inferir a identidade de uma pessoa (Acharjya & Ahmed, 2016; Kaisler et al., 2013; Wigan & Clarke, 2013). Assim sendo, mesmo que não existam identificadores pessoais, quando os dados são suficientemente ricos, é possível deduzir certas caraterísticas dos utilizadores (Wigan & Clarke, 2013). Para Chen et al. (2014) a privacidade de *Big Data* inclui a proteção da privacidade pessoal durante a aquisição de dados dos utilizadores (por exemplo, interesses e hábitos) e durante o armazenamento, transmissão e utilização dos mesmos. A qualidade dos dados em *Big Data* também influencia a sua utilização apropriada e segura (Chen et al., 2014). A segurança de *Big Data* pode ser aprimorada utilizando técnicas de autenticação, autorização e criptografia, contudo, a criptografia, é altamente influenciada pela escala e pela variedade dos dados (Acharjya & Ahmed, 2016; Chen et al., 2014).

- **Políticas**: Facilmente é possível copiar, integrar e utilizar recorrentemente dados (C. Costa & Santos, 2017a). Uma vez que atualmente há uma quantidade significativa de dados sensíveis sobre os indivíduos, as políticas relativas à segurança dos dados são significativamente relevantes (Manyika & Chui, 2011). A propriedade intelectual, a propriedade de dados e a responsabilidade em relação a dados imprecisos precisam de especial atenção para a formulação de políticas adequadas (Manyika & Chui, 2011). É necessário analisar a adequação das leis e regulamentos atuais para proteger adequadamente os dados sobre os indivíduos (Hashem et al., 2015).
- **Propriedade dos Dados**: Outro tópico relevante é a propriedade dos dados, por exemplo, o conceito é frequentemente abordado em relação às redes sociais, pois apesar dos servidores destas redes armazenarem os dados dos seus utilizadores eles não são os seus proprietários (Chandarana & Vijayalakshmi, 2014). Existem certas legislações que permitem que as organizações não excluam permanentemente os dados dos utilizadores, mesmo que estes o solicitem, pelo que muitas das vezes elas supõem que detêm os direitos sobre esses dados (Wigan & Clarke, 2013). Para além disso, atualmente, algumas organizações discutem formas de partilhar ou vender dados sem perderem o controlo sobre os mesmos (Jagadish et al., 2014).

Mudança Organizacional

Nesta última categoria estão inseridos os desafios de *Big Data* que implicam mudanças nas organizações, decorrentes da sua implementação, adoção e utilização.

- **Valor e Competências de *Big Data***: A maioria dos líderes não entendem o valor de *Big Data* nem como extraí-lo, contudo, este tende a ser cada vez mais atraente para as organizações (Manyika & Chui, 2011). Em diversas áreas, as organizações precisam de monitorizar as tendências e obter vantagens em relação aos concorrentes, no entanto, muitas delas não têm o talento, nem os fluxos de trabalho, nem os incentivos necessários para lidar com *Big Data* (C. Costa & Santos, 2017a; Manyika & Chui, 2011). Assim, é necessário entenderem o valor que *Big Data* poderá trazer, assim como reconhecerem a necessidade de deterem especialistas competentes com os conhecimentos adequados para lidar com *Big Data* e com as tecnologias associadas, ou até mesmo a importância de dar as devidas formações (Katal et al., 2013; Manyika & Chui, 2011).
- **Cooperação**: A análise de *Big Data* é uma abordagem multidisciplinar, que requer que especialistas de diferentes áreas cooperem para obter o máximo potencial de *Big Data*. É necessária uma arquitetura em rede para auxiliar os intervenientes a perceber e utilizar diferentes tipos de dados, de áreas distintas da sua, mas que são dominadas por outro

interveniente com quem estão a cooperar, o que irá permitir otimizar as suas capacidades, de modo a contribuir para o alcance dos objetivos (Chen et al., 2014). Logo, as iniciativas de *Big Data* exigem a colaboração entre profissionais com diferentes competências, a fim de fornecer resultados úteis para a organização, ainda que para isso seja fundamental que ocorra mudanças organizacionais significativas (C. Costa & Santos, 2017a; Jagadish et al., 2014).

2.2.4 Processamento de Dados

No contexto tradicional o processamento de dados pode ser definido como a recolha, o processamento e a gestão de dados que tem como resultado informação para os consumidores finais. No processamento de dados transacionais primeiramente os dados são analisados, é criado um modelo de dados e posteriormente uma estrutura de base de dados para os processar. Os dados recolhidos têm uma natureza estruturada e são discretos em volume, sendo que todo o processo é pré-definido com base em requisitos conhecidos. Para além disso, tarefas como a qualidade e a limpeza dos dados não são importantes, já que são tratadas nos sistemas de origem (Krishnan, 2013).

O processamento de *Big Data*, ilustrado na Figura 6, difere amplamente do processamento tradicional, pois em ambientes de *Big Data*, inicialmente, os dados são recolhidos, carregados e armazenados numa plataforma destino, de seguida são aplicados os metadados e é criada uma estrutura de dados, só posteriormente é que os dados são transformados e analisados (Krishnan, 2013).

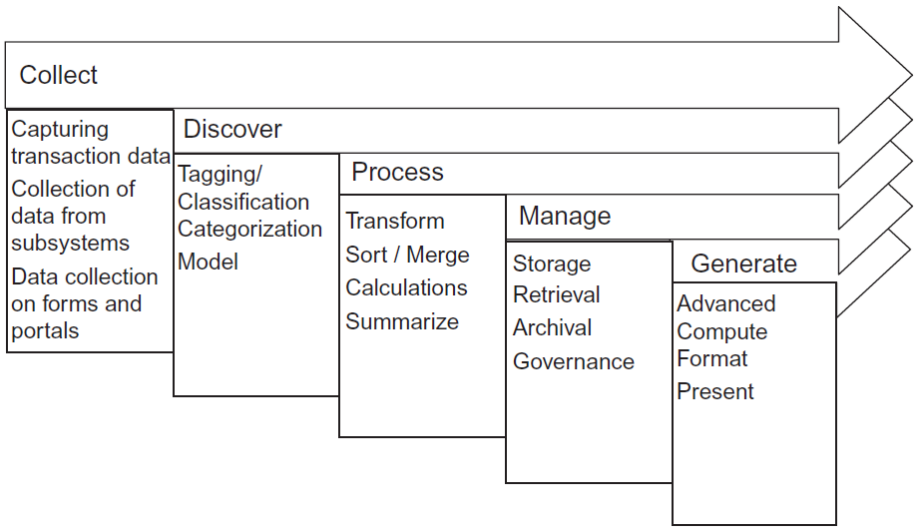

Figura 6. Ciclo de processamento de Big Data. Retirado de Krishnan (2013).

Segundo Krishnan (2013), em contextos de *Big Data*, para processar dados de forma flexível é mais adequada uma arquitetura orientada a ficheiros, visto que terá um desempenho superior ao

processar maior volume e complexidade de dados. Concludentemente, o autor apresenta um fluxo de processamento de *Big Data* constituído por quatro fases principais, conforme representado na Figura 7.

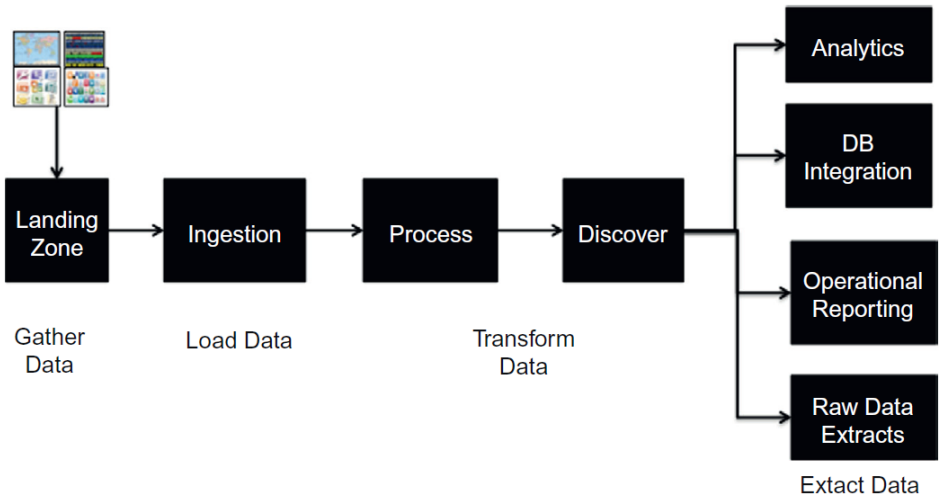

Figura 7. Fases do processamento de Big Data. Retirado de Krishnan (2013).

1. **Recolha de Dados** (*Gather Data*): Inicialmente os dados são recebidos de diferentes origens, carregados e armazenados num sistema de ficheiros denominado por área de estágio (*Landing Zone*);
2. **Carregamento dos Dados** (*Load Data*): Nesta fase os dados são carregados através da aplicação dos metadados, e com a aplicação de algum tipo de estrutura, ficando preparados para a transformação. O processo de carregamento divide os dados em ficheiros mais pequenos, criando um catálogo de ficheiros. Dependendo dos requisitos do utilizador e do processamento, nesta fase também é possível particionar os dados de forma horizontal ou vertical. No particionamento horizontal divide-se uma tabela em várias tabelas, sendo que cada uma contém o mesmo número de colunas, mas menos linhas. O particionamento vertical divide uma tabela em várias tabelas, mas que contêm menos colunas;
3. **Transformação dos Dados** (*Transform Data*): Esta fase diz respeito ao tratamento dos dados, pela aplicação de regras de negócio e processamento do seu conteúdo. Dentro desta fase encontram-se várias etapas, pelo que a sua gestão pode tornar-se complexa. A etapa de processamento produz resultados intermédios que podem ser armazenados para análise posterior;
4. **Extração dos Dados** (*Extract Data*): Na última fase, o conjunto de dados resultantes pode ser extraído para processamento adicional, incluindo análises, criação de relatórios operacionais, integração num DW e para propósitos de visualização.

São estas as principais fases para a obtenção dos dados a serem armazenados num DW sobre o qual poderão ser executadas consultas aos dados. Importa agora estudar os tipos de processamento existentes. Segundo Du (2015) os dados podem ser processados de três formas diferentes:

1. **Processamento em *batch***: Utilizado para processar os dados em lote. Um conjunto de dados é lido da fonte de dados sendo em seguida processado e finalmente escrito na base de dados ou no ficheiro de destino. É caraterizado por apenas ser capaz de executar uma tarefa de cada vez e de forma sequencial;
2. **Processamento em tempo real**: Consiste em processar os dados e obter os resultados quase que imediatamente. Carateriza-se por executar as tarefas em paralelo de maneira a conseguir apresentar os resultados prontamente;
3. **Processamento em *stream***: Processa os dados continuamente e atua sobre eles em *live stream* para obter os resultados, isto é, obtém-se resultados à medida que surgem os novos dados.

2.3 *Big Data Warehouses*

Tal como estudado anteriormente a abordagem mais frequente para a gestão de conjuntos de dados de larga escala consiste num DW, responsável por armazenar dados dos sistemas operacionais, sendo um recurso fundamental para suportar as decisões da organização. O volume de dados é um grande desafio para o *Data Warehousing*, tendo em consideração as suas tecnologias e, para além disso, os atuais tipos e formatos de dados também são um problema, pois põem em causa as regras de processamento do DW, visto que essas não podem ser aplicadas a textos, imagens, vídeos ou dados de sensores (Krishnan, 2013).

Assim, *Big Data Warehouse* (BDW) pode ser definido como um sistema escalável, de elevado desempenho, de armazenamento e processamento flexível que é capaz de lidar com o crescente volume, variedade e velocidade dos dados, ou seja, *Big Data*. Sabendo que *Big Data* impõe severas dificuldades às tecnologias tradicionais, os BDWs surgem para superar os seus desafios (C. Costa, Andrade, & Santos, 2018).

Portanto, um BDW será substancialmente diferente de um DW tradicional, pois deve-se basear em novos modelos lógicos mais flexíveis, cujas tecnologias possuem níveis mais altos de desempenho, escalabilidade e tolerância a falhas (Philip Chen & Zhang, 2014; Tria et al., 2014).

Assim, com o aparecimento do BDW surgiram novas caraterísticas importantes para os DWs tradicionais, que foram identificadas por uma comunidade de autores, sendo algumas destacadas por

outros (C. Costa & Santos, 2017b; Mohanty, Jagadeesh, & Srivatsa, 2013; M. Y. Santos, Costa, et al., 2017):

- Armazenamento flexível para suportar dados provenientes de múltiplas fontes;
- Escalabilidade para acomodar dados cada vez maiores, assim como utilizadores e análises, tudo isto a baixo custo;
- Alto desempenho através de respostas rápidas e quase que em tempo real;
- Processamento massivamente paralelo e distribuído;
- Cargas trabalho analíticas e complexas (tais como, consultas *ad hoc*, *data mining*, *text mining*, análise exploratória, análise geoespacial, vistas materializadas, simulações);
- Operações em tempo real (processamento em *stream*, baixa latência e atualizações frequentes);
- Interoperabilidade na gestão de múltiplas tecnologias;
- Capacidade de gerir compensações entre consistência, disponibilidade e tolerância a partições.

A metodologia para projetar um BDW deve ser (Tria et al., 2014):

- **Híbrida**, para considerar os benefícios das abordagens dos DWs tradicionais, orientadas a dados e requisitos;
- **Incremental**, com base em abordagens ágeis;
- E **automática**, a fim de tornar o desenvolvimento mais rápido, mesmo que várias fontes sejam consideradas. Então, espera-se adquirir rapidez na execução e na reação às mudanças frequentes nos requisitos do negócio.

Outros referem que a metodologia de conceção de um BDW deve ser uma abordagem ágil e iterativa que pela integração de diversas fontes de dados, sejam elas internas ou externas, a definição ou não dos modelos de dados e a execução de algoritmos especializados, permitam uma compreensão e perceção eficiente dos dados (Mohanty et al., 2013).

Para além disso, segundo outros autores a projeção dos BDWs deve-se concentrar na camada física, que diz respeito à infraestrutura tecnológica e na camada lógica que se centraliza no modelo de dados, componentes lógicos e fluxos de dados (Russom, 2014).

Na Tabela 2 encontra-se a síntese das caraterísticas associadas aos BDWs, assim como a comparação com as caraterísticas dos sistemas operacionais e analíticos, já apresentadas na Tabela 1.

Tabela 2. Sistemas Operacionais vs DW vs BDW. Baseado em H. Fang (2015), Sá (2010) e M. Santos & Ramos (2006).

Sistemas Operacionais	Sistemas Analíticos (DW)	*Big Data Warehouses* (BDWs)
Destina-se a administradores de sistemas e aplicações informáticas operacionais;	Destina-se a analistas de negócio, gestores e executivos;	Destina-se a analistas de negócio, gestores e executivos;
Operações diárias, registo de transações diárias (OLTP);	Suporte à tomada de decisão, transações analíticas (OLAP);	Suporta operações analíticas e operacionais;
Orientado às aplicações;	Orientado aos assuntos;	Orientado aos dados e aos requisitos;
Dados correntes, atualizados, atómicos, relacionais (normalizados) e isolados;	Dados históricos, sumarizados, multidimensionais e integrados;	Dados estruturados, semiestruturados e não estruturados;
Acesso por transações simples predefinidas, repetitivas e rotineiras;	Acesso por questões *ad hoc*, *queries* complexas;	Cargas trabalho analíticas e complexas, tais como, consultas *ad hoc*, *data mining*, *text mining*, análise exploratória, análise geoespacial, vistas materializadas, simulações;
Otimizado para acesso a poucos registos de cada vez (1 a 3 tabelas envolvidas);	Otimizado para acesso a muitos registos de cada vez (várias tabelas envolvidas);	Otimizado para acesso a muitos registos de cada vez (várias tabelas envolvidas);
Acessos de leitura e escrita;	Acessos só de leitura;	Acessos de leitura;
Dados atualizados em tempo real;	Carregamento periódico de mais dados, permite analisar o desempenho do negócio;	Operações em tempo real, processamento em *stream*, baixa latência e atualizações frequentes;
Estrutura otimizada para atualizações.	Estrutura otimizada para o processamento de questões.	Processamento escalável de dados.

O BDW pode ser implementado utilizando duas estratégias:

1. A estratégia de ***lift and shift***, em que as capacidades dos DWs tradicionais e relacionais são aumentadas a partir das tecnologias de *Big Data* (por exemplo, NoSQL), para resolver casos de uso específicos seguindo uma abordagem orientada ao assunto ao invés de uma abordagem de modelação de dados, o que, normalmente, provoca possíveis silos de dados não coordenados (Clegg, 2015; Philip Chen & Zhang, 2014).
2. Por outro lado, existe a estratégia ***rip and replace*** em que os DWs tradicionais são totalmente substituídos pelas tecnologias de *Big Data*, sendo uma arquitetura completamente moderna, podendo, contudo, ser prejudicial em alguns cenários, por exemplo, para as empresas que investiram durante muitos anos nas tecnologias tradicionais (Mohanty et al., 2013; R. Murthy & Goel, 2012; Russom, 2014).

As bases de dados cada vez mais utilizadas nos contextos de *Big Data* são as NoSQL, as quais irão ser abordadas de seguida. Estas bases de dados são muitas das vezes referidas como *schema-free*,

pois o esquema dos dados pode mudar ao longo do tempo consoante as necessidades analíticas. Contudo, apesar de serem *schema-free*, em algum momento os dados precisam de ser estruturados, isto é, as bases de dados acabam por precisar de um modelo de dados para apoiar os processos de tomada de decisão, através de mecanismos de análise e de armazenamento adequados (C. Costa et al., 2018; Durham, Rosen, & Harrison, 2014).

Tal como já referido, as metodologias de BDW devem adotar novos modelos lógicos, tais como aqueles que são utilizados pelas bases de dados NoSQL, de forma a assegurar escalabilidade, flexibilidade e desempenho (Tria et al., 2014).

2.4 Armazenamento de Dados

Um dos grandes desafios de *Big Data* é o armazenamento dos dados, tal como estudado na secção 2.2.. Assim sendo, na presente secção serão apresentados os conceitos e caraterísticas associadas às bases de dados SQL, NoSQL e NewSQL, assim como a comparação entre elas.

2.4.1 Bases de Dados SQL, NoSQL e NewSQL

Os sistemas tradicionais de bases de dados baseiam-se na linguagem SQL e são fundamentados num modelo de dados relacional, cuja estrutura é normalizada, no entanto, nos últimos anos tem aumentado o interesse por bases de dados não relacionais, sendo estas conhecidas por *Not only SQL* (NoSQL) (Alekseev et al., 2016; Fatima & Wasnik, 2016; Li & Manoharan, 2013). Esse fenómeno mudou a forma como as bases de dados são projetados atualmente, embora os sistemas relacionais estejam em conformidade com as propriedades ACID (Bonnet, Laurent, Sala, Laurent, & Sicard, 2011):

- **Atomicidade** (*Atomicity*), o que significa que a falha de apenas uma parte de uma operação resulta na falha de toda a operação, ou seja, toda a operação deve ser bem-sucedida;
- **Consistência** (*Consistency*), onde erros de leitura e escrita são evitados, isto é, uma operação não pode deixar a base de dados num estado inconsistente;
- **Isolamento** (*Isolation*), sendo que várias operações geradas ao mesmo tempo não têm impacto umas nas outras;
- **Durabilidade** (*Durability*), uma operação completa deve permanecer na base de dados, mesmo depois do sistema reiniciar ou falhar.

Os sistemas de gestão de bases de dados relacionais fornecem mecanismos poderosos para armazenar e consultar dados estruturados, garantindo a consistência nestes processos e atingindo um

nível incomparável de confiabilidade, estabilidade e suporte durante vários anos (Gessert, Wingerath, Friedrich, & Ritter, 2017).

Com o aumento da acessibilidade à *internet* e da disponibilidade de armazenamento, enormes quantidades de dados estruturados, semiestruturados e não estruturados são apreendidos para uma variedade de aplicações, sendo esses dados designados de *Big Data*, tal como já constatado em pontos anteriores (Fatima & Wasnik, 2016; Li & Manoharan, 2013). O processamento de *Big Data* exige velocidade, esquemas flexíveis e bases de dados distribuídas, isto é, não centralizadas (Li & Manoharan, 2013).

As bases de dados *Not only SQL* (NoSQL), são a nova abordagem para gerir as grandes quantidades de dados. A sua designação "NoSQL" poderá gerar mal-entendidos, na medida em que dá a entender que se exclui o *SQL*, no entanto, embora alguns sistemas NoSQL sejam totalmente não relacionais outros simplesmente evitam algumas funcionalidades relacionais, como esquemas fixos ou operações de junção (Philip Chen & Zhang, 2014).

As tecnologias de bases de dados evoluíram significativamente de forma a lidar com volumes de dados de diferentes escalas, pois as abordagens tradicionais (bases de dados SQL) já não são capazes de se adaptar ao *Big Data*, assim surgiram e tornaram-se populares as bases de dados NoSQL. Por conseguinte, este novo tipo de bases de dados, com a devida escalabilidade, fornecem mecanismos para gerir grandes volumes de dados distribuídos, assim como para suportar as necessidades de acesso aleatório aos dados (Hashem et al., 2015; Philip Chen & Zhang, 2014).

Então, as bases de dados NoSQL surgiram para oferecer escalabilidade horizontal e maior disponibilidade relativamente às bases de dados relacionais, sacrificando, contudo, a consistência das bases de dados tradicionais (Gessert et al., 2017). Escalabilidade horizontal significa adicionar mais nós ao sistema, tais como adicionar um novo servidor e um sistema de *software* que permita a distribuição do trabalho entre múltiplas máquinas (Cattell, 2011; Vaish, 2013).

Estas bases de dados são vistas com sistemas de armazenamento distribuídos, escaláveis, elásticos e tolerantes a falhas, pelo que apresentam alta disponibilidade, assim, no caso de um dos nós falhar, os dados são replicados pelas várias máquinas (Kambatla, Kollias, Kumar, & Grama, 2014).

As bases de dados NoSQL possuem certas caraterísticas e vantagens face às tradicionais (Cassavia et al., 2014; Cattell, 2011; Leavitt, 2010; Vaish, 2013):

- Distribuídas, escaláveis horizontalmente e não dependem de recursos de *hardware*;
- Grande parte destas bases dados são *open-source*, implicando custos reduzidos;
- Armazenamento de dados em grande escala sem necessidade de definir uma estrutura;

- Modelos simples e flexíveis;
- Tempo de desenvolvimento reduzido, pois não é necessário lidar com consultas SQL complexas;
- Taxa de transferência maior;
- Replicam e particionam os dados em muitos servidores;
- Adicionam dinamicamente novos atributos aos registos;
- Eventualmente consistentes/BASE, não são capazes de suportar as propriedades ACID, evitando a complexidade desnecessária;
- Suportam principalmente *queries* e poucas atualizações.

Leavitt (2010) destaca ainda algumas desvantagens associadas aos sistemas NoSQL:

- Podem implicar **sobrecarga** e **complexidade**, pois como não funcionam com a linguagem SQL, pois exigem programação "manual", o que poderá ser uma tarefa mais custosa para algumas pessoas;
- Não oferecem o nível de **fiabilidade** que as propriedades ACID oferecem, pelo que para se obter tais funcionalidades é necessária programação adicional;
- Como não suportam as transações ACID, podem comprometer a **consistência**, contudo, permite melhorar o desempenho e a escalabilidade, mas isto é um problema para alguns setores, como por exemplo, o bancário;
- A maioria das organizações **desconhece a tecnologia** ou não está familiarizada com o NoSQL, por isso não tem capacidades para as usar;
- Ao contrário das bases de dados relacionais as aplicações NoSQL, muitas das vezes demonstram **falta de suporte** ou de ferramentas de gestão necessárias ao cliente.

Como é possível aferir, as bases de dados NoSQL, ao contrário das bases de dados SQL, não suportam as propriedades ACID (atomicidade, consistência, isolamento e durabilidade), contudo, suportam um conjunto de outras propriedades, as BASE (Vaish, 2013):

- **Disponibilidade básica** (*Basic Availability*) o sistema garante a disponibilidade, pelo que para cada pedido há resposta;
- **Estado flexível** (*Soft state*), o estado do sistema pode mudar ao longo do tempo;
- **Eventual consistência** (*Eventual consistency*), o sistema acabará por se tornar consistente, apesar de poder estar momentaneamente inconsistente.

Como um sistema distribuído que é, o NoSQL segue as considerações do teorema da consistência, disponibilidade e tolerância à partição (CAP): "qualquer sistema de dados compartilhados

numa rede pode ter no máximo duas das três propriedades" (Brewer, 2012). O teorema pode ser consultado na Figura 8, e a definição de cada componente apresenta-se posteriormente.

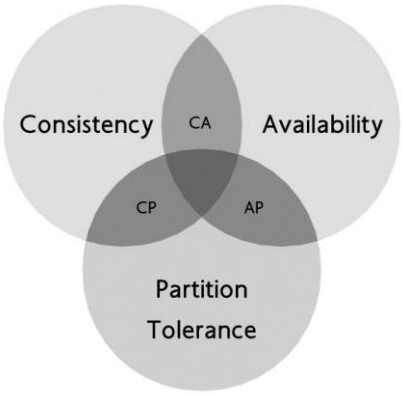

Figura 8. Teorema CAP de Eric Brewer. Adaptado de Han, E, Le e Du (2011).

- **Consistência** (C – *Consistency*): Após consulta, todos os nós, isto é, todos os utilizadores, terão acesso à mesma visão sobre os dados, mesmo com atualizações simultâneas. Esta caraterística também está muito relacionada com a atomicidade e o isolamento das propriedades ACID (Cassavia et al., 2014; Gessert et al., 2017);
- **Disponibilidade** (A – *Availability*): Todos os utilizadores poderão a qualquer altura ler e gravar dados, ou seja, qualquer pedido deve sempre resultar numa resposta. Um sistema apresenta disponibilidade se for projetado e implementado de uma forma que permita que funcione adequadamente, mesmo que algum nó do *cluster* falhe ou no caso de algum componente do *hardware* ou *software* ficar inativo devido a operações de manutenção (Cassavia et al., 2014; Gessert et al., 2017);
- **Tolerância à Partição** (P – *Partition Tolerance*): A base de dados pode ser dividida em várias máquinas, continuando a funcionar face a quebras de rede, ou seja, o sistema continua em execução mesmo que a rede seja particionada. Só no caso de falha total da rede é que o sistema deixa de funcionar. No caso de um sistema particionado, não é possível manter a consistência e a disponibilidade simultaneamente. Assim, mesmo que continue a execução apesar da partição, algum dos nós perdeu contacto com os outros, pelo que tem que se decidir dar continuidade ao processamento dos pedidos, garantindo a disponibilidade, ou se, por outro lado, se rejeitam os pedidos, de modo a manter a consistência (Cassavia et al., 2014; Gessert et al., 2017).

Tendo como base o teorema CAP representado na Figura 8, na Figura 9 é possível analisar o cruzamento das suas caraterísticas, assim como o exemplo de algumas bases de dados NoSQL que podem ser classificadas segundo as duas propriedades que conseguem responder.

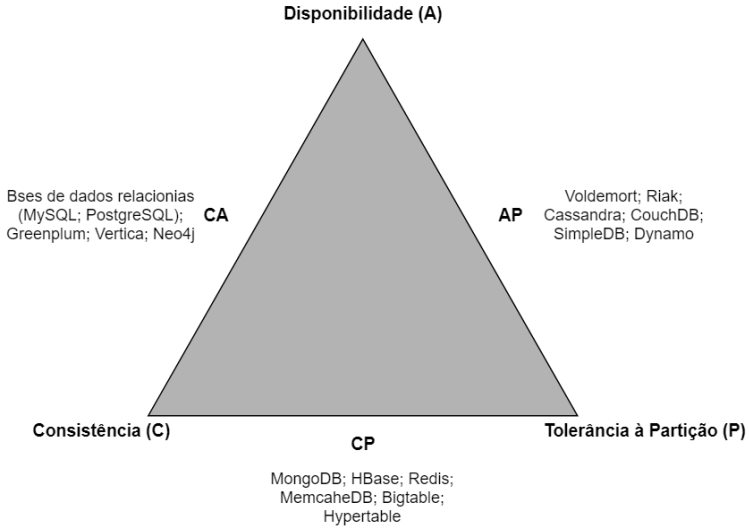

Figura 9. Teorema CAP com exemplos de algumas bases de dados NoSQL. Adaptado de Bonnet et al. (2011).

Portanto, tal como verificado anteriormente as bases de dados NoSQL com algumas caraterísticas das relacionais, apenas conseguem garantir a consistência e a disponibilidade, pelo que as restantes para garantirem a tolerância à partição, terão de abdicar da consistência ou da disponibilidade.

Existe uma diversidade de bases de dados NoSQL, pelo que surgiu a necessidade de as dividir em quatro tipos, conforme ilustrado na Figura 10, sendo que cada um dos tipos se encontra descrito posteriormente.

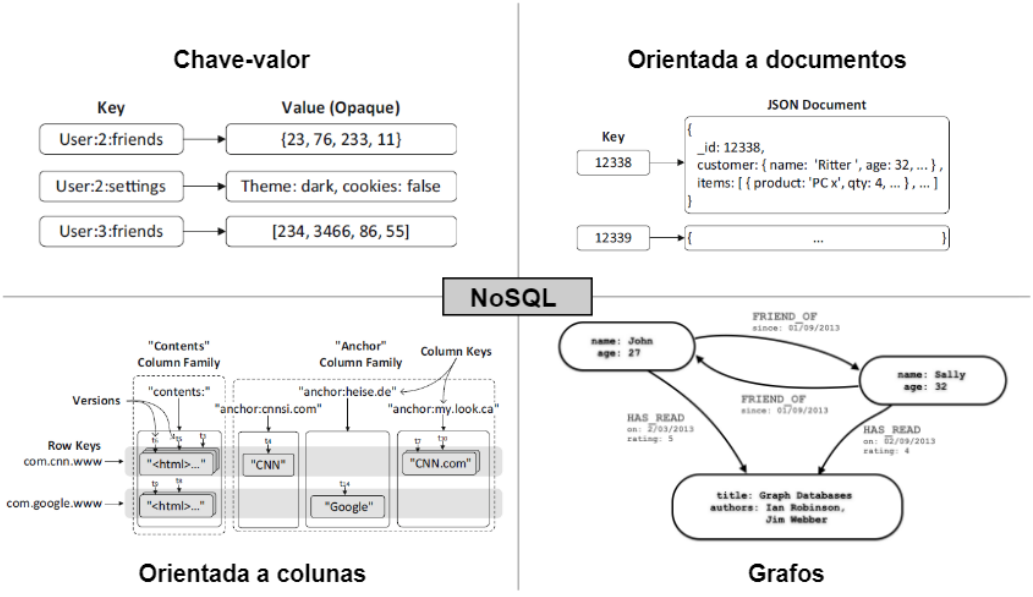

Figura 10. Tipos de bases de dados NoSQL. Adaptado de Bhogal e Choksi (2015) e Gessert et al. (2017).

1. **Bases de dados chave-valor** (*Key-value*): Usam uma tabela *hash* onde existe uma chave única e um apontador para um determinado conjunto de valores (Bhogal & Choksi, 2015). São conhecidos por serem *schema-free*, mas podem não ter a capacidade de representar adequadamente relações ou estruturas, uma vez que as consultas são asseguradas pela chave (Grolinger, Higashino, Tiwari, & Capretz, 2013). Cada chave é única e as consultas estão estritamente ligadas às chaves (Chen et al., 2014). Este modelo apresenta simplicidade na medida em que facilita a partição e a consulta dos dados e, com isso, permite que as base de dados apresentem baixa latência e altas taxas de transferência (Gessert et al., 2017). Exemplos de bases de dados que utilizam este tipo de modelo são: Voldemort[8], Dynamo[9] e Redis[10] (C. Costa & Santos, 2017a);
2. **Bases de dados orientada a colunas** (*Column-oriented*): Este modelo pode ser visto como uma extensão do modelo chave-valor, tornando a consulta mais poderosa, devido à adição de linhas, colunas e famílias de colunas (Grolinger et al., 2013; Krishnan, 2013). Cada linha é composta por um conjunto de famílias de colunas, sendo que cada linha representa uma determinada entidade (Gessert et al., 2017; Grolinger et al., 2013). Da mesma forma que o modelo chave-valor, a chave da linha é semelhante à chave e o conjunto de famílias de colunas equivale ao valor representado pela chave da linha. No entanto, cada família de colunas também atua como uma chave para uma ou mais colunas que contém (Grolinger et al., 2013). Exemplos de bases de dados que são representadas com este modelo: HBase[11], Cassandra[12], Hypertable[13] (C. Costa & Santos, 2017a);
3. **Bases de dados orientada a documentos** (*Document*): é outra extensão do modelo chave-valor e é adequado para representar dados em formato de documentos, tais como JSON (*JavaScript Object Notation*), XML (*eXtensible Markup Language*) e YAML (*Yet Another Markup Language*) (Gessert et al., 2017; Kaur & Rani, 2013). Pode conter estruturas complexas, tais como objetos *nested* e índices secundários fornecendo maior flexibilidade na consulta do que o modelo chave-

[8] https://www.project-voldemort.com/voldemort/

[9] https://aws.amazon.com/pt/dynamodb/

[10] https://redis.io/

[11] https://hbase.apache.org/

[12] http://cassandra.apache.org/

[13] http://www.hypertable.org/

valor (Grolinger et al., 2013). Exemplos de bases de dados NoSQL deste tipo: CouchDB[14], MongoDB[15] (C. Costa & Santos, 2017a);

4. **Bases de dados de grafos** (*Graph*): Os objetos podem ser representados por nós e os relacionamentos entre eles representados por arestas (Krishnan, 2013). Este tipo é especializado e eficiente no armazenamento de dados com diversos relacionamentos entre as entidades, armazenando as relações entre os diferentes nós (Grolinger et al., 2013). Comparando este com o modelo relacional, um nó corresponde a uma entidade, uma propriedade do nó a um atributo e o relacionamento entre nós corresponde ao relacionamento entre entidades (Kaur & Rani, 2013). As bases de dados de grafos são adequadas para encontrar relacionamentos dentro de grandes quantidades de dados de uma forma mais rápida, já que os *joins* não são executados (Kaur & Rani, 2013). Quase todas as bases de dados de grafos têm capacidade de armazenar informação semiestruturada (Kaur & Rani, 2013). Por exemplo, as bases de dados Neo4j[16] e GraphDB[17] utilizam este tipo de modelo (C. Costa & Santos, 2017a).

Contudo, a tendência é combinar os benefícios dos sistemas de armazenamento SQL e NoSQL (Cuzzocrea, Song, & Davis, 2011). Para tal, um novo conceito de bases de dados esta a surgir, é ele o NewSQL, este combina os modelos relacionais com os benefícios da escalabilidade fornecidos pelas NoSQL (Grolinger et al., 2013). Este tipo de bases de dados utilizam a linguagem SQL, para além de que, tal como as NoSQL, são projetadas para escalar cargas de trabalho OLTP (*Online Transaction Processing*) sobre vários nós, atendendo aos requisitos de ambientes com milhões de operações simples, ao mesmo tempo que mantêm as propriedades ACID das bases de dados tradicionais. São exemplos deste novo tipo de bases de dados: VoltDB[18], Clustrix[19] e NuoDB[20] (Cattell, 2011; C. Costa & Santos, 2017a; Fatima & Wasnik, 2016).

[14] http://couchdb.apache.org/

[15] https://www.mongodb.com/

[16] https://neo4j.com/

[17] http://graphdb.ontotext.com/

[18] https://www.voltdb.com/

[19] https://www.clustrix.com/

[20] https://www.nuodb.com/

2.4.2 Modelos de dados

Os DWs, como peças centrais em BI e *Business Analytics*, são suportados por modelos de dados que permitem a análise dos dados sob diferentes perspetivas. Os modelos de dados são uma peça central no desenvolvimento de sistemas de informação, pois garantem que as necessidades analíticas dos dados sejam devidamente consideradas (M. Y. Santos & Costa, 2016a).

Segundo Elmasri e Navathe (2010) uma caraterística fundamental da abordagem de bases de dados é que esta fornece um certo nível de abstração dos dados. Esta abstração refere-se à supressão de detalhes do armazenamento e ao destaque dos recursos essenciais para uma melhor compreensão dos dados por parte dos utilizadores. Um modelo de dados pode ser definido como um conjunto de conceitos que podem ser utilizados para descrever a estrutura de uma base de dados, fornecendo os meios necessários para alcançar a dita abstração.

Elmasri e Navathe (2010) propõem três modelos de dados categorizados de acordo com os tipos de conceitos que utilizam para descrever a estrutura da base de dados. Os vários modelos considerados encontram-se descritos de seguida (Elmasri & Navathe, 2010; Vaisman & Zimányi, 2014):

1. **Modelo de Dados Conceptual** (*High-level or Conceptual data model*): Também designado como modelo de alto nível, fornece conceitos que se aproximam do modo como os utilizadores entendem os dados, não incluindo pormenores da implementação. Utiliza conceitos tais como entidades, atributos e relações;
2. **Modelo de Dados Lógico** (*Representational or Implementation data model*): Ou modelo de implementação, fornece conceitos que podem ser facilmente compreendidos pelos utilizadores finais, apesar de apresentarem detalhes sobre a forma como os dados são armazenados e organizados no modelo da base de dados. Um dos conceitos mais utilizados é a estrutura dos registos que depende do sistema de base de dados a utilizar;
3. **Modelo de Dados Físico** (*Low-level or Physical data model*): Ou modelo de baixo nível, fornece conceitos que descrevem os detalhes de como os dados são armazenados, sendo este tipo destinado a especialistas e não a utilizadores no geral. Este modelo, parte do modelo lógico e customiza-o para uma plataforma de gestão de base de dados específica. Formatos dos registos e organização dos registos são conceitos que, normalmente, estão associados a este modelo.

Dehdouh, Bentayeb, Boussaid e Kabachi (2015) também distinguem esses três níveis de abstração, conforme ilustrado na Figura 11.

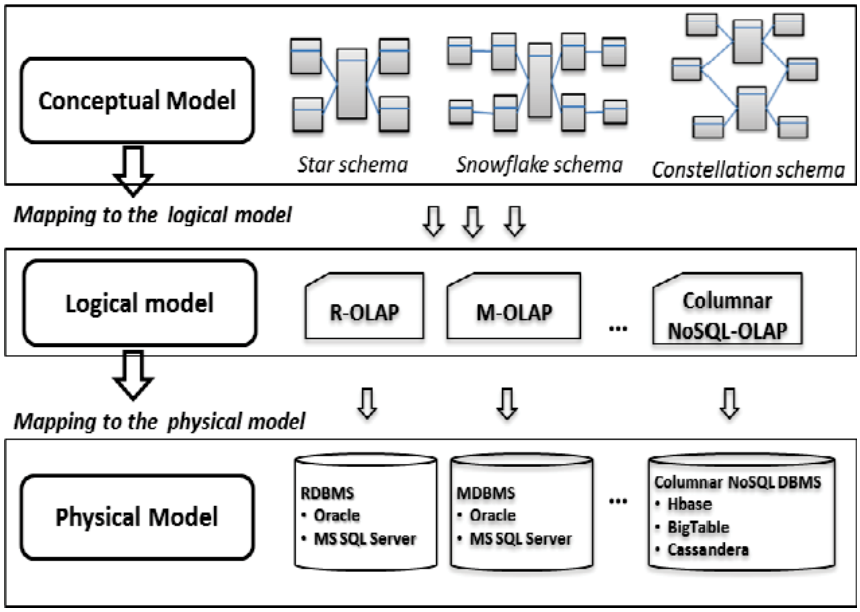

Figura 11. Modelos utilizados no processo de implementação de um Data Warehouse. Retirado Dehdouh et al. (2015).

O modelo conceptual altamente aceite como referência para a construção de DWs é o esquema multidimensional, sendo que os mais conhecidos são os que se encontram representados na primeira componente da Figura 11 sendo descritos posteriormente (Dehdouh et al., 2015; Inuwa & N. D. Oye, 2015; Kimball & Caserta, 2011; Kimball & Ross, 2013):

- **Modelo em estrela** é constituído por tabelas de factos e dimensões através de relações de chave primária ou secundária, que possibilitam as análises sobre diversas perspetivas;
- **Modelo em floco de neve** que engloba dimensões normalizadas, os atributos de menor cardinalidade aparecem como tabelas secundárias dessa dimensão. Sendo por isso um esquema mais complexo, pelo que dificulta a navegação pelos dados;
- **Modelo em constelação** que é constituído por múltiplas tabelas de factos que partilham dimensões. Corresponde à união de um ou mais modelos em estrela.

O modelo lógico visa reorganizar os dados de acordo com a arquitetura de armazenamento mais adequada, pelo que se encontra entre os modelos concetuais e os modelos físicos dos dados. Por outras palavras, fornece maior detalhe do que o modelo conceptual relativamente à estruturação dos dados e às suas relações, mas também prepara a transição para o nível físico. Isso faz com que seja o modelo mais decisivo no processo de modelação (Dehdouh et al., 2015).

O mapeamento entre o modelo conceptual e o modelo lógico é efetuado de acordo com três abordagens: **ROLAP** (*Relational*-OLAP), **MOLAP** (*Multidimensional*-OLAP) e **HOLAP** (*Hybrid*-OLAP). Os modelos de dados relacionais são os mais utilizados, sendo definidos segundo regras muito específicas

relativamente aos requisitos que estes modelos devem considerar. No entanto, tal como já observado em pontos antecedentes, com o aparecimento de *Big Data*, a modelação lógica das abordagens tradicionais utilizada na construção de DWs já não é capaz de se adaptar a um ambiente caraterizado por essa abundância de dados. Pelo que a adoção de *BDW*s implica a adesão a novos modelos lógicos, como o NoSQL orientado a colunas. Para além disso, pela observação da Figura 11, os modelos "Columnar NoSQL-OLAP" são responsáveis pela associação dos modelos concetuais tradicionais e as bases de dados NoSQL adotadas no modelo físico (Dehdouh et al., 2015; M. Y. Santos & Costa, 2016a). Em contextos de *Big Data* onde as bases de dados NoSQL são normalmente utilizadas, os modelos lógicos dos dados são *schema-free*, contudo, na prática estes modelos têm um esquema de dados, mas a sua definição segue uma abordagem diferente. Ao invés de refletir as entidades relevantes de um certo domínio assim como os relacionamentos entre elas, os esquemas são então definidos tendo em consideração as consultas que precisam de resposta, sendo os dados replicados quantas vezes necessárias. Portanto, numa fase inicial a recolha de dados pode ser menos rígida, mas a certa altura, quando as tarefas analíticas assim o exigem, é necessária alguma estrutura nos dados (M. Y. Santos & Costa, 2016a).

Devido ao crescimento do volume de dados, surge a necessidade de se transferir as bases de dados relacionais para bases de dados colunares. Até mesmo certos contextos exigem a rápida transição do ambiente tradicional para *Big Data*, pelo que seria benéfico para os utilizadores a existência de regras e modelos para suportar esta transição (M. Y. Santos & Costa, 2016a). Assim, é proposto por M. Y. Santos e Costa (2016a) um conjunto de regras para a transição automática entre estes ambientes. Os modelos podem ser transformados: num modelo de dados baseado em colunas, por exemplo, no HBase, para suportar necessidades operacionais, isto num contexto de NoSQL; ou num modelo de dados tabular, como é o caso do Hive[21], para suportar as necessidades analíticas, sendo este último associado ao contexto de BDWs (M. Y. Santos & Costa, 2016a, 2016b).

2.5 *Data Governance*

Partindo dos desafios identificados na subsecção 2.2.3, referentes ao Ciclo de Vida de *Big Data*, nomeadamente o desafio de Governança de *Big Data*, surge a necessidade de explorar o conceito de Governança de Dados com maior detalhe.

[21] https://hive.apache.org/

A Governança de Dados descreve o modo como as organizações gerem todos os dados que passam pelos seus processos de negócio. A governança não é um projeto único, mas sim um programa constante. Esta governança incentiva o comportamento desejável na avaliação, criação, armazenamento, utilização, catalogação e exclusão de dados. Inclui processos, funções, padrões e métricas que garantem a utilização eficaz e eficiente dos dados para permitir que a organização atinja os seus objetivos (Smallwood, 2014).

A fim de preparar efetivamente a transição para *Big Data*, as organizações que aderem a uma estrutura de Governança de Dados estão melhor posicionadas para serem bem-sucedidas. Governança de Dados consiste num sistema que descreve quem pode executar determinadas ações, quando o podem fazer, em que circunstâncias e com que métodos. Assim, a Governança de Dados inclui políticas relacionadas com a otimização e a privacidade de *Big Data*. A governança de *Big Data* pode auxiliar na criação e utilização correta dos dados para a tomada de determinadas decisões de negócio, independentemente de onde a informação vem, de que tipo é, ou quão rápido se move (Soares, 2013a).

A estrutura de Governança de Dados é definida como um conjunto de processos que garante que os ativos dos dados sejam geridos por toda a empresa (Sarsfield, 2009). A governança pode ser estruturada de várias formas em diferentes organizações, por exemplo, Khatri e Brown (2010) apresentam uma estratégia de Governança de Dados projetada para orientar a tomada de decisão organizacional e reduzir os riscos para as organizações. Estes organizam a Governança de Dados em cinco domínios interrelacionados, constituindo uma infraestrutura a partir da qual a Governança de Dados é implementada. Khatri e Brown (2010) descrevem como as organizações devem aderir a esses domínios e o impacto que implicam quando aplicados:

- Princípios dos dados: As organizações devem estabelecer os princípios dos dados, que são um ativo de toda a empresa e, portanto, que políticas, normas e diretrizes são apropriadas. Os princípios fornecem um meio para as organizações reconhecerem o significado dos dados como um ativo;
- Qualidade dos dados: As organizações devem estabelecer padrões para garantir que os dados estejam atualizados, completos e confiáveis. Os padrões de qualidade reduzem os impactos financeiros da má qualidade dos dados;
- Metadados: Os metadados fornecem um mecanismo para uma descrição concisa e consistente da representação dos dados, ajudando a interpretar o seu significado. Os metadados fornecem também um meio sólido para que as organizações realizem eficientemente pesquisas e análises aos seus dados;

- **Acesso aos dados**: As organizações devem controlar o acesso aos ativos dos dados, de modo a gerirem o risco e a conformidade. Os padrões de acesso estabelecem regras para as organizações definirem que dados podem ser acedidos e por quem;
- **Ciclo de Vida dos dados**: As organizações devem entender e documentar como os dados progridem através dos seus processos de negócio. O ciclo de vida fornece às organizações os meios para compreender a forma como os dados provêm de várias fontes, tanto de dentro como de fora da organização.

Sabendo que os dados estão a surgir de uma forma exponencial, provenientes de diferentes fontes, com diferentes caraterísticas, propósitos e restrições, a governança implica o controlo sobre as regras relacionadas a essas grandes quantidades de dados (Hashem et al., 2015). Tal como referido por vários autores, devido às exigências provocadas pela heterogeneidade dos dados, será fundamental a capacidade de gerar de forma automática os metadados sobre os dados (Chandarana & Vijayalakshmi, 2014; Katal et al., 2013; Labrinidis et al., 2012). Subentende-se que metadados são dados sobre os dados.

Associado ao conceito de Governança de Dados existe um *framework* designado por *DAMA International's Guide to the Data Management Body of Knowledge* (*DAMA-DMBOK Guide*), que se encontra ilustrado na Figura 12. Este reúne um conjunto de processos e áreas de conhecimento, geralmente aceites como as melhores práticas de Governança de Dados (DAMA, 2014).

Figura 12. Roda das Áreas de Conhecimento de Data Governance. Adaptado de DAMA (2014).

Governança de Dados é um termo abrangente que descreve os processos empregues para planear, especificar, ativar, criar, adquirir, controlar e limpar dados, sendo que todos estes processos interagem dentro de cada área de conhecimento (DAMA, 2014).

Analisando a Figura 12 e os vários conceitos que envolve e tendo em consideração o foco deste trabalho, identificar os metadados sobre os dados, subentende-se que as componentes *tagging* e *lineage* (etiquetagem e rastreio) de fontes de dados estão inseridas na área de conhecimento Gestão de ***Metadata***. Esta área traduz-se em recolher, categorizar, manter, integrar, controlar e gerir os metadados. Para além desta área, o ***Data Quality*** está diretamente relacionado com ***Data Profiling*** e consiste em definir, monitorizar e manter a integridade dos dados, pelo que se definem métricas, regulamentos ou padrões para a obtenção do conhecimento sobre os dados. Assim, o *Data Profiling*, como uma subárea de *Data Quality*, irá auxiliar na obtenção de alguns metadados.

2.5.1 *Data Quality*

A Qualidade dos Dados desempenha um papel importante na Governança de Dados (Dai et al., 2016). Qualidade dos Dados pode ser definida como a atividade de detetar e corrigir anomalias nos dados. Assim, os dados são considerados de elevada qualidade se forem adequados às operações às quais estão destinados (Rodrigues, 2017). Quando isto não acontece, significa que existe um conjunto de problemas na qualidade dos dados tanto ao nível do esquema, como nas respetivas instâncias (Rodrigues, 2017). No entanto, parte deles podem ser eliminados através de procedimentos e tecnologias tradicionais, mas outra parte não (Barateiro & Galhardas, 2005).

Algumas dimensões de Qualidade dos Dados que se concentram na qualidade da informação para a Governança de Dados são enumeradas de seguida:

- **Acessibilidade** (*accessibility*): Capacidade de acesso aos dados. Extensão na qual os dados estão disponíveis ou podem ser recuperados com relativa facilidade e rapidez (Batini & Scannapieco, 2006; Pipino, Lee, & Wang, 2002);
- **Precisão** (*accuracy*): Grau em que os dados descrevem corretamente o objeto ou evento, ou seja, grau em que são equivalentes aos seus valores reais. Por exemplo, em valores numéricos, com vista a verificar se estão incluídos num intervalo esperado ou se são *outliers*, neste último caso não são precisos/corretos (Ardagna, Cappiello, Samá, & Vitali, 2018; Batini & Scannapieco, 2006);
- **Completude** (*completeness*): Mede a proporção de dados armazenados em relação ao total. Ou seja, avalia a proporção entre a quantidade de valores atualmente disponíveis no conjunto de

dados e a quantidade esperada. A quantidade esperada de valores considera valores nulos e registos em falta (Ardagna et al., 2018; Batini & Scannapieco, 2006; Pipino et al., 2002);

- **Consistência** (*consistency*): Refere-se à não verificação das regras de semântica definidas sobre um conjunto de dados. Contudo, apenas pode ser calculada se houver disponibilidade de dados que apresentem dependências entre os atributos (Ardagna et al., 2018; Batini & Scannapieco, 2006);
- **Atualidade** (*timeliness*): Grau em que os dados são suficientemente atualizados para uma certa tarefa, isto é, se estão disponíveis antes do tempo de utilização planeado (Ardagna et al., 2018; Batini & Scannapieco, 2006; Pipino et al., 2002);
- **Mais-valia** (*value-added*): Modo como os dados são benéficos e fornecem vantagens a quem os utiliza (Batini & Scannapieco, 2006; Pipino et al., 2002);
- **Segurança** (*security*): Até que ponto o acesso aos dados é restrito, para manter a segurança dos mesmos de forma apropriada (Pipino et al., 2002).

A execução de uma estratégia de *Data Quality* envolve várias atividades, cuja finalidade será melhorar a qualidade dos dados, tendo em conta as dimensões identificadas num determinado contexto. Algumas das atividades mais citadas são (Juddoo, 2015):

- ***Data Profiling***: Consiste na análise de fontes de dados com o objetivo de gerar dados e informação sobre os dados;
- **Definição de regras de qualidade dos dados**: Regras a utilizar para determinar ou descobrir determinados problemas nos dados, ou que seriam utilizadas para a atividade de limpeza de dados (*Data Cleansing*);
- ***Data Cleansing***: Com base nas regras de negócio especificadas, aplica um conjunto de técnicas que têm como propósito aperfeiçoar ou transformar os dados.

Dada a relevância do *Data Profiling* para este trabalho, a próxima subsecção caracteriza em mais detalhe este conceito e algumas das suas tarefas.

2.5.2 *Data Profiling*

A componente de *Data Profiling* é definida como uma das subatividades do processo de *Data Quality*, permitindo melhorar a qualidade dos dados (Dai et al., 2016). O processo de *Data Profiling* tem como objetivo detetar sistematicamente erros, inconsistências, redundâncias e dados incompletos. Depois de analisar os dados, o *Data Profiling* produz um conjunto de relatórios que contem informação sobre os dados (Rodrigues, 2017).

Normalmente, o *Data Profiling* também é associado a um conjunto de atividades e processos para determinar os metadados sobre um determinado conjunto de dados. Existe uma série de metadados que são mais simples de calcular, podendo ser designados por estatísticas simples, por exemplo, o número de valores nulos, valores distintos, o tipo de dados ou os padrões mais frequentes, isto relativamente a uma determinada coluna. Contudo, existem outros metadados que são mais complexos, pois implicam a correlação entre várias colunas (Abedjan, Golab, & Naumann, 2015).

Assim, segundo Abedjan et al. (2015), o *Data Profiling* implica três desafios:

- **Gerir o *input*** (*managing the input*): Diz respeito à especificação do resultado esperado, isto é, quais as tarefas de *Data Profiling* que devem ser aplicadas a certas partes dos dados. Algumas abordagens exigem o detalhe do que é suposto examinar, sendo que outras executam certas tarefas descobrindo os metadados de forma automática;
- **Realizar a computação** (*performing the computation*): A complexidade da computação dos algoritmos de *Data Profiling* depende do número de linhas e do número de colunas. Muitas das tarefas necessitam de analisar todas as combinações de colunas, ou seja, são exponenciais no número de colunas. A escalabilidade dos métodos de *Data Profiling* é importante, pois como os dados estão em constante crescimento exigem processamento distribuído;
- **Gerir o *output*** (*managing the output*): Qualquer metadado refere-se apenas à instância de dados que é fornecida e não pode ser usado para derivar propriedades estruturais e semânticas, tais como domínios de valor, chaves primárias ou relações de chave estrangeira, pelo que é necessária a interpretação efetiva dos resultados do *Data Profiling* por parte de um especialista.

O *Data Profiling* reúne um conjunto de casos de uso, sendo destacado por Abedjan et al. (2015) aqueles que contribuem para a gestão e análise de dados:

- **Exploração de Dados** (*Data Exploration*): Como a tendência do volume dos dados é aumentar, sejam dados oriundos por via *web*, ou de bases de dados históricas ou mais recentemente adquiridas, existe a necessidade de proceder à sua exploração. Muitas das vezes os dados chegam com esquemas incompletos, ou até mesmos inexistentes e sem documentação, pelo que, inicialmente, tem que se entender como são efetivamente estruturados, qual o seu significado, bem como a sua quantificação. Esta exploração pode ser suportada por técnicas de *Data Profiling*. Apesar de algumas consultas SQL mais simples já revelarem algumas tendências, como valores distintos e nulos, existe a necessidade de descobrir de forma eficiente e sistemática os metadados (Abedjan et al., 2015). Morton, Balazinska, Grossman e Mackinlay (2014)

destacam a possível suposição de que os dados já estão limpos, tratados e num formato relacional bem estruturado, no entanto, concluem que, normalmente, não podem ser analisados como estão;

- **Gestão de Bases de Dados** (*Database Management*): Uma forma básica de *Data Profiling* é a análise de colunas individuais numa determinada tabela (Abedjan et al., 2015). Tal como já referido, existem metadados que correspondem a estatísticas mais simples, sendo usados para executar outras etapas de otimização de bases de dados (Mannino, Chu, & Sager, 1988). Contudo, existem técnicas mais avançadas que utilizam histogramas, dependências funcionais e combinações de colunas para otimização de consultas (Poosala, Haas, Ioannidis, & Shekita, 1996);
- **Engenharia Inversa de Bases de Dados** (*Database Reverse Engineering*): Identifica os atributos e as relações das bases de dados, bem como a semântica, tais como chaves estrangeiras e cardinalidades. O resultado deste caso pode ser um modelo entidade/relacionamento ou um esquema lógico para auxiliar na manutenção, integração e consulta de bases de dados (Abedjan et al., 2015);
- **Integração de Dados** (*Data Integration*): A grande abundância de dados disponíveis e o desejo de integrá-los com dados locais amplificam a necessidade da sua exploração, pois os conjuntos de dados que chegam são muitas vezes desconhecidos (Abedjan et al., 2015). Um caso concreto de *Data Profiling* é o *schema matching*, isto é, encontrar correspondências semanticamente corretas entre os elementos de dois esquemas (Euzenat & Shvaiko, 2013). Assim, serão obtidas certas caraterísticas dos atributos como o tipo de dados, comprimento médio e padrões para se fazer a comparação entre elas e alinhar os atributos com os seus correspondentes (Madhavan, Bernstein, & Rahm, 2001; Nauman, Ho, Tian, Haas, & Megiddo, 2002). Além de explorar fontes individuais o *Data Profiling* pode evidenciar como dois conjuntos de dados podem ser integrados, por exemplo, nas dependências de inclusão entre tabelas de diferentes fontes, sugerindo como podem ser combinadas com uma operação de junção (Abedjan et al., 2015);
- **Qualidade de Dados** (*Data Quality*): O *Data Profiling*, como ferramenta de avaliação e monitorização da qualidade dos conjuntos de dados, permite desmascarar erros e incoerências, tais como formatação inconsistente, valores ausentes, *outliers* ou até registos que não se enquadram nas restrições estabelecidas (Abedjan et al., 2015; Kandel, Parikh, Paepcke, Hellerstein, & Heer, 2012; Pipino et al., 2002);

- ***Big Data Analytics***: *Big Data*, com seu elevado volume, elevada velocidade e elevada variedade, diz respeito a dados que não podem ser geridos com técnicas tradicionais (Laney, 2001). Portanto, se os dados apenas estiverem num repositório é improvável conseguir utilizá-los, pelo que é útil conhecer as propriedades dos dados e perceber o que se deve integrar e como se deve integrar, para além de avaliar a sua qualidade (Abedjan et al., 2015; Labrinidis et al., 2012).

Por fim, tal como citado por Abedjan et al. (2015), a Governança de Dados e a gestão do seu ciclo de vida estão a tornar-se cada vez mais relevantes para as organizações. Pelo que, mais uma vez, o *Data Profiling* pode auxiliar a determinar que ações devem ser executadas e em que dados.

Abedjan et al. (2015) apresentam um conjunto de tarefas de *Data Profiling*, tal como representado na Figura 13. Como se constata, os autores optaram por atribuir uma classe de *profiling* especifica para deteção de dependências (*dependency*), apesar desta poder ser incluída no *profiling* de várias colunas (*multi-column*). De seguida, as várias classes são caraterizadas.

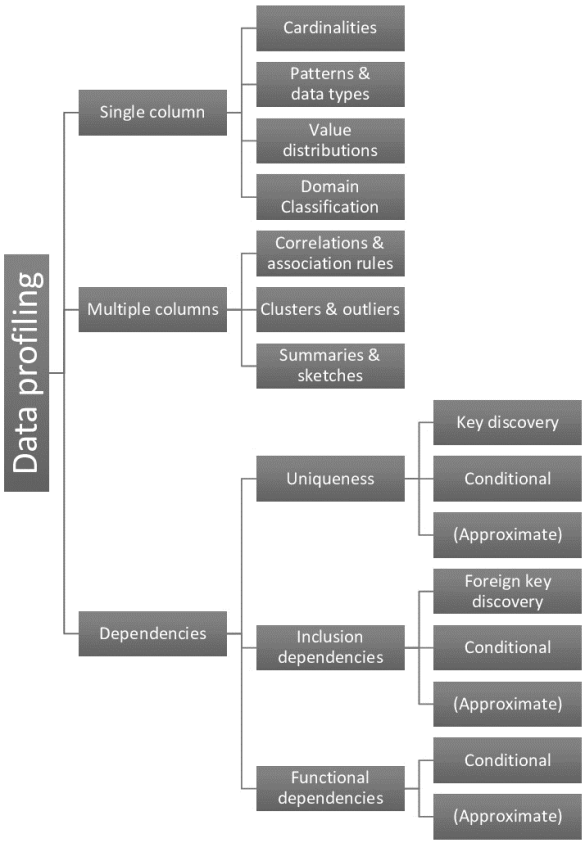

Figura 13. Classificação das tarefas de Data Profiling. Retirado de Abedjan et al. (2015).

- ***Single-column profiling***: É uma forma básica de *Data Profiling* e corresponde à análise de colunas individuais num determinado conjunto de dados. Geralmente os metadados contêm a verificação de várias contagens (***cardinalities***), desde o número de valores, número de nulos e número de valores únicos, para além da identificação dos valores máximo e mínimo (Abedjan et al., 2015).

Existem ainda técnicas mais avançadas que originam histogramas de distribuição de valores (***value distribution***), os quais armazenam frequências dentro de grupos definidos, normalmente dividindo os valores num conjunto fixo de intervalos (Raman & Hellerstein, 2001). Para além disso, é extraído o tipo de dados (*string*, *numeric* ou *date*) e são identificados padrões típicos (***types and patterns***) (Abedjan et al., 2015). Detetar padrões nos valores que podem estar incompletos (***data completeness***) é algo difícil, porém a solução passa por examinar cada coluna de cada vez e, para cada valor possível, calcular a distribuição dos outros valores do atributo (Hua & Pei, 2007);

- ***Multi-column profiling***: Abrange simultaneamente várias colunas e identifica dependências e similaridades entre colunas. Identifica relacionamentos entre valores das colunas através de padrões frequentes ou regras de associação (***correlations*** *and* ***association rules***). As abordagens de ***clustering*** segmentam os registos em grupos homogéneos com determinadas caraterísticas, pelo que os registos que não se encaixam em nenhum grupo são designados de ***outliers***. A deteção de valores *outliers* é utilizada na limpeza dos dados, pois podem indicar valores incorretos. Igualmente, os resumos (***summaries***) de dados relacionam-se ao *profiling* entre colunas (Abedjan et al., 2015);
- ***Dependencies***: A deteção de dependências entre colunas implica algumas dificuldades, na medida em que devem ser analisados conjuntos de colunas com vastos conjuntos de registos, para além de que a eventual existência de uma dependência não indica que esta seja significativa (Abedjan et al., 2015). Uma vez que as combinações de colunas únicas (***uniqueness***) foram descobertas, isto é, conjuntos de colunas cujos valores identificam exclusivamente linhas, é necessário identificar a chave primária pretendida (Abedjan et al., 2015; Heise, Quiané-Ruiz, Abedjan, Jentzsch, & Naumann, 2013). Por outro lado, a descoberta de chaves estrangeiras associada à dependência de inclusão (***inclusion dependencies***) declara que todos os valores ou combinações de valores de um conjunto de colunas também aparecem noutro conjunto de colunas. A dependência funcional (***functional dependencies***) indica que os valores num conjunto de colunas determinam funcionalmente o valor de outra coluna. Logo, as dependências funcionais permitem a normalização do esquema enquanto que as dependências de inclusão podem sugerir associações entre as diferentes fontes de dados (Abedjan et al., 2015). Dentro de cada tipo de dependência existem outros subtipos que são descritos posteriormente. As dependências parciais (***partial***) são adequadas apenas para um subconjunto de registos, exemplificando, são válidas para 95% dos registos ou para todos exceto 10 registos (Abedjan et

al., 2015; Stonebraker et al., 2013). Após detetar a dependência parcial é interessante caraterizar os registos que esta contém, de modo a encontrar uma condição que satisfaça essa dependência, ou seja, que selecione esses registos, surgindo assim as dependências condicionais (***conditional***). Por último, existem as dependências aproximadas (***approximate***) que são descobertas utilizando amostras e outras técnicas de sumarização (Abedjan et al., 2015).

Para além do que já foi referido, a abordagem da estrutura KAYAK proposta por Maccioni & Torlone (2018) faz uma distinção interessante relativamente ao *profiling* dos metadados, recolhendo metadados dentro do conjunto de dados *(intra-dataset)* e entre conjuntos de dados *(inter-dataset)*:

- Metadados ***intra-dataset*** formam o *profile* associado a cada conjunto de dados, incluindo metadados descritivos, estatísticos e estruturais;
- Metadados ***inter-dataset*** especificam relacionamentos entre conjuntos de dados diferentes ou entre os seus atributos. Incluem restrições de integridade, como por exemplo, dependências de inclusão e outras propriedades como *joinability* e *affinity*. *Joinability* mede a percentagem de valores comuns entre os atributos dos dois conjuntos de dados, enquanto que *affinity* mede a força semântica do relacionamento entre os atributos.

2.5.3 Trabalhos Relacionados

Realizada a revisão de literatura e após procura de trabalhos relacionados, são de seguida destacados os que se consideram mais relevantes para o trabalho a desenvolver nesta dissertação.

Tendo em conta que as áreas de *governance*, *tagging* e *lineage*, aplicadas e exploradas, explicitamente e diretamente ao BDW, são inexistentes, é necessário pesquisar e analisar estes conceitos no contexto mais tradicional (DW) e no contexto de *Data Lakes* (DLs), para posteriormente se estender o estudo ao contexto de BDW.

Similaridade Semântica no Processo de *Schema Matching* num *Data Warehouse*

Seguindo a abordagem proposta por Banek, Vrdoljak e Tjoa (2007), a integração do DW consiste em mapeamentos que interligam as componentes do DW que são compatíveis. Devem ser definidas funções de similaridade que comparem os nomes e as subestruturas desses elementos para tornar a integração semiautomática. Certos conflitos relacionados com o esquema podem surgir devido à utilização de nomes e/ou estruturas diferentes para descrever os mesmos dados, por exemplo, a dimensão que descreve um "Hospital" pode ter esta designação num dos componentes do DW e, noutro, ser designada "Clínica". As funções de semelhança semântica (*semantic similarity*) são utilizadas para solucionar conflitos no *schema matching* e representam o grau de relacionamento entre duas palavras,

podendo ser expostas como uma função de probabilidade. Por exemplo, para duas palavras diferentes com o mesmo significado, isto é, sinónimos, o valor da função é 1 e o valor de 0 significa que as duas palavras não estão relacionadas.

Banek et al. (2007) apresentam uma nova técnica de semelhança semântica baseada em *edge counting*, que combina as ontologias WordNet[22] e as ontologias de domínio escritas em OWL (*Web Ontology Language*)[23]. Enquanto que o WordNet é capaz de produzir similaridade semântica para qualquer par de palavras, as ontologias de domínio contêm um número muito menor de palavras, mas que descrevem a sua relação de forma mais precisa, restringindo o significado das palavras exclusivamente a esse domínio em específico e não em relação ao seu significado mais padronizado. Ou seja, o OWL fornece um vocabulário comum, uma gramática para publicar dados e uma descrição semântica dos dados.

O objetivo do processo de correspondência é determinar que componentes multidimensionais (factos e medidas, por exemplo) de um DW são mutuamente equivalentes, mas somente aqueles do mesmo tipo são compatíveis para correspondência.

O algoritmo de correspondência consiste em duas etapas (Banek et al., 2007):

1. **Comparação dos componentes multidimensionais**: Determina-se o nível de semelhança entre duas estruturas multidimensionais, comparando os seus nomes e subestruturas. A similaridade dos atributos e medidas é calculada pela similaridade semântica entre os seus nomes. Por outro lado, a similaridade de elementos complexos (dimensões, níveis de agregação e factos), pode ser calculada pela soma ponderada da semelhança dos nomes e da subestrutura;
2. **Criação de mapeamentos**: Nesta etapa são produzidos mapeamentos que indicam que elementos do DW representam a mesma informação, provenientes de diferentes fontes heterogéneas e que podem ser unidos.

Modelo para a Gestão de Metadados

Com o enorme crescimento da quantidade, variedade e velocidade de dados armazenados, surgiu o conceito de *Big Data*, tal como já abordado. Os repositórios que suportam *Big Data* podem ser conhecidos por DLs, que armazenam dados no seu formato original e por BDWs, que armazenam dados no seu formato estruturado (Terrizzano, Schwarz, Roth, & Colino, 2015). Para dar suporte aos cientistas

[22] https://wordnet.princeton.edu/

[23] https://www.w3.org/OWL/

de dados na análise dos dados, deve haver um controlo sobre os mesmos, de modo a descrever e perceber o seu conteúdo, utilizando-se para isso os metadados. Por conseguinte, os metadados podem ser explorados para detetar e descobrir conjuntos de dados, sejam eles duplicados, relacionados (isto é, atributos "*joinable*") ou *outliers* (Alserafi, Abelló, Romero, & Calders, 2016; Morton et al., 2014).

Atualmente, a descoberta de informação para identificar, localizar, integrar e restruturar os dados consome cerca de 70% do tempo de um projeto de análise de dados, sendo necessário diminuir esse valor (Terrizzano et al., 2015). Para lidar com este desafio, o trabalho apresentado por Alserafi et al. (2016) propõe um processo que consiste em catalogar o esquema dos dados existentes no DL, assim como implica a extração sistemática, gestão e exploração de metadados sobre o conteúdo dos conjuntos de dados e a sua relação por meio de *schema matching* (correspondência de esquema). A estrutura proposta integra técnicas automáticas para apoiar a descoberta do conteúdo do DL, ou seja, compreende um processo formalizado de Gestão de Metadados para impedir que o DL se torne um *swamp* de dados, por outras palavras, para evitar que seja mal gerido, não conseguindo manter a qualidade apropriada dos dados (Walker & Alrehamy, 2015).

O tradicional *schema extraction* e o *Data Profiling* envolvem a análise de dados no seu formato original para detetar padrões estruturais e distribuições estatísticas (Naumann, 2014). Atualmente, existe a necessidade de utilização de técnicas de *profiling* de alto nível, que envolve a análise de informação sobre os relacionamentos dos esquemas (tipos de atributos e dependências) e de instâncias (valores dos atributos) entre diferentes conjuntos de dados, que se define especificamente *como* ***Information Profiling***. Isso implica a análise dos metadados e do esquema que são extraídos dos dados sem qualquer tratamento, utilizando técnicas de alinhamento de ontologias (Bernstein, Madhavan, & Rahm, 2011; M. Suchanek, Abiteboul, & Senellart, 2011). Assim, estas técnicas exploram o esquema e o *data profile* de metadados para combinar diferentes conjuntos de dados, gerando *information profiling*. Um *schema profile* descreve o esquema dos conjuntos de dados, por exemplo, quantos atributos, os seus tipos de dados e os seus nomes (Abedjan et al., 2015). Os *data profiles* descrevem os valores do conjunto de dados, ou seja, as estatísticas de um atributo (Naumann, 2014). Portanto, *information profiling* explora os padrões de *Data Profiling* e de *Data Schema* (Varga, Romero, Pedersen, & Thomsen, 2014).

O conteúdo dos metadados é a representação de todos os tipos de *profiles*, pelo que o interesse é aumentar os metadados que descrevem o conteúdo informativo dos conjuntos de dados, a fim de facilitar a exploração de um DL. Várias pesquisas têm sido feitas para a extração do esquema (*schema extraction*) e do conteúdo dos metadados (*metadata content*), fornecendo uma visão geral de técnicas, algoritmos e abordagens para extrair e combinar esquemas (*schema matching*) e encontrar padrões no

conteúdo dos dados (K. Murthy et al., 2012; Touma, Romero, & Jovanovic, 2015). Existem também pesquisas para detetar relações entre dados (*cross-data relationships*) que visam encontrar conjuntos de dados similares com conceitos semelhantes (Bykau, Kiyavitskaya, Tsinaraki, & Velegrakis, 2010; Moawed, Algergawy, Sarhan, Eldosouky, & Saake, 2014). O alinhamento de ontologias e as técnicas de correspondência de esquemas (*schema matching*), que se baseiam em encontrar similaridades entre esquemas de dados e instâncias de dados, também podem ser utilizadas para integrar conjunto de dados (Bernstein et al., 2011). Isso pode ser obtido através da extração do esquema e da ontologia dos dados e, em seguida, a aplicação de técnicas de correspondência (*matching*) (Touma et al., 2015).

A lacuna atual da investigação sobre a Gestão de Metadados no DL demonstra que as técnicas disponíveis não são formalmente definidas como um processo sistemático de Governança de Dados, sendo apenas aplicáveis a DWs relacionais, para além de não manipularem o registo automático do conteúdo informativo dos conjuntos de dados presentes no DL (Alserafi et al., 2016). A estrutura proposta por Alserafi et al. (2016) baseia-se num processo automático de gestão do conteúdo de metadados que fornece uma abordagem sistemática para a Governança de Dados. Neste trabalho são identificadas as tarefas e as atividades para a gestão do conteúdo dos metadados.

Os autores, Alserafi et al. (2016), sugerem uma estrutura que é implementada tendo por base um modelo do processo BPMN (*Business Process Model and Notation*) para Gestão de Metadados, o qual se encontra ilustrado na Figura 14. Este facilita a recolha e manutenção sistemática dos metadados, ao longo da vida útil do DL.

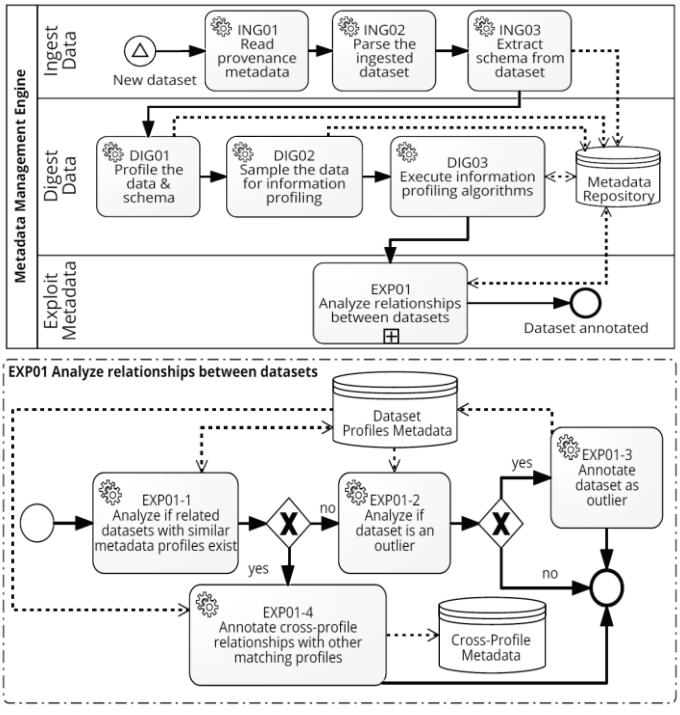

Figura 14. Modelo para a Gestão de Metadados. Retirado de Alserafi et al. (2016).

O processo *Metadata Management Engine* presente na Figura 14 está dividido em três fases principais que englobam várias atividades, tal como descrito de seguida (Alserafi et al., 2016):

1. A primeira fase corresponde à ***Data Ingestion***, consiste em analisar os dados e extrair o seu esquema, bem como armazenar os dados no DL com os metadados associados ao seu esquema (*metadata schema*) (Touma et al., 2015);
    - Quando um novo conjunto de dados é disponibilizado, o processo de armazenamento é iniciado, indicando que um novo conjunto é carregado no DL. Na atividade **ING01**, o conjunto de dados é localizado utilizando os seus metadados de origem. Em seguida, na atividade **ING02** é analisado o conjunto de dados para verificar os padrões da sua estrutura. Na atividade **ING03** o conjunto de dados é analisado para extrair e armazenar a semântica do esquema, sendo isto executado utilizando técnicas de extração de ontologias (Touma et al., 2015). Os metadados gerados são armazenados num repositório de metadados que guarda a semântica identificada;
2. ***Data Digestion*** representa a segunda fase do processo e consiste em analisar os dados que fluem para o DL, de modo a descobrir os conceitos informacionais e as caraterísticas dos dados. Esta fase inclui técnicas de *Data Profiling* e alinhamento de ontologias para extrair os *profiles* da informação. Os conjuntos são armazenados juntamente com os seus *profiles* no repositório de metadados (*metadata repository*);
    - O conjunto de dados é então explorado para extrair o conteúdo dos metadados. Começando pela atividade **DIG01** que cria o *profile* dos dados e o *profile* dos esquemas usando técnicas estatísticas simples e algoritmos de *profiling* (Naumann, 2014). A atividade **DIG02** exibe amostras de instâncias de dados para melhorar a eficiência dos algoritmos de *information profiling* na próxima atividade. Na atividade **DIG03** o conjunto de dados e os seus *profiles* são comparados a outros conjuntos de dados e aos *profiles* correspondentes, utilizando técnicas de alinhamento de ontologias. O conjunto de dados é analisado para extrair o *profile* da informação;
3. A fase três inclui ***Metadata Exploitation*** que implica a descoberta de relacionamentos entre os conjuntos de dados. Portanto, isto envolve *information profiling* que explora o conteúdo dos metadados provenientes do processo de *data digestion*, para detetar e armazenar relacionamentos entre conjuntos de dados que estão associados e que, por isso, podem ser analisados em conjunto (Varga et al., 2014). Estes são chamados de metadados *cross-dataset relationship*;

- A atividade **EXP01** deteta relacionamentos com outros conjuntos de dados, utilizando o conteúdo dos metadados armazenados no repositório (*Metadata Repository*). Tal como pode ser analisado na parte inferior da Figura 14 (EXP01 *Analyze relationships between datasets*) essa atividade inclui a subatividade **EXP01-1** que verifica se existem atributos semelhantes presentes noutros conjuntos de dados, comparando a similaridade entre os atributos dos conjuntos de dados. Se for detetada a similaridade entre atributos, o fluxo seguirá para a subatividade **EXP01-4** que armazenará os *cross-profile relationships* e irá guardá-los no *Cross-Profile Metadata*. Em EXP01-4 também são descobertos conjuntos de dados duplicados, ou seja, conjuntos com os mesmos *profiles*, que incluem o mesmo esquema e que têm o mesmo número de atributos. Caso contrário, se não houver conjuntos de dados relacionados que sejam detetados em EXP01-1, o conjunto de dados será verificado na atividade **EXP01-2** para averiguar se corresponde a um valor *outlier* (Abedjan et al., 2015). O conjunto de dados é um *outlier* se não tiver atributos correspondentes (*matching attributes*) a outros conjuntos de dados que se encontram no repositório de metadados, sendo então armazenado como um valor *outlier* em **EXP01-3**;

Constance: Sistema *Data Lake*

O *Constance*, um sistema DL para a Gestão de Metadados, é proposto por Hai, Geisler e Quix (2016). Os DLs ingerem dados brutos no seu formato original a partir de fontes de dados heterogêneas, cumprindo a sua função como repositórios de armazenamento e integração de dados, permitindo aos utilizadores realizarem a consulta e exploração dos dados. Sabendo que as informações sobre o esquema, os mapeamentos e outras restrições não são explicitamente definidas ou não são requeridas inicialmente para um DL, é importante extrair os metadados a partir das fontes de dados, isto durante a fase de ingestão. Sem metadados o DL é pouco útil, pois a estrutura e a semântica dos dados não são conhecidas, o que o transforma num *swamp* de dados (Hai et al., 2016).

O *Constance* apresenta as seguintes caraterísticas (Hai et al., 2016):

- Fornece um sistema de Gestão de Metadados estruturais e semânticos;
- Concentra-se, principalmente, em dados estruturados e semiestruturados, extraindo metadados explícitos ou implícitos;
- É composto por anotações semânticas, modelação e *record linkage* (localiza registos em conjunto de dados que se refiram à mesma entidade em diferentes fontes de dados);

- Fornece meios para enriquecer os metadados com técnicas de *schema matching* e sumarização de esquemas, para melhoria da qualidade dos dados;
- Também fornece mecanismos básicos de segurança e proveniência de dados.

A Figura 15 representa a arquitetura de *Constance*, que se divide em três camadas, conforme descrito.

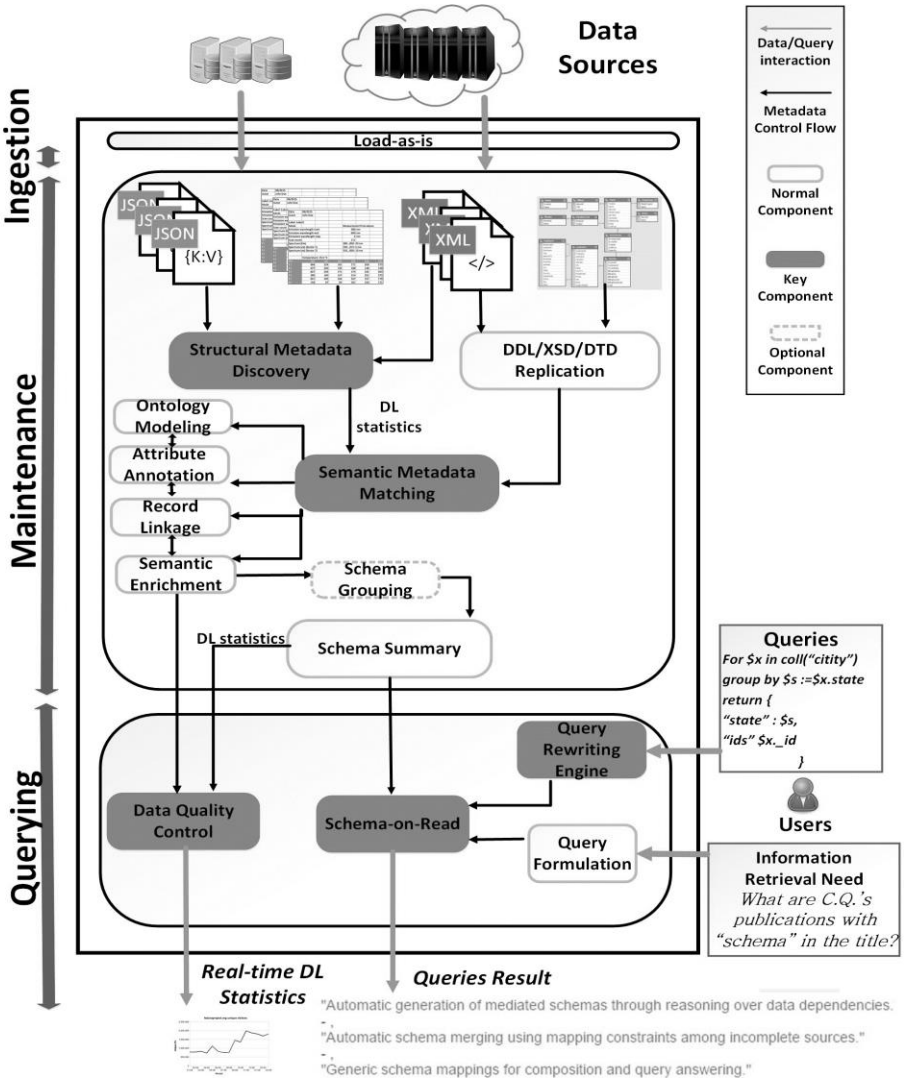

Figura 15. Arquitetura de Constance. Retirado de Hai et al. (2016).

- **Ingestão** (*Ingestion*): Implementa a interface entre as fontes de dados e o *Constance*. Importa dados de fontes heterogéneas e carrega os mesmos no *Constance* no seu formato original;
- **Manutenção** (*Maintenance*): Constituída por componentes que dinamicamente e incrementalmente extraem e resumem os metadados do DL e fornecem uma interface de consulta eficiente. O componente *Structural Metadata Discovery* assume a responsabilidade de descobrir metadados implícitos (por exemplo, tipos de entidade, tipos de relacionamento e restrições) a partir de dados semiestruturados. Outro componente presente é o *Semantic*

Metadata Matching, que consiste na modelação ontológica, anotação de atributos e *record linkage*;

- **Consulta** (*Querying*): é nesta camada que ocorre a interação principal com o utilizador. Todas as funcionalidades das camadas anteriores permitem a recuperação de informações, para posteriormente se obter respostas às consultas. O *Constance* oferece duas opções principais para a consulta, permitindo consultas utilizando uma linguagem específica ou consultas a partir de palavras-chave.

IBM *Data Governance Solutions*

Atualmente, as organizações da União Europeia encontram-se sujeitas ao cumprimento do Regulamento Geral de Proteção de Dados, assim sendo, o *lineage* de dados da IBM permite ter o conhecimento sobre os ativos dos dados e a informação sobre o fluxo de dados (Wróbel, Komnata, & Rudek, 2017). Algumas instituições, principalmente as financeiras, são bastante regulamentadas, devido a esse facto a IBM desenvolveu um fluxo de trabalho para a administração diária do processo de Governança de Dados (Soares, 2013b).

O *IBM InfoSphere Information Governance Catalog* descreve o *lineage* dos dados, tendo em conta os sistemas de origem. Conhecer a origem dos dados e o que acontece quando eles se movem ao longo de vários sistemas é fundamental para estabelecer a confiança na integridade e na qualidade desses dados. A validação dos dados e a rastreabilidade são um fator crítico no cumprimento de regulamentações que exigem que as empresas confirmem a veracidade dos seus dados (Soares, 2013b; Wróbel et al., 2017).

As iniciativas de governança abordam as preocupações da organização relativamente à qualidade e à confiabilidade dos dados, em particular, respondem a algumas questões tais como: "Qual a origem dos dados?"; "Para onde fluem os dados?"; "Os dados estão atualizados?"; "Quem tem acesso esses dados?"; "Os dados são confiáveis?" (Soares, 2013b; Wróbel et al., 2017).

Para construir o fluxo dos metadados a IBM desenvolveu e implementou a seguinte abordagem (Soares, 2013b; Wróbel et al., 2017):

- Recolher metadados das fontes de origem, como bases de dados ou qualquer objeto que fornece metadados e participe no fluxo de dados;
- Normalizar os metadados recebidos;
- Corresponder os metadados recebidos com os objetos existentes;
- Produzir algoritmos de interação entre dados para os comparar com base em múltiplos critérios;

- Enriquecer os relacionamentos entre os conjuntos de dados através de ações adicionais baseadas no conhecimento do utilizador;
- Validar os relacionamentos entre os conjuntos;
- Atualizar constantemente os relacionamentos através da monitorização de mudanças nos dados.

Um exemplo de tal fluxo referido encontra-se exposto na Figura 16, que ilustra um dos fluxos que é possível representar no IBM *InfoSphere Information Governance Catalog*.

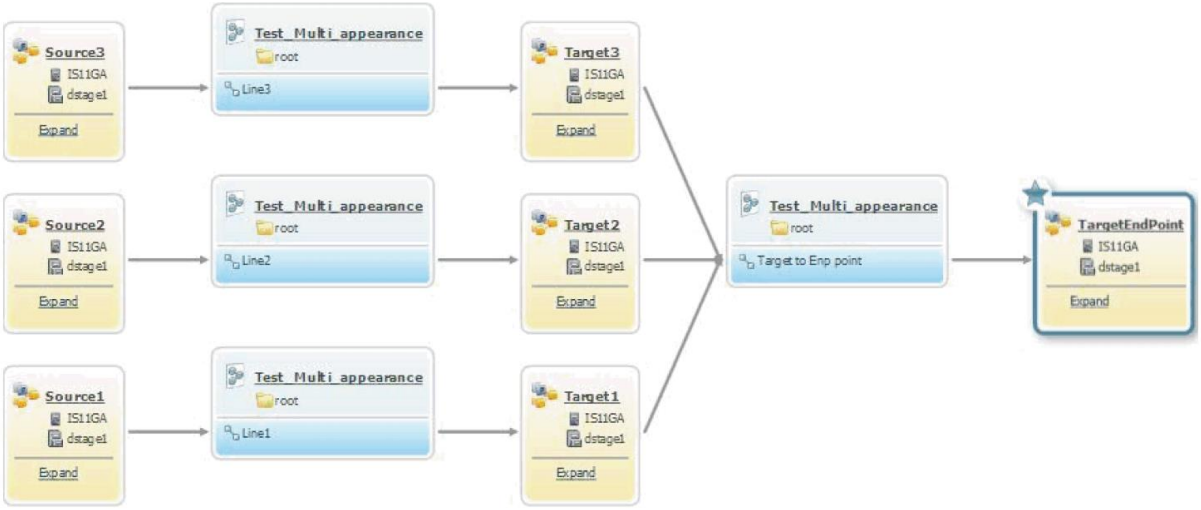

Figura 16. Data Lineage no IBM InfoSphere Information Governance Catalog. Retirado de Wróbel et al. (2017).

A ferramenta permite às organizações ter os dados catalogados num repositório central que identifica relacionamentos entre os objetos dos dados, termos de negócio e metadados. Esse repositório fornece às empresas um método consistente de rastrear a utilização e a qualidade dos dados, a fim de atender aos requisitos especificados por determinado setor. O *Information Governance Catalog* (IGC) é utilizado para verificar como se define e governa a terminologia do negócio, relacionada à gestão de processos. Assim, a utilização do IGC permite validar e aumentar a confiança nos relatórios de BI. Tal como se constata na Figura 16, este catálogo é utilizado para se verificar que fonte de dados alimenta o processo e também para se saber que relatórios e outros repositórios são alimentados pelo *output* do processo. Uma caraterística crucial para a governança é a capacidade de rastrear de onde vêm os dados, como foram obtidos e como são utilizados, portanto, o IGC proporciona essas funcionalidades. É possível constatar a presença de vários objetos envolvidos no *lineage* ilustrado na Figura 16, tal como as transformações aos dados, sendo possível expandir as caixas existentes para analisar com maior detalhe a lógica do negócio e os mapeamentos que estão a ser aplicados (Soares, 2013b).

Rastreio de Dados e Semântica de Negócio num *Data Warehouse*

Para Tomingas, Järv e Tammet (2018) a integração da terminologia do negócio com os recursos técnicos é um desafio para as organizações, especialmente em contextos de DW e BI. As múltiplas fontes de dados heterogéneas, assim como a ausência de informações sobre os esquemas, fluxos de dados complexos, domínios diferentes e padrões de nomenclatura, são apenas algumas das dificuldades técnicas sentidas na integração dos recursos num modelo semântico. Assim, algumas questões são colocadas:

- De onde vêm ou para onde vão os dados (para/de uma coluna, tabela, *view* ou *report*)?
- Como se pode mapear os nomes técnicos para um glossário comercial?
- Quando é que os dados foram carregados, atualizados ou calculados, numa determinada coluna, tabela, *view* ou *report* ?
- Quais componentes (*reports*, *queries*, carregamentos e estruturas) são afetados quando outros componentes são alterados?
- Por quem e quando certos dados, estruturas ou *reports* são utilizados?

O objetivo da investigação dos autores mencionados é desenvolver métodos para a descoberta automática das dependências entre componentes, assim como o rastreamento dos dados e os mapeamentos para o glossário do negócio. A definição de um modelo semântico integrado ou a definição de ontologias de negócio permitirá gerir os conceitos associados ao mesmo, assim como irá fornecer o significado necessário para a governança dos dados.

A integração da terminologia do negócio é baseada em duas fontes (Tomingas et al., 2018):

- Informações sobre os esquemas e as estruturas dos dados (por exemplo, base de dados, sistema ETL ou *reports* sobre as estruturas e dependências), pelo que fornece a base para a terminologia;
- Rastreamento de dados, através da semântica das *queries* capturada pelo grafo de rastreamento dos dados (*data lineage*), portanto fornecem dependências estruturais e semânticas adicionais, não disponíveis nas definições dos esquemas.

O método da integração da terminologia do negócio é ilustrado na Figura 17 com duas fontes de dados complementares, S1 e S2, que são integradas num modelo semântico M e que podem ser materializadas como uma ontologia de negócio aprendida, O1, ou combinadas como uma ontologia de negócio importada, O2.

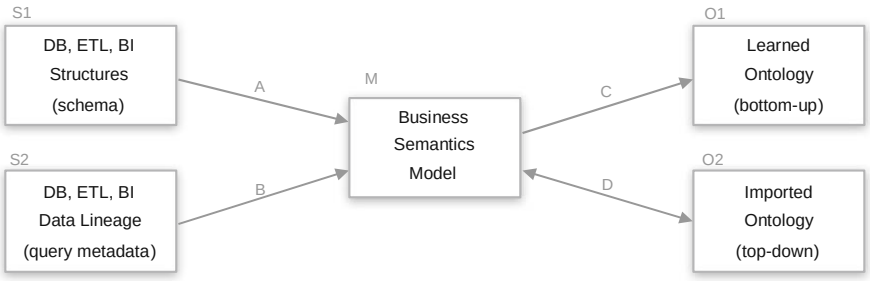

Figura 17. Integração da Semântica do Negócio. Retirado de Tomingas et al. (2018).

O método de integração semântica do negócio captura e combina informações de várias origens diferentes. A fonte de dados **S1** contém informações sobre o esquema de várias bases de dados, a transformação de dados ou sistemas de *reporting*, que não estão necessariamente interligados tecnicamente uns aos outros. A fonte de dados **S2** contém metadados extraídos de várias integrações de dados e sistemas de *queries*, armazenados num grafo de dependência, sendo utilizada como uma fonte de dados adicional para interligar os recursos em S1.

Os itens dos dados de S1 e S2 são transformados, através de um conjunto de técnicas, e integrados num modelo de semântica de negócio, **M**. O modelo M pode ser materializado ou exportado como uma ontologia de negócios aprendida, **O1** (abordagem *bottom-up*), ou seja, geração de ontologias de negócio com a estrutura interna e anotações de ativos ou a correspondência e combinação baseada nos termos e na estrutura da ontologia de negócio externa e importada, **O2** (abordagem *top-down*), adicionando anotações de ativos e/ou novos candidatos a conceitos de negócio.

Este método permite a integração de diferentes origens e estruturas de dados numa única ontologia de negócio, o que geralmente é o caso de uso para ambientes DW e BI com várias fontes e destinos.

Os trabalhos anteriormente apresentados dizem respeito a abordagens que se baseiam nos contextos de DWs e DLs, pelo que apresentam sistemas e conceitos que permitem a governança e o rastreio (*lineage*) dos dados existentes nesses ambientes. O contributo destes estudos para a presente dissertação é perceber algumas das questões que são necessárias responder para se atingir a eficiência na governança dos dados. Assim, como o objetivo deste trabalho é desenvolver um sistema para a catalogação e governança de dados de um BDW, é necessário definir-se um conjunto de metadados que respondam às questões que são colocadas. Esses metadados deverão ser definidos para as várias componentes que o BDW integra, tais como bases de dados, tabelas, colunas, entre outras. Logo, o grafo que suporta o sistema reunirá, entre outras, as caraterísticas dessas componentes (tais como, tipo e nome), as suas estatísticas de qualidade (por exemplo, número de nulos), as permissões associadas, até à forma como estas estão relacionadas (similaridade).

3. ENQUADRAMENTO TECNOLÓGICO

Neste capítulo será apresentado o ecossistema Hadoop, assim como as suas principais componentes e algumas das tecnologias que inclui. Posteriormente, expõe-se as caraterísticas do Hive, bem como a sua arquitetura. De seguida, é apresentada com maior detalhe a tecnologia Atlas, assim como é descrita a base de dados incorporada na sua arquitetura.

3.1 Ecossistema Hadoop

O Hadoop[24] consiste numa plataforma *open-source* que utiliza *commodity hardware*, sendo mais célere, pois possui escalabilidade, robustez, flexibilidade e processamento paralelo, de modo a lidar com as características que os dados apresentam em contextos de *Big Data* (Goss & Veeramuthu, 2013; Krishnan, 2013). Portanto, é normalmente reconhecido como um sistema de *Big Data* que satisfaz os seus requisitos de armazenamento e processamento (Clegg, 2015).

É uma estrutura de *software* baseada em Java para gestão e processamento de dados, pelo que contém o seu próprio sistema de ficheiros distribuído, o HDFS (*Hadoop Distributed File System*) e utiliza o modelo de programação MapReduce (Fan, Han, & Liu, 2014).

O Hadoop é inspirado no sistema de ficheiros da Google (GFS – *Google File System*) e no seu paradigma de programação MapReduce, pelo que foi concebido com a capacidade de escalar de pequenos servidores para grandes conjuntos de máquinas, garantindo uma elevada disponibilidade. É responsável por executar um conjunto de tarefas, por distribuir as tarefas em tarefas mais pequenas (com cargas de trabalho menores) e por armazenar os dados de forma paralela e distribuída (Zikopoulos & Eaton, 2011). Portanto, o Hadoop é considerado um ambiente de computação construído sobre um sistema de ficheiros que oferece uma forma de paralelizar e executar programas num *cluster* de máquinas distribuído, concebido para operações que envolvem dados em larga escala (Krishnan, 2013; Zikopoulos & Eaton, 2011).

Holmes (2012) apresenta uma arquitetura de alto nível para o Hadoop, estando esta representada na Figura 18. Este ecossistema é caraterizado por apresentar uma arquitetura distribuída que consiste num sistema de ficheiros distribuído (HDFS) para o armazenamento, e MapReduce para as capacidades computacionais. Para lidar com o volume de dados, o Hadoop fornece caraterísticas de

[24] http://hadoop.apache.org/

particionamento de dados e de computação paralela, através dos *slave nodes*. Assim sendo, o HDFS do *master node* é responsável por particionar o armazenamento ao longo dos *slave nodes*, assim como por manter um registo da localização dos ficheiros através de metadados. O MapReduce do *master node* é responsável por organizar e agendar o trabalho computacional pelos vários *slave nodes*.

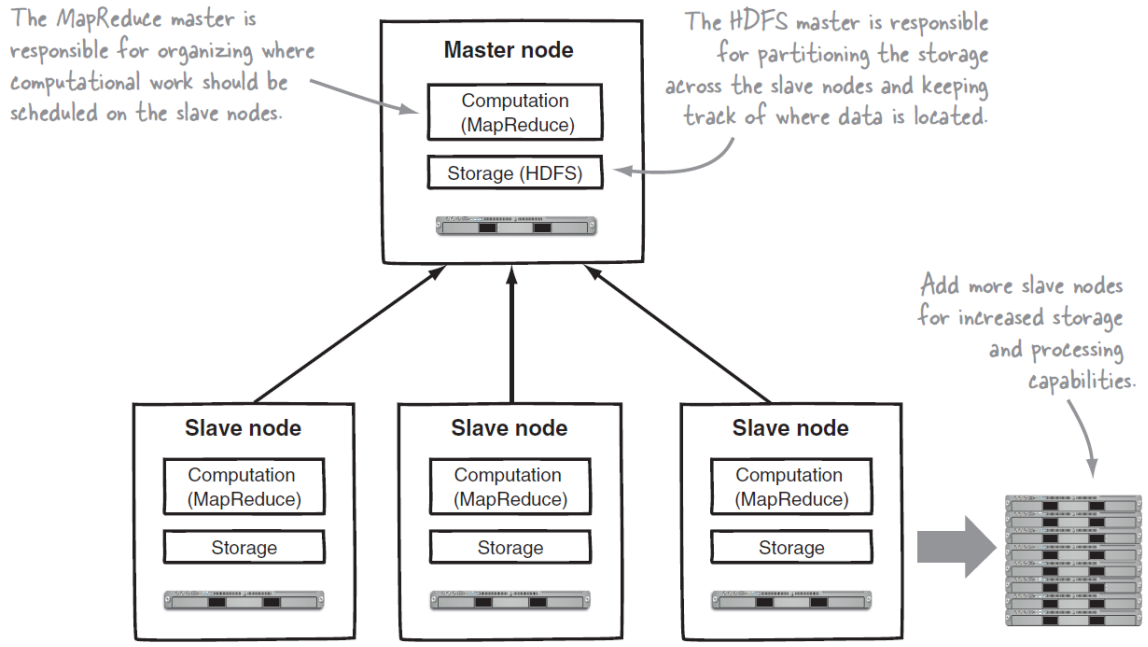

Figura 18. Arquitetura do Hadoop. Retirado de Holmes (2012).

Segundo Gupta (2015), as especificidades do Hadoop envolvem:

- **Acessibilidade**: O Hadoop é executado em grandes *clusters* de máquinas comuns, pelo que ultrapassa as barreiras da distância e fornece um acesso fácil a todos os sistemas;
- **Robustez**: Pode facilmente superar problemas no funcionamento das máquinas, devido ao facto de ser projetado com o pressuposto de possuir tolerância a falhas de *hardware*;
- **Escalabilidade**: Linearmente escalável de forma lidar com grandes quantidades de dados, adicionando mais nós ao *cluster*;
- **Simplicidade**: Consiste na escrita rápida e eficiente de programas paralelos, possibilitando aos programadores a vantagem de utilizar qualquer linguagem, por exemplo, Java ou Python;
- **Custo-benefício**: Económico, pois utiliza *commodity hardware* e servidores de baixo custo.

No ecossistema do Hadoop existem vários componentes, onde cada um detém diferentes responsabilidades, sendo capazes de armazenar e processar grandes quantidades de dados (Thusoo et al., 2010).

Os módulos principais que o Hadoop apresenta são os seguintes (Hadoop, 2018; Krishnan, 2013):

- **Hadoop Common**: Contém um conjunto de serviços comuns e a estrutura base que dá suporte às demais tecnologias do Hadoop;
- **Hadoop Distributed File System**: Permite o armazenamento e a transferência de grandes conjuntos de dados a partir de máquinas de baixo custo. Possui mecanismos que o caracterizam como um sistema altamente tolerante a falhas;
- **Hadoop YARN** (*Yet Another Resource Negotiator*): É uma estrutura que permite o agendamento de tarefas e a gestão de recursos do *cluster*;
- **Hadoop MapReduce**: Sistema baseado em YARN para o processamento paralelo de grandes conjuntos de dados. Abstrai toda a computação paralela em apenas duas funções: Map e Reduce.

Além destes componentes, existem outras tecnologias no ecossistema Hadoop que lhe adicionam certas funcionalidades, algumas delas são referidas de seguida (Bhardwaj, Vanraj, Kumar, Narayan, & Kumar, 2015; Hadoop, 2018; Krishnan, 2013):

- **Ambari**[25]: Interface *web* para o suporte, gestão e monitorização de outros módulos do Hadoop como, por exemplo, HDFS, MapReduce, Hive, HBase, ZooKeeper, entre outros;
- **Cassandra**[26]: Base de dados escalável e de elevada disponibilidade, sem pontos de falha;
- **Chukwa**[27]: Sistema de recolha de dados para gerir sistemas distribuídos;
- **HBase**[28]: Base de dados distribuída, escalável e orientada a colunas, sendo estas colunas organizadas em famílias de colunas. Suporta o armazenamento de grandes quantidades de dados em tabelas, possibilitando um acesso rápido e não sequencial aos dados, permitindo ainda a realização de *queries* e de transações;

[25] https://ambari.apache.org/

[26] http://cassandra.apache.org/

[27] http://chukwa.apache.org/

[28] https://hbase.apache.org/

- **Hive**[29]: DW distribuído que gere grandes quantidades de dados armazenados no HDFS e que permite a realização de consultas aos mesmos. As suas tabelas são uma abstração do HDFS, o que torna mais fácil o carregamento dos dados;
- **Spark**[30]: Fornece um modelo de programação simples que suporta, por exemplo, ETL, *machine learning* e processamento em *streaming*;
- **ZooKeeper**[31]: Serviço de coordenação centralizado e de alto desempenho para aplicações distribuídas.

Contudo, existem muitas outras tecnologias que se encontram incluídas no Hadoop e que são significativamente relevantes em contextos de *Big Data*, mas que não foram citadas devido ao facto de serem vastas e de não serem utilizadas neste trabalho.

3.2 Hive

O Apache Hive é uma infraestrutura de DW, construída sobre o Hadoop, que facilita a consulta e a gestão de grandes conjuntos de dados em armazenamento distribuído. O Hive fornece um mecanismo para mapear a estrutura de dados na camada de armazenamento e para consultar dados. Utiliza para as consultas uma linguagem baseada no SQL, o HiveQL, que é traduzida pelo mecanismo de execução MapReduce. Para cada base de dados, as tabelas são criadas, sendo cada tabela armazenada numa diretoria no HDFS (Cassavia et al., 2014; Thusoo et al., 2010).

O modelo de dados do Hive fornece uma estrutura em tabelas de alto nível sobre o HDFS, suportando três tipos de estruturas de dados (Krishnan, 2013; M. Y. Santos & Costa, 2016b; Thusoo et al., 2010):

- **Tabelas**: São estruturas comuns, constituídas por colunas e linhas que são armazenadas numa diretoria no HDFS, podendo ser filtradas, projetadas e unidas;
- **Partições**: São definidas sobre as tabelas para dividir horizontalmente os dados e melhorar o desempenho no processamento de *queries*. As partições são armazenadas dentro da diretoria de uma tabela, sendo possível ter várias partições associadas a uma tabela;

[29] https://hive.apache.org/

[30] https://spark.apache.org/

[31] https://zookeeper.apache.org/

- ***Buckets***: Podem ser associados a tabelas ou partições, sendo armazenados num ficheiro na diretoria da partição ou da tabela, dependendo do facto de a tabela ser particionada ou não. São utilizados como uma técnica para agrupar grandes conjuntos de dados verticalmente, agrupando-os por determinado atributo, para além de permitir otimizar o desempenho das *queries*. Quando uma tabela é criada, o utilizador pode especificar o número de *buckets* necessários e a coluna que será usada para agrupar os dados.

O Hive suporta a maior parte dos tipos de dados primitivos: inteiros, *floats*, *doubles*, *strings*, assim como suporta tipos complexos sejam eles mapas, listas ou estruturas (Thusoo et al., 2010).

Os componentes principais presentes na arquitetura do Hive encontram-se ilustrados na Figura 19 e são descritos posteriormente (Hive, 2018; Krishnan, 2013; Thusoo et al., 2010).

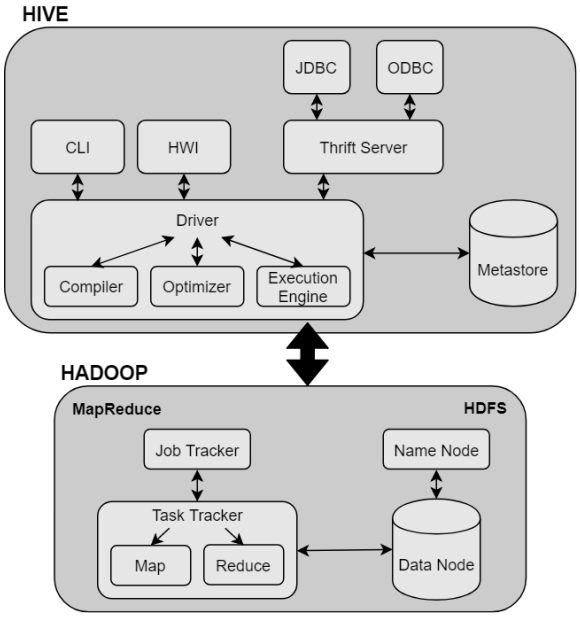

Figura 19. Arquitetura do Hive. Adaptado de Hive (2018), Krishnan (2013) e Thusoo et al. (2010).

- ***Metastore***: Armazena o catálogo do sistema e os metadados sobre as tabelas, tais como o seu tamanho, as suas partições, os esquemas, as colunas, os seus tipos, entre outros. Esta informação pode ser consultada ou modificada usando uma interface *thrift* e como resultado pode ser acedida por clientes que se baseiam em diferentes linguagens de programação. Esta informação pode ser utilizada para a otimização de *queries*;
- ***Driver***: Gere os detalhes da sessão, como é assegurada, as suas estatísticas e gere o ciclo de vida de uma instrução HiveQL, tal como esta é processada através do Hive;
    - ***Compiler***: Componente que compila o HiveQL, decompondo a *query* e realizando a análise semântica da mesma, e produz um plano de execução de acordo com os dados do *metastore*;

- ***Optimizer***: componente que otimiza o plano criado pelo *compiler*;
- ***Execution Engine***: processa e executa o plano produzido pelo compilador de acordo com uma ordem de dependência. Gere as dependências entre as diferentes fases do plano e executa essas fases nos componentes apropriados do sistema. Assim, gere todas as interações entre o compilador e o Hadoop;

- ***Thrift server***: fornece uma interface para os serviços de cruzamento de linguagens (*cross-language*), pelo que assim um servidor escrito numa certa linguagem consegue suportar clientes baseados noutras linguagens;
- **JDBC/ODBC**: APIs para integração do Hive com outras aplicações;
- **CLI (*Comand Line Interface*)** e **HWI (*Hive Web Interface*)**: Dizem respeito a duas interfaces do cliente que permitem a comunicação com o HDFS através do *driver*. A interface CLI permite a execução da linha de comandos e a interface HWI é uma interface *web*.

A partir da Figura 19 é possível perceber os fluxos de dados no Hive. Inicialmente o utilizador executa comandos HiveQL, por meio das interfaces. Assim, já no driver, envia-se a *query* para o *compiler* que extrai os metadados do *Metastore* e produz um plano de execução, de seguida, o plano é otimizado no *optimizer*, sendo depois enviado para o *execution engine* que traduz o plano em múltiplas fases, executando-as nos componentes apropriados do sistema. É executado no *Job Tracker*, se inclui tarefas de MapReduce, ou no *Metastore*, se envolve operações aos metadados, ou, ainda, no *Name Node* se inclui operações diretas ao HDFS (Hive, 2018; Krishnan, 2013; Thusoo et al., 2010).

3.3 Atlas

O Atlas é um conjunto escalável e extensível de serviços de Governança de Dados, possibilitando o cumprimento eficiente dos requisitos de conformidade do Hadoop e permitindo a integração com os dados organizacionais (Hortonworks, 2017). O Apache Atlas[32] inclui capacidades de Gestão de Metadados e de governança para as organizações criarem um catálogo dos seus ativos de dados e para procederem à sua classificação, fornecendo suporte aos cientistas e analistas de dados.

Assim, o Atlas disponibiliza (Atlas, 2018; Hortonworks, 2017):

- Tipos predefinidos para os metadados, permitindo ainda definir novos tipos;

[32] https://atlas.apache.org/

- Uma interface para visualizar o *lineage* dos dados à medida que estes se movem através dos vários processos, mantendo um histórico das fontes de dados e da forma com os dados foram gerados;
- Capacidade de criar dinamicamente as classificações sobre os dados, por exemplo classificação da qualidade dos dados;
- Propagação das classificações via *lineage*, garantindo que, de forma automática, estas seguem os dados à medida que estes passam por vários processos;
- Uma interface intuitiva para pesquisar entidades por tipo, classificação, valor do atributo, utilizando o SQL como linguagem de consulta;
- Segurança no acesso aos metadados, permitindo controlar o acesso a instâncias e a operações como adicionar, atualizar ou remover;
- Permuta de metadados com outras ferramentas.

O Atlas é constituído por um conjunto de componentes que podem ser agrupados em determinadas categorias. Assim, a arquitetura de alto nível do Atlas encontra-se ilustrada na Figura 20 (Atlas, 2018). Os componentes são de seguida descritos.

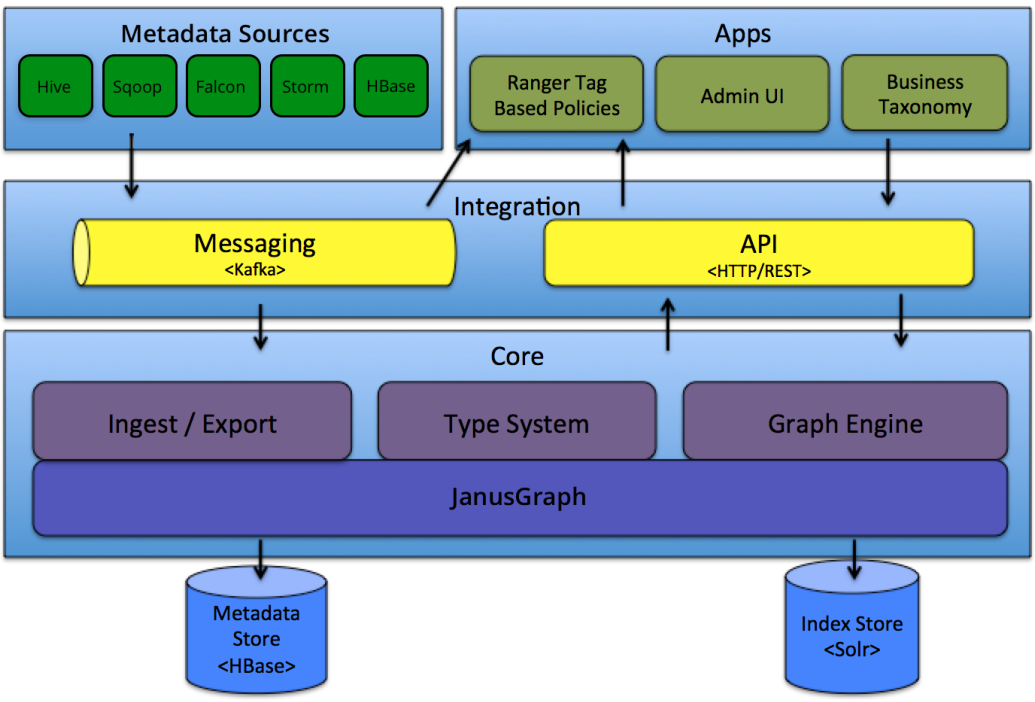

Figura 20. Arquitetura do Atlas. Retirado de Atlas (2018).

- ***Core***: A componente central do Atlas é constituída por:
  - ***Ingest/Export***: O componente *Ingest* permite que os metadados sejam adicionados ao Atlas. Por outro lado, o componente *Export* expõe as alterações dos metadados

detetadas pelo Atlas. Desta forma, as alterações dos metadados podem ser verificadas em tempo real através de eventos;

- ***Type System***: O Atlas permite que seja definido um modelo para os objetos dos metadados que se deseja gerir. O modelo é composto por "tipos" e pelas instâncias dos "tipos" que são chamadas de "entidades", representando os objetos dos metadados que são geridos. Portanto, o *Type System* permite definir e gerir os tipos e as entidades, assim, todos os objetos dos metadados geridos pelo Atlas, tais como as tabelas Hive, são modelados da forma indicada. A modelação no Atlas permite que sejam definidos metadados técnicos e metadados de negócio, para além de permitir definir relacionamentos entre os dois;
- ***Graph Engine***: Internamente o Atlas utiliza um modelo de grafos para gerir os objetos dos metadados. Esta abordagem fornece grande flexibilidade e permite a manipulação dos relacionamentos entre os objetos dos metadados. O componente *Graph Engine* é responsável pela tradução entre os tipos e as entidades do sistema e o modelo de grafos subjacente. Este mecanismo permite também ciar índices para os objetos dos metadados para que estes possam ser procurados com relativa eficiência. Assim, o Atlas incorpora no seu sistema a base de dados de grafos JanusGraph[33] para armazenar os metadados, utilizando dois tipos de armazenamento: o *Metadata Store*, configurado como padrão para o HBase e o *Index Store*, configurado para Solr[34];

- ***Integration***: Os metadados podem ser geridos recorrendo-se a dois métodos:
  - ***API*** : Todas as funcionalidades do Atlas são apresentadas aos utilizadores finais através de uma REST (*REpresentational State Transfer*) API que permite que os tipos e as entidades sejam criados, atualizados ou excluídos. É o mecanismo responsável por consultar e descobrir os tipos e as entidades geridos pelo Atlas;
  - ***Messaging***: Os utilizadores podem optar por integra-se com o Atlas utilizando uma interface de mensagens baseada no Kafka. Esta abordagem pode ser útil para a correspondência de objetos dos metadados para o Atlas e para consumir os eventos relacionados às alterações dos metadados;

[33] https://janusgraph.org/

[34] http://lucene.apache.org/solr/

- ***Metadata Sources***: O Atlas suporta a integração com múltiplas fontes, atualmente suporta a Gestão de Metadados provenientes do HBase[35], Hive[36], Sqoop[37], Storm[38] e Kafka[39]. Existem modelos de metadados que o Atlas define nativamente para representar objetos e, por outro lado, existem componentes no Atlas que possibilitam a ingestão de objetos dos metadados dessas fontes, sejam eles provenientes em tempo real ou em *batch*;
- ***Applications***: Os metadados que são geridos no Atlas são utilizados por várias aplicações de governança:
    - ***Atlas Admin UI***: É uma aplicação baseada na *web* que permite aos administradores e cientistas de dados descobrir e anotar os metadados. É importante a existência de uma interface de pesquisa e uma linguagem semelhante à SQL para consultar os tipos e os objetos dos metadados geridos pelo Atlas. Esta componente baseia-se na utilização da REST API do Atlas;
    - ***Tag Based Policies***: O Apache Ranger[40] é uma solução para a gestão da segurança no ecossistema Hadoop, podendo ser integrada com diversas ferramentas. Ao integrarem o Atlas com o Ranger os administradores da segurança podem definir políticas orientadas por metadados, proporcionando uma governança mais eficaz;
    - ***Business Taxonomy***: Permite que os utilizadores definam um conjunto hierárquico de termos de negócio que representam o domínio do negócio e os associe às entidades dos metadados geridos pelo Atlas.

A base de dados **JanusGraph**, utilizada pelo Atlas, é baseada no modelo de grafos e apresenta uma elevada escalabilidade para dados em constante crescimento. Os modelos em grafo são apropriados quando se pretende compreender os relacionamentos entre grandes volumes de dados. Como tal, suporta o processamento de grafos significativamente extensos, ou seja, que possuem bilhões de nós e arestas e que, por isso, requerem o armazenamento e capacidades computacionais que vão muito além do que uma única máquina é capaz de suportar. Portanto, os benefícios principais associados a esta

[35] https://hbase.apache.org/

[36] https://hive.apache.org/

[37] http://sqoop.apache.org/

[38] http://storm.apache.org/

[39] https://kafka.apache.org/

[40] http://ranger.apache.org/

base de dados estão relacionados com a escalabilidade no processamento de travessias de grafos (*graph traversal*) em tempo real e no processamento de *queries* analíticas (He, 2018; JanusGraph, 2018).

Além disso, o JanusGraph apresenta mais alguns benefícios e capacidades (He, 2018; JanusGraph, 2018):

- Suporta um elevado número de transações em simultâneo e o processamento operacional de grafos;
- A capacidade transacional está de acordo com o número de máquinas no *cluster* e executa *queries* complexas em milissegundos;
- Suporta a análise global de grafos, *reporting*, ETL e o processamento em *batch* por meio do ecossistema Hadoop e de algumas das suas ferramentas, como o Spark;
- Suporta o modelo de dados de grafos exposto pelo Apache TinkerPop[41];
- Suporta a linguagem Gremlin[42] para a travessia de grafos, apresentando uma fácil integração com o servidor Gremlin;
- Fornece consultas ao nível das arestas através de índices centrados nas mesmas;
- Suporta procuras geográficas, numéricas e de texto, através de índices que aceleram e permitem consultas complexas, tais como ElasticSearch[43], Apache Solr e Apache Lucene[44];
- Suporta a integração com algumas bases de dados como o Cassandra e o HBase;
- Baseia-se nas propriedades BASE. O sistema terá que escolher entre reduzir o rendimento, ou seja, parar de responder a solicitações (mantendo a consistência) ou reduzir a precisão dos resultados, isto é, fornecer respostas com base em dados incompletos (mantendo a disponibilidade). Essa decisão deve ser baseada nos requisitos do negócio e depende das bases de dados a que se integra.

No contexto do Atlas, o JanusGraph utiliza um armazenamento baseado em grafos para guardar metadados que se encontram indexados e para facilitar os mecanismos de procura da API. Este tipo de base de dados é utilizado para representar os relacionamentos entre as fontes de dados, os conjuntos de dados que possuem, o seu significado comercial, a sua classificação em termos de qualidade e

[41] http://tinkerpop.apache.org/

[42] http://tinkerpop.apache.org/gremlin.html

[43] https://www.elastic.co/

[44] http://lucene.apache.org/

confiabilidade e ainda para saber que pessoas e processos estão a utilizar esses dados e para quais fins (He, 2018).

A arquitetura do JanusGraph, que se encontra ilustrada na Figura 21, demonstra como as aplicações podem interagir com esta base de dados, sendo possível realizar esta interação de duas formas distintas (JanusGraph, 2018):

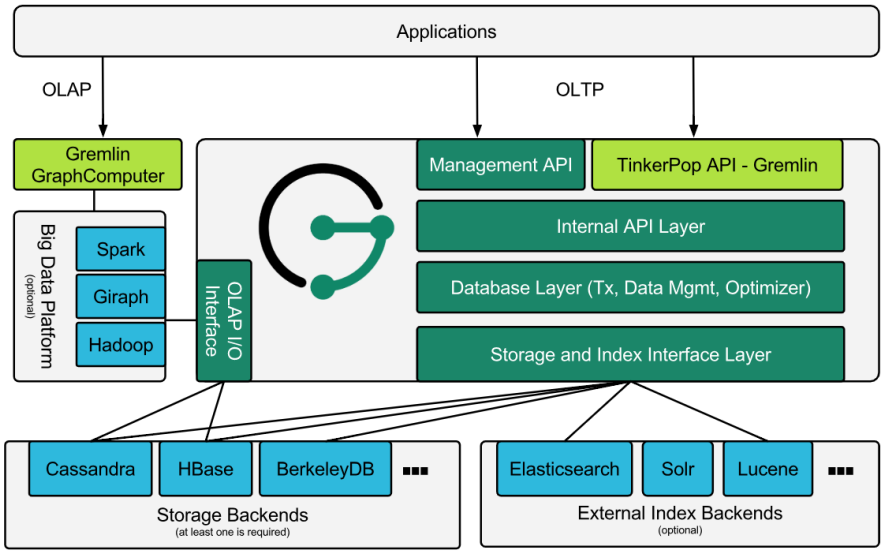

Figura 21. Arquitetura JanusGraph. Retirado de JanusGraph (2018).

- O JanusGraph pode ser diretamente incorporado com as aplicações através da execução de consultas Gremlin dentro da JVM (*Java Virtual Machine*). Assim, a execução de consultas e a gestão de transações ocorrem na mesma JVM do que da aplicação, enquanto que o armazenamento de dados que provêm das ferramentas, por exemplo, do HBase, pode ser feito de forma local ou remota;
- É possível interagir com as aplicações de forma local ou remota enviando consultas ao servidor Gremlin. O JanusGraph suporta integração com o servidor Gremlin através do Apache TinkerPop.

4. GRAFO PARA CATALOGAÇÃO E GOVERNANÇA DE DADOS DE UM *BIG DATA WAREHOUSE*

Realizada a revisão de literatura, neste capítulo irá propor-se o grafo que será incluído no sistema de catalogação e governança de dados de um BDW. Para iniciar este capítulo, primeiramente será necessário definir os requisitos a que o grafo deverá responder para permitir a eficiente governança do BDW, ou seja, de modo a que este seja definido com os metadados apropriados. Assim, na Figura 22 segue-se a classificação dos requisitos tendo em conta as vertentes que se tenciona analisar.

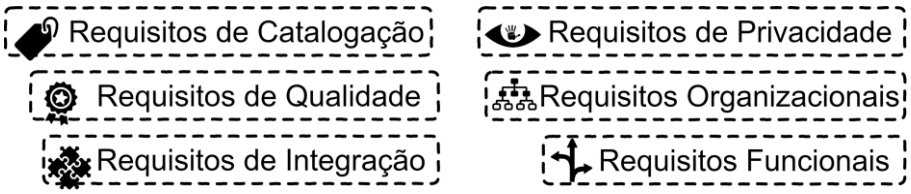

Figura 22. Requisitos do Sistema.

De seguida são explicados os requisitos e formuladas as questões que se pretende responder em cada uma das vertentes expostas anteriormente:

- **Requisitos de Catalogação**: Relacionados com o registo de metadados mais básicos relativos às propriedades dos vários componentes do BDW, desde nomes, tipos, *tags*, termos, entre outros:
    1. Como são classificadas as entidades (tabelas, colunas, entre todas as outras), a fim de organizá-las através de palavras chave?
    2. Quais termos são utilizados na organização e como as entidades estão associadas a eles?
    3. Com base no modelo BDW, qual é o tipo de cada tabela e dos atributos?
- **Requisitos de Qualidade**: Permitem avaliar a qualidade:
    4. Como os dados são avaliados em termos de qualidade?
- **Requisitos de Integração**: Detalha como os dados estão relacionados ao nível da similaridade:
    5. Como é possível integrar diferentes tabelas, considerando diferentes medidas de similaridade?
- **Requisitos de Privacidade**: Permitem avaliar os acessos aos dados:
    6. Que permissões estão definidas para determinados utilizadores e determinados dados?
- **Requisitos Organizacionais**: Apresenta os processos de negócio e os indicadores de desempenho da organização:
    7. Que processos de negócio estão incluídos no BDW e que dados envolve cada um?

8. Quais os indicadores de desempenho da organização e que dados são utilizados no seu cálculo?

- **Requisitos Funcionais**: Descreve a forma como o sistema reage a determinadas entradas e ações específicas e como se comporta em determinadas situações:
    9. Que ações são executadas sobre qualquer entidade?
    10. Como fluem os dados no BDW?

O grafo irá ser implementado na ferramenta Atlas que se adequa perfeitamente ao problema, pelo facto de ser um projeto de código aberto e altamente personalizável. Assim, com base naquilo que o Atlas já armazena na sua base de dados (JanusGraph) irá ser proposto o modelo do grafo, que se apresenta como o contributo desta dissertação. No entanto, o modelo também reunirá outros metadados que são considerados importantes armazenar, considerando os requisitos definidos no início deste capítulo, ou seja, é necessário criar e estender os tipos de metadados já existentes no Atlas.

A Figura 23 resume as várias fontes e tecnologias que interagem na modelação do catálogo de metadados do BDW que irá ser definido ao longo deste capítulo.

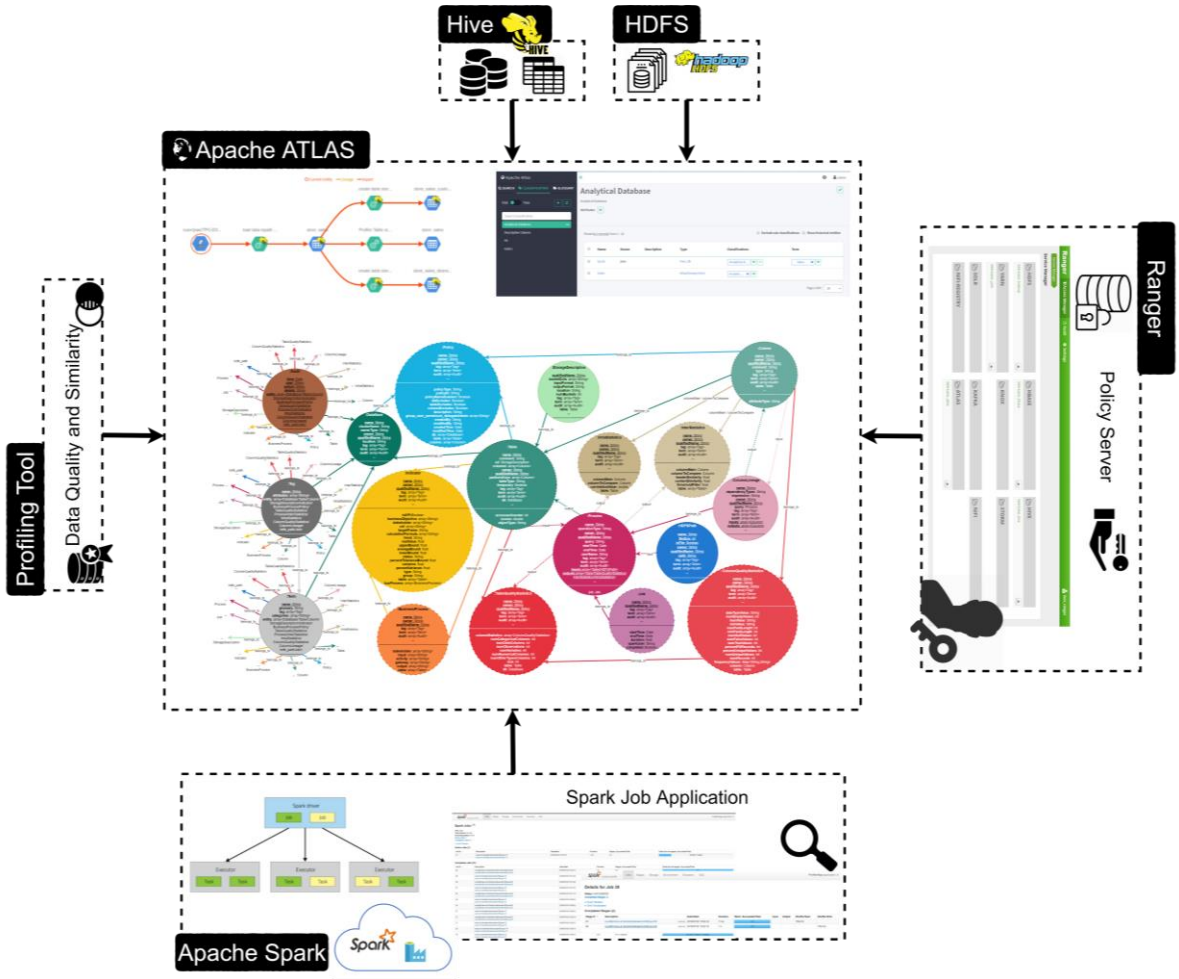

Figura 23. Tecnologias envolvidas na Modelação do Catálogo de Metadados do Big Data Warehouse.

Portanto, o grafo inclui no seu *core* o Atlas, na medida em que todos os metadados dos diferentes tipos são adicionados e editados a partir da REST API dessa ferramenta. Posteriormente, no topo da figura, encontram-se as fontes Hive e HDFS, uma vez que muitos dos tipos já presentes no Atlas provêm dessas fontes. No lado esquerdo encontra-se a ferramenta de *profiling* que foi desenvolvida noutra dissertação por Magalhães (2019b, 2019a) e basicamente permite a monitorização da qualidade dos dados, através de estatísticas de qualidade e dos algoritmos de similaridade que fornece. Do lado direito da figura está presente a tecnologia Ranger, onde a partir da sua REST API os dados relativos às políticas/permissões de acesso são recolhidos e reproduzidos para o Atlas. Por último, na parte inferior da figura, encontra-se a aplicação do Spark que interage com a ferramenta de *profiling* e é a partir dela que são obtidos os metadados sobre os *Jobs* que são despoletados e que estão associados aos processos de *profiling* ("ProfilerProcesses").

Assim, é realizado o levantamento dos termos e conceitos utilizados pelo Atlas, de forma a entender o seu comportamento e organização em termos de tipos, entidades e atributos. A Tabela 3, que se segue, apresenta uma descrição mais sucinta da organização dos tipos no Atlas[45].

Tabela 3. Organização dos Tipos no Atlas.

Type: Um tipo indica como um determinado tipo de objetos dos metadados são armazenados e acedidos.	**Metatype**: Um tipo tem um determinado metatipo.	**Basic metatypes**: Por exemplo: *Int*, *String*, *Boolean*, etc..
		Enum metatypes. Por exemplo: "USER", "ROLE" ou "GROUP". Exemplificando, quando o tipo de dados é "hive_principal_type" e sabendo que este tipo é um *enum*, o atributo só pode ter o valor das três *strings* anteriormente referidas.
		Collection metatypes: Por exemplo: *Array*, *Map*.
		Composite metatypes: Coleção de Atributos. Por exemplo: *Entity, Struct, Classification, Relationship.*
	Supertype: Um tipo pode "estender" um tipo pai, "supertipo", e em virtude disso, também incluirá os atributos definidos no supertipo.	**Referenceable**: Este tipo representa todas as entidades que podem ser procuradas utilizando um atributo exclusivo (*unique attribute*) designado ***qualifiedName***.
		Asset: Este tipo contém atributos, como por exemplo, ***name***, ***description*** e ***owner***.
		Infrastructure: Tipo comum para objetos dos metadados de infraestrutura como *clusters*, *hosts*, etc.
		DataSet: Utilizado para representar um tipo que armazena dados. No Atlas, as tabelas Hive, as tabelas Sqoop RDBMS, entre outras, são tipos que se estendem do DataSet. Espera-se que os tipos que estendem o DataSet tenham um esquema no sentido de terem um atributo que define os atributos desse conjunto de dados. Por exemplo, o atributo *columns* no *hive_table*. Também as entidades dos tipos que estendem do DataSet participam na transformação dos dados sendo essa capturada pelo Atlas através do *lineage*.

[45] http://atlas.apache.org/1.1.0/TypeSystem.html

		Process: Utilizado para representar qualquer operação de transformação de dados. Por exemplo, um processo ETL que transforma os dados brutos de uma tabela Hive noutra tabela Hive. O tipo *process* possui dois atributos específicos, ***inputs*** e ***outputs***. Tanto os *inputs* como os *outputs* são *arrays* das entidades **DataSet**. Assim, uma instância do tipo *process* pode utilizar esses *inputs* e *outputs* para capturar como o *lineage* de um DataSet evolui.
Entities: Uma "entidade" é um valor ou uma instância específica de um tipo e, portanto, representa um objeto específico dos metadados.		
Attributes: São definidos dentro de metatipos compostos, pelo que têm um nome, um determinado metatipo e mais algumas propriedades.		

Na Tabela 4 estão expostos os metadados dos tipos específicos do Hive[46], os quais o Atlas já armazena, sendo este o ponto de partida para a realização do modelo e catálogo de metadados do BDW. De notar que os atributos a negrito não são definidos aquando da criação dos tipos, mas estão por defeito presentes no Atlas mesmo que não preenchidos.

Tabela 4. Metadados do Hive presentes no Atlas.

Nome do Tipo	Descrição
hive_db	**Supertypes**: Asset
	Attributes: **owner**; ownerType; **replicatedTo**; **replicatedFrom**; **qualifiedName**; clusterName; **name**; **description**; location; parameters
	System Attributes: **createdBy**; **modifiedBy**; **createdTime**; **modifiedTime**
hive_table	**Supertypes**: DataSet
	Attributes: **owner**; temporary; lastAccessTime; aliases; **replicatedTo**; **replicatedFrom**; **qualifiedName**; columns; **description**; viewExpandedText; sd; tableType; createTime; **name**; comment; partitionKeys; parameters; db; retention; viewOriginalText
	System Attributes: **createdBy**; **modifiedBy**; **createdTime**; **modifiedTime**
hive_storagedesc	**Supertypes**: Referenceable
	Attributes: **replicatedTo**; **replicatedFrom**; **qualifiedName**; inputFormat; bucketCols; sortCols; storedAsSubDirectories; location; compressed; outputFormat; parameters; table; serdeInfo; numBuckets
	System Attributes: **createdBy**; **modifiedBy**; **createdTime**; **modifiedTime**
hive_column	**Supertypes**: DataSet
	Attributes: **owner**; **replicatedTo**; **replicatedFrom**; **qualifiedName**; **name**; **description**; comment; position; type; table
	System Attributes: **createdBy**; **modifiedBy**; **createdTime**; **modifiedTime**
hive_process	**Supertypes**: Process
	Attributes: **owner**; **outputs**; queryGraph; **replicatedTo**; **replicatedFrom**; recentQueries; **qualifiedName**; **inputs**; **description**; username; queryId; clusterName; **name**; queryText; startTime; operationType; queryPlan; endTime
	System Attributes: **createdBy**; **modifiedBy**; **createdTime**; **modifiedTime**
hive_column_lineage	**Supertypes**: Process
	Attributes: **owner**; **outputs**; expression; **replicatedTo**; **replicatedFrom**; **qualifiedName**; **inputs**; query; **name**; **description**; dependencyType
	System Attributes: **createdBy**; **modifiedBy**; **createdTime**; **modifiedTime**

[46] http://atlas.apache.org/1.1.0/Hook-Hive.html

Nome do Tipo	Descrição
hdfs_path	**Supertypes**: fs_path
	Attributes: **owner**; **modifiedTime**; **replicatedTo**; **isFile**; numberOfReplicas; **replicatedFrom**; **qualifiedName**; **description**; extendedAttributes; nameServiceId; **path**; **posixPermissions**; **createTime**; **fileSize**; clusterName; **name**; **isSymlink**; **group**
	System Attributes: **createdBy**; **modifiedBy**; **createdTime**; **modifiedTime**

Assim sendo, na Figura 24 está presente a legenda da Figura 25 e da Figura 26, permitindo interpretar o modelo do grafo proposto.

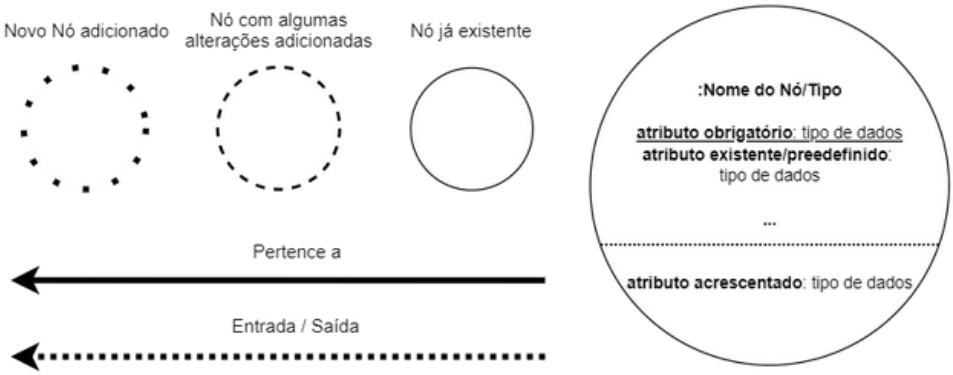

Figura 24. Legenda do Modelo de Metadados do Catálogo do Big Data Warehouse.

Na Figura 25 encontra-se o modelo do grafo de metadados pré-existente no Atlas, antes de qualquer extensão, indicando os seus nós e os relacionamentos existentes entre eles.

Quando à Figura 26 esta expõe a representação geral do modelo de metadados do catálogo do BDW já com as alterações que resultam do contributo desta dissertação. Assim, o último modelo inclui os nós já disponíveis no Atlas, alguns dos quais são estendidos com novos atributos, enquanto os outros são aqui propostos para atender aos requisitos definidos previamente. Após as figuras, segue-se a descrição mais detalhada de cada um dos seus nós.

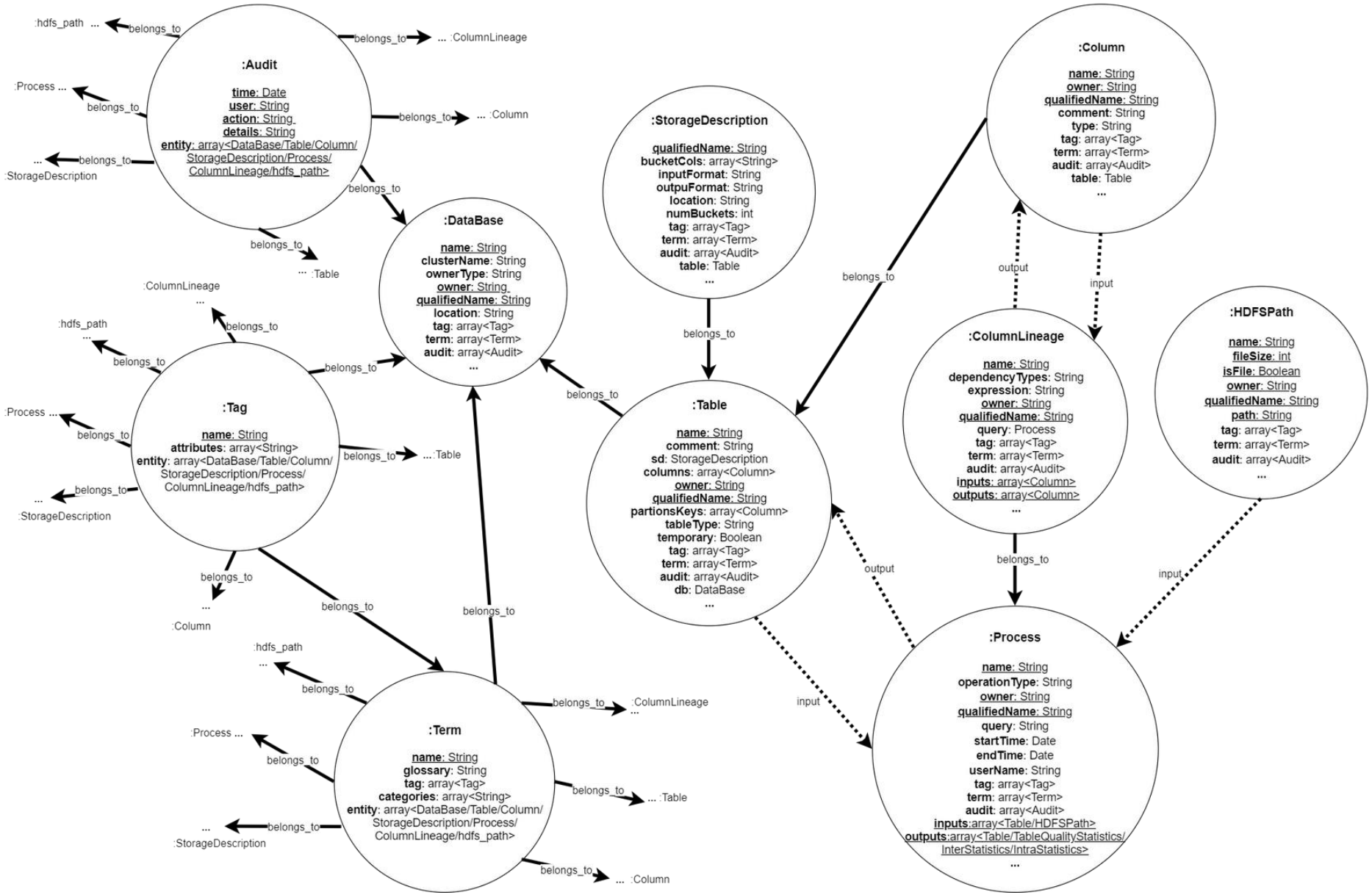

Figura 25. Modelo de Metadados do Big Data Warehouse pré-existente no Atlas.

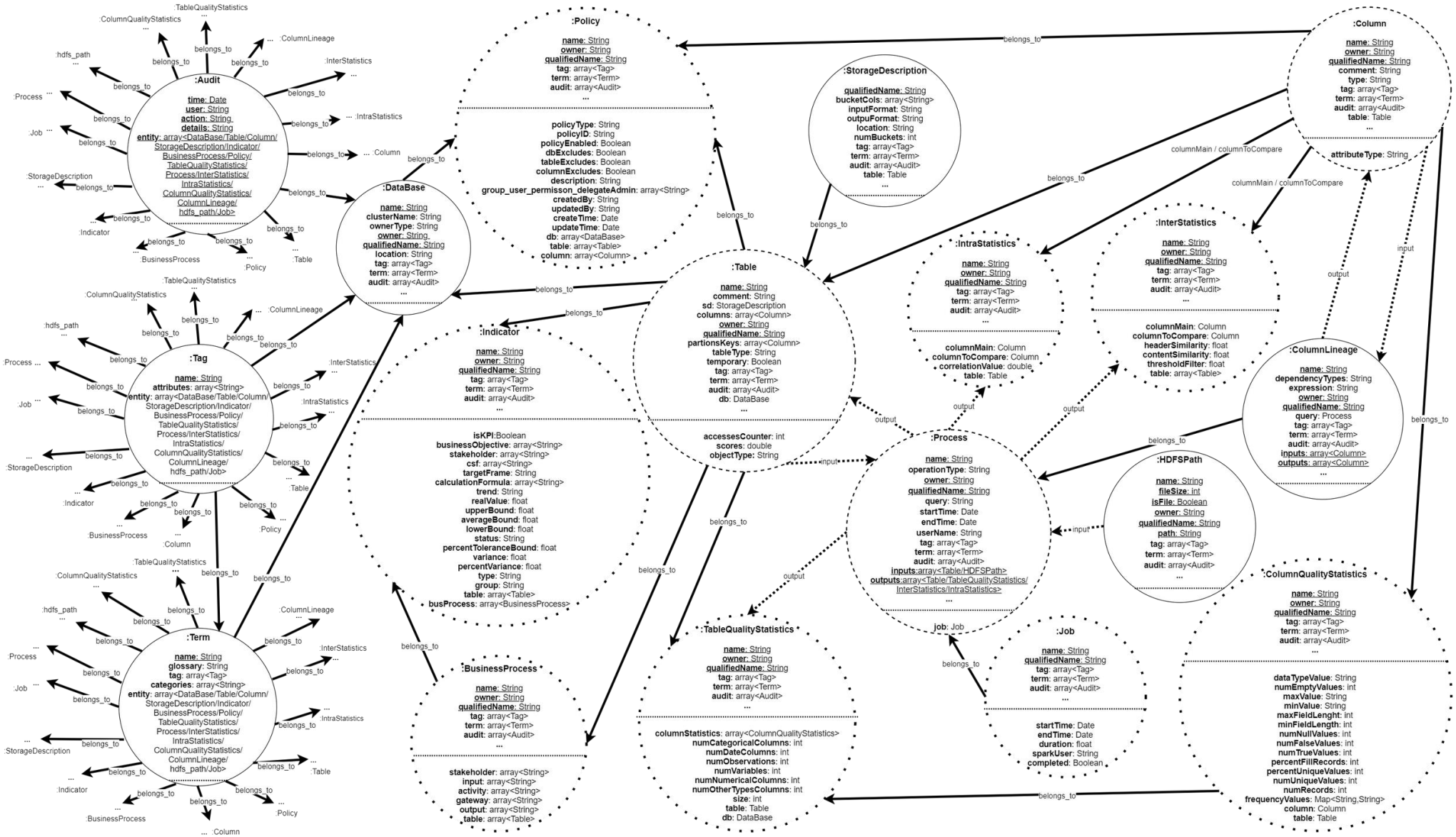

Figura 26. Modelo de Metadados do Catálogo do Big Data Warehouse.

Em cada nó do modelo encontram-se os metadados correspondentes, sendo cada um dividido em duas partes. A parte inferior de cada nó contém os metadados que foram acrescentados tendo por base os requisitos necessários e a revisão de literatura já realizada. Na parte superior encontram-se alguns dos metadados que são obtidos a partir da ferramenta Atlas e, entre esses, existem atributos que são obrigatórios (encontram-se a sublinhado). Entre outros, destacam-se os seguintes:

- *name*: Nome da base de dados, tabela, estatística, entre outros;
- *owner*: Proprietário da base de dados, do processo, da tabela, entre outros;
- *qualifiedName*: Por exemplo, nome do *storage* da tabela. Normalmente este atributo é requerido quando se insere alguma entidade, por exemplo, quando se cria um processo de negócio ("Business Process") as tabelas a associar têm que ser pesquisadas pelo seu "qualifiedName", contudo este nó será abordado com maior detalhe mais adiante.

Para além disso, no Atlas, sempre que se cria uma classificação (classification/*tag*) ou termo (*term*) é possível associá-los a cada um dos nós do grafo, ou seja, a qualquer entidade do Atlas.

Como tal, é apresentado o nó **Tag**, conforme descrito na Tabela 5. Uma *tag* corresponde a uma classificação que visa ordenar e organizar informações através de palavras chave, dentro de uma hierarquia de informações, permitindo relacionar significados semelhante. Torna-se crucial quando existem muitos dados, pois permite encontrar com relativa facilidade aquilo que se pretende. Uma classificação permite catalogar metadados de uma forma mais rápida pelo que identifica os dados em si, quer isto dizer, exemplificando, indica se os dados são sensíveis (por exemplo, *tag*: SENSITIVE), ou se são referentes à qualidade dos dados (por exemplo. *tag*: DATA_QUALITY);

Tabela 5. Nó Tag. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
attributes	array<String>	Conjunto de atributos associados à *tag* que podem ser de vários tipos.
entity	array<DataBase/Table/Column/StorageDescription/Indicator/ BusinessProcess/Policy/TableQualityStatistics/ Process/InterStatisticsIntraStatistics/ ColumnQualityStatistics/ColumnLineage/HDFSPath/Job>	Cada *tag* pode ser associada a várias entidades que passam a estar relacionadas. Ou seja, cada nó dos vários tipos pode ter uma ou várias *tags*.

Está presente ainda o nó **Term**, descrito na Tabela 6, que ilustra um termo de um glossário. Trata-se de uma palavra útil para uma organização, pelo que está relacionado com o negócio e com as suas regras. Cada termo deve ter um único *qualifiedName*, no entanto pode haver termos com o mesmo nome, mas quando assim é, não podem pertencer ao mesmo glossário.

Tabela 6. Nó Term. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
glossary	String	Glossário a que pertence o termo. Um Glossário fornece o vocabulário aos utilizadores do negócio e permite que os termos sejam categorizados e relacionados entre si para que possam ser compreendidos em diferentes contextos.
tag	array<Tag>	Os termos podem ser classificados através das *tags* e, assim, a mesma classificação é aplicada às entidades com as quais o termo está associado.
categories	array<String>	Um termo pode pertencer a zero ou mais categorias.
entity	array<DataBase/Table/Column/StorageDescription/Indicator/ BusinessProcess/Policy/TableQualityStatistics/ Process/InterStatisticsIntraStatistics/ ColumnQualityStatistics/ColumnLineage/HDFSPath/Job>	Cada termo pode ser associado a várias entidades, ou seja, cada nó dos vários tipos pode ter um ou vários termos.

A Tabela 7 expõe o nó **Audit** que representa cada ação executada sobre qualquer entidade, demonstrando, por exemplo, os detalhes da sua criação ou alteração, informações essas armazenadas no Atlas.

Tabela 7. Nó Audit. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
time	Date	Corresponde ao identificador do *audit*, que se encontra no tipo data.
user	String	Utilizador que executou determinada ação (*audit*).
action	String	Identifica a ação que foi feita, podendo ser, por exemplo, "Entity Updated", "Term Added", "Entity Created" e "Classification Added".
details	String	Descreve os detalhes do que foi efetuado.
entity	array<DataBase/Table/Column/StorageDescription/Indicator/ BusinessProcess/Policy/TableQualityStatistics/ Process/InterStatisticsIntraStatistics/ ColumnQualityStatistics/ColumnLineage/HDFSPath/Job>	Cada *audit* pode ser associado a uma entidade, ou seja, cada nó dos vários tipos pode ter um ou vários *audits*.

O nó **DataBase** do modelo contém alguns metadados relativos às bases de dados, conforme descrito na Tabela 8 que se segue. Todos esses metadados são obtidos a partir da ferramenta Atlas.

Tabela 8. Nó DataBase. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
clusterName	String	Nome do *cluster* onde se encontra alojada a base de dados.
ownerType	String	Tipo do proprietário da base de dados, por exemplo: "USER", "ROLE" ou "GROUP". O tipo de dados é hive_principal_type, sabendo que este tipo é um *enum*, o atributo só pode ter o valor das três *strings* anteriormente referidas.
location	String	Localização da base de dados.

Cada base de dados tem associada um conjunto de tabelas, assim o nó **Table** descreve as caraterísticas de cada uma, conforme descrito na Tabela 9. De entre os metadados deste nó que foram obtidos por meio do Atlas, o atributo *objectType* é adicionado, com o intuito de fornecer uma informação mais explícita do tipo de tabela que se trata e, para além desse, também são acrescentados os atributos *accessesCounter* e *scores* que fornecem um indicador de utilização da tabela.

Tabela 9. Nó Table. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
comment	String	Descrição e comentário adicional à tabela.
sd	StorageDescription	*Storage* da tabela.
columns	array<Column>	Conjunto de colunas pertencentes à tabela.
partitionKeys	array<Column>	Colunas particionadas na tabela.
tableType	String	Tipo de tabela. Por exemplo: "EXTERNAL_TABLE", "MANAGED_TABLE". As *Managed Tables* podem apenas ser alteradas através do Hive, pelo que este possui os dados destas tabelas e gere o seu ciclo de vida. Pelo contrário, o Hive não gere os dados das *External Tables*, pelo que estas são utilizadas quando os ficheiros já estão presentes em locais remotos, mantendo-se presentes, mesmo que a tabela seja excluída.
temporary	Boolean	Se a tabela é temporária retorna "true", caso contrário retorna "false".
accessesCounter	int	Contador de atualizações à tabela, ou seja, contabiliza o número de *audits*.
scores	double	Corresponde à média de utilização da tabela (escala de 0 a 5 pontos).

Metadado	Tipo de Dados	Descrição
objectType	String	Uma tabela pode corresponder a um (M. Y. Santos & Costa, 2019): • **Objeto analítico** (*Analytical Object*): é uma tabela de factos totalmente desnormalizada que contém atributos descritivos ou atributos analíticos, isto é, conjuntos de factos ou de atributos preditivos. Os atributos descritivos são divididos em *descriptive families*, enquanto que os atributos analíticos são divididos em *analytical families*, quando aplicável; • **Objeto analítico complementar** (*Complementary Analytical Object*): é uma combinação entre uma tabela de factos desnormalizada e uma dimensão que proporciona não só valor analítico, mas também valor descritivo para outros objetos analíticos. Assim, este tipo de objetos combina tabelas de factos e dimensões, aparecendo relacionados (por um ou mais atributos que permitam *joins*) com vários objetos analíticos. Por exemplo uma tabela "clientes" poderá ser deste tipo, pois pode ter atributos analíticos em relação aos clientes (tal como, média de vendas) e ao mesmo tempo ser necessário saber-se mais detalhes sobre o cliente que comprou determinados produtos; • **Objeto analítico materializado** (*Materialized Analytical Object*): é essencialmente a mesma coisa que o conceito de vista materializada no contexto de bases de dados, onde uma tabela agrega informação referente a uma ou mais tabelas, pois, normalmente, a sua computação demora demasiado tempo. Em contextos de bases de dados uma vista materializada é um objeto que contém os resultados de uma *query* (*view*). Tipicamente pode ser um conjunto ou um subconjunto de linhas e colunas de uma tabela, ou o resultado de uma função *join* ou de agregação; • **Objeto de data e tempo** (*Date and Time Object*): são essencialmente as dimensões "calendário" e "tempo" utilizadas em DWs tradicionais, que irão ser utilizadas pelos objetos analíticos que têm cariz temporal; • **Objeto espacial** (*Spatial Object*): são utilizados para padronizar informação geoespacial de alto nível, tais como países e cidades, que fornecem informação espacial aos objetos analíticos sem que essa informação esteja armazenada de forma redundante nos mesmos, podendo isto provocar por vezes inconsistências nos dados (por exemplo, o local da venda apesar de ser o mesmo local da devolução, não tem associado a mesma loja, pois possivelmente houve inconsistência na inserção de dados).
db	DataBase	Base de dados a que pertence a tabela.

Um dos metadados das tabelas é o *StorageDescription* que descreve o *storage* de cada uma, apresentado uma série de atributos. O *storage description* está relacionado com o armazenamento das tabelas, mais especificamente das *Managed Tables* (MANAGED_TABLE), fornecendo, nomeadamente, informação sobre a localização das mesmas (*path*) no repositório (*file system*). Assim, para esse efeito,

é exposto o presente nó, **StorageDescription**, tal como descrito na Tabela 10. Mais uma vez, os metadados são obtidos a partir do Atlas.

Tabela 10. StorageDescription. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
bucketCols	array<String>	Retorna o nome das colunas do *bucket*.
inputFormat	String	Formato do *input*.
outputFormat	String	Formato do *output*.
location	String	Localização do *storage*.
numBuckets	int	Número de *buckets* da tabela.
table	Table	Tabela a que se encontra associado o *storage*.

O nó **Job** surge para executar determinado processo, assim é possível, por exemplo, averiguar o seu estado e duração, conforme descrito na Tabela 11. Cada *job* é identificado pelo seu "id" que corresponde ao seu nome. Estes metadados foram obtidos a partir da aplicação Spark Job[47].

Tabela 11. Job. Baseado na aplicação Spark Job.

Metadado	Tipo de Dados	Descrição
startTime	Date	Data de criação (inicialização) do *job*.
endTime	Date	Data de finalização do *job*.
duration	float	Duração do *job* em milissegundos.
sparkUser	String	Utilizador do Spark.
completed	Boolean	Corresponde ao estado do *job*, se falhou indica "false" ou se foi bem-sucedido retorna "true".

O nó **Process** é criado para descrever através dos seus metadados um processo que ocorreu, este é baseado nas caraterísticas que é possível obter pelo Atlas, sendo estas descritas na Tabela 12. É importante realçar que cada processo pode ter um conjunto de *inputs* e *outputs* de vários tipos, desde tabelas, *hdfs paths*, entre outros, conforme as ligações de *input* e *output* representadas no modelo. O supertipo *Process* inclui o tipo de processo "hive_process". Para além disso, é criado o tipo de processo "ProfilerProcesses" que irá ter as caraterísticas de um "Process" comum, pois é esse o seu supertipo, mas o *input* que recebe pode ser, por exemplo, uma tabela (Table) e o *output* as estatísticas dessa tabela (TableQualityStatistics), sendo estas últimas explicadas posteriormente com maior detalhe. Além disso, um processo desse tipo pode ter associado um *job* específico. Esta componente de *profiling* foi elaborada

[47] https://www.cloudera.com/documentation/enterprise/5-6-x/topics/cdh_ig_spark_apps.html

em colaboração com outra dissertação realizada por Magalhães (2019b, 2019a), pois nessa é gerado o processo de *data profiling* no Spark, através dos metadados definidos na presente dissertação.

Tabela 12. Process. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
operationType	String	Tipo da operação associada ao processo. Por exemplo: "CREATETABLE_AS_SELECT", "LOAD", "CREATETABLE", "Profiler Quality Operation", entre outras.
query	String	Descreve detalhadamente o processo, o que foi criado, por exemplo.
startTime	Date	Data de criação (inicialização) do processo.
endTime	Date	Data de finalização da criação do processo.
userName	String	Nome do utilizador do processo. Por exemplo: "hive".
inputs	array<Table/HDFSPath >	*Input* que alimenta o processo e que sofre uma determinada operação.
outputs	array<Table/ TableQualityStatistics/ InterStatistics/ IntraStatistics>	Através da execução de determinada operação a um *input*, no final do processo é obtido um *output*.
job	Job	*Job* que está associado a um determinado processo. Os jobs só se encontram associados a processos de *profiling* (tipo de processo "ProfilerProcesses").

Tal como referido um dos tipos de *input* e *output* de um processo pode ser um *hdfs path*, daí a formulação do nó **HDFSPath**. O Atlas inclui os metadados que descrevem este *path*, tal como apresentado na Tabela 13.

Tabela 13. HDFSPath. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
fileSize	int	Tamanho do ficheiro.
isFile	Boolean	Se é um ficheiro retorna "true", caso contrário retorna "false".
path	String	Caminho (*path*) correspondente.

Com base na revisão de literatura já realizada na secção 2.5, algumas caraterísticas relativas à qualidade dos dados foram definidas. Um processo do tipo "ProfilerProcesses" quando é despoletado origina o tipo "TableQualityStatistics" e, consequentemente, o tipo "ColumnQualityStatistics". Assim, o nó **TableQualityStatistics** inclui os metadados pré-definidos pelo Atlas para cada tipo (nomeadamente, o *qualifiedName*, *name* e o *owner*, tal como todos os nós que são descritos incluem) e descreve as caraterísticas gerais de qualidade de uma determinada tabela, sendo estas apresentadas na Tabela 14 que se segue (Abedjan et al., 2015).

Tabela 14. TableQualityStatistics.

Metadado	Tipo de Dados	Descrição
columnStatistics	array<ColumnQualityStatistics>	Conjunto das estatísticas das colunas associadas à tabela em questão.
numcategoricalColumns	int	Número de colunas do tipo nominal.
numDateColumns	int	Número de colunas do tipo data.
numObservations	int	Número de linhas da tabela.
numVariables	int	Número de colunas da tabela.
numNumericalColumns	int	Número de colunas do tipo numérico.
numOtherTypesColumns	int	Número de colunas de outro tipo de dados.
size	int	Tamanho da tabela em megabytes.
table	Table	Tabela à qual as estatísticas estão associadas.
db	DataBase	Base de dados à qual as estatísticas estão associadas.

Cada uma das tabelas contém um conjunto de colunas. As colunas podem ser descritas através da definição de vários metadados presentes no nó **Column**, apresentado na Tabela 15. Para além dos metadados retirados a partir do Atlas, é acrescentado outro, a propriedade *attributeType*, cuja definição se baseia no trabalho de Santos & Costa (2019) e permite registar se o atributo é de caráter descritivo ou analítico.

Tabela 15. Column. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
comment	String	Descrição e comentário adicional à coluna.
type	String	Tipo de dados associado à coluna.
attributeType	String	O atributo pode ser do tipo "Descriptive", "Factual", "Predictive" ou "Identifier". Atributos **descritivos** (*Descriptive*) fornecem uma forma de interpretar atributos analíticos por meio de diferentes perspetivas, utilizando operações de agregação ou filtragem. Estes podem ser associados aos atributos encontrados nas dimensões tradicionais de um DW. Os atributos analíticos fornecem valores numéricos (às vezes incorporados em estruturas de dados complexas ou *nested* que também contêm dados de texto) que podem ser analisados através da utilização de diferentes atributos descritivos, incluindo atributos preditivos e factuais. Atributos **factuais** (*Factual*) representam evidências numéricas de algo que aconteceu num registo específico do objeto analítico e podem ser associados aos factos de uma tabela de factos tradicional. Os atributos **preditivos** (*Predictive*) não representam evidências numéricas de algo que aconteceu, mas sim uma estimativa do que aconteceu ou uma previsão do que pode acontecer no futuro (Kimball & Ross, 2013; M. Y. Santos & Costa, 2019). As *Surrogate Keys* podem ser, ou não, *Primary Keys*, incluídas em dimensões, pelo que correspondem à chave utilizada para ligar a dimensão ao facto, assim, o valor que este metadado toma é "Identifier".
table	Table	Tabela a que pertence a coluna.

O nó **ColumnLineage** descreve as caraterísticas e o percurso de uma coluna, isto é, através do metadado *query* é possível analisar que operação e processo ocorreram em determinada coluna. Todos os metadados que se encontram descritos na Tabela 16 são obtidos a partir da ferramenta Atlas.

Tabela 16. ColumnLineage. Baseado no Atlas.

Metadado	Tipo de Dados	Descrição
dependencyType	String	Tipo de dependência. Por exemplo: "EXPRESSION", "SIMPLE".
expression	String	No caso de o tipo de dependência ser "EXPRESSION" retorna a expressão correspondente.
query	Process	Descreve detalhadamente o processo ao qual o *lineage* da coluna está associado.
inputs	array<Column>	Colunas *input* do *lineage*.
outputs	array<Column>	Colunas *output* do *lineage*.

Ainda, no seguimento das caraterísticas relativas à qualidade dos dados, é definido o nó **ColumnQualityStatistics** que descreve as caraterísticas gerais de qualidade de uma determinada coluna, apresentadas na Tabela 17 que se segue (Abedjan et al., 2015).

Tabela 17. ColumnQualityStatistics.

Metadado	Tipo de Dados	Descrição
dataTypeCol	String	Tipo de dados associado à coluna.
numEmptyValues	int	Número de valores vazios.
maxValue	String	Valor máximo da coluna.
minValue	String	Valor mínimo da coluna.
maxFieldLength	int	Número máximo de carateres da coluna.
minFieldLength	int	Número mínimo de carateres da coluna.
numNullValues	int	Número de valores nulos.
numFalseValues	int	Se o tipo de dados da coluna é *boolean* retorna o número de valores "false".
numTrueValues	int	Se o tipo de dados da coluna é *boolean* retorna o número de valores "true".
percentFillRecords	float	Percentagem de valores preenchidos.
percentUniqueValues	float	Percentagem de valores únicos.
numUniqueValues	int	Número de valores únicos.
numRecords	int	Número de linhas.
frequencyValues	Map<String,String>	Frequência de determinados valores da coluna.
column	Column	Coluna à qual as estatísticas estão associadas.
table	Table	Tabela à qual as estatísticas estão associadas.

Mais uma vez de acordo com a revisão de literatura já efetuada anteriormente, como forma de analisar o *joinability* entre colunas de tabelas diferentes, é criado o nó **InterStatistics** que reúne duas métricas de similaridade, identificadas como as melhores no cálculo da similaridade, em determinados contextos específicos, devido aos resultados obtidos na dissertação de Magalhães (2019b, 2019a), já

anteriormente mencionada. Os metadados associados a este nó encontram-se descritos na Tabela 18 e, essencialmente, permitem perceber a percentagem de valores comuns entre os atributos de dois conjuntos de dados, isto é, entre colunas de tabelas diferentes (Abedjan et al., 2015; Maccioni & Torlone, 2018).

Tabela 18. InterStatistics.

Metadado	Tipo de Dados	Descrição
columnMain	Column	Coluna a ser comparada.
columnToCompare	Column	Coluna com a qual se irá comparar.
headerSimilarity	float	Calcula a similaridade (entre 0 e 1) pela comparação dos cabeçalhos das colunas. Neste contexto, a métrica identificada como mais eficiente no cálculo da similaridade pelo cabeçalho é a *Cosine*. Esta métrica calcula a semelhança entre dois vetores diferentes que compartilham a mesma representação, ou seja, mede o ângulo entre os dois vetores, por outras palavras, permite a análise da similaridade entre duas *strings* (Alodadi & Janeja, 2015).
contentSimilarity	float	Calcula a similaridade (em percentagem) pela comparação do conteúdo das colunas. A métrica utilizada para este caso é definida pelo autor da dissertação anteriormente indicada. Assim, esta calcula a similaridade com base na interseção e união de valores.
thresholdFilter	float	Valor que se define como o limite em que se filtra a similaridade dos *headers* (cabeçalhos).
table	array<Table>	Tabelas às quais as estatísticas estão associadas.

Por outro lado, como forma de analisar o relacionamento (correlação) entre colunas da mesma tabela é criado o nó **IntraStatistics**. Os metadados associados a este nó encontram-se descritos na Tabela 19 e, essencialmente, permitem perceber a percentagem de valores comuns entre colunas (Abedjan et al., 2015; Maccioni & Torlone, 2018).

Tabela 19. IntraStatistics.

Metadado	Tipo de Dados	Descrição
columnMain	Column	Coluna a ser comparada.
columnToCompare	Column	Coluna com a qual se irá comparar.
correlationValue	double	Valor da correlação entre as colunas. Permite determinar a "intensidade" da relação que existe entre dois conjuntos de dados. O valor da correlação pode variar entre -1 e 1 e o resultado obtido define se a correlação é negativa ou positiva. Logo, se o coeficiente for igual a 0 significa que as colunas não dependem uma da outra.
table	Table	Tabela à qual as estatísticas estão associadas.

Outro dos desafios definidos para *Big Data* está associado com a privacidade e as políticas definidas para os dados, pelo que é necessário ter em conta as autorizações dadas aos mesmos (C. Costa & Santos, 2017a). Assim, de acordo com o que é necessário definir na criação de uma política (permissão) e com base na ferramenta Ranger[48], com o objetivo de transcrever os dados dessa ferramenta para a interface do Atlas, é criado o nó **Policy**, associado a certos conjuntos de dados e com uma série de metadados cuja descrição pode ser consultada na Tabela 20.

Tabela 20. Policy. Baseado no Ranger.

Metadado	Tipo de Dados	Descrição
policyType	String	Tipo de política, por exemplo, "Access".
policyID	String	Identificador da política.
policyEnabled	Boolean	Se a política está ativa (*enabled*) retorna "true" e "false", caso contrário, neste último caso, a política está desativada (*disabled*). No último caso faz-se o processo inverso, ou seja, ao invés de se habilitar a política, desabilita-se a mesma, por outras palavras, realiza-se a negação da política que se definir.
dbExcludes	Boolean	Se as bases de dados estão incluídas (*include*) retorna "false" e "true", caso contrário, neste último caso, está a excluir-se as bases de dados (*exclude*) da política.
tableExcludes	Boolean	Se as tabelas estão incluídas (*include*) retorna "false" e "true", caso contrário, neste último caso, está a excluir-se as tabelas (*exclude*) da política.
columnExcludes	Boolean	Se as colunas estão incluídas (*include*) retorna "false" e "true", caso contrário, neste último caso, está a excluir-se as colunas (*exclude*) da política.
description	String	Descrição e comentário adicionais à política.
group_user_permisson_delegateAdmin	array<String>	Reúne o(s) grupo(s) de utilizadores e o(s) utilizador(es) aos quais se dá permissão, assim como as permissões definidas, por exemplo: "select", "update", "Create", "Drop", "Alter", "Index", "Lock", entre outras e, ainda, indica se a permissão tem associada a função de administrador (*delegateAdmin*).
createdBy	String	Utilizador que criou a política no Ranger.
updatedBy	String	Utilizador que editou a política no Ranger.
createTime	Date	Data de criação da política no Ranger.
updateTime	Date	Data de edição da política no Ranger.

[48] http://ranger.apache.org/

Metadado	Tipo de Dados	Descrição
db	array<DataBase>	Bases de dados associadas à política.
table	array<Table>	Tabelas associadas à política.
column	array<Column>	Colunas associadas à política.

Uma vez que um processo de negócio envolve determinados dados, pois está relacionado a certo assunto e inclui um conjunto de atividades que produzem um serviço ou produto específico para determinados *stakeholders*, é acrescentado um novo nó designado **BusinessProcess**. Este nó é constituído por metadados que descrevem o processo de negócio no geral (Dijkman, Dumas, & Ouyang, 2008), tal como verificado na Tabela 21.

Tabela 21. BusinessProcess.

Metadado	Tipo de Dados	Descrição
stakeholder	array<String>	*Stakeholder* é a pessoa que intervém, é interessada ou influenciada por determinado processo de negócio.
input	array<String>	*Inputs* associados ao processo, basicamente dizem respeito àquilo que motiva e alimenta a execução e existência do processo.
activity	array<String>	Uma atividade pode ser uma tarefa ou um subprocesso. Uma tarefa é uma atividade detalhada e representa o trabalho a ser executado. Existem sete tipos de tarefas: *service*, *receive*, *send*, *user*, *script*, *manual* e *reference*. Um subprocesso é uma atividade composta e é definido como um fluxo de outras atividades.
gateway	array<String>	Ao longo dos processos podem existir vários *gateways*. Existem *gateways* paralelos (AND-split) para criar fluxos simultâneos (sequência), *gateways* de junção paralela (AND-join) para sincronizar fluxos simultâneos, *gateways* de decisão XOR baseados em dados ou eventos para selecionar um conjunto de fluxos alternativos onde a escolha é baseada nos dados do processo (baseado em dados, ou seja, XOR-split) ou evento externo (baseado em eventos, ou seja, *deferred choice*), *gateways* de junção XOR (XOR-join) para unir um conjunto de alternativas e *gateways* de decisão OR inclusivos (OR-split) para selecionar um qualquer número de ramificações entre todos os fluxos de saída.
output	array<String>	Os *outputs* correspondem àquilo que o processo origina no final.
table	array<Table>	Tabelas associadas ao processo de negócio.

Com base no trabalho de Imhoff (2019) é averiguada a importância de incluir o nó **Indicator** para posteriormente o classificar KPI (*Key Performance Indicator*) conforme a importância para o negócio e o processo em questão. Este nó permite representar os indicadores, classificados como KPIs, que permitem medir o desempenho da organização durante uma determinada janela temporal. Assim, com base na leitura e revisão de alguns documentos científicos é identificado e definido um conjunto de metadados associados aos *Indicators*, conforme ilustra a Tabela 22.

Tabela 22. Indicator.

Metadado	Tipo de Dados	Descrição
isKPI	Boolean	Se o *Indicator* for considerado relevante para o negócio poderá ser identificado como um KPI (isKPI: true, caso contrário, isKPI: false). Um KPI possibilita a medição do desempenho de uma organização ou de um processo de negócio, ou seja, é um indicador-chave do desempenho (Kerzner, 2017): • *Key*: É o principal contribuidor para o sucesso ou fracasso; • *Performance*: Pode ser medida, quantificada, ajustada e controlada, para assim melhorar o desempenho; • *Indicator*: Representação do desempenho presente e futuro.
businessObjective	array<String>	O objetivo de negócio corresponde à meta que se pretende alcançar a longo prazo.
stakeholder	array<String>	*Stakeholder* é a pessoa que intervém ou é interessada por determinado *Indicator*.
cfs	array<String>	Para que qualquer empresa tenha futuro, há certos elementos que não podem falhar. Estes são os fatores críticos de sucesso, e são determinados por vários fatores, que vão desde o mercado no qual a empresa se insere, até ao que a diferencia. Os CSF (*Critical Success Factor*) são fundamentais no planeamento estratégico, pois determinam quais são os elementos cruciais para que a empresa venha a ter sucesso (Turner, 2009).
targetFrame	String	Janela temporal associada ao *Indicator*. Por exemplo, "Anual", "Semestral", "Mensal", entre outras.
calculationFormula	array<String>	Forma de cálculo do *Indicator*.
trend	String	A tendência mede o desempenho em relação a um período de tempo.
realValue	float	O valor atual especifica o valor nesse determinado período de tempo.
upperBound	float	Limite máximo que o *Indicator* pode atingir, ou seja, este valor corresponde ao valor objetivo que se pretende alcançar ao fim de determinado período.
averageBound	float	Limite médio que o *Indicator* pode atingir.
lowerBound	float	Limite mínimo que o *Indicator* pode atingir. O valor do *Indicator* deve situar-se entre o *upperBound* e *lowerBound*.

Metadado	Tipo de Dados	Descrição
status	String	O estado mede o desempenho do *Indicator* em relação ao objetivo e é geralmente representado como um semáforo (Eckerson, 2009). Pode tomar a cor "Red" se não for cumprido, ou seja, o *realValue* está abaixo do *lowerBound*. É "Yellow" se for cumprido, pelo que o *realValue* tem que ser superior ao *lowerBound* ou inferior a este, neste último caso, tendo em conta a percentagem de tolerância definida (*percentToleranceBound*), ou seja, ainda é cumprido se for ligeiramente inferior ao *lowerBound* (*lowerBound – lowerBound * percentToleranceBound*). O estado é "Green" se o *realValue* for superior ao *averageBound* e inferior ao *upperBound* ou, ainda, se o *realValue* for ligeiramente superior ao *upperBound*, isto é, superior ao *upperBound* tendo em conta a percentagem de tolerância (*upperBound + upperBound * percentToleranceBound*). Pode ainda o estado ser "Blue" se o *realValue* for superior ao *upperBound*.
percentToleranceBound	float	No caso de o valor do *Indicator* não se situar entre o *upperBound* e *lowerBound*, mas, no entanto, ser inferior ao *lowerBound* em determinada percentagem de tolerância considera-se que o *Indicator* foi cumprido (o *status* é "Yellow"). Para além disso, se o valor do *Indicator* for superior ao *upperBound*, em determinada percentagem de tolerância, significa que o seu *status* é ainda "Green".
variance	float	A variância mede a diferença entre o valor atual (*realValue*) e o objetivo (*upperBound*).
percentVariance	float	A percentagem de variância divide a variância em relação ao *upperBound* (*variance/ upperBound* *100).
type	String	São distinguidos dois tipos de *Indicators* (Meier, Lagemann, Morlock, & Rathmann, 2013): • **Absolutos**: Número individual, soma, diferença e média; • **Relativos**: Cotas, números de referência e números de índice.
group	String	Os *Indicators* podem ser divididos em dois grupos (Chan & Chan, 2004): • **Objetivos**: Tempo, custo e quantidade; • **Subjetivos**: Qualidade, funcionalidade e satisfação.
table	array<Table>	Tabelas às quais o *Indicator* está associado.
busProcess	array<BusinessProcess>	Processos de negócio aos quais o *Indicator* está associado.

5. Demonstração do Grafo e Validação do Sistema para Catalogação e Governança de Dados

Tendo em conta o que é apresentado no capítulo anterior, no presente capítulo será implementado no Atlas o grafo anteriormente proposto, nomeadamente os tipos/nós que foram estendidos ou criados. Como tal, inicia-se com a apresentação da infraestrutura física utilizada e a exposição dos dados que irão ser utilizados. Na secção da instanciação, utiliza-se os dados indicados para preencher o modelo do grafo anteriormente definido. Na secção da implementação, é exposta a representação dos vários tipos no Atlas. Por último, na secção da validação o sistema que inclui o grafo é avaliado tendo em conta a forma como responde às questões anteriormente colocadas aos requisitos.

5.1 Ambiente de Implementação e Dados Utilizados

5.1.1 Infraestrutura Física

A infraestrutura utilizada é constituída por um *Cluster* Hadoop que integra 5 nós (1 HDFS *Name Node* e 4 HDFS *Data Nodes*), sendo que cada um deles detém as seguintes caraterísticas:

- Intel i5 *quad core* com velocidade entre 3.1-3.3 GHz;
- 32GB de *Random Access Memory* (RAM), com 24GB disponíveis para processamento de *queries*;
- Samsumg 850 EVO 500GB *Solid State Drive* (SSD), com capacidade de leitura de 540MB/s e de escrita de 520MB/s;
- 1 *gigabit Ethernet card*.

Em todos os nós o sistema operativo utilizado é o CentOS 7 com um sistema de ficheiros XFS. A versão do Hadoop instalada é a *Hortonworks Data Platform* (HDP) 3.1.0.

Para além disso, é adicionado um novo nó que basicamente apresenta as mesmas caraterísticas já apresentadas para os outros nós, contudo, apenas se altera a componente de memória RAM e o número de *cores* presentes no processador, ou seja, este apresenta 16GB de RAM e um processador Intel i5, *dual core*. Consequentemente, neste novo nó foi instalada a versão 1.1.0 do Apache Atlas onde estão alocadas as componentes *Atlas Metadata Client* e *Atlas Metadata Server*.

5.1.2 Conjunto de Dados

Na seleção do conjunto de dados a utilizar é necessário ter em conta que o seu modelo de dados permita diferentes perspetivas de modelação. Após definição do grafo, surge necessidade de realizar a sua instanciação e implementação no Atlas, assim sendo é utilizado o *TPC Benchmark DS* (TPC-DS). Utiliza-se o seu modelo de dados original, transformando-o em um modelo BDW com base nos princípios definidos no capítulo anterior e implementando esse modelo para posteriormente o utilizar para alimentar o modelo do grafo para a catalogação e governança. Este *benchmark*, desenvolvido por Nambiar e Poess (2006), modela vários aspetos geralmente aplicáveis a um sistema de suporte à decisão, incluindo consultas e a manutenção de dados. O esquema do *benchmark* é em floco de neve, pelo que consiste em várias dimensões e tabelas de factos. Cada dimensão tem uma *surrogate key*, assim sendo as tabelas de factos associam-se às dimensões usando esse tipo de chave de cada dimensão.

O TPC-DS utiliza o modelo de negócio de uma empresa de revenda com vários canais de vendas, desde lojas físicas, até à venda de mercadorias por meio de catálogos e via *Internet*. Por conseguinte, juntamente com as tabelas para modelar as vendas e as devoluções associados, está incluído um sistema de inventário e um sistema de promoção. Inclui sete tabelas de factos, seis dessas tabelas focam as vendas (*sales*) e devoluções (*returns*) de produtos para cada um dos três canais de venda e uma única tabela de factos que modela o inventário (*inventory*) para os canais de vendas por catálogo e pela *Internet*. Além disso, o esquema do TPC-DS inclui dezassete tabelas de dimensão associadas a todos os canais de vendas. No entanto, a componente de instanciação e implementação desta dissertação tem por base a tabela de factos "store_sales", assim como algumas dimensões que esta inclui, nomeadamente a dimensão "customer", logo exibe-se na Figura 27 a primeira estrela do modelo. De realçar que a tabela "store_sales" possui 2 880 404 linhas e a tabela "customer" detém 100 000 linhas.

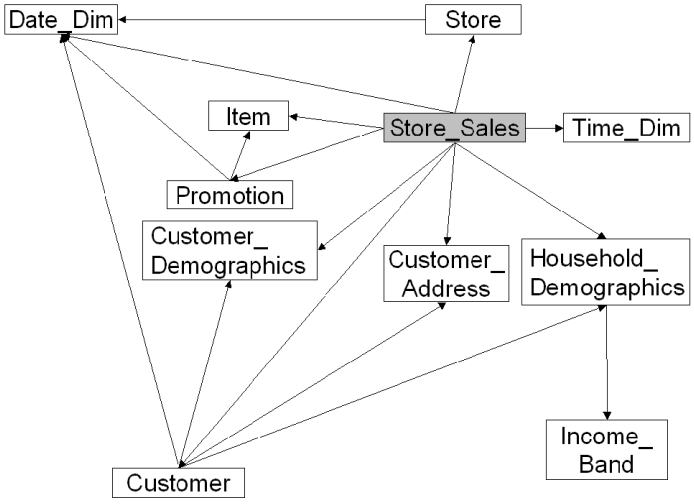

Figura 27. Primeira Estrela do modelo do TPC-DS. Retirado de Nambiar & Poess (2006).

Antes de instanciar o grafo, é importante referir as tarefas necessárias que são executadas para a preparação do ambiente. Após criação do *cluster* com as configurações já expostas em 5.1.1 (tarefa executada pela orientadora), são gerados os dados do TPC-DS com o fator de escala 10 e armazenados no HDFS e posteriormente carregados no Hive. Uma vez que o TPC-DS contém um número elevado de estrelas, é necessário selecionar-se as estrelas que se pretendem gerar, sendo que neste caso apenas são utilizados alguns componentes da primeira estrela representada na Figura 27, tal como já indicado.

5.2 Instanciação do Grafo ao TPC-DS

Seguidamente, é apresentado um conjunto de exemplos de cenários do grafo instanciado nesta dissertação, tendo por base a tabela de factos "store_sales" e a dimensão "customer" do TPC-DS. Estes cenários reúnem uma variedade de metadados relativos aos tipos possíveis de representar no Atlas.

Na Figura 28 está representado o cenário do grafo numa perspetiva geral, ou seja, com todos os nós, relacionamentos entre eles e os seus principais metadados, já preenchidos com valores reais, relativos à tabela "store_sales".

Tal como já foi esclarecido para a Figura 26, a parte inferior de cada nó contém os metadados que foram acrescentados tendo por base os requisitos definidos, enquanto que na parte superior encontram-se alguns dos metadados que são adquiridos a partir da ferramenta Atlas e, entre esses, existem atributos que são obrigatórios, sendo estes os que se encontram sublinhados (por exemplo, *qualifiedName*).

De forma a simplificar a representação do grafo, todos os nós apresentam determinada cor que ilustra o tipo a que correspondem. Para além disso, para evitar a sua excessiva extensão, é demonstrado pelo menos um exemplo de cada tipo, portanto em alguns casos utilizam-se reticências, indicando que existem mais nós de determinado tipo associados ao nó do tipo em questão.

As figuras que seguem a Figura 28 ilustram a instanciação do grafo à Tabela de Factos "store_sales", mas por cenários, isto é, cada um apresenta apenas alguns nós de determinados tipos que estão relacionados, para facilitar a compreensão da lógica associada ao modelo. Assim, o cenário geral é dividido em sete cenários, cada um reunindo vários tipos.

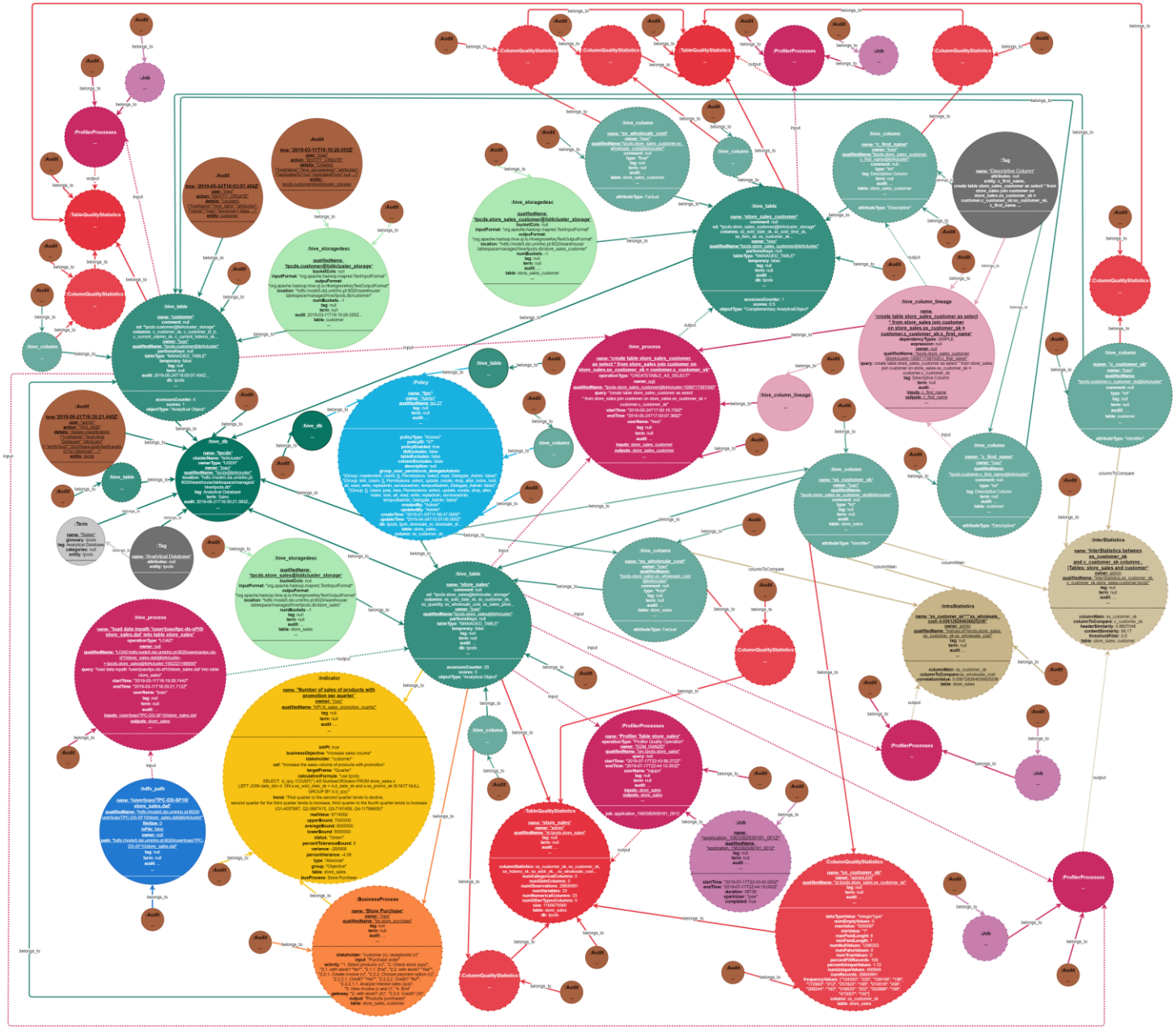

Figura 28. Cenário Geral da Instanciação do Grafo à Tabela de Factos "store_sales" do TPC-DS.

A Figura 29 ilustra o cenário 1 que consiste na organização dos tipos “hive_db”, “hive_table”, “hive_storagedesc” e “hive_column”.

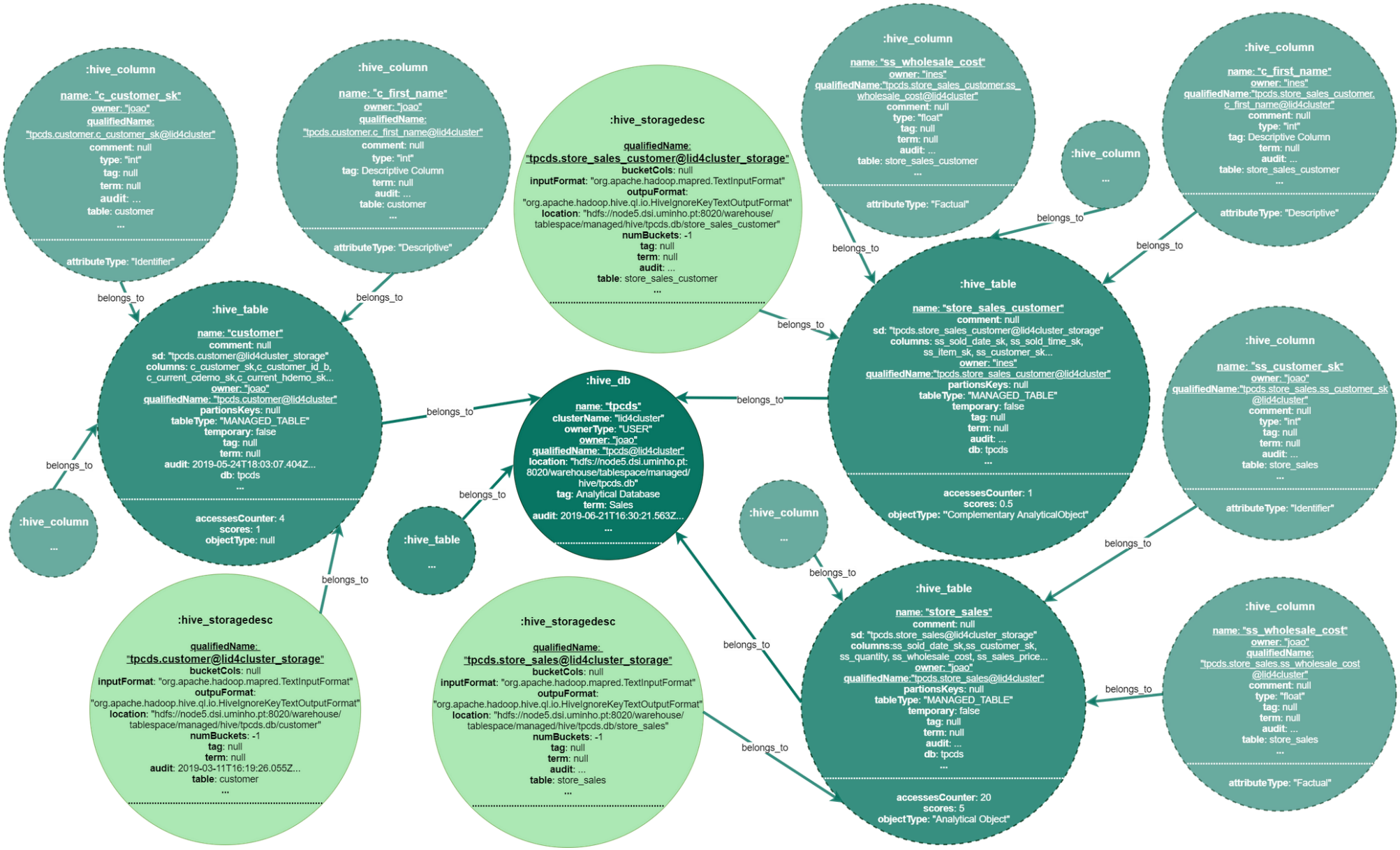

Figura 29. Cenário 1 – Tipos “hive_db”, “hive_table”, “hive_storagedesc” e “hive_column”.

Partindo do “hive_db", que neste caso corresponde à base de dados cujo nome é “tpcds”, esta é constituída por um conjunto de tabelas, entre quais três delas são representadas na Figura 29, portanto está presente o tipo “hive_table” através das tabelas “store_sales”, “customer” e “store_sales_customer”. Cada linha da tabela “store_sales” representa uma venda realizada pelo canal da loja, assim sendo, esta acomoda uma tabela de factos que corresponde a um objeto analítico, tal como permite concluir com o atributo “objectType: Analytical Object”, para além de que é possível averiguar que o número de acessos à tabela é 20 (accessesCounter: 20) e a pontuação da mesma é 5 (scores: 5). Quanto à tabela “customer” cada uma das suas linhas representa um cliente, ou seja, contém a descrição detalhada dos clientes, e ainda representa o número de acessos, que é 4 (accessesCounter: 4) e a pontuação é 1 (scores: 1). Por último, a tabela “store_sales_customer” é uma tabela criada pelo *join* de outras duas tabelas, como irá ser descrito num cenário mais adiante, sendo criada para exemplificar um tipo diferente de objeto da tabela. Esta última corresponde a um objeto analítico complementar (objectType: Complementary Analytical Object), o seu número de acessos é 1 (accessesCounter: 1) e a pontuação é 0.5 (scores: 0.5). Todas as tabelas são do tipo “MANAGED_TABLE” (tableType: "MANAGED_TABLE") e não são tabelas temporárias (temporary: false).

Relativamente ao tipo “hive_storagedesc”, cada uma das tabelas do tipo “hive_table” que são *managed* (podem apenas ser alteradas através do hive) tem um atributo “sd” que carateriza o seu *storage*. Destaca-se o atributo “bucketCols” que indica as colunas a que serão utilizadas para agrupar os dados e o atributo “numBuckets” que indica o número de *buckets* necessários.

Para além disso, cada tabela tem um conjunto de colunas. A coluna “ss_customer_sk” e a coluna “ss_wholesale_cost” estão associadas à tabela “store_sales”, sendo a primeira uma chave (attributeType: Identifier) e a segunda do tipo factual (attributeType: Factual). A coluna “c_customer_sk” que pertence à tabela “customer” também é do tipo chave (attributeType: Identifier), enquanto que a coluna “c_first_name” que pertence à mesma tabela é do tipo descritivo (attributeType: Descriptive e tag: “Descriptive Column”). Para terminar, as outras duas colunas, “ss_wholesale_cost” e “c_first_name”, pertencem à tabela “store_sales_customer” e são do tipo factual (attributeType: Factual) e descritivo (attributeType: Descriptive e tag: “Descriptive Column”), respetivamente.

O cenário 2 tal como ilustrado na Figura 30 traduz a representação dos tipos “hive_process” e “hive_column_lineage”.

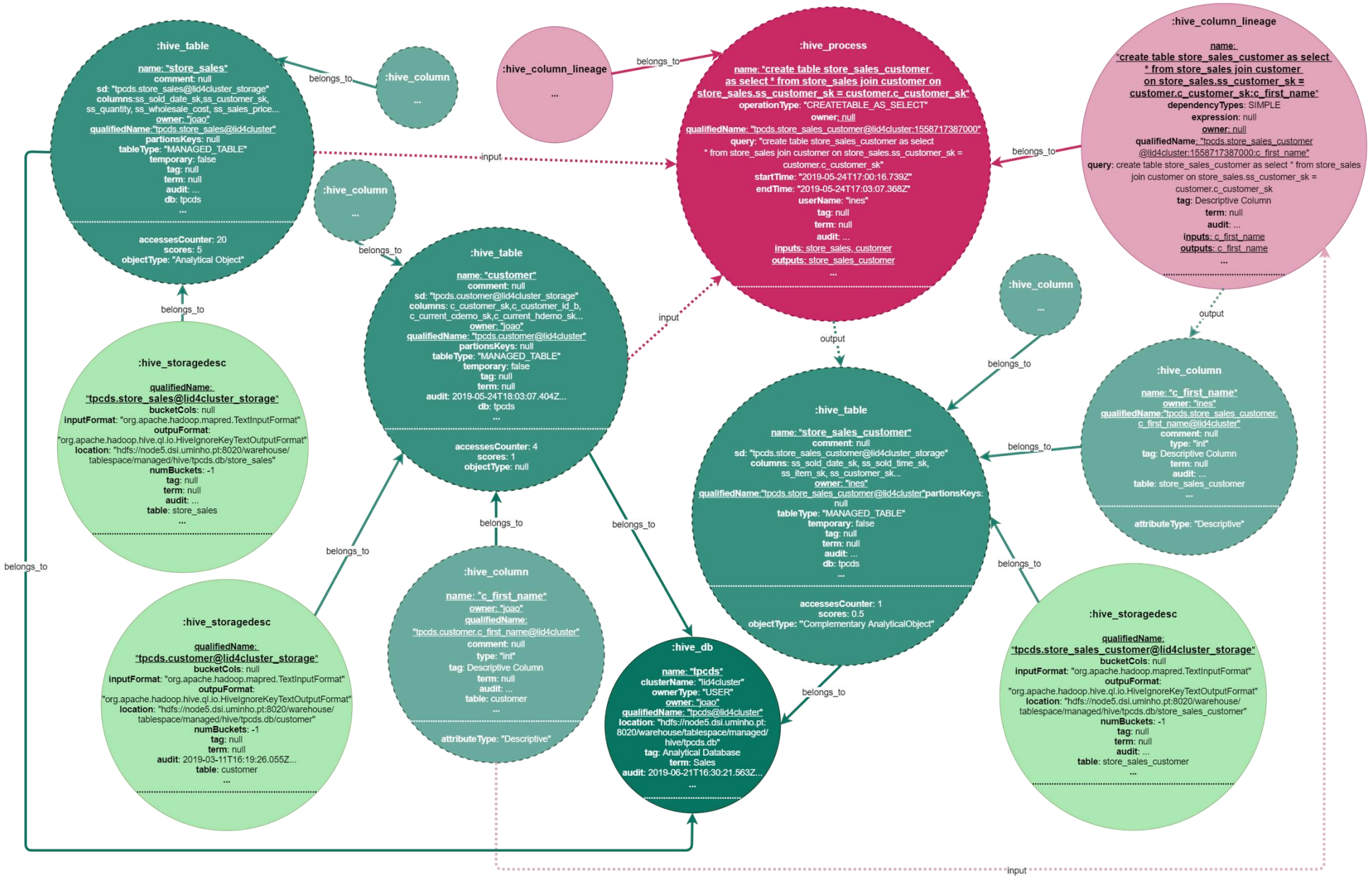

Figura 30. Cenário 2 – Tipos “hive_process” e “hive_column_lineage”.

Para simular este cenário é criada a tabela "store_sales_customer" através do *join* das tabelas "store_sales" e "customer". Tal como já averiguado, um "hive_process" é um tipo do supertipo *Process*. Assim, o "hive_process" apresentado detém o "qualifiedName" "tpcds.store_sales_customer @lid4cluster:1558717387000" e é proveniente de uma operação do tipo "CREATETABLE_AS_SELECT", ou seja, é executada a *query* da Figura 31 que cria a tabela selecionando e fazendo *join* das outras duas tabelas.

```
create table store_sales_customer as select *
from store_sales join customer on store_sales.ss_customer_sk = customer.c_customer_sk
```

Figura 31. Query de Criação da Tabela "store_sales_customer".

Todos os tipos do supertipo "Process" têm associado um ou vários *inputs* e *outputs*, neste caso os *inputs* são as tabelas "store_sales" e "customer" que são selecionadas e que sofrem *join* e o *output* é apenas a tabela que é criada, "store_sales_customer". Adicionalmente, cada "hive_process" tem uma data de início (startTime: "2019-05-24T17:00:16.739Z") e de fim (endTime: "2019-05-24T17:03:07.368Z"), onde a primeira corresponde à data de criação do processo e a última à data de finalização da criação do processo.

Devido à criação da nova tabela "store_sales_customer" foi traçado um *lineage* das colunas associadas às tabelas do "hive_process" em questão, um exemplo desse *lineage* encontra-se exibido nos nós cujo tipo é "hive_column_lineage". O "qualifiedName" do "hive_column_lineage" apresentado é "tpcds.store_sales_customer@lid4cluster:1558717387000:c_first_name" e pertence ao "hive_process", anteriormente indicado, pelo que inclui a *query* já exposta na Figura 31. Portanto, inclui como *input* a coluna "c_first_name", que pertence à tabela já existente "customer", e como *output* a coluna "c_first_name", incluída na nova tabela "store_sales_customer".

A Figura 32 exibe o cenário 3 que exemplifica essencialmente os tipos “hive_process”, “hdfs_path” e “Policy”.

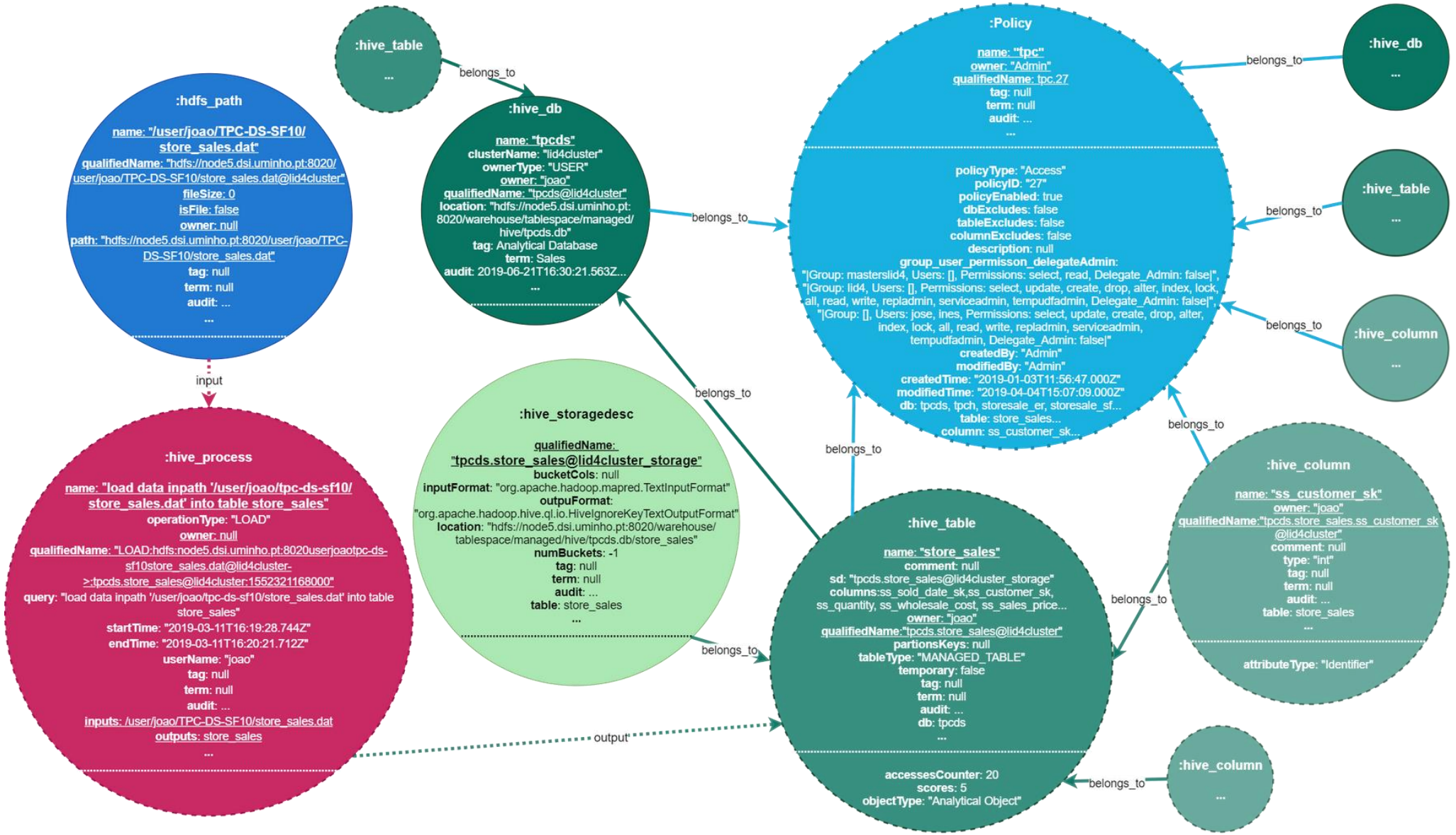

Figura 32. Cenário 3 – Tipos “hdfs_path” e “Policy”.

Na Figura 32 está presente o nó "hdfs_path" designado por "/user/joao/TPC-DS-SF10/store_sales.dat" cujo *path* é "hdfs://node5.dsi.uminho.pt:8020/user/joao/TPC-DS-SF10/store_sales.dat" e origina um "hive_process". O "hive_process" "load data inpath '/user/joao/tpc-ds-sf10/store_sales.dat' into table store_sales" é derivado do carregamento de dados da "store_sales", pelo que a operação é do tipo "LOAD", assim como tem como *input* o "hdfs_path" mencionado, "/user/joao/TPC-DS-SF10/store_sales.dat", e como *output* a tabela "store_sales".

Quanto ao nó "Policy", este foi criado tendo por base a ferramenta Ranger com o objetivo de reproduzir os dados sobre as políticas definidas nessa ferramenta para o Atlas, políticas essas nomeadamente relativas ao Hive, assim uma série de metadados foram adicionados. O nó desse tipo, presente na figura, carateriza a política designada por "tpc", cujo criador e modificador é o "admin", é do tipo "Access", o seu identificador é o "27" e está ativa/habilitada (policyEnabled: true). A política está associada e inclui ("dbExcludes: false", "tableExcludes: false" e "columnExcludes: false") as bases de dados "tpcds", "tpch", "storesale_er", "storesale_sf", as tabelas "store_sales", "customer" e as colunas "ss_customer_sk", "ss_wholesale_cost", entre todas as outras tabelas e colunas de cada uma das bases de dados (pois quando a política é definida no Ranger com "*" no campo da base de dados, tabela ou coluna, indica que tudo articulado a elas é selecionado). Numa política podem existir vários conjuntos de permissões associadas, podendo ser selecionado um grupo ou vários grupos de utilizadores a quem se quer dar acesso, pode ainda ser selecionado o utilizador ou os utilizadores em específico aos quais se dá permissão. Finalmente, escolhem-se as permissões, por exemplo "select", para além disso pode-se indicar se a permissão tem associada a função de administrador (*delegateAdmin*). Todos os metadados explicitados anteriormente estão expostos num dos atributos presentes no nó das políticas, sendo esse o "group_user_permisson_delegateAdmin". As permissões definidas para a política vigente são melhor detalhadas na Tabela 23.

Tabela 23. Detalhes da Política – Allow Condition. Retirado do Ranger.

Group	User	Permissions	Delegate Admin
masterslid4	–	select, read	false
lid4	–	select, update, create, drop, alter, index, lock, all, read, write, repladmin, serviceadmin, tempudfadmin	false
–	jose, ines	select, update, create, drop, alter, index, lock, all, read, write, repladmin, serviceadmin, tempudfadmin	false

O cenário 4 retrata particularmente os tipos “BusinessProcess” e “Indicator”, tal como ilustrado na Figura 33.

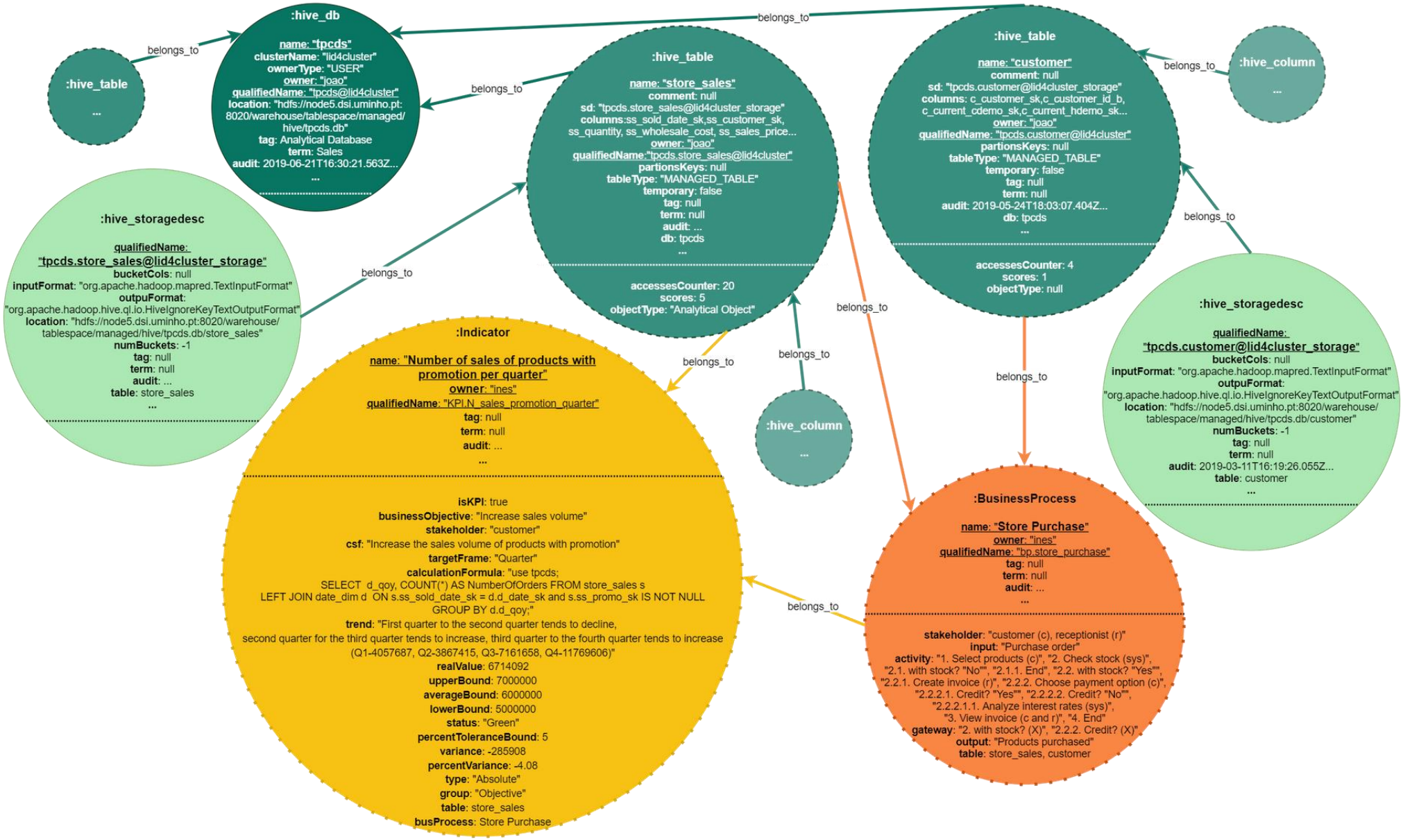

Figura 33. Cenário 4 – Tipos “BusinessProcess” e “Indicator”.

O nó “BusinessProcess” exemplificado é designado por “Store Purchase” e traduz uma compra efetuada na loja. Os *stakeholders* deste *business process* são o "customer (c)" e o "receptionist (r)". Para despoletar um processo é necessário a existência de *inputs*, que neste caso corresponde apenas a um pedido de compra (input: “Purchase order”). As atividades associadas ilustram as tarefas que são realizadas durante o processamento do pedido, inicialmente são selecionados os produtos por parte do cliente (“1. Select products (c)”), posteriormente o sistema verifica o *stock* (“2. Check stock (sys)”) e quando não tem *stock* (“2.1. with stock? “No”””) o processo termina (“2.1.1. End”), caso tenha *stock* (“2.2. with stock? “Yes”””) é criada a fatura pelo rececionista (“2.2.1. Create invoice (r)”) e de seguida é escolhido o método de pagamento (“2.2.2. Choose payment option (c)”), se for por meio de crédito (“2.2.2.1. Credit? “Yes”””) são analisadas as taxas de juro (“2.2.2.1.1. Analyze interest rates (sys)”), se não for por crédito (“2.2.2.2. Credit? “No”””) o processo segue diretamente para a atividade de visualizar a fatura (“3. View invoice (c and r)”) e depois o processo termina (“4. End”). Quanto a *gateways* o presente processo de negócio inclui o *gateway* que verifica se existe *stock* (“2. with stock? (X)”) e o que verifica se o método de pagamento é crédito (“2.2.2. Credit? (X)”). Todos os processos de negócio no final originam *outputs*, neste caso em concreto os *outputs* são os produtos comprados (“Products purchased”). Este processo de negócio pertence à tabela “store_sales” e “customer”. Na Figura 34 subsequente está patenteado o processo de negócio, “Store Purchase”, explicado anteriormente.

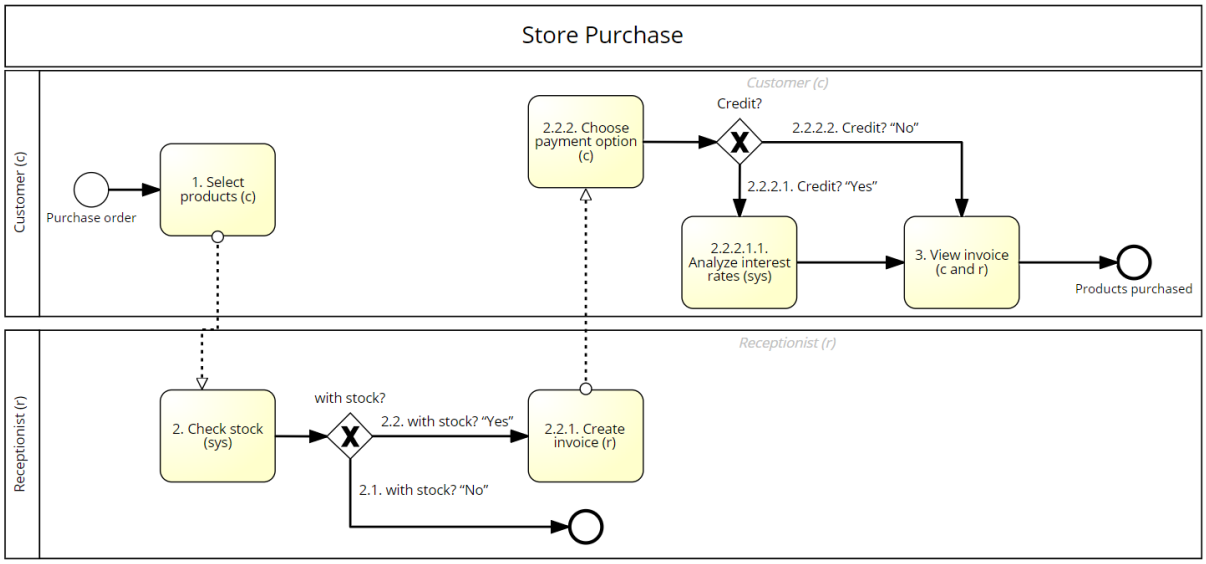

Figura 34. Processo de Negócio “Store Purchase”.

O tipo “Indicator” exposto no nó correspondente designa-se “Number of sales of products with promotion per quarter”. Este *Indicator* possui como objetivo de negócio aumentar o volume de vendas (businessObjective: “Increase sales volume”), o seu *stakeholder* é o “customer”, o fator crítico de sucesso associado é aumentar o volume de vendas de produtos com promoção (csf: “Increase the sales

volume of products with promotion") e a sua janela temporal é ao trimestre (targetFrame: "Quarter"). Se o Indicador for considerado relevante para o negócio poderá ser identificado como um KPI, pelo que neste caso representa exatamente um KPI (isKPI: true). Assim sendo, para calcular o *Indicator* foi necessário executar a *script* da Figura 35 que como resultado retorna os valores das vendas com promoção para cada trimestre.

```
use tpcds;
SELECT  d_qoy, COUNT(*) AS NumberOfOrders
FROM store_sales s
LEFT JOIN date_dim d  ON s.ss_sold_date_sk = d.d_date_sk and s.ss_promo_sk IS NOT NULL GROUP BY d.d_qoy;
```

Figura 35. CalculationFormula do Indicador "Number of sales of products with promotion per quarter".

A tendência (*trend*) basicamente traduz a evolução dos valores das vendas de produtos com promoção por trimestre, como se de um gráfico semelhante ao da Figura 36 se tratasse.

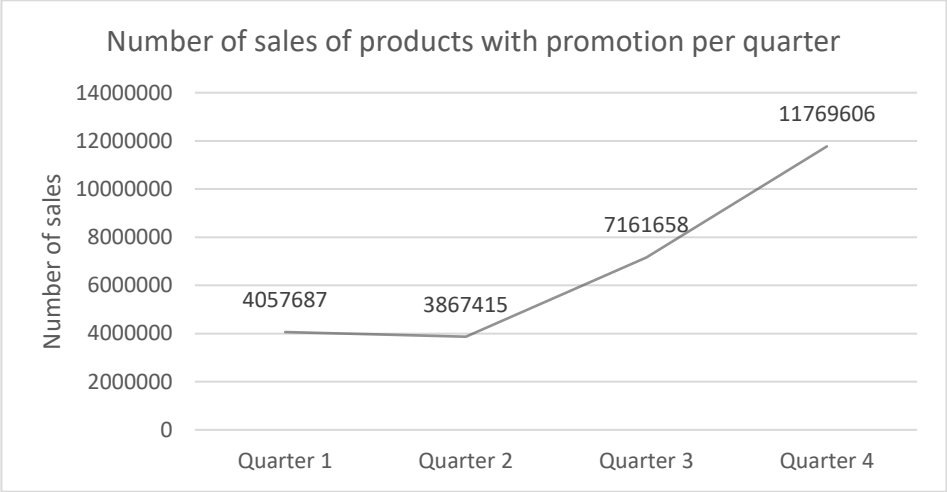

Figura 36. Trend do Indicador "Number of sales of products with promotion per quarter".

O valor real/atual do *Indicator* é "6714092", o limite superior é "7000000", o limite intermédio é "6000000", o limite inferior é "5000000", a percentagem de tolerância é "5", assim a partir destes valores enumerados é possível verificar em que medida é que o *Indicator* está ou não a ser cumprido. Concluindo, o estado atual do *Indicator* é "Green" quer isto dizer que é cumprido, pois o *realValue* é superior ao *averageBound* e inferior ao *upperBound*, contudo também é cumprido se o *realValue* for ligeiramente superior ao *upperBound*, isto é, superior ao *upperBound* tendo em conta a percentagem de tolerância (upperBound + upperBound * percentToleranceBound), todos os outros casos possíveis podem ser analisados na descrição presente na Tabela 22. Para além disso, também está patente a variância e a percentagem de variância, que é "-285908" e "-4.08%", respetivamente. Mais uma vez, tal como já explicitado na Tabela 22, os *Indicators* pertencem a um determinado grupo e tipo, neste caso o grupo é "Objective" e o tipo "Absolute". Para concluir, o *Indicator* pertence à tabela "store_sales" e está associado ao processo de negócio "Store Purchase".

O cenário 5 encontra-se representado na Figura 37 e expõe os tipos “ProfilerProcesses”, “Job”, “TableQualityStatistics” e “ColumnQualityStatistics”.

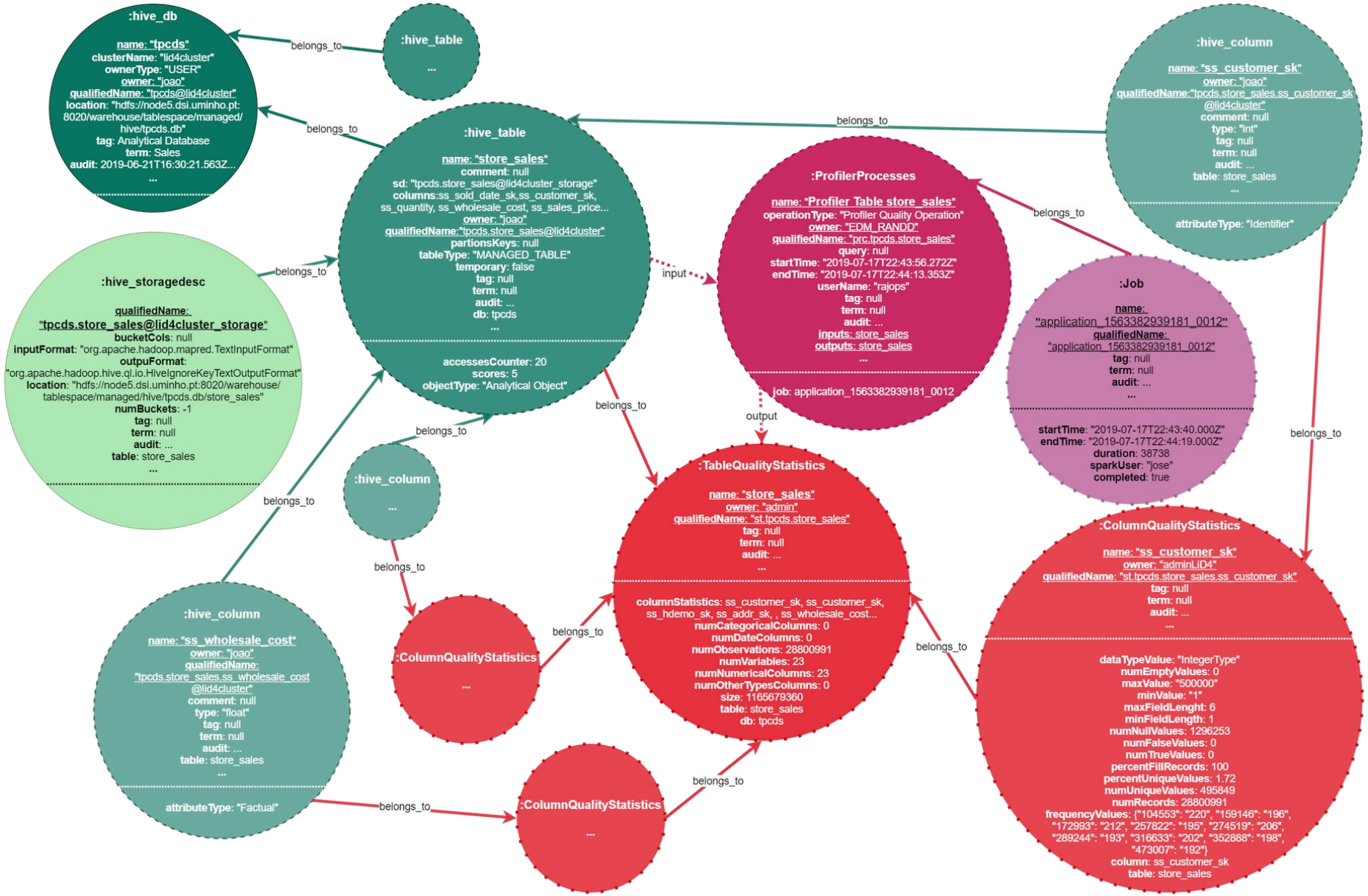

Figura 37. Cenário 5 – Tipos “ProfilerProcesses”, “TableQualityStatistics” e “ColumnQualityStatistics”.

O nó do tipo "ProfilerProcesses", cujo supertipo é o *Process*, é despoletado e origina os nós dos tipos "TableQualityStatistics" e "ColumnQualityStatistics". Portanto, o "ProfilerProcesses" é gerado de um processo de *data profiling* executado no Spark, assim o nó desse tipo tem o nome "Profiler Table store_sales" e traduz uma operação do tipo "Profiler Quality Operation" que tem como *input* a tabela "store_sales" e *output* as estatísticas de qualidade dessa tabela, também denominadas "store_sales". Este processo tem associado o "Job" com o nome "application_1563382939181_0012", cujo estado é "completo" (completed: true) e a duração correspondente é "38738" milissegundos.

O *output* do "Profiler Table store_sales" do tipo "TableQualityStatistics" mencionado, são as estatísticas da tabela "store_sales". Estas estatísticas de qualidade reúnem um conjunto de metadados que descrevem as caraterísticas gerais da tabela desde o número total colunas dessa tabela (numVariables: 23) e quantas são do tipo data (numDateColumns: 0), ou que são categóricas (numCategoricalColumns: 0), ou numéricas (numNumericalColumns: 23), ou de outro qualquer tipo (numOtherTypesColumns: 0). Neste caso em específico conclui-se que todas as colunas da tabela são numéricas. É possível ainda averiguar qual o número de observações (numObservations), ou seja, qual o número de linhas da tabela, neste cenário a tabela tem "28800991" linhas, tal como se verifica que o tamanho em *megabytes* da mesma é "116567936". É de realçar que ao gerar as estatísticas de qualidade de uma tabela, juntamente são geradas as estatísticas de qualidade de cada uma das suas colunas, assim o nó "TableQualityStatistics" reúne ainda as várias estatísticas das colunas associadas (columnStatistics). Posto isto, o nó "ColumnQualityStatistics" descreve as caraterísticas de qualidade das colunas, assim sendo, o nó deste tipo ilustrado no presente cenário designa-se "ss_customer_sk". Através destas estatísticas é possível analisar que a coluna "ss_customer_sk" é do tipo inteiro (dataTypeValue: "IntegerType"), não possui valores vazios (numEmptyValues: 0), no entanto apresenta "1296253" valores nulos (numNullValues: 1296253). O valor mínimo que possui é "1" (minValue: "1") e o máximo é "500000" (maxValue: "500000") e, neste contexto, o número mínimo de carateres da coluna corresponde ao número de carateres do valor mínimo que a coluna toma (minFieldLength: 1) e o número máximo de carateres da coluna corresponde ao número de carateres do valor máximo que a coluna toma (maxFieldLenght: 6). Uma vez que é do tipo inteiro não possui valores falsos (numFalseValues: 0) nem verdadeiros (numTrueValues: 0) e ainda todos os registos são preenchidos (percentFillRecords: 100). O número de registos é "28800991" (numRecords: 28800991), tal como o número de linhas da tabela anteriormente explorada, assim a percentagem de valores únicos é "1.72" (percentUniqueValues: 1.72). Para terminar, ainda são facultadas as frequências dos valores (FrequencyValues: {"104553": "220", "159146": "196",...}).

De seguida, na Figura 38 é instanciado o cenário 6 que se centra nos tipos “InterStatistics” e “IntraStatistics”.

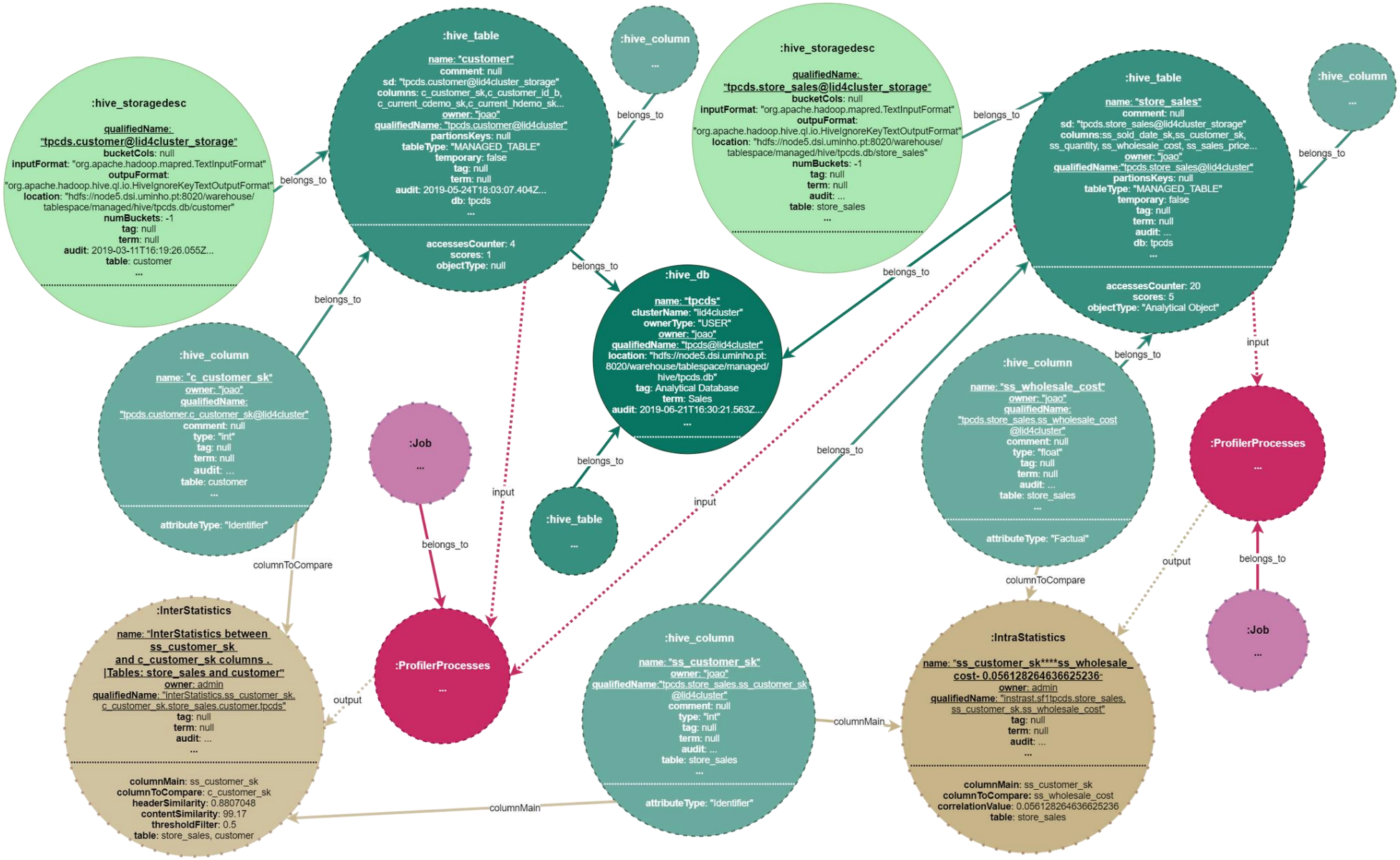

Figura 38. Cenário 6 – Tipos “InterStatistics” e “IntraStatistics”.

Quando um processo do tipo "ProfilerProcesses" é despoletado também podem ser geradas as "IntraStatistics" de uma tabela. Pelo que, neste cenário, o processo tem como *input* a tabela "store_sales" e *output* as "IntraStatistics" dessa tabela, denominadas "ss_customer_sk****ss_wholesale_cost- 0.056128264636625236".

Ao gerar as "IntraStatistics" de uma tabela é selecionada uma coluna principal (columnMain) e outra com a qual se irá comparar (columnToCompare). Neste caso, a coluna "ss_customer_sk" (columnMain: ss_customer_sk) é comparada com a coluna "ss_wholesale_cost" (columnToCompare: ss_wholesale_cost) e obtém-se "0.056128264636625236" (correlationValue: 0.056128264636625236) que corresponde ao valor da correlação entre essas duas colunas. É possível concluir que, neste caso, o valor da correlação está muito próximo de 0, significando que as colunas não dependem significativamente uma da outra.

Quanto às "InterStatistics" estas são geradas quando outro processo do tipo "ProfilerProcesses" é despoletado. Nesta situação, o processo tem como *inputs* as tabelas "store_sales" e "customer" e *output* as "InterStatistics" dessas tabelas, denominadas "InterStatistics between ss_customer_sk and c_customer_sk columns . |Tables: store_sales and customer".

Nas "InterStatistics" também é selecionada uma coluna principal (columnMain) e outra com a qual se irá comparar (columnToCompare). Neste panorama, a coluna "ss_customer_sk" (columnMain: ss_customer_sk), pertencente à tabela "store_sales", é comparada com a coluna "c_customer_sk" (columnToCompare: c_customer_sk), que pertence à tabela "customer". Assim, obtém-se o valor da similaridade entre os cabeçalhos que corresponde a "0.8807048" (headerSimilarity: 0.8807048) e o valor da similaridade do conteúdo das colunas que é "99.17"% (contentSimilarity: 99.17). Logo, as colunas são similares, pois o valor da similaridade dos cabeçalhos está próximo de 1 e o valor da similaridade do conteúdo está próximo de 100%.

Para terminar, na Figura 39 é instanciado o cenário 7 que se ilustra os tipos “Audit”, “Tag” e “Term”.

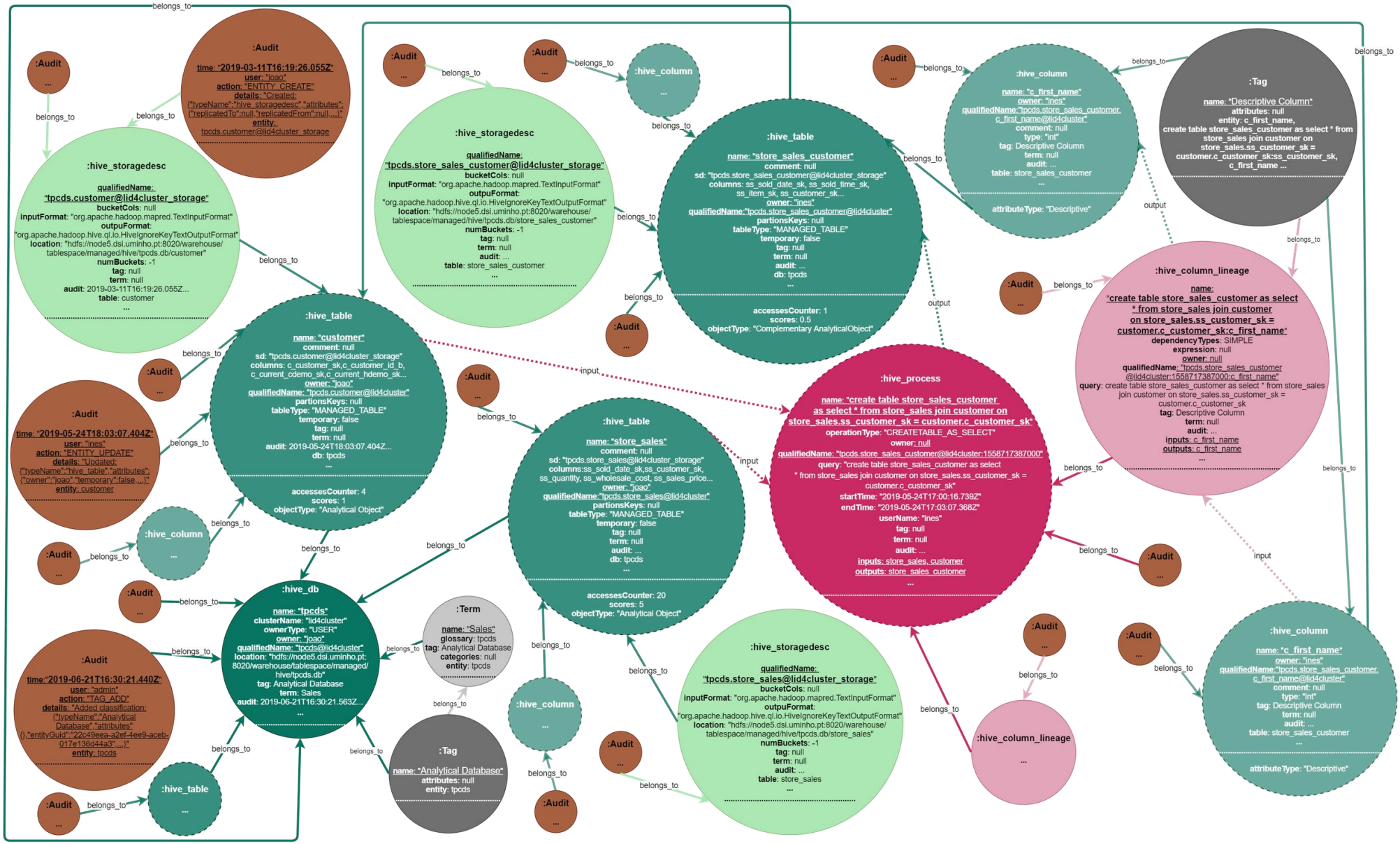

Figura 39. Cenário 7 – Tipos “Audit”, “Tag” e “Term”.

Partindo do tipo "hive_db", que neste caso corresponde à base de dados cujo nome é "tpcds", é de realçar que tem uma *tag* associada que indica que se trata de uma "Analytical DataBase" e tem associado o *term* "Sales" que por sua vez também tem associada a *tag* "Analytical DataBase", no entanto não pertence a nenhuma categoria (categories: null).

Sabendo que a coluna "c_first_name" pertence à tabela "customer" e se trata de uma tabela descritiva é de realçar que tem associada a *tag* "Descriptive Column". Adicionalmente, o "hive_column_lineage" também possui a *tag* "Descriptive Column" que é propagada da classificação atribuída inicialmente à coluna "c_first_name" da tabela "customer". Igualmente, a coluna "c_first_name" pertencente à tabela "store_sales_customer" tem associada a *tag* "Descriptive Column".

Qualquer entidade a partir do momento em que é criada tem associado uma ou mais *audits*, ou seja, patenteia cada ação executada sobre qualquer entidade, demonstrando, por exemplo, os detalhes da sua criação. Como se pode observar na Figura 39 estão simbolizados com detalhe três exemplos de *audits*, cada um representando uma ação diferente. O nó do tipo "hive_db" que ilustra a base de dados "tpcds" tem o identificador *time* (time: 2019-06-21T16:30:21.440Z) que indica a data em específico em que a entidade sofreu a ação, neste caso corresponde a uma ação onde um determinada *tag* é adicionada, pois é do tipo "TAG_ADD" (action: TAG_ADD), sabe-se ainda que quem realizou a ação foi o utilizador "admin" e ainda se consegue verificar os detalhes (details) associados. Relativamente à tabela "customer" está presente um *audit* que descreve uma alteração (action: ENTITY_UPDATE) que esta entidade sofreu, originada pelo utilizador "ines" na data "2019-05-24T18:03:07.404Z". Por último, no que diz respeito ao *storage* da tabela "customer" (tpcds.customer@lid4cluster_storage) está associado um *audit* cuja ação consiste no momento de criação (action: ENTITY_CREATE) desta entidade pelo utilizador "joao".

Como forma de facilitar e melhorar a visualização das figuras anteriormente apresentadas relativas ao grafo, na pasta denominada "Graph_Demonstration_Scenarios" disponibilizada por Costa (2019) encontra-se a ilustração do grafo modelado, da figura ilustrativa das tecnologias envolvidas na modelação do catálogo de metadados e de cada cenário devidamente identificado.

5.3 Implementação do Grafo ao TPC-DS no Atas

Para a implementação no Atlas dos metadados anteriormente indicados é necessário o desenvolvimento de um ficheiro JSON para criar ou editar cada tipo através da REST API do Atlas. Mais uma vez, na implementação utiliza-se a tabela de factos "store_sales" e a dimensão "customer" do TPC-DS que alimentam os nós do grafo proposto com os seus metadados. Na Tabela 24 são apresentados

os componentes e atributos que devem ser incluídos nos ficheiros JSON para se acrescentar os novos tipos.

Tabela 24. Metadados (Metatypes) do Atlas.

Types (enumTypes, classTypes, structTypes, traitTypes)	**superTypes**: DataSet/Asset/Process/Referenceable/fs_path	
	hierarchicalMetaTypeName	
	typeName	
	typeDescription	
	attributeDefinitions	Cada atributo contêm as seguintes especificações: **name**: *string*, **dataTypeName**: *string*, **multiplicity**: *enum*, **isComposite**: *boolean*, **isUnique**: *boolean*, **isIndexable**: *boolean*, **reverseAttributeName**: *string*

Consequentemente, recorre-se a alguns métodos HTTP (Hypertext Transfer Protocol) para serviços REST, nomeadamente os métodos POST (criar) e PUT (editar), conforme o tipo de ação que deve ser executada. Assim, na pasta designada por "Atlas Type Structure" disponibilizada por Costa (2019) encontram-se todos os ficheiros JSON utilizados para editar e criar alguns tipos.

Relativamente ao tipo "hive_table", tal como referido, são adicionados quatro novos metadados, são eles os atributos "accessesCounter", "scores" e "objectType". Para esse fim, é desenvolvido o ficheiro JSON, cujo extrato do mesmo se encontra representado na Figura 40.


```
"enumTypes": [],
"structTypes": [],
"traitTypes": [],
"classTypes": [{
        "superTypes": ["DataSet"],
        "hierarchicalMetaTypeName": "org.apache.atlas.typesystem.types.ClassType",
        "typeName": "hive_table",
        "typeDescription": null,
        "attributeDefinitions": [
            {
                "name": "acessesCounter",
                "dataTypeName": "int",
                "multiplicity": "optional",
                "isComposite": false,
                "isUnique": false,
                "isIndexable": false,
                "reverseAttributeName": null,
                "defaultValue": null,
                "description": null,
                "options": null
            },
            {
                "name": "scores",
                "dataTypeName": "double",
                "multiplicity": "optional",
                "isComposite": false,
                "isUnique": false,
                "isIndexable": false,
```


Figura 40. JSON definido para o tipo "hive_table".

Posteriormente, é utilizado o método PUT para editar o tipo "hive_table" já existente no Atlas, assim foi executado o seguinte comando: "curl -i -X PUT -H 'Content-Type: application/json' -H 'Accept: application/json' -u admin:adminatlaslid4 'node6.dsi.uminho.pt:21000/api/atlas/types' -d

@Json/hive_table.json". Portanto, vai-se recolher o ficheiro JSON à diretoria indicada e depois edita-se o tipo lá mencionado. O atributo "accessesCounter" conta os *audits* de uma tabela, sendo preenchido e modificado quando as estatísticas de qualidade da tabela são despoletadas ou atualizadas. Numa primeira abordagem os restantes atributos também iriam ser preenchidos de forma automática, não sendo possível devido a limitações da ferramenta. Logo, para colmatar esta situação, uma solução passará por o utilizador preencher estes atributos manualmente a partir da interface do Atlas, uma vez que possivelmente se encontra contextualizado com o negócio em questão. Igualmente, de momento o valor da pontuação (score) só pode ser inserido manualmente pelos utilizadores do Atlas, pois a ferramenta não permite que consoante a pontuação atribuída por cada utilizador o sistema calcule automaticamente a média e retorne o valor da mesma, contudo no futuro é expectável que tal aconteça.

No que diz respeito ao tipo "hive_column" também se recorreu ao método PUT para editar este tipo. Assim é executado o comando: "curl -i -X PUT -H 'Content-Type: application/json' -H 'Accept: application/json' -u admin:adminatlaslid4 'node6.dsi.uminho.pt:21000/api/atlas/types' -d @Json/hive_column.json" para se adicionar o novo atributo da coluna, designado "attributeType". Comparativamente aos atributos adicionados no tipo "hive_table", os novos atributos do "hive_column" também deverão ser preenchidos manualmente pelo utilizador, tendo em conta a mesma razão anteriormente explicada. Na Figura 41 é representada a coluna "ss_customer_sk" da tabela "store_sales", implementada no Atlas.

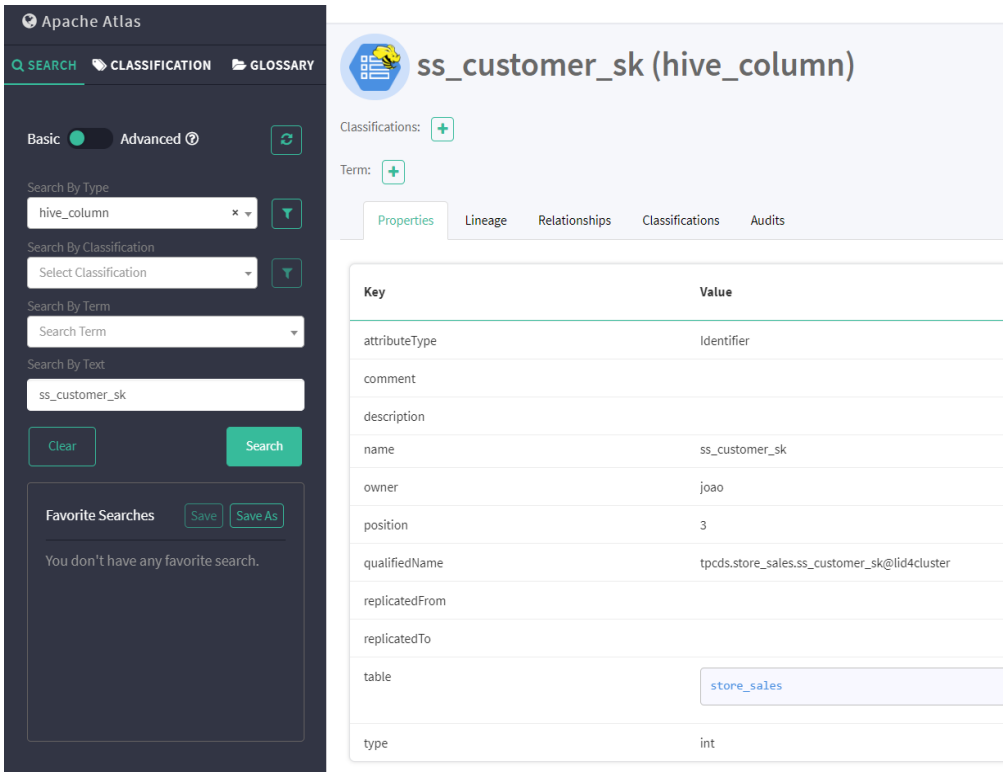

Figura 41. Implementação da Coluna "ss_customer_sk" no Atlas.

Os tipos “TableStatistics” e “ColumnStatistics”, assim como as “IntraStatistics” e InterStatistics” são adicionados no Atlas, através de outro método POST. Posteriormente, os seus metadados são preenchidos recorrendo-se à ferramenta de *profiling* desenvolvida na dissertação de Magalhães (2019b, 2019a). Na Figura 42 são expostas as estatísticas de qualidade da tabela “store_sales” no Atlas, já com os metadados preenchidos.

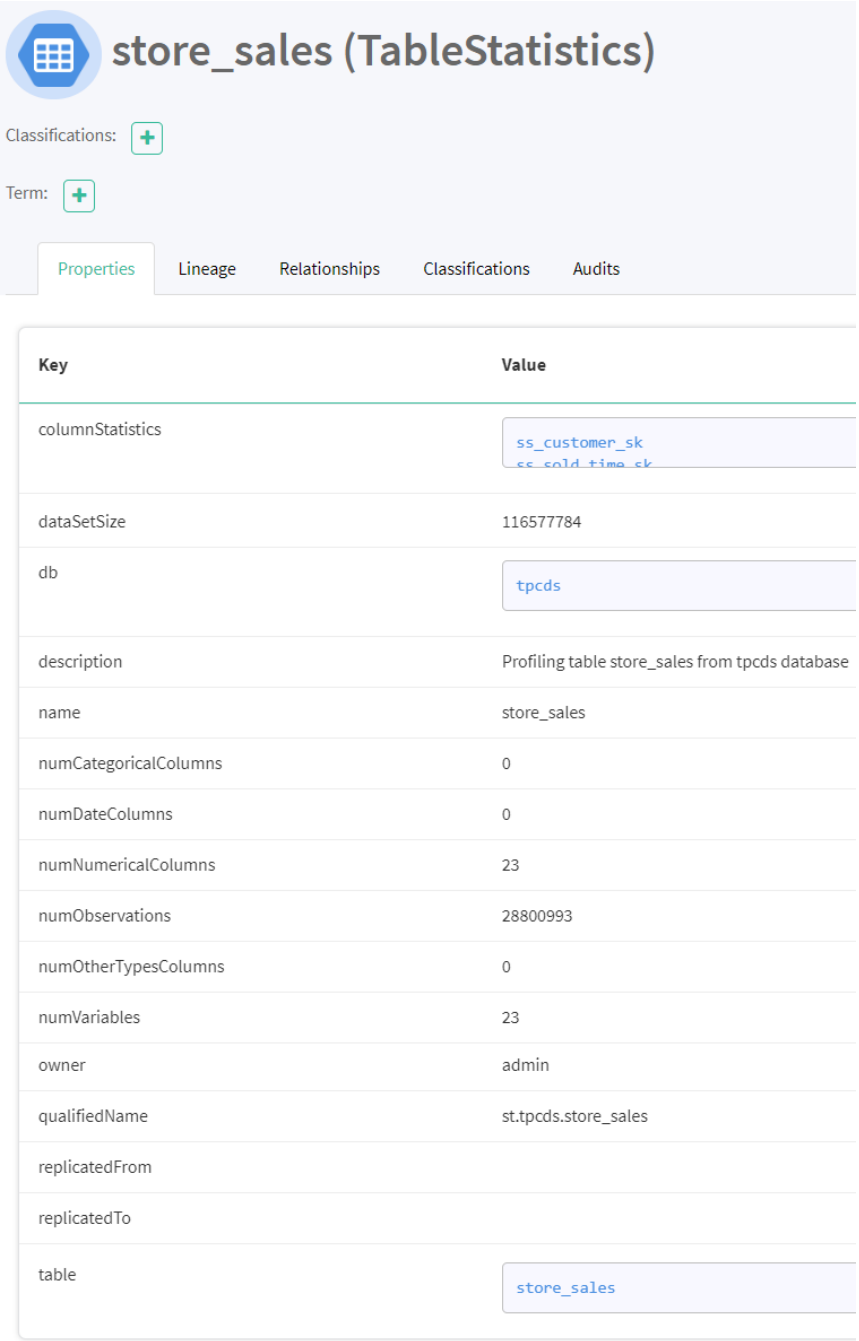

Figura 42. Implementação das Estatísticas de Qualidade da Tabela “store_sales” no Atlas.

Quanto à Figura 43, estão representadas as estatísticas de qualidade da coluna “ss_customer_sk” que pertence à tabela “store_sales”.

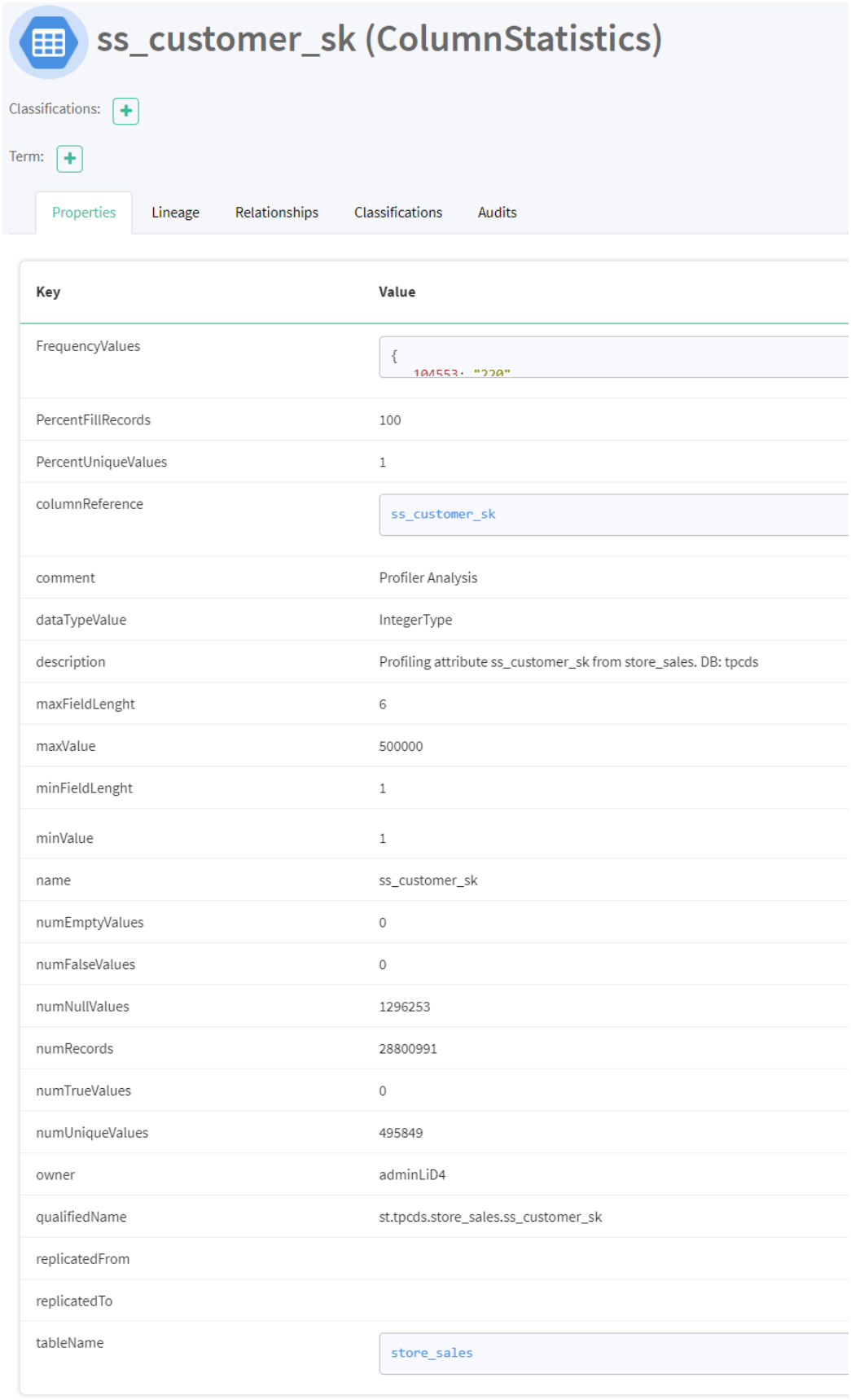

Figura 43. Implementação das Estatísticas de Qualidade da Coluna "ss_customer_sk" no Atlas.

Sempre que um processo de *profiling* (tipo de processo "ProfilerProcesses") é despoletado origina as estatísticas e consequentemente tem associado um *job* em particular. O tipo "Job" é criado no Atlas pelo método POST e os seus atributos são preenchidos invocando-se a REST API da aplicação Spark Job. Logo, o processo "Profiler Table store_sales" que originou as estatísticas anteriormente descritas é associado ao *job* designado por "application_1563382939181_0012" e a representação no Atlas desse processo encontra-se na Figura 44.

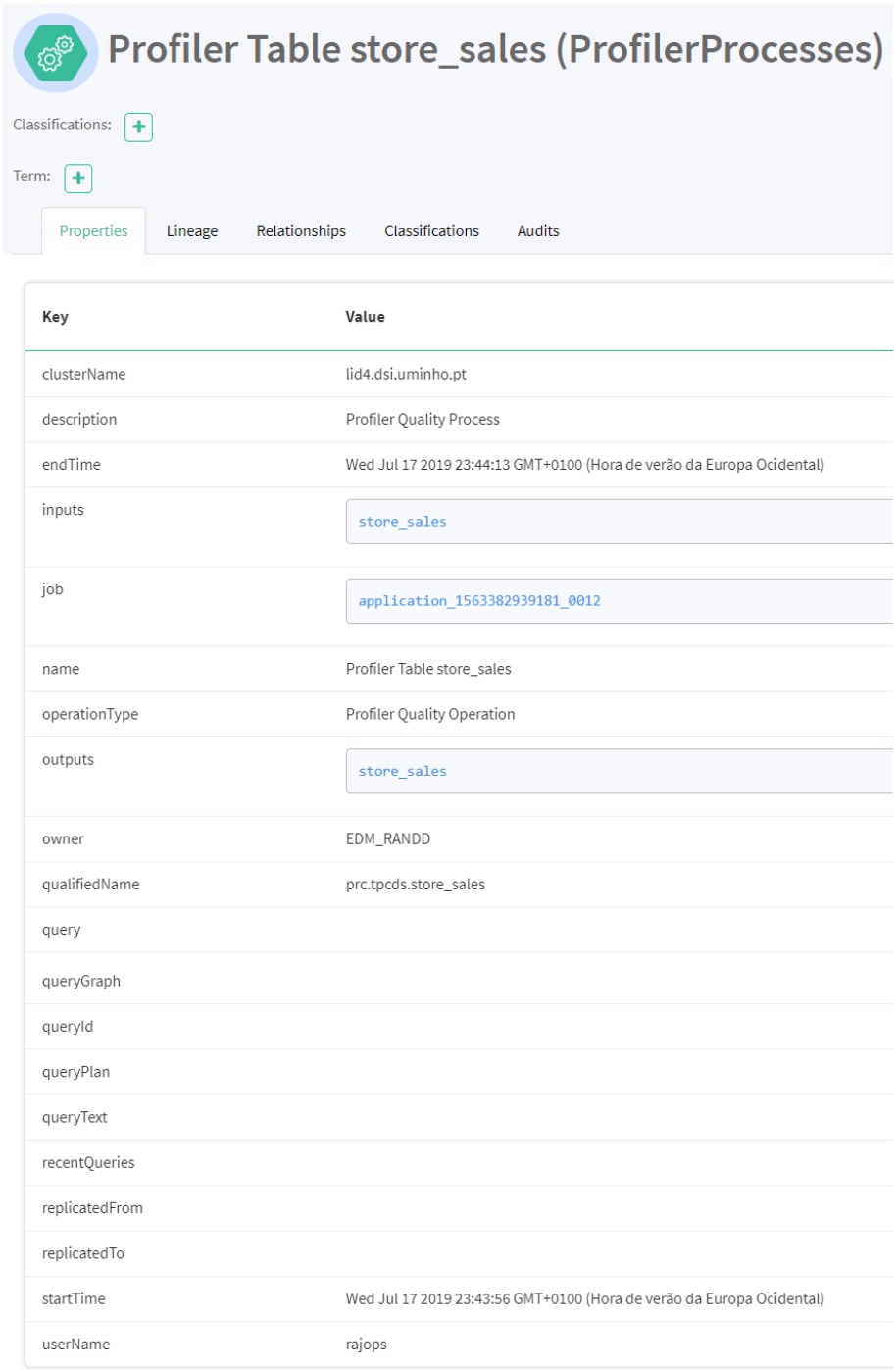

Figura 44. Implementação do Processo de Profiling "Profiler Table store_sales" no Atlas.

Enquanto que a exposição do *job* "application_1563382939181_0012" se encontra na Figura 45.

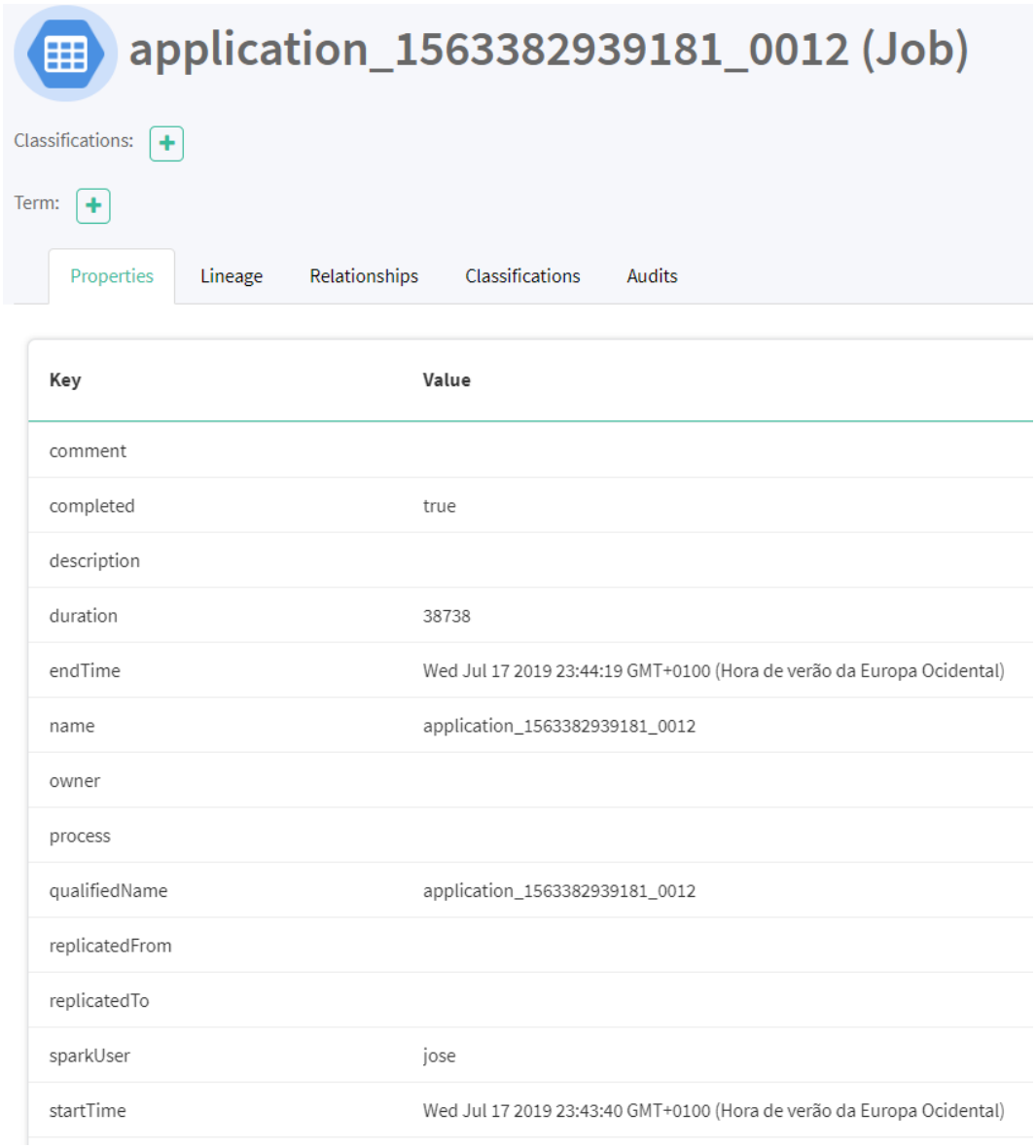

Figura 45. Implementação do Job "application_1563382939181_0012" no Atlas.

À semelhança das outras estatísticas os tipos das "IntraStatistics" e "InterStatistics" são produzidos pelo método POST e preenchidos pela ferramenta de *profiling*. Posteriormente, são geradas as "IntraStatistics" que comparam a correlação entre as colunas "ss_customer_sk" e "ss_wholesale_cost", e as "InterStatistics" que comparam a similaridade entre as colunas "ss_customer_sk" e "c_customer_sk".

O tipo "Policy" é um tipo adicionado de raiz ao Atlas que reúne as informações das políticas do Ranger. Assim sendo, para criar este tipo (pelo método POST) é executado o seguinte comando: "curl -i -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' -u admin:adminatlaslid4 'node6.dsi.uminho.pt:21000/api/atlas/types' -d @Json/Policy.json". Posteriormente, as informações

das políticas são transcritas do Ranger para o tipo "Policy", criado no Atlas, a partir da REST API do Ranger que permite analisar e retirar os metadados do XML que retorna. Assim, é implementada no Atlas a política "tpc".

Tal como no tipo anterior, o tipo "BusinessProcess" é adicionado ao Atlas, contudo, a diferença é que os atributos aqui definidos não são originados de nenhuma ferramenta. Para se criar este tipo utiliza-se novamente o método POST, através do seguinte comando: "curl -i -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' -u admin:adminatlaslid4 'node6.dsi.uminho.pt:21000/api/atlas/types' -d @Json/ BusinessProcess.json". Estando já o tipo criado é necessário preencher os seus atributos. Neste caso, é inevitável que os atributos sejam preenchidos manualmente, pois as informações não provêm de nenhuma ferramenta, tal como já indicado, pelo que tem que ser o utilizador a preencher os seus atributos, evidentemente pelo facto de ter conhecimento do negócio. Assim, é implementado e devidamente preenchido no Atlas o processo de negócio "Store Purchase".

Para criar o tipo "Indicator" o processo seguido é semelhante ao do tipo "BusinessProcess", pois os metadados não derivam de nenhuma aplicação. O "Indicator" é adicionado recorrendo-se ao método POST: "curl -i -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' -u admin:adminatlaslid4 'node6.dsi.uminho.pt:21000/api/atlas/types' -d @Json/ Indicator.json". Quanto ao preenchimento deste tipo também é efetuado manualmente pelo utilizador. Por último, é representado no Atlas o indicador "Number of sales of products with promotion per quarter".

Contudo, os sobrantes metadados que não foram mencionados em exceções anteriores são preenchidos de forma automática.

Os restantes exemplos de implementação no Atlas dos tipos anteriormente referidos, cujas ilustrações não foram apresentadas, podem ser consultadas nos Apêndices 1.

5.4 Validação do Sistema

Nesta secção é verificado em que medida a solução proposta, ou seja, o grafo proposto, responde aos requisitos. Através das figuras relativas à implementação, pode-se verificar que efetivamente os nós respondem aos requisitos, pois apresentam os metadados devidamente preenchidos. Posteriormente, seguem os requisitos definidos e a explicação da forma como o grafo atende a cada um, ou seja, é indicado o nó ou os nós que existem ou que foram criados para responder ao requisito:

1. **Como são classificadas as entidades (tabelas, colunas, entre todas as outras), a fim de organizá-las através de palavras chave?**
   - Uma *tag* visa classificar as entidades através de palavras chave. Assim, um nó do tipo "Tag", pode estar associado a várias entidades, pelo que indica que de alguma forma os dados dessas entidades estão relacionados. Como se pode verificar no grafo cada um dos nós pode ter uma ou mais *tags* nos seus atributos.
2. **Quais termos são utilizados na organização e como as entidades estão associadas a eles?**
   - Um termo encontra-se integrado num determinado glossário de negócio, por esse facto é concedido o nó "Term". Como se pode observar cada nó tem associado o atributo "term", ou seja, um termo pode estar presente e ser associado a várias entidades.
3. **Com base no modelo BDW, qual é o tipo de cada tabela e dos atributos?**
   - Os atributos acrescentados ao nó "Table", nomeadamente o atributo "objectType" permite averiguar qual o tipo de objeto que envolve. Por outro lado, são acrescentados também metadados ao nó "Column", particularmente o atributo "attributeType" que distingue os atributos descritivos e analíticos. Assim, estes novos metadados possibilitam concluir o tipo de tabelas e colunas que a organização/negócio possui de modo a compreender como se pode utilizar ou aplicar os dados.
4. **Como os dados são avaliados em termos de qualidade?**
   - *Data Quality* e *Data Profiling* são, essencialmente, dos conceitos base presentes e abordados nesta dissertação. Tal como já constatado, o *Data Profiling* consiste em atividades e processos que analisam fontes de dados com o objetivo de gerar informação e determinar metadados sobre um conjunto de dados, sendo definido como uma das subatividades de *Data Quality*. Assim, tem como objetivo detetar sistematicamente erros, inconsistências, redundâncias e dados incompletos. Existe uma série de metadados que são mais simples de calcular, sendo designados por estatísticas simples, por exemplo, o número de valores nulos, valores distintos, o tipo de dados ou os padrões mais frequentes. Assim, quando um processo de *profiling* é despoletado origina os nós "TableQualityStatistics" e "ColumnQualityStatistics", pelo que estes são desenvolvidos para responder a esta questão e englobam as estatísticas de qualidade de tabelas e colunas, respetivamente.
5. **Como é possível integrar diferentes tabelas, considerando diferentes medidas de similaridade?**
   - Contudo, existem outros metadados que são mais complexos, pois implicam a correlação e associação entre colunas. Posto isto, foi delineado o nó "IntraStatistics" para verificar o valor

da correlação entre colunas da mesma tabela e o nó "InterStatistics" para analisar a percentagem de valores comuns entre colunas de tabelas distintas. Portanto, estes nós identificam dependências e similaridades entre colunas.

6. Que permissões estão definidas para determinados utilizadores e determinados dados?
   - Particularmente o nó "Policy" por si só já permite responder a esta questão, na medida em que reúne as caraterísticas que permitem averiguar as permissões e os acessos fornecidos às bases de dados, tabelas e colunas e, ainda, que utilizadores ou grupos têm articulados esses privilégios. Exemplificando, o utilizador "ines" que pertence ao grupo "masterslid4", pode ter associadas as permissões de "select", "read" e "update" sobre todas as tabelas e colunas da base de dados "tpcds".
7. Que processos de negócio estão incluídos no BDW e que dados envolve cada um?
   - Adicionalmente, existe o nó "BusinessProcess" que descreve os processos de negócio da organização, e a forma como estão associados ao BDW, ou seja, que atividades envolve e quais dados manipula, podendo ser associado a uma determinada tabela.
8. Quais os indicadores de desempenho da organização e que dados são utilizados no seu cálculo?
   - Como forma de medir o desempenho da organização é necessário ter em conta certos indicadores, indicadores esses que podem ser classificados KPIs. Portanto, o nó "Indicator" responde a este requisito pelo facto de ser medido numa certa janela temporal e envolver determinados dados no seu cálculo, pois pode ser associado a várias tabelas.
9. Que ações são executadas sobre qualquer entidade?
   - Uma das propriedades de cada nó é o atributo "audit" a partir de onde se liga ao nó "Audit" que guarda um histórico das ações executadas sobre a entidade em questão. Existe vários tipos de ação, podendo ser, por exemplo, "Entity Created", "Term Added", "Entity Updated" e "Classification Added". Exemplificando, o tipo "Entity Created" permite averiguar qual a data de criação e quem criou a entidade em questão. Logo, a primeira ação executada sobre qualquer entidade ocorre quando está é criada, pelo que os *audits* guardam todos os detalhes dessa ação.
10. Como fluem os dados no BDW?
    - Sabe-se que uma base de dados pode ter uma ou mais tabelas associadas, tal como uma tabela pode ter uma ou mais colunas associadas, assim sendo, no grafo, pode-se observar a que base de dados corresponde cada tabela e a que tabela está relacionada cada coluna;

- Aumentando a complexidade, por exemplo, um processo do tipo “hive_process” pode executar uma operação de carregamento ou de criação, carregando ou originando uma nova tabela, sendo no primeiro caso a tabela proveniente de um dado *path* e no segundo, proveniente do *join* entre tabelas. Portanto, um processo tem determinadas entradas (*inputs*), continuando o exemplo anterior, pode ser um *path* ou tabelas e também tem saídas (*outputs*), podendo ser uma tabela, em ambos os casos. Consequentemente, o *lineage* de cada coluna é traçado pelo tipo “hive_column_lineage”, por exemplo, quando uma certa tabela é criada por um determinado processo, o *lineage* das colunas indica as colunas dos *inputs* e dos *outputs* e é associado a esse processo.

6. CONCLUSÕES

A realização dos primeiros capítulos deste documento tem como objetivo fundamental perceber o contexto em que se insere esta dissertação e os potenciais benefícios e contributos que o seu desenvolvimento pode trazer para a área em estudo.

No capítulo do enquadramento conceptual procedeu-se à revisão de literatura sobre *Data Warehouses*, *Big Data* e *Big Data Warehouses*, assim como a abordagem aos diferentes tipos de bases de dados existentes. Desse modo, foi possível "arrumar" as ideias mais precisamente, de modo a enquadrar e relacionar o presente tema com estes conceitos. O ponto de partida fundamental para se compreender a pertinência da dissertação deve-se, essencialmente, ao facto de um dos desafios identificados no *Big Data* ser a Governança de Dados. Posteriormente, fez-se uma revisão do estado da arte da Governança de Dados e das áreas de conhecimento que esta abrange, bem como alguns dos estudos associados a este assunto. Por conseguinte, é possível constatar que existe uma grande discrepância entre a informação disponível acerca das temáticas que foram alvo de revisão. Portanto, Governança de Dados é sem dúvida um tema recente e pouco explorado, ainda para mais aplicado ao conceito de *Big Data Warehouses*.

No capítulo posterior, relativo ao enquadramento tecnológico, foi exibida a ferramenta base que irá sustentar o trabalho futuro desta dissertação. Essa ferramenta encontra-se presente no ecossistema Hadoop, sendo designada por Atlas. O Atlas apresenta uma arquitetura capaz de satisfazer as necessidades e objetivos que foram propostos, pois apresenta a capacidade de integrar múltiplas fontes de dados para a Gestão de Metadados, por exemplo, provenientes do Hive, tecnologia também existente no Hadoop. Para a Gestão de Metadados o Atlas apresenta uma peculiaridade interessante de realçar, que diz respeito ao facto de incorporar no seu sistema uma base de dados de grafos, o JanusGraph, também explorada neste capítulo. Logo, será esta a base de dados onde será armazenado o catálogo com os metadados do *Big Data Warehouse* (por exemplo, tabelas, atributos, entre outros).

No capítulo do *design* e desenvolvimento do grafo para a catalogação e Governança de Dados de um Big Data Warehouse, com base na revisão de literatura realizada em capítulos anteriores, definem-se os requisitos que devem ser respondidos por esse grafo. Posteriormente, procede-se à exploração do Atlas, pois é importante perceber o que este já armazena relativamente a metadados para de seguida serem definidos os componentes de etiquetagem e rastreio do grafo. Averiguou-se que o Atlas incorpora uma REST API a partir de onde é possível ler, editar e criar novos tipos de dados. Finalmente, o grafo é

modelado e definido, pelo que inclui os tipos adicionados (novos), os tipos que se baseiam em outras ferramentas e os tipos incorporados no Atlas, assim como a extensão de alguns deles.

Na demonstração do grafo, primeiramente faz-se a caracterização da infraestrutura física que suporta o ambiente de implementação do grafo e realiza-se uma breve apresentação dos dados selecionados. Ainda neste capítulo, são instanciados alguns cenários do grafo que são possíveis de representar com os dados utilizados. Estes cenários modelam alguns dos tipos e entidades presentes no Atlas e outros que são adicionados nas fases de *design* e desenvolvimento do grafo. Por fim, na fase de validação, realiza-se a validação do grafo, ou seja, verifica-se de que modo é que este responde aos requisitos que foram definidos. Posto isto, após cada requisito segue a resposta que inclui os nós que patenteiam exatamente o que é pedido.

6.1 Trabalho Realizado

Um sistema de catalogação e governança de dados que se baseia num grafo é a proposta desta dissertação, tendo como finalidade a governança de um *Big Data Warehouse*, de modo a permitir que se lide melhor com o crescente aumento dos dados e dos processos de negócio de uma determinada organização. Desta forma, pode-se dizer que os objetivos propostos para este trabalho foram atingidos, pois analisando esses mesmos objetivos e comparando com o resultado final é possível concluir-se que:

- Foram definidos os componentes de etiquetagem e rastreio (*tagging* e *lineage*) do sistema de catalogação e governança, e como estes devem ser representados e armazenados numa base de dados de grafos;
- Como o objetivo principal da presente dissertação é essencialmente recolher e reunir o catálogo de metadados do *Big Data Warehouse*, não faz sentido desenvolver uma ferramenta ou API própria, uma vez que já existe a ferramenta Atlas, pelo que nem o tempo nem o esforço investido estariam associados a uma dissertação deste tipo. O Atlas incorpora uma REST API a partir de onde é possível ler, editar e criar novos tipos de dados, sendo que já permite a etiquetagem e rastreio tendo em consideração informação que deriva de várias fontes, por exemplo do Hive. Todavia, como alguns componentes foram acrescentados são adicionados novos tipos, que podem advir de outras ferramentas, assim como são estendidos outros por via dessa REST API;
- O Atlas incorpora a base de dados de grafos JanusGraph, pelo que todos os componentes, sejam eles os novos ou os já existentes, estão armazenados nessa base de dados, ou seja a base de dados contém o catálogo do *Big Data Warehouse* com os metadados das várias entidades.

Em conclusão, pode-se afirmar que a dissertação traz um contributo importante para a área de Governança de Dados, tópico de pesquisa bastante atual, com quase nenhuma contribuição em *Big Data Warehousing*. Assim, o resultado da dissertação oferece um conjunto de componentes que poderão ser utilizadas e aplicados a vários contextos de negócio.

6.2 Dificuldades, Limitações e Trabalho Futuro

Ao longo do desenvolvimento desta dissertação a maior dificuldade sentida esteve sempre associada à escassez de informação relativamente à área em estudo. Como tal, foi necessário um grande espírito crítico até se chegar ao resultado final.

Por outro lado, apesar do Atlas ser a ferramenta fundamental neste trabalho, também foi a geradora de algumas dificuldades, nomeadamente no momento da sua exploração, pois foi trabalhoso compreender o seu modo de funcionamento, ou seja, a forma como armazena e modela as informações, mais uma vez devido à falta de informação.

Outra dificuldade prende-se com o facto de que alguns os metadados adicionados não serem preenchidos de forma automática no Atlas, devido a limitações da ferramenta. Logo, para colmatar esta situação, uma solução passará por o utilizador preencher estes atributos manualmente a partir da interface do Atlas, uma vez que possivelmente se encontra contextualizado com o negócio em questão. Contudo, no futuro, é expectável que tais metadados sejam preenchidos de forma automática.

Ainda relativamente ao Atlas, foi constatado que quando se inserem novas linhas a tabelas o Atlas não considera essa ação, ou seja, não está a considerar que houve um "update" à entidade, tal como guarda os detalhes quando se cria uma nova tabela pelo que uma provável melhoria será estender os tipos de operações dos processos e assim será possível guardar outros tipos de "updates".

Contudo, será importante no futuro estender e completar a ferramenta Atlas no sentido de serem introduzidas algumas melhorias, especialmente as que foram mencionadas, mas também de forma a tornar a ferramenta mais automática e interativa, por exemplo, permitindo a representação de gráficos.

Para além disso, é necessário avaliar de que forma este modelo é fiável e cumpre os requisitos e necessidades dos especialistas. Para executar esta avaliação, deverá ser realizado uma espécie de questionário a vários especialistas da área e posterior análise dos resultados obtidos. Ainda assim, o grafo também poderá e deverá evoluir com o tempo no sentido de englobar novos conceitos e teorias, tornando-se cada vez mais abrangente.

Em conclusão, pode-se afirmar que o trabalho traz um contributo importante para a área de Governança de Dados, conceito tão atual, mas muito imaturo, nomeadamente quando se emprega ao

contexto de *Big Data Warehouse*. Assim, o resultado da dissertação oferece um conjunto de componentes que poderão ser utilizadas e aplicados a vários contextos de negócio.

Referências Bibliográficas

Abedjan, Z., Golab, L., & Naumann, F. (2015). Profiling Relational Data: A Survey. *The VLDB Journal*, *24*(4), 557–581. https://doi.org/10.1007/s00778-015-0389-y

Acharjya, D. P., & Ahmed, K. (2016). A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *7*(2). https://doi.org/10.14569/IJACSA.2016.070267

Alekseev, A. A., Osipova, V. V., Ivanov, M. A., Klimentov, A., Grigorieva, N. V., & Nalamwar, H. S. (2016). Efficient data management tools for the heterogeneous big data warehouse. *Physics of Particles and Nuclei Letters*, *13*(5), 689–692. https://doi.org/10.1134/S1547477116050022

Alodadi, M., & Janeja, V. P. (2015). Similarity in Patient Support Forums Using TF-IDF and Cosine Similarity Metrics. *2015 International Conference on Healthcare Informatics*, 521–522. https://doi.org/10.1109/ICHI.2015.99

Alserafi, A., Abelló, A., Romero, O., & Calders, T. (2016). Towards Information Profiling: Data Lake Content Metadata Management. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 178–185. https://doi.org/10.1109/ICDMW.2016.0033

Ardagna, D., Cappiello, C., Samá, W., & Vitali, M. (2018). Context-aware data quality assessment for big data. *Future Generation Computer Systems*, *89*, 548–562. https://doi.org/10.1016/j.future.2018.07.014

Atlas. (2018). Apache Atlas. Obtido 18 de Dezembro de 2018, de https://atlas.apache.org/

Banek, M., Vrdoljak, B., & Tjoa, A. M. (2007). Using Ontologies for Measuring Semantic Similarity in Data Warehouse Schema Matching Process. *2007 9th International Conference on Telecommunications*, 227–234. https://doi.org/10.1109/CONTEL.2007.381876

Barateiro, J., & Galhardas, H. (2005). A Survey of Data Quality Tools. *Datenbank-Spektrum*, *14*, 15–21.

Baru, C. K., Bhandarkar, M., Nambiar, R. O., Pöss, M., & Rabl, T. (2013). Benchmarking Big Data Systems and the BigData Top100 List. *Big data*, *1*(1), 60–64. https://doi.org/10.1089/big.2013.1509

Batini, C., & Scannapieco, M. (2006). *Data Quality: Concepts, Methodologies and Techniques*. Springer Science & Business Media.

Bernstein, P., Madhavan, J., & Rahm, E. (2011). Generic Schema Matching, Ten Years Later. *PVLDB*, *4*, 695–701.

Bhardwaj, A., Vanraj, Kumar, A., Narayan, Y., & Kumar, P. (2015). Big data emerging technologies: A CaseStudy with analyzing twitter data using apache hive. *2015 2nd International Conference on Recent Advances in Engineering Computational Sciences (RAECS)*, 1–6. https://doi.org/10.1109/RAECS.2015.7453400

Bhogal, J., & Choksi, I. (2015). Handling Big Data Using NoSQL. *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, 393–398. https://doi.org/10.1109/WAINA.2015.19

Bonnet, L., Laurent, A., Sala, M., Laurent, B., & Sicard, N. (2011). Reduce, You Say: What NoSQL Can Do for Data Aggregation and BI in Large Repositories. *2011 22nd International Workshop on Database and Expert Systems Applications*, 483–488. https://doi.org/10.1109/DEXA.2011.71

Brewer, E. (2012). CAP twelve years later: How the «rules» have changed. *Computer*, *45*(2), 23–29. https://doi.org/10.1109/MC.2012.37

Bykau, S., Kiyavitskaya, N., Tsinaraki, C., & Velegrakis, Y. (2010). Bridging the Gap between Heterogeneous and Semantically Diverse Content of Different Disciplines. *2010 Workshops on Database and Expert Systems Applications*, 305–309. https://doi.org/10.1109/DEXA.2010.67

Cabibbo, L., & Torlone, R. (1998). Querying multidimensional databases. Em S. Cluet & R. Hull (Eds.), *Database Programming Languages* (pp. 319–335). https://doi.org/10.1007/3-540-64823-2_18

Cassavia, N., Dicosta, P., Masciari, E., & Saccà, D. (2014). *Data Preparation for Tourist Data Big Data Warehousing*. 419–426. https://doi.org/10.5220/0005144004190426

Cattell, R. (2011). Scalable SQL and NoSQL Data Stores. *SIGMOD Rec.*, *39*(4), 12–27. https://doi.org/10.1145/1978915.1978919

Chan, A. P. C., & Chan, A. P. L. (2004). Key performance indicators for measuring construction success. *Benchmarking: An International Journal*, *11*(2), 203–221. https://doi.org/10.1108/14635770410532624

Chandarana, P., & Vijayalakshmi, M. (2014). Big Data analytics frameworks. *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, 430–434. https://doi.org/10.1109/CSCITA.2014.6839299

Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, *19*(2), 171–209. https://doi.org/10.1007/s11036-013-0489-0

Cho, V., & Ngai, E. W. T. (2003). Data mining for selection of insurance sales agents. *Expert Systems*, *20*(3), 123–132. https://doi.org/10.1111/1468-0394.00235

Clegg, D. (2015). Evolving Data Warehouse and BI Architectures: The Big Data Challenge. *Business Intelligence Journal*, *20*(1), 19–24.

Costa, C., Andrade, C., & Santos, M. Y. (2018). Big Data Warehouses for Smart Industries. Em S. Sakr & A. Zomaya (Eds.), *Encyclopedia of Big Data Technologies* (pp. 1–11). https://doi.org/10.1007/978-3-319-63962-8_204-1

Costa, C., & Santos, M. Y. (2017a). Big Data: State-of-the-art concepts, techniques, technologies, modeling approaches and research challenges. *IAENG International Journal of Computer Science*, *43*, 285–301. Obtido de http://hdl.handle.net/1822/46855

Costa, C., & Santos, M. Y. (2017b). The SusCity Big Data Warehousing Approach for Smart Cities. *Proceedings of the 21st International Database Engineering & Applications Symposium*, 264–273. https://doi.org/10.1145/3105831.3105841

Costa, C., & Santos, M. Y. (2018). Evaluating Several Design Patterns and Trends in Big Data Warehousing Systems. Em J. Krogstie & H. A. Reijers (Eds.), *Advanced Information Systems Engineering* (pp. 459–473). https://doi.org/10.1007/978-3-319-91563-0_28

Costa, M. I. (2019). Data Source Tagging and Lineage in a Big Data Warehouse. Obtido de https://gitlab.com/Thesis3_Lid4/atlas-integration

Cuzzocrea, A., Song, I.-Y., & Davis, K. C. (2011). Analytics over Large-scale Multidimensional Data: The Big Data Revolution! *Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP*, 101–104. https://doi.org/10.1145/2064676.2064695

Dai, W., Wardlaw, I., Cui, Y., Mehdi, K., Li, Y., & Long, J. (2016). Data Profiling Technology of Data Governance Regarding Big Data: Review and Rethinking. Em S. Latifi (Ed.), *Information Technology: New Generations* (pp. 439–450). Springer International Publishing.

DAMA, I. (2014). *DAMA-DMBOK: Data Management Body of Knowledge* (Second edition). Basking Ridge, New Jersey: Technics Publications.

Davenport, T. H., Barth, P., & Bean, R. (2012). How Big Data Is Different. *MIT Sloan Management Review*, *54*, 43–46. Obtido de https://www.hbs.edu/faculty/Pages/item.aspx?num=43026

De Mauro, A., Greco, M., & Grimaldi, M. (2014, Setembro 7). *What is Big Data? A Consensual Definition and a Review of Key Research Topics*. https://doi.org/10.13140/2.1.2341.5048

Dehdouh, K., Bentayeb, F., Boussaid, O., & Kabachi, N. (2015). *Using the column oriented NoSQL model for implementing big data warehouses*. 469–475.

Dijcks, J.-P. (2012). Oracle: Big data for the enterprise. *Oracle White Paper, (June)*. Obtido de https://www.oracle.com/technetwork/topics/bigdata/articles/index.html

Dijkman, R. M., Dumas, M., & Ouyang, C. (2008). Semantics and analysis of business process models in BPMN. *Information and Software Technology*, *50*(12), 1281–1294. https://doi.org/10.1016/j.infsof.2008.02.006

Du, D. (2015). *Apache Hive Essentials*. Obtido de https://books.google.pt/books/about/Apache_Hive_Essentials.html?id=4S7WBgAAQBAJ&redir_esc=y

Dumbill, E. (2013). *Big Data: Making Sense of Big Data*. Obtido de https://www.liebertpub.com/doi/abs/10.1089/big.2012.1503

Durham, E. A., Rosen, A., & Harrison, R. W. (2014). A model architecture for Big Data applications using relational databases. *2014 IEEE International Conference on Big Data (Big Data)*, 9–16. https://doi.org/10.1109/BigData.2014.7004462

Eckerson, W. W. (2009). Performance Management Strategies: How to Create and Deploy Effective Metrics. *Transforming Data with Intelligence*. Obtido de https://tdwi.org/research/2009/01/bpr-1q-performance-management-strategies.aspx

Elmasri, R., & Navathe, S. (2010). *Fundamentals of Database Systems* (6th ed.). USA: Addison-Wesley Publishing Company.

Euzenat, J., & Shvaiko, P. (2013). Classifications of Ontology Matching Techniques. Em J. Euzenat & P. Shvaiko (Eds.), *Ontology Matching* (pp. 73–84). https://doi.org/10.1007/978-3-642-38721-0_4

Fan, J., Han, F., & Liu, H. (2014). Challenges of Big Data analysis. *National Science Review*, *1*(2), 293–314. https://doi.org/10.1093/nsr/nwt032

Fang, H. (2015). Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 820–824. https://doi.org/10.1109/CYBER.2015.7288049

Fang, H., Zhang, Z., Wang, C. J., Daneshmand, M., Wang, C., & Wang, H. (2015). A Survey of Big Data Research. *IEEE Network*, *29*(5), 6–9. https://doi.org/10.1109/MNET.2015.7293298

Fatima, H., & Wasnik, K. (2016). Comparison of SQL, NoSQL and NewSQL databases for internet of things. *2016 IEEE Bombay Section Symposium (IBSS)*, 1–6. https://doi.org/10.1109/IBSS.2016.7940198

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, *35*(2), 137–144. https://doi.org/10.1016/j.ijinfomgt.2014.10.007

Gartner. (2017). Gartner of information technology IT definitions and glossary. Obtido 11 de Novembro de 2018, de https://www.gartner.com/it-glossary/

Gessert, F., Wingerath, W., Friedrich, S., & Ritter, N. (2017). NoSQL database systems: A survey and decision guidance. *Computer Science - Research and Development*, *32*(3), 353–365. https://doi.org/10.1007/s00450-016-0334-3

Golfarelli, M., & Rizzi, S. (2009). *Data Warehouse Design: Modern Principles and Methodologies* (1.ª ed.). New York, NY, USA: McGraw-Hill, Inc.

Goss, R. G., & Veeramuthu, K. (2013). Heading towards big data building a better data warehouse for more data, more speed, and more users. *ASMC 2013 SEMI Advanced Semiconductor Manufacturing Conference*, 220–225. https://doi.org/10.1109/ASMC.2013.6552808

Grolinger, K., Higashino, W. A., Tiwari, A., & Capretz, M. A. (2013). Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, *2*(1), 22. https://doi.org/10.1186/2192-113X-2-22

Gupta, A. (2015). Big Data analysis using Computational Intelligence and Hadoop: A study. *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 1397–1401.

Hadoop. (2018). Apache Hadoop. Obtido 18 de Dezembro de 2018, de https://hadoop.apache.org/

Hai, R., Geisler, S., & Quix, C. (2016). Constance: An Intelligent Data Lake System. *Proceedings of the 2016 International Conference on Management of Data*, 2097–2100. https://doi.org/10.1145/2882903.2899389

Han, J., E, H., Le, G., & Du, J. (2011). Survey on NoSQL database. *2011 6th International Conference on Pervasive Computing and Applications*, 363–366. https://doi.org/10.1109/ICPCA.2011.6106531

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Ullah Khan, S. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, *47*, 98–115. https://doi.org/10.1016/j.is.2014.07.006

He, J. C. (2018). Apache Atlas and JanusGraph—Graph-based Meta Data Management. *IBM Developer*. Obtido de https://developer.ibm.com/articles/apache-atlas-and-janusgraph-graph-based-meta-data-management/

Heise, A., Quiané-Ruiz, J.-A., Abedjan, Z., Jentzsch, A., & Naumann, F. (2013). Scalable Discovery of Unique Column Combinations. *Proc. VLDB Endow.*, *7*(4), 301–312. https://doi.org/10.14778/2732240.2732248

Hive. (2018). Apache Hive. Obtido 18 de Dezembro de 2018, de https://cwiki.apache.org/confluence/display/Hive/Home

Holmes, A. (2012). *Hadoop in Practice*. Greenwich, CT, USA: Manning Publications Co.

Hortonworks. (2017). *Hortonworks Data Platform—Data Governance*. 55.

Hua, M., & Pei, J. (2007, Agosto 12). *Cleaning disguised missing data: A heuristic approach*. 950–958. https://doi.org/10.1145/1281192.1281294

Imhoff, C. (2019). Advanced Data Lineage: The #1 Key to Removing the Chaos in Modern Analytical Environments: Asset Page. Obtido 19 de Março de 2019, de Transforming Data with Intelligence website: https://tdwi.org/whitepapers/2019/01/diq-all-octopai-advanced-data-lineage/asset.aspx

Inmon, W. H. (2005). *Building the Data Warehouse* (4.ª ed.). New York, NY, USA: John Wiley & Sons, Inc.

Inuwa, I., & N. D. Oye, D. (2015). Design of a Data Warehouse Model for a University Decision Support System. *Journal of Information and Knowledge Management*, *Vol.5*. Obtido de https://www.researchgate.net/publication/289247040

Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., & Shahabi, C. (2014). Big Data and Its Technical Challenges. *Commun. ACM*, *57*(7), 86–94. https://doi.org/10.1145/2611567

JanusGraph. (2018). JanusGraph: Distributed graph database. Obtido 20 de Dezembro de 2018, de http://janusgraph.org/

Juddoo, S. (2015). Overview of data quality challenges in the context of Big Data. *2015 International Conference on Computing, Communication and Security (ICCCS)*, 1–9. https://doi.org/10.1109/CCCS.2015.7374131

Kaisler, S., Armour, F., Espinosa, J. A., & Money, W. (2013). Big Data: Issues and Challenges Moving Forward. *2013 46th Hawaii International Conference on System Sciences*, 995–1004. https://doi.org/10.1109/HICSS.2013.645

Kambatla, K., Kollias, G., Kumar, V., & Grama, A. (2014). Trends in big data analytics. *Journal of Parallel and Distributed Computing*, *74*(7), 2561–2573. https://doi.org/10.1016/j.jpdc.2014.01.003

Kandel, S., Parikh, R., Paepcke, A., Hellerstein, J. M., & Heer, J. (2012). Profiler: Integrated Statistical Analysis and Visualization for Data Quality Assessment. *Proceedings of the International Working Conference on Advanced Visual Interfaces*, 547–554. https://doi.org/10.1145/2254556.2254659

Katal, A., Wazid, M., & Goudar, R. H. (2013). Big data: Issues, challenges, tools and Good practices. *2013 Sixth International Conference on Contemporary Computing (IC3)*, 404–409. https://doi.org/10.1109/IC3.2013.6612229

Kaur, K., & Rani, R. (2013). Modeling and querying data in NoSQL databases. *2013 IEEE International Conference on Big Data*, 1–7. https://doi.org/10.1109/BigData.2013.6691765

Keim, D. A., Panse, C., Sips, M., & North, S. C. (2004). Visual data mining in large geospatial point sets. *IEEE Computer Graphics and Applications*, *24*(5), 36–44. https://doi.org/10.1109/MCG.2004.41

Kerzner, H. (2017). Key Performance Indicators. Em *Project Management Metrics, KPIs, and Dashboards* (pp. 121–171). https://doi.org/10.1002/9781119427599.ch4

Khan, M. A., Uddin, M. F., & Gupta, N. (2014). Seven V's of Big Data understanding Big Data to extract value. *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education*, 1–5. https://doi.org/10.1109/ASEEZone1.2014.6820689

Khan, N., Yaqoob, I., Hashem, I. A. T., Inayat, Z., Mahmoud Ali, W. K., Alam, M., ... Gani, A. (2014). Big Data: Survey, Technologies, Opportunities, and Challenges [Research article]. https://doi.org/10.1155/2014/712826

Khatri, V., & Brown, C. V. (2010). Designing Data Governance. *Commun. ACM*, *53*(1), 148–152. https://doi.org/10.1145/1629175.1629210

Kimball, R., & Caserta, J. (2011). *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. John Wiley & Sons.

Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). Wiley Publishing.

Krishnan, K. (2013). *Data warehousing in the age of big data*. Amsterdam: Morgan Kaufmann is an imprint of Elsevier.

Kubina, M., Varmus, M., & Kubinova, I. (2015). Use of Big Data for Competitive Advantage of Company. *Procedia Economics and Finance*, *26*, 561–565. https://doi.org/10.1016/S2212-5671(15)00955-7

Labrinidis, A., Agrawal, D., Bertino, E., Davidson, S., Dayal, U., Franklin, M., ... Widom, J. (2012). *Challenges and Opportunities with Big Data*. Obtido de https://cra.org/ccc/resources/ccc-led-whitepapers/

Laney, D. (2001). 3D Data Management: Controlling Data Volume, Velocity, and Variety. *Application Delivery Strategies*. Obtido de https://www.bibsonomy.org/bibtex/263868097d6e1998de3d88fcbb7670ca6/sb3000

Leavitt, N. (2010). Will NoSQL Databases Live Up to Their Promise? *Computer*, *43*(2), 12–14. https://doi.org/10.1109/MC.2010.58

Li, Y., & Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 15–19. https://doi.org/10.1109/PACRIM.2013.6625441

Liu, Z., Yang, P., & Zhang, L. (2013). A Sketch of Big Data Technologies. *2013 Seventh International Conference on Internet Computing for Engineering and Science*, 26–29. https://doi.org/10.1109/ICICSE.2013.13

M. Suchanek, F., Abiteboul, S., & Senellart, P. (2011). PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *Proc VLDB Endow*, *5*. https://doi.org/10.14778/2078331.2078332

Maccioni, A., & Torlone, R. (2018). KAYAK: A Framework for Just-in-Time Data Preparation in a Data Lake. Em J. Krogstie & H. A. Reijers (Eds.), *Advanced Information Systems Engineering* (pp. 474–489). https://doi.org/10.1007/978-3-319-91563-0_29

Madhavan, J., Bernstein, P. A., & Rahm, E. (2001). Generic Schema Matching with Cupid. *Proceedings of the 27th International Conference on Very Large Data Bases*, 49–58. Obtido de http://dl.acm.org/citation.cfm?id=645927.672191

Magalhães, J. (2019a). Abordagem Semântica para a Integração de Dados em Big Data Warehouses. Obtido de (to post)

Magalhães, J. (2019b). Smart Crawler for Big Data Integration. Obtido de https://gitlab.com/lid4_uminho/smart-crawler-big-data-integration

Mannino, M. V., Chu, P., & Sager, T. (1988). Statistical Profile Estimation in Database Systems. *ACM Comput. Surv.*, *20*(3), 191–221. https://doi.org/10.1145/62061.62063

Manyika, J., & Chui, M. (2011). *Big data: The next frontier for innovation, competition, and productivity* (M. G. Institute, Ed.). New York: McKinsey Global Institute.

Meier, H., Lagemann, H., Morlock, F., & Rathmann, C. (2013). Key Performance Indicators for Assessing the Planning and Delivery of Industrial Services. *Procedia CIRP*, *11*, 99–104. https://doi.org/10.1016/j.procir.2013.07.056

Michael, K., & Miller, K. W. (2013). Big Data: New Opportunities and New Challenges [Guest editors' introduction]. *Computer*, *46*(6), 22–24. https://doi.org/10.1109/MC.2013.196

Michael Schroeck, Shockley, R., Smart, J., Romero-Morales, D., & Tufano, P. (2012). Analytics: The real-world use of big data - How innovative enterprises extract value from uncertain data. *IBM Institute for Business Value*.

Moawed, S., Algergawy, A., Sarhan, A., Eldosouky, A., & Saake, G. (2014). A Latent Semantic Indexing-Based Approach to Determine Similar Clusters in Large-scale Schema Matching. Em B. Catania, T. Cerquitelli, S. Chiusano, G. Guerrini, M. Kämpf, A. Kemper, ... A. Vakali (Eds.), *New Trends in Databases and Information Systems* (pp. 267–276). https://doi.org/10.1007/978-3-319-01863-8_29

Mohanty, S., Jagadeesh, M., & Srivatsa, H. (2013). *Big Data Imperatives: Enterprise 'Big Data' Warehouse, 'BI' Implementations and Analytics*. Apress.

Morton, K., Balazinska, M., Grossman, D., & Mackinlay, J. (2014). Support the Data Enthusiast: Challenges for Next-generation Data-analysis Systems. *Proc. VLDB Endow.*, *7*(6), 453–456. https://doi.org/10.14778/2732279.2732282

Murthy, K., Deshpande, P. M., Dey, A., Halasipuram, R., Mohania, M., Deepak, P., ... Schumacher, S. (2012). Exploiting Evidence from Unstructured Data to Enhance Master Data Management. *Proc. VLDB Endow.*, *5*(12), 1862–1873. https://doi.org/10.14778/2367502.2367524

Murthy, R., & Goel, R. (2012). Peregrine: Low-latency Queries on Hive Warehouse Data. *XRDS*, *19*(1), 40–43. https://doi.org/10.1145/2331042.2331056

Nambiar, R. O., & Poess, M. (2006). The Making of TPC-DS. *Proceedings of the 32Nd International Conference on Very Large Data Bases*, 1049–1058. Obtido de http://dl.acm.org/citation.cfm?id=1182635.1164217

Nauman, F., Ho, C.-T., Tian, X., Haas, L., & Megiddo, N. (2002). Attribute classification using feature analysis. *Proceedings 18th International Conference on Data Engineering*, 271-. https://doi.org/10.1109/ICDE.2002.994725

Naumann, F. (2014). Data Profiling Revisited. *SIGMOD Rec.*, *42*(4), 40–49. https://doi.org/10.1145/2590989.2590995

Oussous, A., Benjelloun, F.-Z., Ait Lahcen, A., & Belfkih, S. (2018). Big Data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences*, *30*(4), 431–448. https://doi.org/10.1016/j.jksuci.2017.06.001

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Philip Chen, C. L., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, *275*, 314–347. https://doi.org/10.1016/j.ins.2014.01.015

Pipino, L. L., Lee, Y. W., & Wang, R. Y. (2002). Data Quality Assessment. *Commun. ACM*, *45*(4), 211–218. https://doi.org/10.1145/505248.506010

Poosala, V., Haas, P. J., Ioannidis, Y. E., & Shekita, E. J. (1996). Improved Histograms for Selectivity Estimation of Range Predicates. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 294–305. https://doi.org/10.1145/233269.233342

Raman, V., & Hellerstein, J. M. (2001). Potter's Wheel: An Interactive Data Cleaning System. *Proceedings of the 27th International Conference on Very Large Data Bases*, 381–390. Obtido de http://dl.acm.org/citation.cfm?id=645927.672045

Rodrigues, A. (2017). Data Profiling: Identification of Data Quality Problems through Data Analysis. *Instituto Superior Técnico - Campus Taguspark*, 10.

Russom, P. (2011). Big data analytics. *TDWI Research*.

Russom, P. (2014). Evolving Data Warehouse Architectures In the Age of Big Data. *The Data Warehouse Institute*.

Sá, J. V. de O. e. (2010). *Metodologia de sistemas de data warehouse*. Obtido de http://repositorium.sdum.uminho.pt/handle/1822/10663

Santos, M., & Ramos, I. (2006). *Business Intelligence: Tecnologias da informação na gestão de conhecimento*. FCA-Editora de Informática.

Santos, M. Y., & Costa, C. (2016a). Data Models in NoSQL Databases for Big Data Contexts. Em Y. Tan & Y. Shi (Eds.), *Data Mining and Big Data* (pp. 475–485). https://doi.org/10.1007%2F978-3-319-40973-3_48

Santos, M. Y., & Costa, C. (2016b). Data Warehousing in Big Data: From Multidimensional to Tabular Data Models. *Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering*, 51–60. https://doi.org/10.1145/2948992.2949024

Santos, M. Y., & Costa, C. (2019). *Big Data: Concepts, Warehousing and Analytics*. Obtido de https://www.fca.pt/pt/catalogo/informatica/bases-de-dados-sistemas-inteligentes/big-data/

Santos, M. Y., Costa, C., Galvão, J., Andrade, C., Martinho, B. A., Lima, F. V., & Costa, E. (2017). Evaluating SQL-on-Hadoop for Big Data Warehousing on Not-So-Good Hardware. *Proceedings of the 21st International Database Engineering & Applications Symposium*, 242–252. https://doi.org/10.1145/3105831.3105842

Santos, M. Y., Oliveira e Sá, J., Andrade, C., Vale Lima, F., Costa, E., Costa, C., … Galvão, J. (2017). A Big Data system supporting Bosch Braga Industry 4.0 strategy. *International Journal of Information Management*, *37*(6), 750–760. https://doi.org/10.1016/j.ijinfomgt.2017.07.012

Santos, M. Y., & Ramos, I. (2017). *Business Intelligence Da Informação ao Conhecimento*. FCA-Editora de Informática.

Sarsfield, S. (2009). *The Data Governance Imperative*. IT Governance Publishing.

Sathi, A. (2012). *Big Data Analytics*. Obtido de https://www.kobo.com/us/en/ebook/big-data-analytics

Schneider, M. (2008). A general model for the design of data warehouses. *International Journal of Production Economics*, *112*(1), 309–325. https://doi.org/10.1016/j.ijpe.2006.11.027

Simoff, S., Böhlen, M. H., & Mazeika, A. (Eds.). (2008). *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*. Obtido de //www.springer.com/la/book/9783540710806

Smallwood, R. F. (2014). *Information Governance: Concepts, Strategies, and Best Practices*. John Wiley & Sons.

Soares, S. (2013a). *Big Data Governance: An Emerging Imperative* (New edition edition). Boise: MC Press.

Soares, S. (2013b). *IBM InfoSphere: A Platform for Big Data Governance and Process Data Governance* (1 edition). Alexandria, Va.: MC Press.

Stonebraker, M., Bruckner, D., Ilyas, I. F., Beskales, G., Cherniack, M., Zdonik, S. B., … Xu, S. (2013). Data Curation at Scale: The Data Tamer System. *CIDR*.

Suthaharan, S. (2014). Big Data Classification: Problems and Challenges in Network Intrusion Prediction with Machine Learning. *SIGMETRICS Perform. Eval. Rev.*, *41*(4), 70–73. https://doi.org/10.1145/2627534.2627557

Terrizzano, I. G., Schwarz, P. M., Roth, M., & Colino, J. E. (2015). Data Wrangling: The Challenging Yourney from the Wild to the Lake. *CIDR*.

Theodoratos, D., & Sellis, T. (1999). Designing data warehouses. *Data & Knowledge Engineering*, *31*(3), 279–301. https://doi.org/10.1016/S0169-023X(99)00029-4

Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Zhang, N., ... Murthy, R. (2010). Hive—A petabyte scale data warehouse using Hadoop. *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, 996–1005. https://doi.org/10.1109/ICDE.2010.5447738

Tomingas, K., Järv, P., & Tammet, T. (2018). Computing data lineage and business semantics for data warehouse. *Communications in Computer and Information Science*, *914*, 101–124. https://doi.org/10.1007/978-3-319-99701-8_5

Touma, R., Romero, O., & Jovanovic, P. (2015). Supporting Data Integration Tasks with Semi-Automatic Ontology Construction. *Proceedings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP*, 89–98. https://doi.org/10.1145/2811222.2811228

Tria, F. D., Lefons, E., & Tangorra, F. (2014). Design process for Big Data Warehouses. *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, 512–518. https://doi.org/10.1109/DSAA.2014.7058120

Türkmen, G. (2007). *Developing a Data Warehouse for a University Decision Support System*. Atılım Üniversitesi.

Turner, J. R. (2009). *Handbook of Project-based Management: Leading Strategic Change in Organizations*. Obtido de https://www.accessengineeringlibrary.com/browse/handbook-of-project-based-management-leading-strategic-change-in-organizations

Vaish, G. (2013). *Getting Started with NoSQL*. Birmingham: Packt Publishing.

Vaisman, A., & Zimányi, E. (2012). Data Warehouses: Next Challenges. Em M.-A. Aufaure & E. Zimányi (Eds.), *Business Intelligence: First European Summer School, eBISS 2011, Paris, France, July 3-8, 2011, Tutorial Lectures* (pp. 1–26). https://doi.org/10.1007/978-3-642-27358-2_1

Vaisman, A., & Zimányi, E. (2014). *Data Warehouse Systems: Design and Implementation*. Obtido de //www.springer.com/us/book/9783642546549

Varga, J., Romero, O., Pedersen, T., & Thomsen, C. (2014, Setembro 2). *Towards Next Generation BI Systems: The Analytical Metadata Challenge*. 89–101. https://doi.org/10.1007/978-3-319-10160-6_9

Villars, R., Olofson, C., & Eastwood, M. (2011, Junho). *Big Data: What It Is and Why You Should Care*.

Walker, C., & Alrehamy, H. (2015). Personal Data Lake with Data Gravity Pull. *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*, 160–167. https://doi.org/10.1109/BDCloud.2015.62

Ward, J., & Barker, A. (2013). *Undefined By Data: A Survey of Big Data Definitions*. Obtido de arxiv.org/abs/1309.5821

Wigan, M. R., & Clarke, R. (2013). Big Data's Big Unintended Consequences. *Computer*, *46*(6), 46–53. https://doi.org/10.1109/MC.2013.195

Wróbel, A., Komnata, K., & Rudek, K. (2017). IBM data governance solutions. *2017 International Conference on Behavioral, Economic, Socio-cultural Computing (BESC)*, 1–3. https://doi.org/10.1109/BESC.2017.8256387

Zikopoulos, P., & Eaton, C. (2011). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data* (1st ed.). McGraw-Hill Osborne Media.

APÊNDICES

Apêndices 1 – Implementações dos novos tipos no Atlas utilizando o TPC-DS

A Figura 46 exibe as "IntraStatistics" no Atlas, sendo estas denominadas "ss_customer_sk****ss_wholesale_cost- 0.056128264636625236".

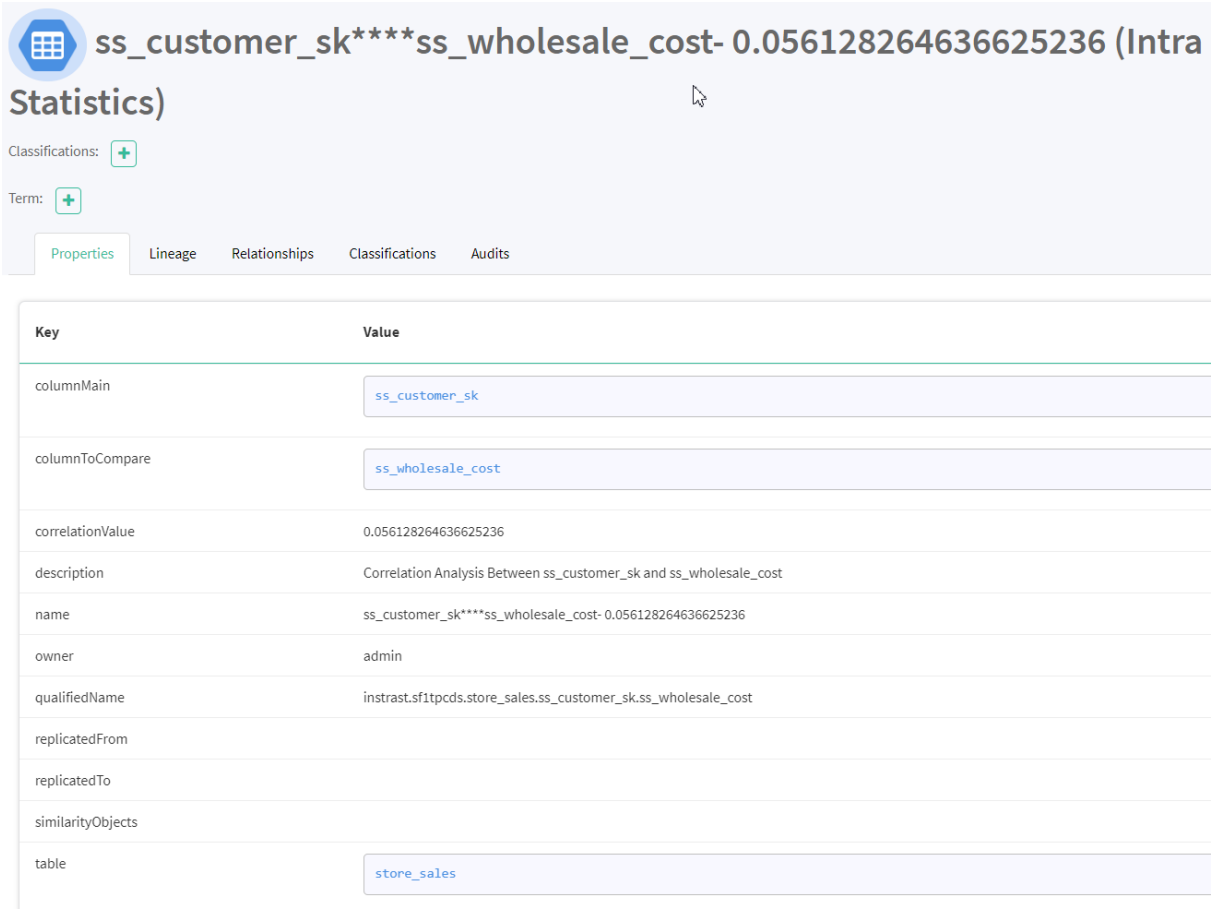

Figura 46. Implementação das "IntraStatistics" no Atlas.

Na Figura 47 estão ilustradas as “InterStatistics” “InterStatistics between ss_customer_sk and c_customer_sk columns | Tables: store_sales and customer” representadas no Atlas.

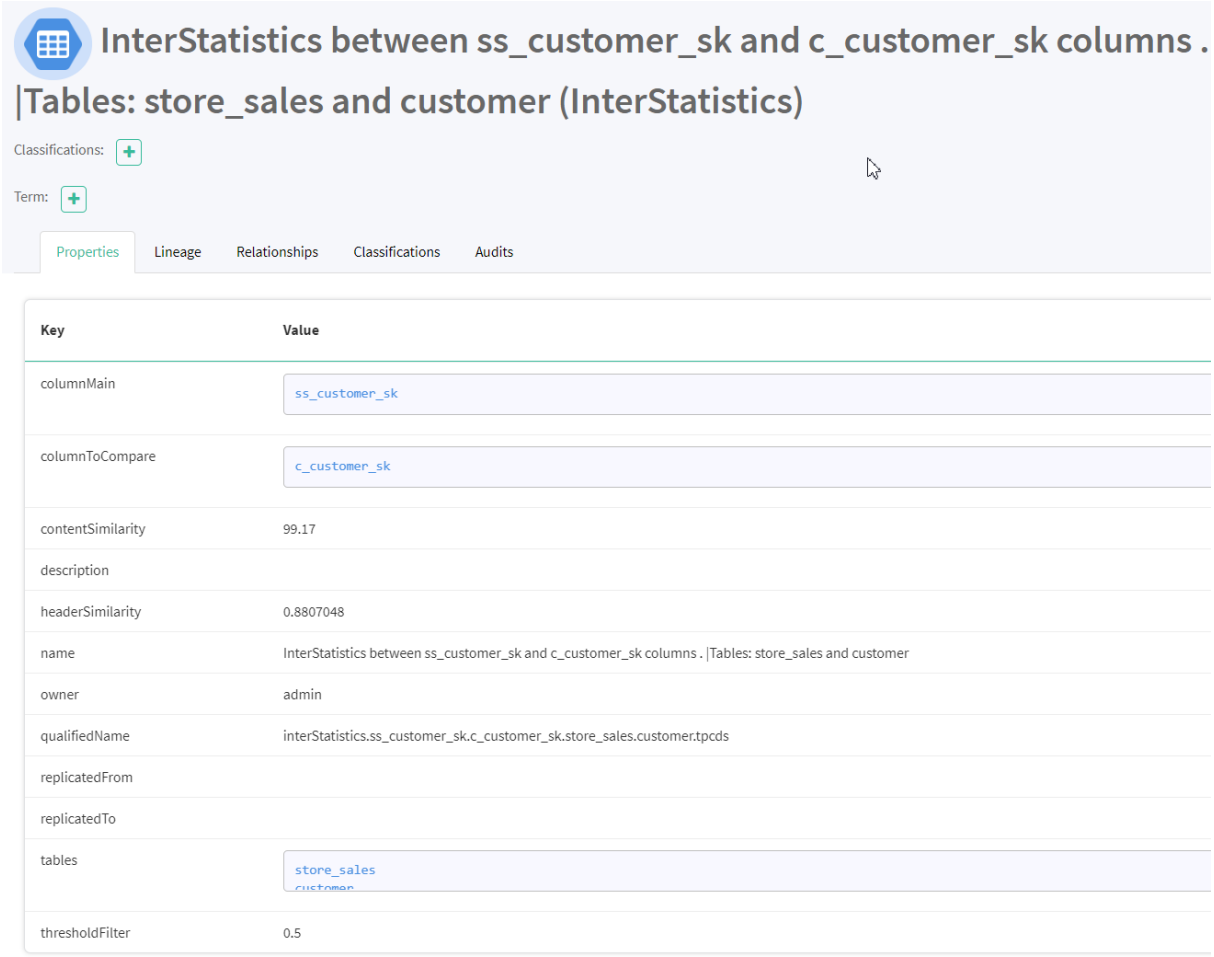

Figura 47. Implementação das “InterStatistcs” no Atlas.

Na Figura 48 que se segue pode-se observar a implementação da “Policy” “tpc” na ferramenta Atlas, já com os atributos devidamente preenchidos.

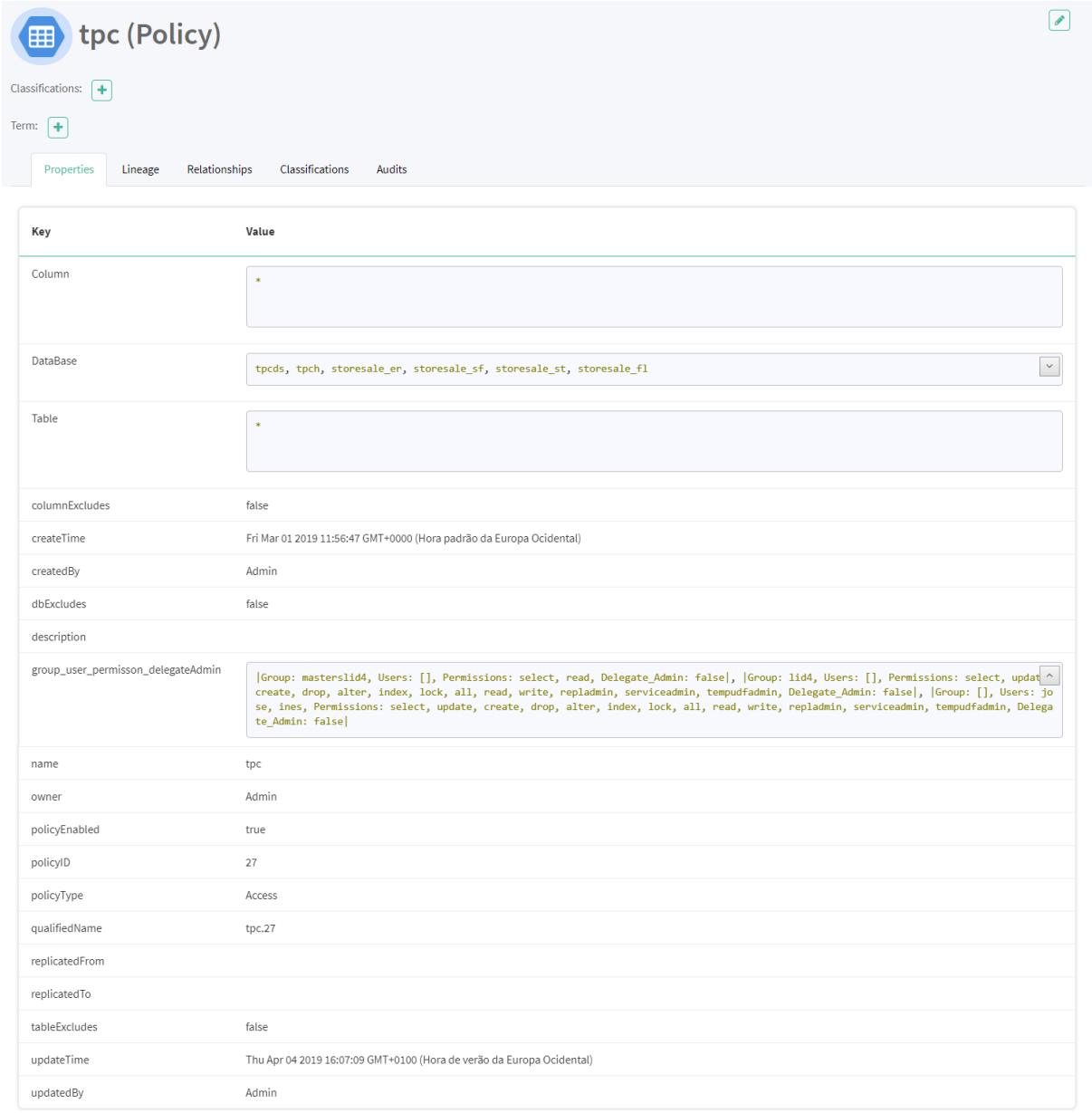

Figura 48. Implementação da Política “tpc” no Atlas.

Na Figura 49 está presente um exemplo do tipo “BusinessProcess”, designado “Store Purchase”.

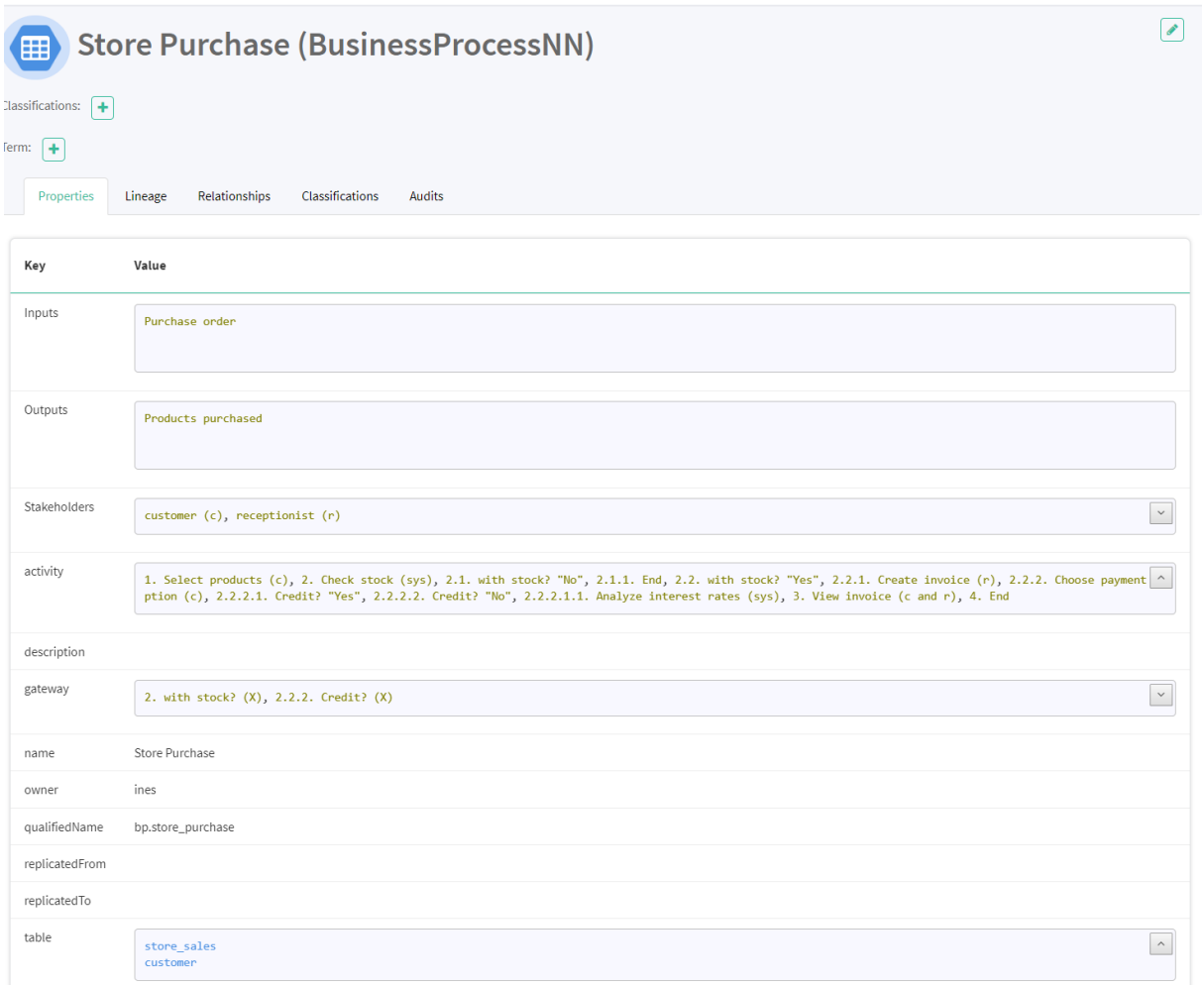

Figura 49. Implementação do Processo de Negócio “Store Purchase” no Atlas.

A Figura 50 expõe o indicador “Number of sales of products with promotion per quarter” presente no Atlas.

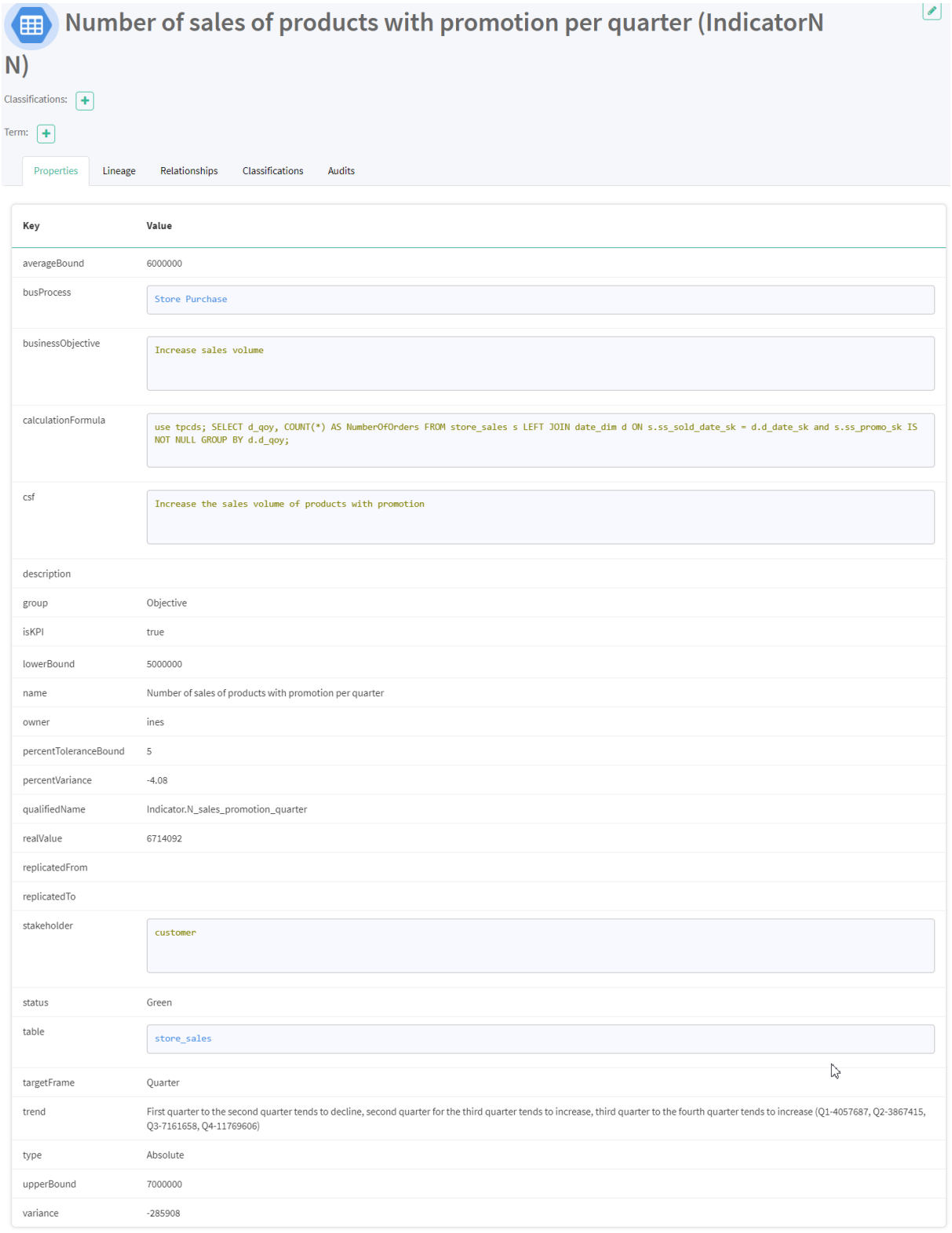

Figura 50. Implementação do Indicador “Number of sales of products with promotion per quarter” no Atlas.