Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ciclo de Vida de Data Warehouse baseado em NoSQL: Adaptações e Análise de Desempenho de Arquiteturas

Beatriz Fragnan P. de Oliveira

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientadora
Prof.a Dr.a Maristela Terto de Holanda

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ciclo de Vida de Data Warehouse baseado em NoSQL: Adaptações e Análise de Desempenho de Arquiteturas

Beatriz Fragnan P. de Oliveira

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof.a Dr.a Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof. Dr.a Célia Ghedini Ralha
CIC/UnB

Prof. Dr. Daniel Cardoso Moraes de Oliveira
UFF

Prof. Dr. Ricardo Pezzuol Jacobi
Coordenador do Programa de Pós-graduação em Informática

Brasília, 23 de fevereiro de 2023

Dedicatória

Dedico esse trabalho ao meu querido avô Anezio e minha mãe, que sempre fizeram todo o possível para me apoiar em todos os aspectos da minha vida. Obrigada minha mãe por sempre incentivar os meus estudos e por nunca medir esforços para que eu me formasse e tivesse minha profissão. Obrigada meu avô, por tudo, por ter sido o meu grande amigo e companheiro para todas as horas. Meu amor e gratidão eternos à você, meu querido e amado vovô. Sou muito feliz por ter sido privilegiada com a sua companhia por quase 37 anos da minha vida. Também dedico ao meu pai, pelo apoio moral à distância e pela compreensão quando nem sempre estive disponível para conversar. Por fim, dedico ao meu esposo Amaro, que sem seu apoio diário e incansável, junto aos nossos 3 filhos, não teria forças físicas para acumular as funções de mãe, esposa e aluna. Os desafios foram muitos, mas você me deu forças para sempre seguir em frente.

Agradecimentos

Gostaria de agradecer a minha orientadora Maristela Holanda, por sua paciência, constância, profissionalismo e preocupação. Suas orientações me guiaram pelo caminho e sua fortaleza me inspirou a enfrentar meus desafios. Também gostaria de agradecer ao professor Márcio Victorino não somente por suas orientações técnicas, mas por sua disponibilidade e pela amizade. Sou muito grata por ter me incentivado a ingressar no mestrado e por ter me apresentado à professora Maristela, que me acolheu prontamente.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

O contexto de *Data Warehouse* (DW) encontra-se em constante transformação nas organizações públicas e privadas. Tendo em vista que os DWs originalmente se apoiavam nos bancos de dados relacionais, com o surgimento do *Big Data*, novas propostas para a gestão de grandes volumes de dados têm sido definidas na literatura, motivando um investimento em soluções alternativas por parte de diversas organizações. Assim, uma das alternativas existentes é utilizar soluções *Not-only* SQL (NoSQL) para processar DW, devido às suas características de flexibilidade e escalabilidade. Tendo em vista o ciclo de vida de DW sugerido por Kimball, foi proposta uma adaptação a esse ciclo de vida convencional para quando se migra para o paradigma NoSQL com diferentes bancos de dados NoSQL pré-selecionados. Em seguida, foi feito um estudo de caso para desenvolver DWs baseados nesses bancos de dados NoSQL (MongoDB e HBase) com a visão do ciclo de vida de Kimball adaptada ao paradigma NoSQL, com o objetivo de validar a proposta e identificar a viabilidade de se trabalhar com a tecnologia NoSQL em DW. No estudo de caso realizou-se uma análise de desempenho da utilização de BDs NoSQL e comparou-se os resultados com os do DW relacional (MySQL). Apesar de o ciclo de vida apresentar diversas etapas, este trabalho enfatizará as etapas de modelagem dimensional e projeto físico, tendo em vista sua relevância no contexto do ciclo de vida de um DW. Com a implementação do estudo de caso foi possível verificar a influência da modelagem de dados no desempenho das consultas selecionadas, assim como de outros parâmetros, fortalecendo as sugestões de adaptação do ciclo de vida de um DW NoSQL para os bancos de dados analisados.

Palavras-chave: Data Warehouse, NoSQL, Big Data, OLAP, NOLAP

Abstract

The context of *Data Warehouse* (DW) is constantly changing in public and private organizations. Bearing in mind that DW were originally based on relational databases, with the emergence of Big Data, new proposals for the management of large volumes of data have been defined in the literature, motivating an investment in alternative solutions on the part of different organizations. Thus, one of the existing alternatives is to use *Not-only* SQL (NoSQL) solutions to process DW, due to its flexibility and scalability characteristics. In view of the DW lifecycle suggested by Kimball, an adaptation to this conventional lifecycle was proposed, for when migrating to the NoSQL paradigm, with different preselected NoSQL databases. Then, a case study was carried out to develop DW's based on these NoSQL databases (MongoDB and HBase) with Kimball's lifecycle vision adapted to the NoSQL paradigm and with the objective of validating the proposal and identifying the feasibility of working with NoSQL technology in DW. In the case study, a performance analysis of the use of NoSQL databases was carried out and the results were compared with those of the relational DW (MySQL). Although the lifecycle has several stages, this work will emphasize the dimensional modeling and physical design stages, considering its relevance in the context of the lifecycle of a DW. With the implementation of the case study, it was possible to verify the influence of data modeling on the performance of the selected queries, as well as other parameters, strengthening the suggestions for adapting the lifecycle of a NoSQL DW for the analyzed databases.

Keywords: Data Warehouse, NoSQL, Big Data, OLAP, NOLAP

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

BD Banco de Dados.

BI *Business Intelligence*.

CRUD Create, Read, Update, Delete.

DM *Data Marts*.

DW *Data Warehouse*.

ETL *Extract-Transform-Load*.

HDFS *Hadoop Distributed File System*.

MD Modelo Dimensional.

MR MapReduce.

MTD Modelo Totalmente Desnormalizado.

NoSQL *Not-only* SQL.

OLAP *Online Analytical Processing*.

OLTP *Online Transaction Processing*.

RAM Random Access Memory.

SAD Sistema de Apoio à Decisão.

SAE Sistemas de Apoio Executivo.

SGBD Sistemas Gerenciadores de Banco de Dados.

SIG Sistemas de Informações Gerenciais.

SPT Sistemas de Processamento de Transações.

SQL *Standard Query Language*.

STC Sistemas de Trabalhadores do Conhecimento.

Capítulo 1

Introdução

Durante a década de 1980, diversas organizações construíram seus sistemas de Banco de Dados (BD) usando o modelo relacional, provocando o surgimento de muitos sistemas de BD com esse paradigma. Logo, durante a década de 1990, as organizações começaram a enfrentar desafios mais complexos em ambientes globais. Essas mudanças provocaram uma transformação no modelo de negócio das empresas, que perceberam a necessidade de um Sistema de Apoio à Decisão (SAD) que pudesse reagir rapidamente às alterações nas condições de negócios [1].

O Banco de Dados (BD) relacional possui uma finalidade diferente do *Data Warehouse* (DW). O primeiro é projetado para otimizar o processamento de consultas que abrangem uma pequena parte do banco, assim como ser rápido em operações Create, Read, Update, Delete (CRUD). Também são estruturados da forma mais eficiente possível, sem informações duplicadas em várias tabelas e geralmente contêm apenas as informações mais atualizadas.

Os DWs foram desenvolvidos para auxiliar na transformação de dados dos sistemas operacionais para os SADs e são otimizados para um número menor de consultas, porém mais complexas, sobre grandes volumes de dados. Os DWs normalmente desnormalizam seus dados, priorizando as operações de leitura sobre as operações de escrita e são projetados desde o início para fins de relatórios e análises. A desnormalização dos dados do DW ocorre para melhorar o seu desempenho, pois a normalização de dados causa um aumento na quantidade de consultas, o que para o DW provoca uma degradação em seu desempenho. Dessa forma, com a chegada de uma nova demanda, surgiram os *Data Warehouses*, capazes de processar consultas complexas de tomada de decisão em dados integrados, históricos e atômicos.

O *Data Warehouse* (DW) é um repositório de dados integrados, obtidos de várias fontes de dados, com o propósito específico de realizar uma análise multidimensional [2], provida pelo processamento analítico *online* ou *Online Analytical Processing* (OLAP). O

projeto de um DW se baseia em sua modelagem dimensional que visa analisar os dados de interesse por meio de medidas e dados descritivos, de forma desnormalizada, e dessa forma, auxiliar no processo de tomada de decisões em um SAD. A abordagem do ciclo de vida de um DW compreende a sequência de passos, em alto nível, por meio dos quais o projeto de desenvolvimento de um DW deve seguir para ser bem-sucedido, abrangendo desde a sua concepção até sua obsolescência [3]. A Figura 1.1 mostra as etapas envolvidas no ciclo de vida de um projeto de DW.

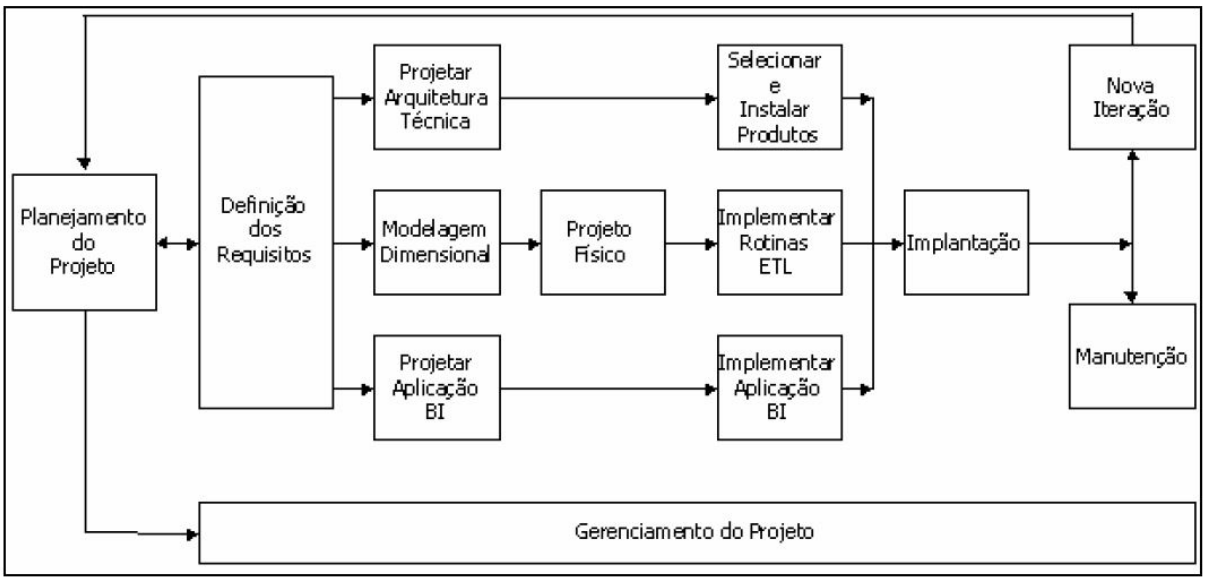

Figura 1.1: Ciclo de vida de um projeto de Data Warehouse. (Fonte: [4]).

A partir do início do Século XXI, com os diversos avanços tecnológicos ocorridos, os processos relacionados aos SADs passaram por transformações, acompanhando novamente a evolução dos modelos de negócio assim como a evolução da geração e tratamento dos dados oriundos de uma organização.

Dessa forma, as transformações provocadas pela emergência do *Big Data* e o contínuo crescimento do volume de dados massivos produzidos por aplicações *web*, *smartphones*, redes sociais, dentre outros, geraram a necessidade de implementar novas tecnologias e formas de armazenamento de dados para as arquiteturas de DW [5], pois junto com essas tecnologias, viu-se não somente o crescimento do volume de dados, mas também o surgimento de novos dados adicionais, como revisão de consumidor e *feedback* sobre experiências, por exemplo, que trouxeram complexidade no processamento e integração de dados. Desde então, uma parte das organizações começou a adotar bases de dados não-relacionais ou *Not-only* SQL (NoSQL) em razão de sua eficiência em processar *Big Data*.

1.1 Definição do Problema

Uma parte importante do ciclo de vida de um DW é a definição da modelagem de dados conceitual e lógica. Essas etapas são tratadas nas atividades da modelagem dimensional e do projeto físico. Quando se trabalha com bancos de dados relacionais essas duas modelagens (conceitual e física) são bem claras e definidas, porém no paradigma NoSQL não. É a partir da modelagem de dados que se implementa o projeto físico e grande parte do desempenho do DW depende da modelagem definida. Logo, as etapas de modelagem dimensional e do projeto físico são muito relevantes para o projeto de um DW como um todo.

Os DWs são uma parte importante dos SAD e, por isso, seu desenvolvimento demanda a utilização de tecnologias que sejam capazes não somente de processar os dados em questão e extrair valor dos mesmos, mas também que propiciem que isso seja feito em tempo hábil para o decisor. A abordagem relacional para modelagem de BD existe desde a década de 1970, é bem consolidada e a mais utilizada até os dias atuais. Devido à sua grande aplicação, assim como a ampla popularidade da linguagem *Standard Query Language* (SQL), é natural que a implementação dos DWs seja feita em sistemas de BDs relacionais. Os DWs relacionais atendem boa parte das demandas existentes, contudo, com o aumento do uso de dados semi-estruturados e com o crescimento contínuo de dados massivos, surgiram requisitos não atendidos completamente pelo modelo relacional existente, devido em parte à normalização dos dados.

A migração para o paradigma NoSQL é uma abordagem promissora que, em alguns cenários, pode melhorar o desempenho dos DWs. Contudo, essa escolha não é trivial, pois nem os bancos relacionais com sua confiabilidade e capacidade de computação, nem os NoSQL com sua velocidade e fácil escalabilidade, ocupam o lugar um do outro por completo. Assim, a mudança para o paradigma NoSQL deve ser um processo planejado.

Apesar de oferecerem maior flexibilidade e escalabilidade horizontal (capacidade de adicionar mais máquinas ao sistema, melhorando seu desempenho), a decisão de que um DW deve ser baseado em uma abordagem NoSQL não é sinônimo de melhora no desempenho do DW. Os trabalhos produzidos até o momento, como, por exemplo, [6] e [7], mostram que o desempenho de um DW NoSQL pode variar de acordo com os esquemas escolhidos que, por sua vez, dependem das consultas a serem realizadas, da categoria e do BD NoSQL escolhidos. A variedade de diferentes Sistemas Gerenciadores de Banco de Dados (SGBD) NoSQL é um fator que também dificulta a escolha de tecnologias adequadas para determinado problema.

A literatura aponta que a flexibilidade do paradigma NoSQL leva a novos problemas, [8], [9],[10] e [11], como, por exemplo, realizar operações OLAP em um DW que não possui

um esquema fixo ou de se definir o melhor método de cálculo de valores agregados. Outro aspecto encontrado que dificulta maior adoção de sistemas NoSQL para DWs recentemente é a inclusão lenta do NoSQL nos *benchmarks* de suporte à decisão existentes [11] e [12].

Trabalhos publicados como [13], [14] e [10], buscam resolver problemas específicos envolvendo a mudança do paradigma relacional para o não-relacional, sem oferecer uma visão de mais alto nível e sem considerar o ciclo de vida ao qual o DW está inserido.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem por objetivo propor uma adaptação ao ciclo de vida de DW, definido por Kimball em [4], quando se migra para o paradigma NoSQL. A validação da proposta foi realizada com um estudo de caso desenvolvendo DWs baseados em bancos de dados NoSQL, com análise de desempenho da utilização de BD NoSQL com a visão do ciclo de vida de Kimball adaptada ao paradigma NoSQL. Apesar do ciclo de vida apresentar diversas etapas, como se verifica na A Figura 1.1, este trabalho enfatiza as etapas de modelagem dimensional e projeto físico, tendo em vista sua relevância no contexto do ciclo de vida de um DW.

1.2.2 Objetivos Específicos

Com a finalidade de propor uma adaptação do ciclo de vida DW quando se migra para o paradigma NoSQL, o presente trabalho busca:

1. Identificar as fases do ciclo de vida do DW que são impactadas pela migração de paradigma e apresentar as possíveis mudanças em cada fase;
2. Especificar as categorias (por exemplo, orientado a documentos, família de colunas, chave-valor ou orientado a grafos) e esquemas de bancos de dados NoSQL de interesse, assim como do BD relacional que será utilizado como referência, para fins de comparação de desempenho com o paradigma não-relacional;
3. Delimitar os dados a serem utilizados, realizar a modelagem dos mesmos e escolher as consultas mais relevantes;
4. Implementar os DWs com os bancos NoSQL e relacional, com os esquemas escolhidos, no mesmo ambiente (nuvem);

5. Realizar uma análise de desempenho e usabilidade de BD NoSQL e compará-la ao DW relacional, para fins de análise da viabilidade de se desenvolver DWs com a tecnologia NoSQL.

1.3 Estrutura do Trabalho

Este trabalho está estruturado da seguinte maneira:

- O Capítulo 2 apresenta a fundamentação teórica envolvendo os principais conceitos e tecnologias utilizadas no desenvolvimento desta pesquisa. São tratados os seguintes assuntos: SAD, Ciclo de Vida de um DW, uma visão geral de Bancos de Dados NoSQL e conceitos introdutórios de *Big Data* e Hadoop.
- O Capítulo 3 descreve sucintamente os principais trabalhos relacionados à pesquisa em questão.
- O Capítulo 4 detalha a proposta de pesquisa, juntamente com os conceitos envolvidos.
- O Capítulo 5 apresenta o estudo de caso, com a especificação do ambiente e do conjunto de dados e a definição da arquitetura de hardware.
- O Capítulo 6 apresenta os resultados do estudo de caso, alinhado com os objetivos da pesquisa.
- O Capítulo 7 apresenta a conclusão desta pesquisa.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta o referencial teórico dos principais conceitos abordados e necessários para a compreensão do trabalho como um todo. A Seção 2.1 apresenta os conceitos relacionados aos Sistemas de Apoio à Decisão e também expõe o ciclo de vida de um DW. Na Seção 2.2 são explicados os conceitos envolvidos em BDs NoSQL, expondo suas vantagens sobre os bancos de dados relacionais no contexto deste trabalho. A Seção 2.3 elucida os conceitos referentes à *Big Data* e, por fim, a Seção 2.4 apresenta o *framework* Apache Hadoop.

2.1 Sistemas de Apoio à Decisão

Sistemas de Apoio à Decisão podem ser definidos, segundo [15], como Sistemas de informação de nível gerencial focados em problemas atípicos e que combinam dados e modelos analíticos mais refinados para apoiar as tomadas de decisões semi-estruturadas e não-estruturadas. O SAD pretende amplificar o conhecimento do decisor. Para ajudar a complementar o entendimento de um SAD é interessante inseri-lo no contexto dos Sistemas de Informação. De maneira sucinta, os Sistemas de Informação são classificados em quatro níveis de decisão de uma organização [15]: operacional, conhecimento, gerencial e estratégico. Suas principais aplicações são relacionadas a seguir. No nível estratégico encontram-se os Sistemas de Apoio Executivo (SAE); no nível gerencial encontram-se os Sistemas de Apoio à Decisão (SAD) e os Sistemas de Informações Gerenciais (SIG), este trabalho focará apenas neste nível de decisão. No nível do conhecimento encontram-se os Sistemas de Trabalhadores do Conhecimento (STC) e os Sistemas de Automação de Escritórios; e, finalmente, no nível operacional estão os Sistemas de Processamento de Transações (SPT) ou Sistemas Transacionais. A Figura 2.1 elucida a compreensão dessa classificação. Tanto o SAD como o SIG atendem ao nível gerencial da organização e auxiliam na tomada de decisão, porém a diferença entre esses sistemas é que o SIG fornece

apoio à decisão de problemas mais rotineiros, utilizando os dados fornecidos pelos SPT e ferramentas usuais como relatórios. Já o SAD visa resolver problemas não rotineiros e mais complexos, respondendo a perguntas não-usuais. Podem consumir informações internas geradas pelos sistemas transacionais e pelos SIG e também dados/informações externas, que objetivam complementar a consulta de interesse. O armazenamento dessas informações ocorre em um repositório multidimensional, chamado de DW, por meio do processo conhecido por *Extract-Transform-Load* (ETL) que extrai, transforma e carrega os dados no DW.

Figura 2.1: Classificação dos sistemas de informação em níveis de decisão. (Fonte: Adaptado de [15]).

2.1.1 *Data Warehouse* e OLAP

O conceito de DW surgiu na década de 1980 para representar o armazenamento de uma grande quantidade de dados oriundos de diversas fontes em uma empresa [5]. Sua concepção ocorreu baseada em Sistemas de Bancos de Dados relacionais e, desde então, com o desenvolvimento de novas tecnologias, essa perspectiva se transformou em um conceito mais amplo que o original. Devido à maturidade do tema, pode-se encontrar diferentes definições na literatura, a seguir serão apresentadas cinco delas.

- O autor William Inmon [16], conhecido como pioneiro na área de DW, o define como uma coleção de dados orientada por assuntos, integrada, não volátil e variável em relação ao tempo, que tem por objetivo apoiar os processos de tomada de decisão;
- A definição dada por Kimball [17] é que um DW é o agrupamento das áreas de *staging* e apresentação do DW de uma organização, onde a cópia dos dados transacionais é estruturada especificamente para o desempenho de consultas e análises, assim como para a facilidade de uso;
- Em [18], Gray e Watson definem DW como sendo tipicamente um sistema de BD dedicado que é separado dos sistemas *Online Transaction Processing* (OLTP) da organização;
- Sen e Jacob [19] definem: DWs são construídos no interesse de suporte à decisão de negócios e contêm dados históricos sumarizados e consolidados provenientes de registros individuais de bancos de dados operacionais;
- O autor Poe [20] define DW como um BD analítico somente leitura, que é usado como base para os SAD.

Dessa forma, apesar das diferentes definições apresentadas, fica evidente o papel central do DW no apoio aos processos de tomada de decisão. A Figura 2.2 apresenta um modelo clássico de DW, definido por [17]. Pode-se observar que a entrada dos dados ocorre pelos sistemas fonte, passando por um processo de ETL, para serem adicionados nos *Data Marts* (DM). A partir disso, os dados podem ser consumidos pelas ferramentas de geração de relatórios/*dashboards* ou até por outros sistemas.

OLAP Outro conceito importante, que está intimamente ligado aos DWs, é o Processamento Analítico Online ou*Online Analytical Processing* (OLAP). Esse possui capacidade de consulta e permite a análise de dados em diversas perspectivas, assim como o cálculo das métricas mais relevantes para o projeto. Enquanto o DW (infraestrutura) armazena os dados de uma maneira eficiente, o OLAP permite a recuperação dos mesmos com rapidez e eficiência [16]. É possível afirmar que ambos os conceitos se complementam, existindo uma relação simbiótica entre eles, tanto que um bom planejamento da apresentação dos dados pela interface OLAP é fundamental para o sucesso do projeto de um DW. Complementando o conceito: OLAP é uma interface com o usuário e não uma forma de armazenamento de dados, porém se utiliza do armazenamento para poder apresentar as informações. É importante o entendimento dessa conexão existente entre OLAP e DW, pois a nomenclatura das diferentes implementações da arquitetura OLAP depende da forma de armazenamento presente no DW [21] e [2]. A seguir, serão apresentadas algumas classificações com seus respectivos métodos de armazenamento.

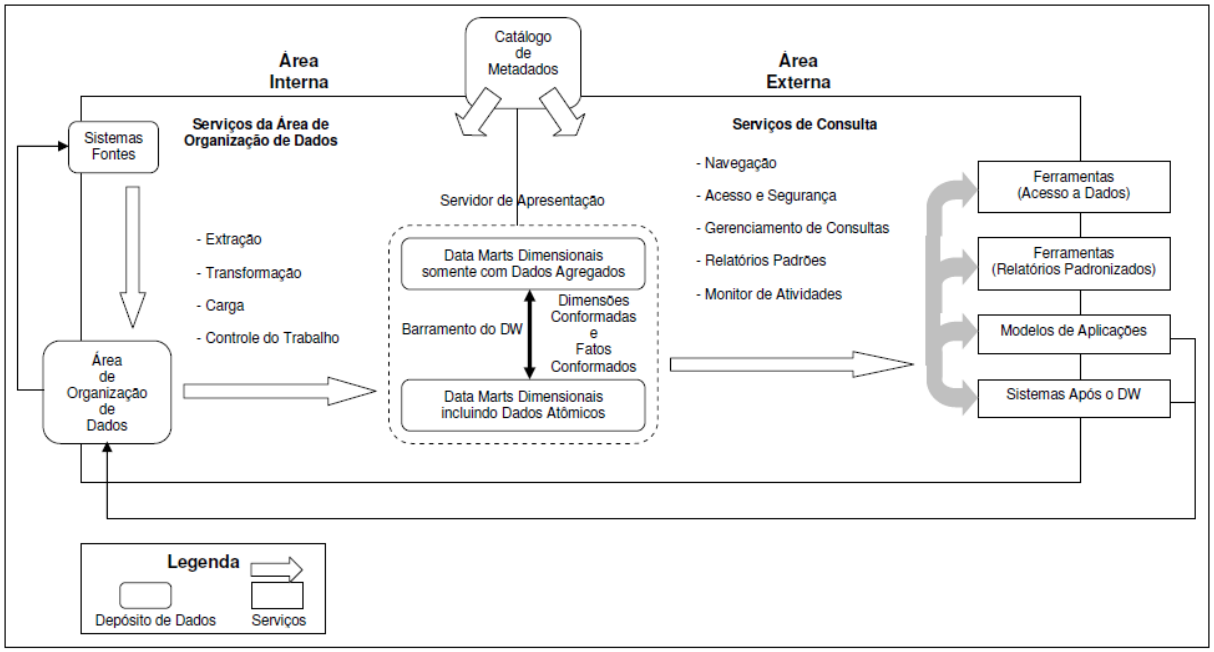

Figura 2.2: Arquitetura técnica de um DW. (Fonte: [17]).

- ROLAP (OLAP Relacional): Os dados são armazenados em um BD relacional.
- MOLAP (OLAP Multidimensional): Os dados são armazenados em um BD multidimensional.
- HOLAP (OLAP Híbrido): Como o próprio nome sugere, é uma combinação dos modelos ROLAP e MOLAP.
- NOLAP (OLAP NoSQL): Os dados são armazenados em um BD NoSQL.
- WOLAP (*Web* OLAP): É a operação de uma ferramenta OLAP de um navegador da Web.
- DOLAP (*Database* OLAP): Fornece portabilidade aos dados. Emite uma consulta para o servidor e recebe as informações de volta para ser analisada localmente.

Cada arquitetura tem suas vantagens e desvantagens decorrentes do BD escolhido. A arquitetura mais utilizada atualmente é a ROLAP [2], pois emprega uma tecnologia estabelecida, com arquitetura padronizada, que pode aproveitar as funcionalidades que a acompanham. Suas vantagens são: possibilidade de ligação com uma grande quantidade de dados, não necessita pré agregação, boa escalabilidade e execução em diferentes plataformas. Sua desvantagem é o baixo desempenho limitado pelas funcionalidades SQL e não possuir funções complexas de análise. A arquitetura NOLAP é pouco conhecida, porém, apresenta vantagens como processamento distribuído, flexibilidade e processamento

de *Big Data*, características diretamente relacionadas aos bancos de dados NoSQL. Os trabalhos relacionados a essa arquitetura serão apresentados no Capítulo 3.

2.1.2 Modelo de Dados

A modelagem dimensional em um DW cumpre um papel decisivo no desempenho das consultas a serem realizadas. Ao modelar um DW, os dados textuais descritivos são separados dos dados numéricos. Assim, os dados textuais relacionados à descrição e caracterização da informação compõem as tabelas conhecidas como dimensões e os dados numéricos as tabelas conhecidas como fato(s) [17], [2] e [16].

As tabelas fato são caracterizadas por uma chave composta pelas chaves estrangeiras das dimensões que se cruzam, envolvidas em um determinado processo. São compostas normalmente por milhões de linhas e podem ser classificadas por suas diferentes características, como: fato transacional, fato agregada, fato consolidada, fato de *snapshot* acumulado, dentre outras.

Já as dimensões são tabelas compostas por grandes campos descritivos. Os atributos da tabela dimensão possuem dois propósitos: filtragem de uma consulta e rotulagem do resultado da consulta. O poder do DW está diretamente relacionado com a qualidade e detalhamento dos atributos da dimensão. Estes devem ser descritivos, verbosos, completos, com poucos valores e de qualidade garantida. As chaves das dimensões podem ser naturais ou artificiais.

Os dois modelos mais conhecidos são o Estrela e o Floco de Neve. No primeiro modelo, as tabelas dimensões são desnormalizadas e não possuem sub-tabelas, enquanto no segundo, as dimensões são normalizadas e podem ter sub-tabelas. O esquema Estrela é uma estrutura otimizada que usa uma tabela fato centralizada que indexa uma ou várias tabelas dimensionais. O esquema estrela é mais fácil de projetar e implementar, além de ser geralmente mais eficiente para consultar, por possuir uma quantidade menor de junções nas tabelas. A Figura 2.3 exemplifica um esquema estrela simplificado. O esquema Floco de Neve é um modelo mais complexo e busca reduzir a redundância no armazenamento, por isso ocupa menos espaço em disco. É caracterizada por dimensões de um esquema que são detalhadas e possuem um relacionamento hierárquico, de modo que as tabelas filhas possuem várias tabelas pai. Seu nome se deve ao desenho que lembra um floco de neve [21]. Esse tipo de esquema dificulta a navegação de dados. A Figura 2.4 mostra um exemplo de esquema floco de neve.

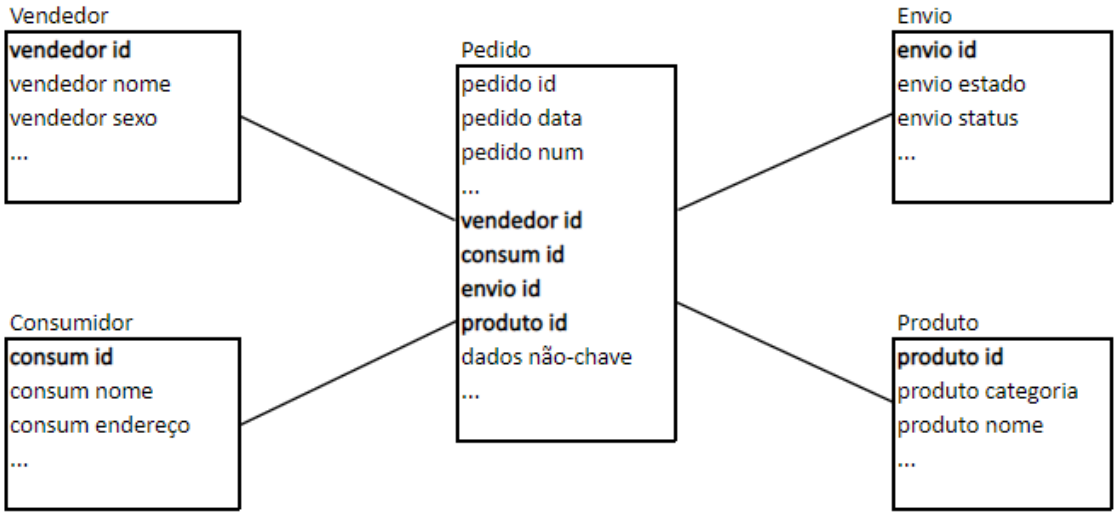

Figura 2.3: Exemplo de esquema Estrela. (Fonte: [4]).

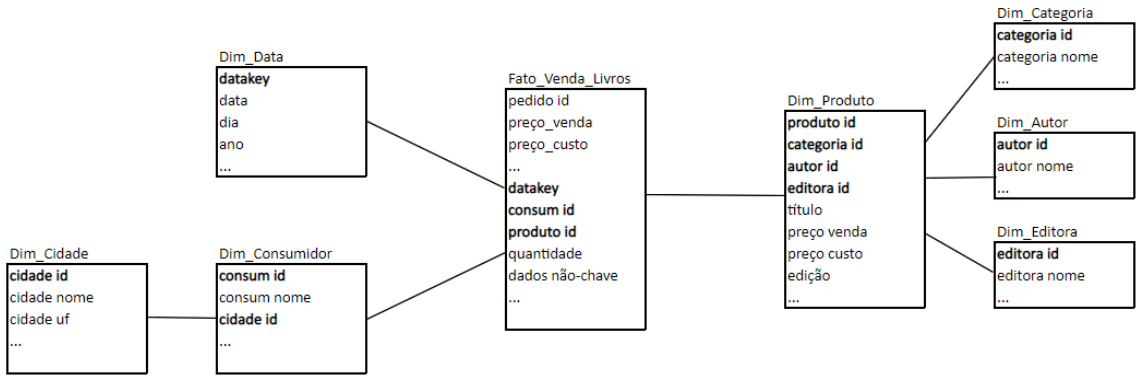

Figura 2.4: Exemplo de esquema Floco de Neve.

2.1.3 Ciclo de Vida de um *Data Warehouse*

Para o desenvolvimento de um DW, Kimball [4] sugere o ciclo apresentado na Figura 2.5. Esse ciclo é composto pelas seguintes atividades:

- Planejamento do Projeto: essa atividade consiste na elaboração do plano do projeto. Esse documento contém o planejamento detalhado da execução de todas as tarefas do projeto.
- Definição dos Requisitos: a definição dos requisitos pode ser dividida em duas tarefas:

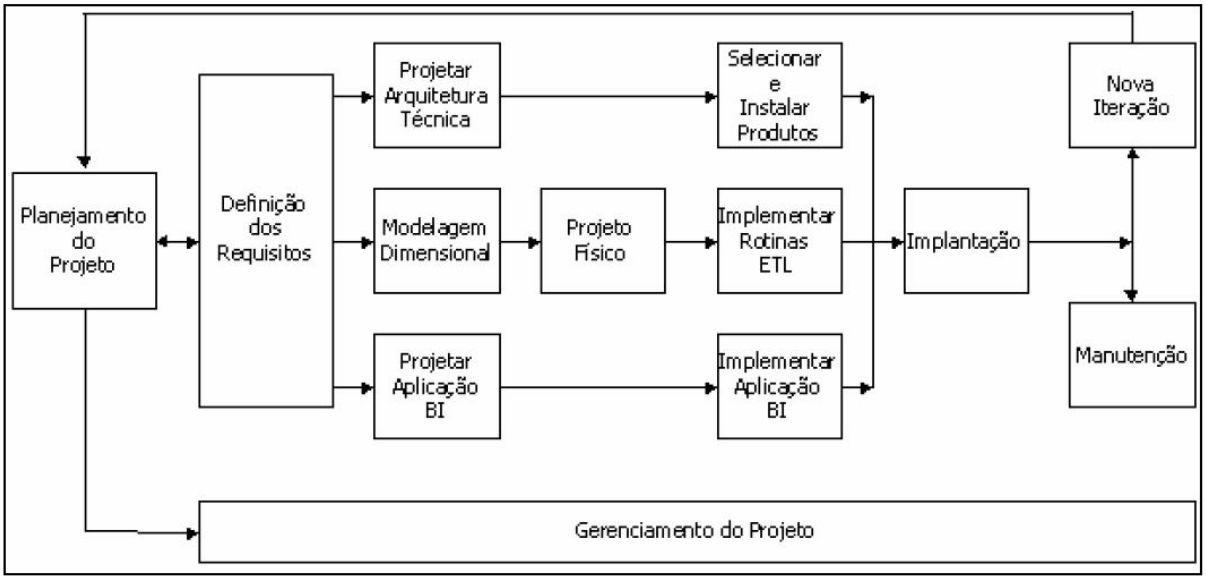

Figura 2.5: Ciclo de vida de um projeto de DW tradicional. (Fonte: [4]).

1. Levantamento das Necessidades do Negócio: consiste no levantamento das carências que o sistema deverá atender. Nesta etapa identifica-se o público-alvo e os requisitos funcionais e não funcionais para o DW.
2. Levantamento das Fontes de Dados: consiste no levantamento e análise inicial das fontes de dados, incluindo o levantamento e análise da documentação existente sobre as fontes identificadas, tais como modelos de dados e correspondentes metadados, documentação sobre detalhes a respeito de regras de negócio, tabelas, arquivos e demais artefatos. Tal atividade promoverá o entendimento sobre o negócio e seus dados, bem como auxiliará no planejamento do projeto e na identificação de riscos correspondentes às fontes de dados.

- Projetar Arquitetura Técnica: a atividade projetar a arquitetura técnica consiste no levantamento dos volumes envolvidos, tanto no que tange às bases de dados quanto aos volumes de processamento e quantidade de usuários simultâneos. Também são organizados os softwares que comporão as camadas da arquitetura OLAP.
- Modelagem Dimensional: consiste na elaboração do modelo dimensional de dados do DW de acordo com o escopo da iteração do projeto. Nessa atividade são identificadas as dimensões e fatos relativos aos temas da iteração. Para cada fato são definidas tanto as dimensões envolvidas quanto o seu conteúdo e o grão dos dados.
- Projetar Aplicação de *Business Intelligence* (BI): essa atividade consiste na definição das consultas previstas para a iteração e a documentação dos indicadores identifi-

cados. Inclui o levantamento das consultas e análises demandadas pelos usuários e que serão a base para a homologação correspondente ao ciclo.

- Projeto Físico: consiste na construção do modelo físico relacional e multidimensional. O modelo físico deriva do modelo lógico e de tratamentos de desempenho e controle; o modelo multidimensional depende das visões e consultas a serem liberadas para os usuários (consultas solicitadas, perfil de usuário etc).
- Selecionar e Instalar Produtos: consiste em preparar o ambiente de desenvolvimento, disponibilizando o software e o hardware configurados.
- Implementar Rotinas ETL: consiste na especificação e documentação dos processos de extração, transformação e carga ETL dos dados. Inclui a obtenção dos dados que alimentarão o DW a partir da extração dos dados das bases transacionais.
- Implementar Aplicação de *Business Intelligence*: consiste na implementação das consultas e das fórmulas que gerarão os indicadores solicitados pelos usuários.
- Implantação: consiste na disponibilização dos temas desenvolvidos na iteração para o DW no ambiente de produção e avaliação de desempenho. Essa fase inclui também a passagem da aplicação do ambiente de desenvolvimento para o ambiente de produção, a documentação do processo e o treinamento dos técnicos que darão sustentação ao aplicativo.
- Nova Iteração: consiste em planejar a próxima iteração levando em conta aspectos ressaltados pelos desenvolvedores ou usuários dos produtos já disponibilizados pelas iterações finalizadas.
- Manutenção: consiste em manutenir algum componente da aplicação instalada que não esteja em conformidade com as especificações do projeto ou evoluir algum requisito por solicitação dos usuários.
- Gerenciamento do Projeto: concentra as tarefas relacionadas ao acompanhamento e controle da execução do projeto.

Conforme mencionado no Capítulo 1, o ciclo de vida convencional de um projeto de DW relacional compreende a sequência de passos, em alto nível, pelos quais o projeto de desenvolvimento do DW deve seguir para ser bem-sucedido. A Figura 2.5 apresenta o ciclo de vida de um projeto de DW convencional. Nela, é possível observar 3 trilhas: a de tecnologia, a de dados e a de aplicação de BI. Enquanto as setas mostram o fluxo das atividades ao longo de cada trilha, ficam implícitas as dependências existentes entre as atividades/tarefas de cada trilha, conforme ilustrado pelo alinhamento vertical de suas caixas. De acordo com Kimball, o maior esforço requerido para se construir um DW

concentra-se na trilha de dados [4] e, por essa razão, este trabalho se concentrará na análise apenas da trilha de dados, com foco nas caixas que tratam a modelagem dimensional e o projeto físico.

2.1.4 Modelagem Dimensional

A modelagem dimensional é uma técnica de *design* lógica para estruturar os dados de maneira que sejam intuitivos para os usuários do negócio e para que forneçam um melhor desempenho nas consultas [4]. O projeto de modelos dimensionais em apoio aos relatórios de negócio e às necessidades analíticas requer uma abordagem diferente daquelas usadas no projeto de bancos transacionais. Na modelagem dimensional é necessário identificar a granularidade da tabela fato, definir as tabelas dimensões e seus atributos relacionados, assim como as tabelas fatos numéricas[4]. As tabelas fato são rodeadas pelas tabelas dimensões e pelo contexto textual relacionado a elas no momento de sua carga. Uma empresa pode ter um modelo dimensional para cada conjunto de processos que deseje analisar [17].

De uma maneira geral, o processo de modelagem dimensional pode ser resumido em quatro passos [4]:

1. Escolha do processo;
2. Declaração do grão;
3. Identificação das dimensões;e
4. Identificação das fatos.

Assim, constrói-se o modelo dimensional baseado em dois fatores: os requisitos do negócio e a realidade dos dados. De acordo com Kimball, modelos dimensionais armazenados em bancos de dados relacionais são chamados de esquemas estrela ou floco de neve e aqueles armazenados em estruturas de processamento analítico online multidimensionais são chamadas de cubos.

2.1.5 Projeto Físico

Na etapa do projeto físico do ciclo de vida convencional, são apresentados todos os elementos para o projeto lógico, assim como são definidos os passos e parâmetros para transformar o projeto lógico no físico. A Figura 2.6 mostra o processo do projeto físico em alto nível. Observa-se a definição de tabelas de *staging*, que são tabelas temporárias com dados a serem transformados, de estruturas de dados adicionais, assim como os índices das tabelas. Verifica-se que o estágio final é o planejamento de detalhes do projeto físico,

como, por exemplo, o particionamento e o *layout* de disco. É possível que no início deste estágio do ciclo de vida realize-se iterações sobre o modelo de dados físico, o que permite a adição de novos elementos relacionados à indexação e segurança ou até um pequeno ajuste de tabelas e colunas. No projeto físico são definidas políticas e padrões do sistema, mapeamento da fonte de dados com o objetivo (DW em si), a ferramenta de modelagem dos dados, tamanho dos índices nas tabelas fato, valores nulos ou não nulos nas tabelas dimensões e em colunas de datas, dentre outros. Em cada caixa definida na figura acima há uma infinidade de tarefas embutidas, elas não serão descritas neste trabalho tendo em vista que o objetivo é o foco no ciclo de vida NoSQL.

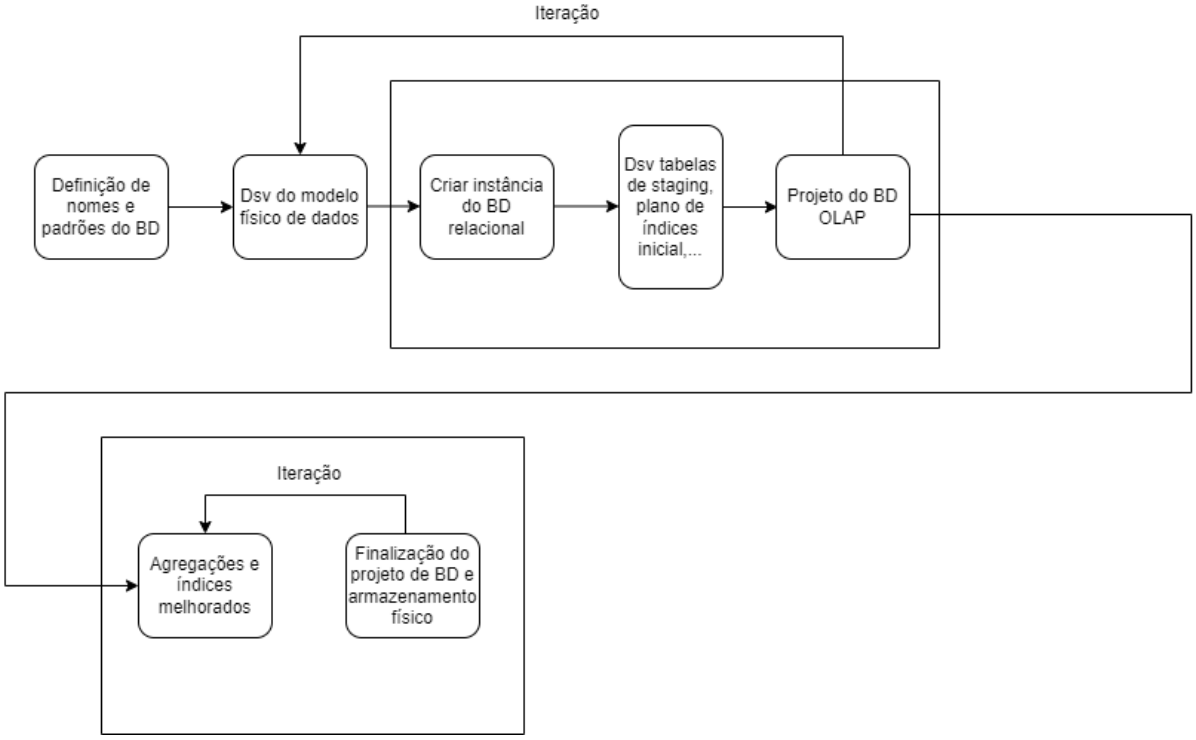

Figura 2.6: Camada física em alto nível. (Fonte: Adaptado de [4]).

2.2 Bancos de Dados NoSQL

Segundo [22] o termo NoSQL trata-se de um neologismo acidental. Originalmente o nome se referia aos BD que não utilizavam SQL como uma linguagem de consulta. Em seu livro, os autores afirmam que não há uma descrição oficial do termo, mas que ele engloba diversos bancos de dados que possuem determinadas características em comum. Em [23], os autores afirmam que os bancos de dados NoSQL são comumente usados para armazenar um grande volume de dados e que existem diferentes categorias desses. Apesar

de distintos, ainda assim apresentam características em comum [24]. Estas características podem ser descritas da seguinte maneira [25]:

1. adotam modelos de dados mais flexíveis que, em sua maioria, não possuem esquemas;
2. as transações de consistência eventual são alcançadas relaxando as propriedades ACID para escalar horizontalmente enquanto alcançam alta disponibilidade e baixa latência;
3. o desempenho da consulta não é alcançado apenas pela co-localização de dados, mas, principalmente, por meio de escalabilidade horizontal e elástica; e
4. os dados podem ser simplesmente replicados e particionados horizontalmente em servidores locais e remotos.

Encontrar um definição única para BDs NoSQL não é trivial, pois existem bancos que não executam em *clusters*, que são de empresas privadas e que possuem esquema. Logo, é razoável analisar-se outra definição, a que afirma que os BDs NoSQL apresentam as seguintes características: flexibilidade, escalabilidade horizontal e consistência eventual.

Seja qual for a definição, são inegáveis os ganhos obtidos com o surgimento do paradigma NoSQL. Na verdade, um relevante ganho foi a persistência poliglota, que significa utilizar diferentes armazenamentos de dados em diferentes circunstâncias. O novo paradigma permitiu que se realizasse uma escolha dos bancos de dados a partir de uma análise da natureza dos dados que se deseja trabalhar, ao invés de escolher o BD mais utilizado por todos [22] . Quanto à sua classificação, os bancos de dados NoSQL podem ser categorizados em grupos de acordo com seu modelo de dados [26]. As principais categorias são listadas a seguir:

- **Chave-valor (*key-value*)** Os bancos do tipo chave-valor armazenam a chave associada a um identificador exclusivo, ou seja, a informação armazenada está ligada diretamente à sua respectiva chave. Sua busca é semelhante à busca em um dicionário. Podem ser utilizados em diversos tipos de aplicações que demandam acesso rápido aos dados como: armazenamento de detalhes de sessão do usuário e suas preferências, assim como recomendações em tempo real e publicidade em aplicações web. Neste modelo, cada par possui um identificador único com um valor associado e é considerado o armazenamento de dados mais simples. Alguns mecanismos de valor chave mantêm dados na memória como Redis, enquanto outros usam discos rígidos como o BD Oracle NoSQL [27];
- **Orientados a documentos (*document-oriented*)** Essa categoria é projetada para armazenar e consultar dados como documentos. Neste tipo de BD o valor

é estruturado e possui hierarquia. Os dados armazenados são comumente do tipo JSON ou XML e são bons para armazenar dados semi-estruturados. Assim como outros BDs NoSQL, não possuem um esquema ou estrutura pré-definida. Nessa categoria o software mais popular é o MongoDB;

- **Família de colunas (*column family* ou *columnar*)** Esses bancos de dados possuem estrutura similar à estrutura tradicional de tabelas dos bancos de dados relacionais, mas são otimizados de modo que as informações armazenadas estão organizadas em colunas ao invés de linhas. São bancos projetados para a recuperação rápida de colunas de dados e para arquiteturas que realizam o processamento distribuído de um grande volume de dados. Essas características o tornam ideal para soluções que executam cálculos massivos no banco, como agregações, também realizam muitas operações de leitura de dados do banco, como soluções de DW, que processam diversas consultas para geração de relatórios em volumes massivos de dados. Tal argumento pode ser ratificado na literatura por [28], [29], [7], [8] e [9], que afirmam que os bancos de dados de família de colunas são os mais indicados para a arquitetura de DW. O que acontece no armazenamento em colunas é que cada chave está associada a um ou mais atributos (colunas) e as informações são guardadas de tal modo que possam ser agregadas rapidamente. Os bancos colunares trabalham naturalmente com o uso da desnormalização através de duplicidade dos dados e isso proporciona um ganho na velocidade de leitura, pois não são utilizados relacionamentos que demandam um trabalho de reconstituição dos dados [22];

- **Grafos (*graph-oriented*)** Os bancos de dados de grafos permitem a criação de entidades com foco em seus relacionamentos. Esses funcionam como arestas que podem ter propriedades de uma rede cujos vértices/nós são as entidades. As arestas têm significância direcional que permitem encontrar diferentes padrões entre os nós. Com estes bancos é possível construir redes complexas de relacionamentos e revelar novos dados através da inferência [22] .

Conforme mencionado anteriormente, os BDs NoSQL possuem distintas características que visam resolver problemas variados. É comum dizer que não existe um BD NoSQL que sirva para qualquer tipo de dado, existe sim uma solução adequada para cada modelo de dados específico.

2.2.1 MongoDB

MongoDB é um BD de código aberto, NoSQL orientado a documentos, sendo o BD NoSQL mais utilizado hoje em dia [30]. Suas principais características são: excepcional

desempenho, facilidade de uso, alta disponibilidade devido ao processo de replicação, escalonamento horizontal simples, suporte a índices secundários genéricos que permitem consultas rápidas variadas, entre outros. O documento é a unidade básica de dados no MongoDB e é semelhante à linha nos BDs relacionais. Analogamente, uma coleção pode ser vista como uma tabela com esquema dinâmico e uma simples instância pode hospedar múltiplos bancos de dados independentes, onde cada um pode ter suas próprias coleções [31]. Outra característica é que cada documento possui uma chave especial, um id, que é único na coleção. Além disso, é provido de um *JavaScript shell* simples, mas poderoso, útil para a administração de instâncias e manipulação de dados. A Tabela 2.1 apresenta uma equiparação entre os dados do MongoDB e um BD relacional.

Tabela 2.1: Equivalência do MongoDB com BD relacional.

MongoDB	BD Relacional
Bancos de Dados	Bancos de Dados
Coleções	Tabelas
Documentos	Linhas
Campos	Colunas

No caso do MongoDB, sua característica de permitir documentos aninhados em uma coleção expande as possibilidades de modelagem para esse banco, pois a escolha dos documentos a serem aninhados e das coleções dependerá das consultas utilizadas. Outra possibilidade de modelo dimensional utilizando mais de uma coleção é aquela em que se cria uma coleção para cada tabela do modelo relacional. O Capítulo 3 apresenta alguns artigos da literatura que utilizaram o MongoDB para aplicações de DW com resultados satisfatórios.

2.2.2 HBase

O Apache HBase é um BD de código aberto NoSQL de família de colunas, escrito em Java, que fornece acesso de leitura e gravação em tempo real a grandes conjuntos de dados. Possui integração nativa com o Apache Hadoop, tornando-o uma excelente alternativa para o armazenamento e processamento de grandes conjuntos de dados. Além disso, é persistente, faz uso eficiente do espaço em disco, suportando algoritmos de compressão que podem ser selecionados com base na natureza dos dados de famílias de colunas específicas. Também fornece esquemas flexíveis, capazes de combinar fontes de dados com variedades

de estruturas e esquemas diferentes, tornando-o adequado para dados semiestruturados e não estruturados. A Figura 2.7 mostra o armazenamento de linhas em um BD relacional em oposição ao design orientado à coluna do HBase. Uma característica específica sua é que os valores NULL no HBase não ocupam espaço, diferentemente dos BDs relacionais.

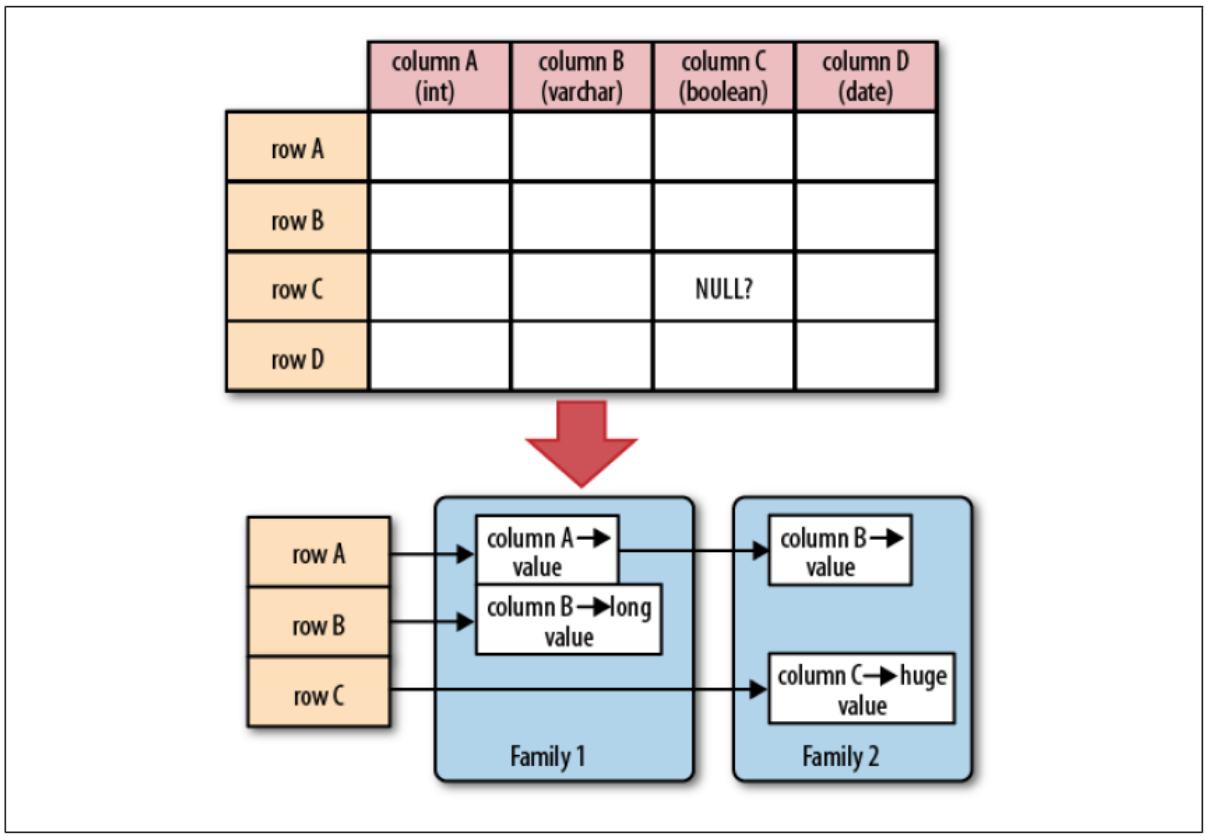

Figura 2.7: Linhas e colunas no HBase. (Fonte: [32]).

Existem três componentes principais para o HBase: a biblioteca cliente, um servidor (nó) mestre e muitos *RegionServers*, ou nós secundários. Os *RegionServers* podem ser adicionados ou removidos enquanto o sistema está instalado e funcionando para acomodar cargas de trabalho variáveis. O mestre é responsável por atribuição de regiões ao *RegionServers* e utiliza o Apache ZooKeeper, um serviço de coordenação persistente e distribuído, confiável e altamente disponível [32]. Cada *RegionServer* atende a um conjunto de regiões, mas uma região pode ser atendida somente por um único *RegionServer*. Sempre que um cliente envia uma solicitação o nó mestre a recebe e a encaminha para o *RegionServer* correspondente.

O HBase exige que todas as tabelas tenham uma chave primária. O espaço da chave está dividido em blocos sequenciais que são então atribuídos a uma região. Os *RegionServers* possuem uma ou mais regiões, de modo que a carga está distribuída uniformemente

em todo o *cluster*. Se as chaves dentro de uma região forem frequentemente acessadas, o HBase pode subdividir a região automaticamente, de modo que o corte manual de dados não é necessário [32]. Ou seja, a unidade mais simples e fundamental da escalabilidade horizontal no HBase é a região, um conjunto contíguo e ordenado de linhas que são armazenadas.

2.3 *Big Data*

Atualmente, a interação e compartilhamento de informação ocorre de diversas maneiras, por meio de texto, imagens, vídeos e utilizando-se plataformas de mídias sociais digitais como Facebook, Twitter e YouTube [5]. Além do contínuo crescimento de dados massivos produzidos por dispositivos como controladores e sensores, aplicações web, *smartphones* e redes sociais, há também o surgimento de novos dados adicionais que trazem mais complexidade ao processamento e integração de dados.

O termo *Big Data* pode ser definido como a capacidade de processamento de dados automatizados que é extremamente rápida, escalável e possui um processamento flexível. Na realidade, quando se trata de *Big Data*, não há uma única definição. Em seu trabalho [33], DeMauro, Greco e Grimaldi apresentam 15 definições existentes de *Big Data*, adaptadas de artigos referenciados. Eles afirmam que, de uma forma geral, essas definições fazem alusão a um ou mais de quatro temas relacionados à área, como: informação, tecnologia, métodos e impacto.

Em [34], Zikopoulos e Eaton definem *Big Data* como uma coleção de grandes conjuntos de dados que não podem ser processados usando técnicas de computação tradicionais. Também afirmam que as definições de um *Big Data* são baseadas, originalmente, em três características principais: volume, variedade e velocidade.

1. Volume é caracterizado pela quantidade de dados que são gerados continuamente por variadas fontes de dados que incluem texto, áudio, vídeo, redes sociais, pesquisas, dados médicos, relatórios de crimes, entre outros [35].

2. A velocidade no *Big Data* é importante na medida em que os dados são analisados em *batches*. O termo velocidade se refere à coleta, processamento e uso dos dados no mais curto intervalo de tempo possível.

3. O terceiro aspecto é a variedade, que refere-se à heterogeneidade de estrutura em um grande conjunto de dados. Existem dados estruturados, não estruturados e semiestruturados. Para [36], a variedade dos dados afeta diretamente a integridade deles, pois, mais variedade implica mais erros.

Atualmente, a quantidade de Vs já é maior que esses três originais [37], [38] e [39]. Como os trabalhos citados existem diversos outros, a publicação de [40] traz um resumo de vários artigos que tratam do tema. Presentemente vive-se um momento em que não somente as tomadas de decisões nas organizações são influenciadas pelos dados como também o é a forma como usamos diversos serviços em nossa vida cotidiana. Portanto, é de suma importância conseguir obter valor destes dados.

2.4 Apache Hadoop

O Apache Hadoop é um *framework open source* para o armazenamento e processamento de dados em larga escala, que oferece como ferramentas principais uma implementação do modelo MapReduce (MR), responsável pelo processamento distribuído, e o *Hadoop Distributed File System* (HDFS), para armazenamento de grandes conjuntos de dados, também de forma distribuída [41].

Os componentes principais da pilha Hadoop são o modelo de programação MapReduce e o sistema de arquivos distribuído HDFS. Ao longo dos anos novos componentes foram incorporados à arquitetura para resolver problemas específicos. A arquitetura do *framework* pode ser definida da seguinte forma [41]:

- Camada de armazenamento: *Hadoop Distributed File System* (HDFS);
- Camada de processamento: MR, Tez ou Spark; e
- Camada de acesso aos dados: ferramentas como Hive, para abstração da linguagem SQL como interface aos comandos MapReduce.

O MapReduce, paradigma de programação introduzido pelo Google com o objetivo de processar e analisar uma quantidade massiva de informações, é uma abordagem que busca dividir os problemas complexos de *Big Data* em pequenas unidades de trabalho, *jobs*, e processá-las em paralelo. Em uma rede de computadores onde o dado encontra-se distribuído pelos nós de computadores que compõem essa rede, o MapReduce pode ser dividido em dois estágios [42] :

- passo de mapeamento: o nó mestre divide os dados em vários subconjuntos menores. Um nó trabalhador, *worker*, processa um subconjunto de dados menor sob o controle de um rastreador de trabalho e armazena o resultado no sistema de arquivos local, onde um redutor será capaz de acessá-lo;
- passo de redução: analisa e reúne os dados de entrada a partir das etapas de mapeamento. Pode haver múltiplas tarefas de redução para paralelizar o processamento.

As etapas supracitadas são executadas nos nós trabalhadores (*workers*) sob o controle do rastreador de trabalho (*Jobtracker*). Assim, o *Jobtracker* é responsável por receber tarefas de MapReduce e submetê-las aos *TaskTrackers*. Ele também deve comunicar-se com o *NameNode*, ou nó mestre, para conseguir os dados a serem processados. O *Jobtracker* deve ser acionado pelos *TaskTrackers* a intervalos regulares, especificando que estes estão efetuando as tarefas.

O Hadoop fornece uma arquitetura para que aplicativos MR funcionem de forma distribuída em um *cluster* de máquinas, organizadas em uma máquina mestre e vários trabalhadores, sendo necessários cinco processos: *NameNode*, *DataNode*, *SecondaryNameNode*, *Jobtracker* e *TaskTracker*. *NameNode*, *DataNode*, *SecondaryNameNode* são integrantes do modelo de programação MapReduce e os dois últimos do sistema de arquivo HDFS [41].

Como evolução do JobTracker, o YARN foi introduzido no Hadoop 2.X. O YARN é um framework que atua sobre o HDFS e realiza o gerenciamento dos recursos do *cluster* como memória RAM e processamento, assim como faz o agendamento de trabalhos (*jobs*).

O Hadoop fornece várias ferramentas de análise, de DW e de consulta, incluindo algoritmos de machine learning. Esta tecnologias surgem como consequência da necessidade de aumentar a eficiência e o desempenho do processamento de dados [43]. A escolha do Hadoop ocorreu por sua integração com as diversas ferramentas pertencentes ao ecossistema Hadoop, como o HBase e o Hive por exemplo, que será explicado na próxima seção. As ferramentas que se integram com o Hadoop fornecem flexibilidade para a análise de dados, pois há diferentes ferramentas para distintos tipos de aplicação.

2.5 Apache Hive

O Hive é uma infraestrutura de DW *open source*, construída sobre o Hadoop, que facilita as consultas e a gestão de grandes volumes de dados armazenados em ambiente distribuído [44][45].

No Hive, não existe um formato único no qual os dados devem ser armazenados, sendo que este suporta arquivos de texto com separação de atributos através de vírgulas ou Tabs (CSV ou TSV), ou outros formatos como Parquet e ORC (*Optimized Row Columnar*).

Podemos classificar a arquitetura do Hive da seguinte forma [43]:

Metastore: é o catálogo do sistema que armazena os metadados sobre tabelas, colunas e partições.

Driver: recebe as consultas e gera os detalhes das sessões, assegura as estatísticas e gera o ciclo de vida de uma instrução HiveQL (semelhante ao SQL). O *driver* compila os

inputs, otimiza a computação e executa os pedidos através de três componentes: *compiler*, *optimizer* e *Executor*.

Thrift server: libera acesso ao Hive através de uma única porta, permitindo conexão programática.

JDBC/ODBC: APIs que fornecem acesso ao Hive através de diferentes aplicações. Por exemplo, o JDBC permite o acesso a aplicações Java e o ODBC a aplicações C++.

Comand Line Interface (CLI) e *Hive Web Interface* (HWI): são duas interfaces do cliente que permitem a comunicação com o HDFS através do *driver*. A primeira permite a execução em linhas de comando e a segunda é uma interface *web* [43].

MR, Tez e Spark Apesar de se ter verificado uma melhoria nos sistemas de DW com a integração do MapReduce, os sistemas de consultas baseados no MapReduce, ficam aquém das expectativas na otimização de consultas e das capacidades dos sistemas de BD. Isto porque, o Hive, na execução de determinada consulta, traduz a consultas num conjunto de trabalhos *map* e *reduce*, assumindo que o utilizador otimizou a consulta antes de introduzir no sistema [46]. Outra característica do MR é que ele persiste os dados de volta no disco após uma ação de *map* ou *reduce*, ou seja, ele lê e escreve no disco, o que torna a computação mais lenta.

O DW do Hive oferece 3 opções de processamento embutidas no ecossistema Hadoop. Essas opções são: MapReduce, Tez e Spark. Naturalmente, cada processamento apresenta pontos fortes e fracos e também há diferenças nos parâmetros de configuração do ambiente para cada escolha. . O Tez possui melhor desempenho que o MR, apesar de também traduzir as consultas em trabalhos *map* e *reduce* e realizar o processamento em disco. Já o Spark processa os dados na memória Random Access Memory (RAM) e não em disco. Em teoria, o Spark deve superar o MR e Tez. No entanto, o Spark precisa de muita memória. Assim como os BDs padrão, o Spark carrega um processo na memória e o mantém lá até novo aviso para fins de armazenamento em cache. Dessa forma, se você executar o Spark no Hadoop YARN e os dados forem muito grandes para caber inteiramente na memória, o Spark poderá sofrer uma grande degradação de desempenho. A vantagem de utilizar-se o HBase ocorre devido à sua integração com o ecossistema Hadoop. Dessa forma, abre-se um leque de opções de ferramentas que podem explorar melhor os dados. Com o dados armazenados no HBase são criadas tabelas externas ao Hive, associadas a esses dados no HBase.

Capítulo 3

Revisão de Literatura

Neste capítulo são apresentados os trabalhos relacionados à pesquisa em questão. Inicialmente foi realizado um mapeamento da literatura acerca do tema DWs baseados em NoSQL, cuja metodologia completa e resultados detalhados encontra-se em [47]. Posteriormente, após uma melhor definição do escopo do trabalho, foi adicionado o tema modelagem de dados para complementar a pesquisa original. O mapeamento da literatura será apresentado na Seção 3.1 e consistiu na investigação de três importantes bases de dados (Scopus, IEEE e Web Of Science), aplicando-se uma metodologia baseada no processo de revisão sistemática da literatura, adaptado dos passos definidos em [48]. Na Seção 3.2 são mencionados os principais trabalhos acerca de NoSQL DW e, por fim, na Seção 3.3, todos os artigos encontrados sobre o tema são apresentados, com informação do BD utilizado e propósito do trabalho.

3.1 Método de Pesquisa: Processo de Mapeamento Sistemático

O objetivo da revisão sistemática é pesquisar os principais tópicos relacionados à fundamentação teórica da pesquisa, assim como identificar os principais trabalhos relacionados ao tema, que, atuando como estudos secundários, auxiliam no desenvolvimento deste trabalho. Com a finalidade de obter os trabalhos existentes na literatura sobre o tema "*Data Warehouse*"e "Bancos de Dados NoSQL", foi desenvolvida uma abordagem sistemática dividida em três etapas: Busca por Trabalhos, Primeiro Ciclo e Segundo Ciclo. O passo a passo está descrito na 3.1. Primeiramente, na fase de planejamento foram definidas as questões de pesquisa, assim como os critérios de inclusão e exclusão.

As questões de pesquisa foram:

- QP1) Qual a quantidade de publicações na área de NoSQL DW por ano?;

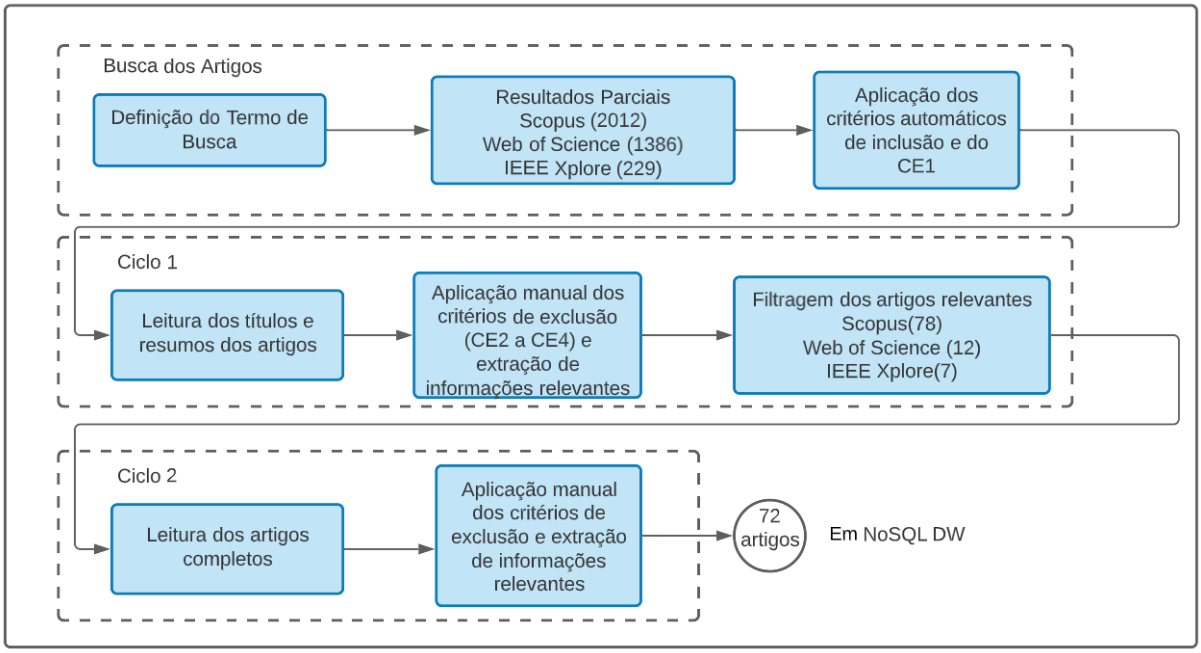

Figura 3.1: Metodologia empregada.

- QP2) Quais as conferências com maior número de publicações na área de NoSQL DW?;
- QP3) Quais as palavras-chave mais utilizadas pelos autores nas publicações referentes à NoSQL DW?;
- QP4) Quais os países que mais publicam na área de NoSQL DW?;
- QP5) Quais as categorias de BDs NoSQL mais estudadas na aplicação em DW?; e
- QP6) Quais os SGBDs mais estudados em NoSQL, com aplicação em DW?

Em seguida, definiu-se os critérios de inclusão e exclusão. Critérios automáticos de inclusão aplicados:

- CI1) Documentos publicados nos últimos 5 anos (2016 a 2020);
- CI2) Documentos nas seguintes áreas de estudo: ciência da computação, ciência da informação, ciências sociais, negócios e matemática;
- CI3) Documentos em inglês, português e espanhol; e
- CI4) Artigos publicados em conferências, jornais e capítulos de livros.

Critérios de exclusão aplicados:

- CE1) Documentos duplicados;

- CE2) Documentos fora do escopo da pesquisa;
- CE3) Documentos com menos de 4 páginas; e
- CE4) Artigos incompletos.

Na Busca por Trabalhos tem-se as atividades de definição do termo de busca, busca nas bases de dados e a aplicação dos critérios de inclusão automáticos. Já os critérios de exclusão manuais foram aplicados no Primeiro Ciclo e no Segundo Ciclo. O termo de busca procurou agrupar termos relacionados a DW e a BD NoSQL, sendo apresentados na Tabela3.1.

Tabela 3.1: Termo de Busca da pesquisa.

Termos Principais	**Termos relacionados**
Data Warehouse	“DW”, "data warehouses”, "datawarehouse”, “datawarehousing”,"online analytical processing”, “olap”, “nolap”, “etl”, “dss”, "decision support", “BI”, "Business Intelligence”, “analytical”, “dashboard”, “datamart”, “data mart”, “dm”.
Sistemas / BD NoSQL	“nosql”, “cassandra”, “neo4J”, “mongoDB”, “couchDB”, “HBase”, “Hypertable”, “Riak”, “columnar”, “column-oriented”, “document-oriented”,” key-value”, “graph-oriented”

Após a definição do termo de busca, este foi aplicado nas bases de dados Scopus (2.012 artigos), Web of Science (1.386 artigos) e no IEEE Xplore (229 artigos). Os critérios de inclusão (CI) automáticos visam remover os resultados falso-positivos da pesquisa e reunir os artigos mais recentes. Já os critérios de exclusão (CE) manuais propõem-se a filtrar e apagar os artigos irrelevantes à pesquisa.

Dessa forma, após a aplicação do termo de busca nas bases supracitadas, foram aplicados neste resultado os critérios de inclusão automáticos e os critérios de exclusão, obtendo-se, assim, os artigos que compuseram o corpus da pesquisa. A partir da metodologia, 72 artigos foram selecionados referentes a NoSQL DW, porém, apenas 59 estão diretamente ligados ao tema desta pesquisa. Assim, com os artigos resultantes descobriu-se quais os SGBDs NoSQL mais importantes e utilizados na área, juntamente com suas vantagens e desvantagens. Um outro resultado interessante foi a descoberta dos termos recorrentemente utilizados na área, mostrados pela nuvem de palavras na Figura 3.2.

Figura 3.2: Nuvem de termos mais utilizados.

3.2 Principais Propostas de Arquitetura de DW com NoSQL

3.2.1 Trabalhos relacionados à implementação de DW utilizando BDs orientados a documentos

Bicevska [49] apresenta as vantagens e possibilidades de usar BD NoSQL orientado a documentos em DW e também descreve detalhes da implementação de um DW utilizando armazenamento de dados NoSQL, especificamente o MongoDB. Destaca simplicidade e transparência fornecidas pela estrutura desnormalizada de um DW baseado em NoSQL, indicando sua aplicação para organizações de médio e pequeno porte devido ao aumento do volume de dados provocado pela duplicação dos mesmos. Além disso, orienta o leitor a obter benefícios da alta flexibilidade do processo de criação de um NoSQL DW, pois seu nível maior de abstração torna mais rápida sua construção. Sobre identificar e atualizar objetos iguais em sistemas de informação de maneira não ambígua, sugere o uso de IDs únicos em cada sistema ou o uso de mineração de dados. Outrossim, comenta sobre confecção de relatórios e sobre a necessidade do uso de metadados, sugerindo uma representação de dados (*Nested Table*) que favorece a construção de relatórios. Por fim,

afirma que os NoSQL DWs possuem potencial para: unir os benefícios da tecnologia de DW clássica com a simplicidade de uso e busca dos dados; e fornecer novos recursos de análise de dados, impossíveis aos sistemas de DWs clássicos.

Chevalier, em 2016, [6] realizou um estudo em que investiga a instanciação de um DW usando um BD NoSQL orientado a documentos, analisando aspectos como modelagem, consulta, carregamento de dados e cuboides OLAP. Os resultados obtidos são comparados entre os modelos orientados a documentos (com e sem normalização) e os modelos de BD relacionais. Por fim, também sugere melhorias de aproveitamento dos recursos orientados a documentos.

Em seu trabalho, Chevalier propôs dois tipos de modelos conceituais para um SAD: DFL (*Document FLat*), correspondente a um modelo plano desnormalizado armazenado em um BD NoSQL orientado a documentos; e DSH (*Document SHattered*), que corresponde a um modelo estrela armazenado em um BD NoSQL orientado a documentos. Os outros dois modelos utilizados para comparação foram RFL e RSH, que são semelhantes aos modelos acima, porém, armazenados em um BD relacional (PostgreSQL).

O BD utilizado foi o MongoDB. Os aspectos analisados foram: espaço de armazenamento de cada modelo, tempo de carregamento de dados e desempenho em três conjuntos de consultas, cuja complexidade foi variada. No MongoDB o desempenho do modelo DFL é superior ao DSH, além disso, os tempos de consulta melhoram com o ambiente distribuído.

3.2.2 Trabalhos relacionados à implementação de DW utilizando BDs orientados a colunas

De maneira semelhante, Chevalier [28], em 2015, também realizou um trabalho com BD orientados a colunas em que utilizou o HBase. As três abordagens estudadas neste trabalho foram: MLC0 (todas as tabelas de dimensões e a tabela de fatos transformadas em uma única tabela); MLC1 (dimensões transformadas em famílias de colunas com dados agrupados) e MLC2 (tabelas separadas para as dimensões e para a tabela de fatos). Como resultado final o modelo MCL2 precisou de menor espaço em disco, mas obteve pior desempenho nas consultas, ao passo que os outros MCL0 e MCL1 obtiveram desempenho semelhantes, com vantagem para MCL1. Ou seja, há relatos na literatura que atestam o bom desempenho da abordagem de transformação de um SAD para o esquema de uma única tabela.

O trabalho [29] é uma extensão de [28], realizado pelos mesmos autores. Assim, em [29], os autores avaliaram 4 modelos conceituais diferentes: M0, que corresponde a um modelo lógico totalmente desnormalizado e plano; M1, que corresponde a um modelo

desnormalizado com metadados; M2, que corresponde a um modelo normalizado estrela; e M3 correspondendo a um modelo híbrido, de documentos de diferentes esquemas em uma coleção. Os modelos M2 e M3 requerem menos espaço em disco. Para consultas muito seletivas não há um modelo ideal, pois, às vezes, o menor tempo foi para um modelo e noutras vezes para outros. Já para consultas menos seletivas os modelos M2 e M3 são muito prejudicados devido aos JOINS, tornando esses modelos não recomendáveis.

Scabora [12] e Dehdouh [7] realizaram trabalhos semelhantes nos quais propuseram soluções para armazenamento de dados distribuídos utilizando os BDs NoSQL orientados a colunas nas aplicações de DWs. Ambos compararam a desempenho de 3 diferentes modelos de dados aplicados na camada física do DW, utilizando o BD HBase. Das abordagens estudadas em cada trabalho, duas delas são iguais.

Os modelos propostos por Dehdouh foram: NLA (abordagem lógica normalizada), DLA (abordagem lógica desnormalizada com dados simples em colunas); e DLA-CF (abordagem lógica desnormalizada usando dados agregados em colunas). A abordagem NLA obteve o pior desempenho em todas as situações. As abordagens DLA e DLA-CF obtiveram resultados semelhantes, sendo a DLA-CF de desempenho um pouco melhor.

Já as propostas de Scabora foram: SameCF (semelhante à DLA), CNSSB (semelhante à DLA-CF) e FactDate. Esta última, que é a proposta de seu trabalho, agrupa as dimensões mais frequentes à tabela FATO, no caso do artigo o autor juntou a dimensão DATE à tabela FATO. A abordagem SameCF obteve o melhor desempenho para consultas que acessam várias dimensões (a partir de 3 dimensões). Por outro lado, as abordagens CNSSB e FactDate obtiveram melhor desempenho em consultas que acessaram uma ou duas dimensões. Já a abordagem FactDate se destacou para consultas com uma dimensão, que acessaram dados da dimensão Data.

Os resultados dos trabalhos supracitados, de uma maneira geral, mostram como os resultados podem variar de acordo com o perfil das consultas a serem utilizadas e indicam que o modelo a ser escolhido deve ser adequado às consultas mais utilizadas pelos usuários.

3.3 Análise Consolidada dos Trabalhos Relacionados

Para realizar o mapeamento da literatura no tema, foi realizada uma pesquisa bibliográfica nas 3 principais bases científicas: Scopus, IEEE Explore e Web of Science.

Foram analisados diversos trabalhos, publicados a partir de 2016, relacionados à implementação de DWs com BD NoSQL e à integração de dados não-estruturados à DW existentes.

Aplicações com BDs orientados a documentos:

Dos trabalhos analisados, 21 utilizaram o MongoDB. Dentre as pesquisas realizadas com MongoDB aplicado à DWs, encontrou-se:

- análise de viabilidade de construção de um DW com uma solução NoSQL, comparando-se a implementação de DWs com BDs NoSQL, como Cassandra e HBase; e
- análise de viabilidade de construção de um DW com o MongoDB, comparando-o com a solução tradicional de DW com banco relacional, mais comumente o BD *PostGreSQL*.

s

Aplicações com BDs orientados a colunas:

A aplicação do Cassandra em DW ocorreu para aplicações em tempo quase real, também com formato de dados de série temporal conforme [50] e[51].

Já o BD HBase foi utilizado nas propostas de migração de um DW em arquitetura ROLAP para uma arquitetura OLAP com BDs NoSQL - NOLAP, por [28] e [52].Além disso, [12] também utiliza o Hbase para implementação de modelos distintos em consultas OLAP.

Aplicações com os BDs orientados a grafos:

Na aplicação de DW utilizando-se BD orientado a grafos, foram encontrados 4 trabalhos no tema. Observou-se que o Neo4j foi utilizado para aplicações em que desejava-se obter vantagem dos relacionamentos existentes entre os dados, como relações sociais, por exemplo.

Em resumo, analisando-se os artigos, os SGBD mais utilizados para implementação de DW baseados em BD NoSQL são o MongoDB e HBase, provavelmente por sua popularidade e até facilidade devido à compatibilidade com as ferramentas Hadoop, no caso do HBase.

Em relação à comparação de modelos de dados aplicados à arquitetura de DW baseado em BD NoSQL, os resultados são variados e diferem de acordo com as consultas a serem realizadas, exigindo uma melhor análise de cada caso específico. Os trabalhos de [6], [13], [12], [53], [54] e [55] mostram com mais detalhes como podem variar os resultados, dependendo do modelo escolhido.

Vale ressaltar que existem mais trabalhos publicados sobre DWs baseados em BD orientados a colunas, como os supracitados na Seção 3.1, porém, esses foram publicados antes de 2016 e não foram contabilizados na análise consolidada.

Tabela 3.2: Resumo dos Trabalhos Pesquisados na área de DW.

Autor/Referência	**BD NoSQL Utilizado**	**Propósito**
Bicevska [56]	MongodB e ClusterPointDB	Requisitos e sugestões para criação de DW.
Chevalier [6]	MongoDB	Comparativo: MongoDB e PostgreSQL.
Bicevska [49]	MongoDB	Sugestões para criação de DW.
Chevalier [13]	MongoDB	Comparativo: MongoDB e PostgreSQL.
Bouaziz [57]	CouchDB, RavenDB ou MongoDB	Projeto de integração de dados não estruturados no DW, utilizando os BD NoSQL orientado a documentos como fonte.
Bouaziz [58]	Não especificado	Estudo do processo de ETL para NoSQL DW e da influência do paradigma NoSQL na criação de DW .
Yangui [59]	MongoDB	Estrutura baseada em ETL para transformar um modelo conceitual multidimensional em um modelo lógico orientado a documentos (NoSQL DW).
Davardoost [10]	Não especificado	Método para extrair cubos OLAP de um BD NoSQL orientado a documentos.
Gallinucci [60]	MongoDB	Abordagem para extração de cubos OLAP de coleções armazenadas em BD NoSQL orientado a documentos.

A continuar na próxima página

Tabela 3.2 – *Continuação da página anterior*

Autor/Referência	**BD NoSQL Utilizado**	**Propósito**
Challal [61]	MongoDB e Cassandra	Comparativo de armazenamento de Cubos OLAP de grafos sociais: MongoDB ou Cassandra.
Gallinucci [62]	MongoDB	Abordagem para consultas multidimensionais e OLAP em fontes sem esquema, em particular em coleções armazenadas em BDs NoSQL orientados a documentos.
Hubail [63]	Couchbase	Descreve a arquitetura e o serviço de Analytics do Couchbase.
Chevalier [64]	MongoDB	Sumarização no contexto de implementações de DWs NoSQL orientados a documentos.
Bicevska [65]	MongodB e ClusterPointDB	Sugestões para criação de DW NoSQL orientado a documentos.
Souibgui [66]	Não especificado	Abordagem ETL para extrair cubos OLAP de um BD NoSQL orientado a documentos.
Gallinucci [67]	MongoDB	Reescrita de consultas OLAP em BD NoSQL orientados a documentos.
Pticek [68]	MongoDB	Integração de dados NoSQL aos DWs tradicionais por meio de aplicação de semântica.
Ferro [69]	MongoDB	DW para dados Geoespaciais.

A continuar na próxima página

Tabela 3.2 – *Continuação da página anterior*

Autor/Referência	**BD NoSQL Utilizado**	**Propósito**
Alami [70]	MongoDB e HBase	Comparativo de algoritmos de desduplicação de dados aplicados em diferentes BDs: MongoDb ou HBase.
Aluva [71]	MongoDB	Comparativo de desempenho de consultas aplicadas em BDW: MongoDB e Relacional.
ElMalki [72]	MongoDB e Cassandra	Proposta de um novo benchmark aplicado à BDW.
Chouder [73]	MongoDB	Abordagem interativa de schema on read para permitir consultas OLAP em BD NoSQl orientado a documentos.
Yangui [54]	MongoDB e Cassandra	Comparativo de tempo de carregamento de dados e desempenho de consultas OLAP para dois modelos distintos de cada categoria.
Prakash [74]	Cassandra	Regras para transformação de um modelo de informação para o modelo lógico NoSql.
Dehdouh [9]	HBase	Construção de cubos OLAP de acordo com a abrdagem colunar e comparativo desta proposta com o cubo automático construído pelo Hive e Kylin.
Mallek [75]	HBase	ETL adaptado para *Big Data*
Scabora [12]	HBase	Comparativo de desempenho de consultas OLAP para três modelos distintos.

A continuar na próxima página

Tabela 3.2 – *Continuação da página anterior*

Autor/Referência	**BD NoSQL Utilizado**	**Propósito**
Murazza [50]	Cassandra	Comparativo de DW quase em tempo real: Cassandra, MySQL ou PostgreSQL.
Jianmin [76]	HBase	ETL adaptado para a BD NoSQL orientado a colunas.
Boumlik [77]	HBase	ETL adaptado para *Big Data*.
Correia [78]	Druid	Avaliação de DW orientado a colunas.
Torres [79]	MonetDB	Comparativo de desempenho entre modelos Estrela e Floco de Neve aplicados em diferentes BDs: MonetDB, MySQL e PostgreSQL.
Boussahoua [80]	HBase	Técnica para melhorar a modelagem lógica de dados para DW de famílias de colunas.
Yang [81]	HBase	Abordagem de DW de 3 camadas para *Big Data*, utilizando Hadoop e MapReduce.
Pereira [82]	HBase	Projeto de ETL e integração de dados não estruturados no DW, utilizando os BDs NoSQL orientado a colunas como fonte.
Dehdouh [8]	HBase	Construção de cubos OLAP utilizando o modelo colunar.

A continuar na próxima página

Tabela 3.2 – *Continuação da página anterior*

Autor/Referência	**BD NoSQL Utilizado**	**Propósito**
Khalil [27]	Oracle NoSQL	Comparativo desempenho: Armazenamento de espaço entre dois modelos utilizando Keyvalue e computação cubo OLAP entre key-value e Oracle SQL.
Sellami [14]	Neo4j	Regras de transformação do modelo conceitual de DW em um modelos lógicos NoSQL para BD orientado a grafos
Vaisman [83]	Neo4j	Comparativo DW modelos Estrela e Floco de Neve: Neo4j ou PostgreSQL.
Guminska [84]	Neo4j	Arquitetura para análise OLAP em DWs baseados em BDs orientados a grafos.
Gomez [85]	Neo4j	Implementação de proposta para DWs baseados em BDs orientados a grafos.
Akid [86]	Não especificado	Proposta para DWs baseados em BDs orientados a grafos para aplicação em *Big Data* Social.
Ngo [87]	MongoDB, Hive e Cassandra	Arquitetura de DW híbrida aplicada à *Big Data* de dados agrícolas, utilizando Cassandra como fonte de dados.
Khabibullina [51]	Cassandra e MongoDB	Proposta de modelo de NoSQL DW que utiliza Spark, MongoDB, Cassandra e PostgreSQL.
Costa [53]	Hadoop-Hive	Comparativo de desempenho de consultas para dois modelos distintos.

A continuar na próxima página

Tabela 3.2 – *Continuação da página anterior*

Autor/Referência	**BD NoSQL Utilizado**	**Propósito**
Oliveira [55]	Hadoop-Hive	Comparativo de desempenho de consultas para dois modelos distintos, associado à variação dos formatos das tabelas (CSV e Parquet).

Tabela 3.3: Resumo de trabalhos sobre modelagem de dados em DW.

Autor/Referência	**BD NoSQL Utilizado**	**Modelos Comparados**
Chevalier [6]	MongoDB	Tabela única desnormalizada e Estrela
Chevalier [13]	MongoDB	Tabela única desnormalizada, desnormalizado com metadados, fatos e dimensões separadas (normalizada), híbrido
Scabora [12]	HBase	Comparativo de desempenho de consultas OLAP para três modelos distintos
Costa [53]	Hadoop-Hive	Tabela única desnormalizada e Estrela
Yangui [54]	MongoDB e Cassandra	Tabela única desnormalizada (fatos e dimensões armazenadas em uma mesma coleção/familia de colunas) e Estrela (Fatos e dimensões armazenadas em coleções/famílias de colunas separadas)
Oliveira [55]	Hadoop-Hive	Tabela única desnormalizada e Floco de Neve.

Diferentemente do que foi apresentado nos trabalhos relacionados, a contribuição dessa dissertação será o acréscimo do ponto de vista do ciclo de vida de um projeto de DW aplicado ao paradigma NoSQL. A partir dos passos sugeridos por Kimball para os bancos relacionais, será proposta uma adaptação para os SGBDs NoSQL MongoDB e HBase, que destacará o que necessita ser feito de diferente do DW tradicional nas fases (originais) de Modelagem Dimensional e Projeto Físico. Assim, com a implementação de um estudo de caso, as sugestões no ciclo de vida poderão ser validadas e, além de ser possível verificar como a modelagem e escolha do BD NoSQL afeta o desempenho de consultas em um DW, também haverá a possibilidade de se identificar a viabilidade da tecnologia NoSQL

aplicada em DW, por meio dos resultados das medições de tempos das consultas aplicadas a cada SGBD.

Capítulo 4

Proposta de Ciclo de Vida de Projeto de um DW NoSQL

Este capítulo apresenta a proposta de pesquisa deste trabalho. A Seção 4.1 exprime um resumo da proposta supracitada. A Seção 4.2 aponta a justificativa de escolha dos SGBDs e indica as propostas de adaptação do ciclo de vida sob o paradigma NoSQL para os BDs escolhidos. Já a Seção 4.3 apresenta as métricas de avaliação mais utilizadas, assim como os critérios para a escolha de um BD NoSQL quando se inicia um projeto de DW NoSQL.

4.1 Cenário Atual e Resumo da Proposta

A abordagem do Ciclo de vida de um projeto de DW, proposto por Kimball em [4], existe há tempos e ainda é plenamente aceita e empregada em diversas organizações que trabalham com DW por todo o mundo. Essa abordagem envolve conceitos que transcenderam décadas de evolução tecnológica justamente por focar na agregação de valor aos negócios de toda a organização e na aceitação destes entregáveis do DW pela equipe de negócios da mesma.

Contudo, apesar de não focar em tecnologia, essa abordagem não deixa de depender da mesma, tendo em vista que a proposta de desenvolvimento do projeto de um DW baseia-se em BDs relacionais [4].

Com o advento do *Big Data* no que tange ao aumento do volume de dados e à necessidade de maiores velocidades de processamento dessas grandes quantidades, os BDs não-relacionais estão sendo cada vez mais demandados. Por possuir características diferentes dos bancos relacionais, os bancos NoSQL necessitam de um tratamento diferenciado para a modelagem dos dados, entre outros ajustes. Dessa forma, serão analisadas as etapas relacionadas à modelagem dimensional e ao projeto físico, tendo em vista que essas

abrangem o modelo conceitual, lógico e físico do DW. Apresenta-se aqui uma abordagem do ciclo de vida para bancos NoSQL orientados a documentos e de família de colunas.

4.2 Ciclo de Vida de DW NoSQL

Nesta Seção apresenta-se uma adaptação ao ciclo de vida convencional, ajustada às peculiaridades dos SGBDs NoSQL analisados. Primeiramente é apresentada a justificativa de escolha dos bancos NoSQL escolhidos. Em seguida são apresentadas as propostas de ajustes nas etapas da modelagem dimensional e do projeto físico.

A Figura 4.1 destaca as etapas envolvidas na proposta de adaptação deste trabalho. As caixas em azul (Modelagem Dimensional e Projeto Físico) são aquelas que já existiam no ciclo de vida de Kimball [4] e que foram analisadas para possíveis sugestões de adaptações, já as caixas do projeto lógico NoSQL e Projeto físico NoSQL, em vermelho, são uma proposta para serem adaptadas ao ciclo de vida do projeto de um DW sob o paradigma NoSQL. Essa separação das etapas se justifica pelas diferenças existentes entre os paradigmas relacional e NoSQL.

Ao analisar as etapas de Modelagem Dimensional e Projeto Físico na abordagem original de Kimball, percebe-se que o nível de abstração lógico é tratado no projeto físico. Dessa forma, para tornar a modelagem lógica mais clara e objetiva, sugere-se separá-la das demais, criando-se novas sub-etapas dentro do projeto físico original de Kimball, que chamaremos de projeto lógico NoSQL e projeto físico NoSQL. Cada etapa trata das atividades relacionadas apenas ao seu nível de abstração. Em resumo, este trabalho concentra-se nas etapas destacadas na Figura 4.1, que abordam exatamente os 3 níveis de abstração da modelagem de dados: Conceitual, Lógico e Físico.

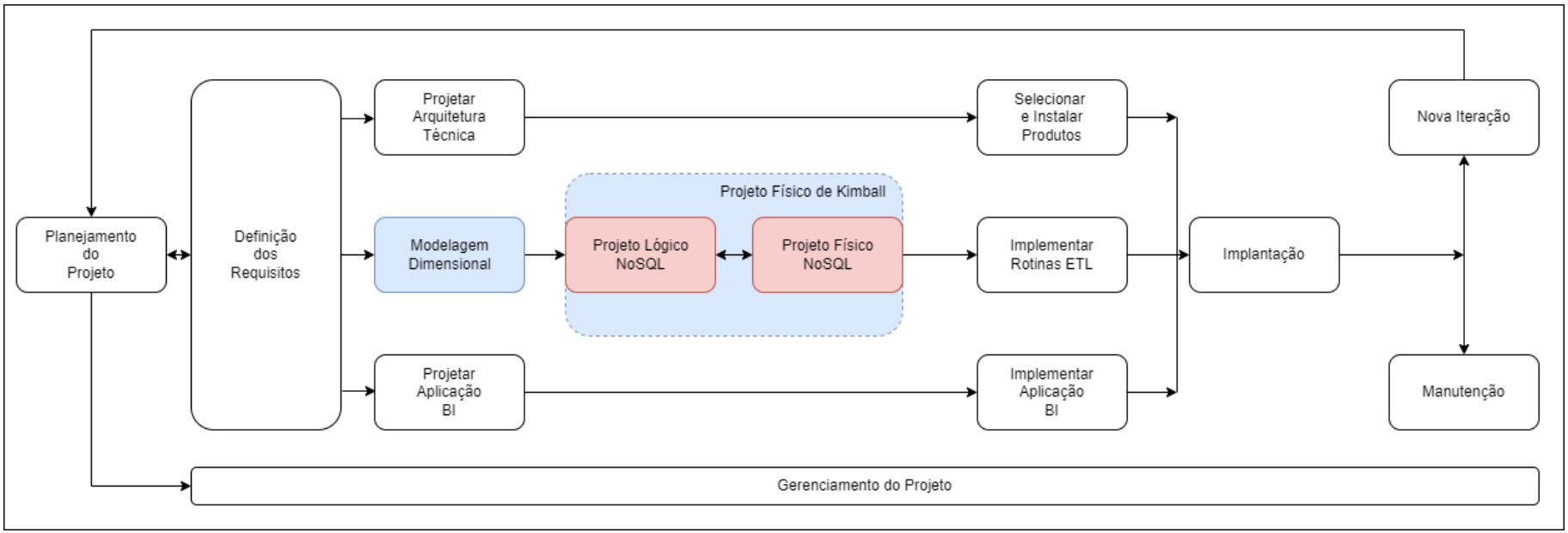

Figura 4.1: Adaptação proposta para o ciclo de vida do projeto de um DW sob o paradigma NoSQL.

A Figura 4.2 mostra uma abordagem para os 3 níveis de abstração supracitados no projeto de um DW sob o paradigma NoSQL, apresentando, no nível lógico, os modelos ROLAP e MOLAP, além do NOLAP. No nível físico são detalhados apenas os modelos referentes aos BDs NoSQL.

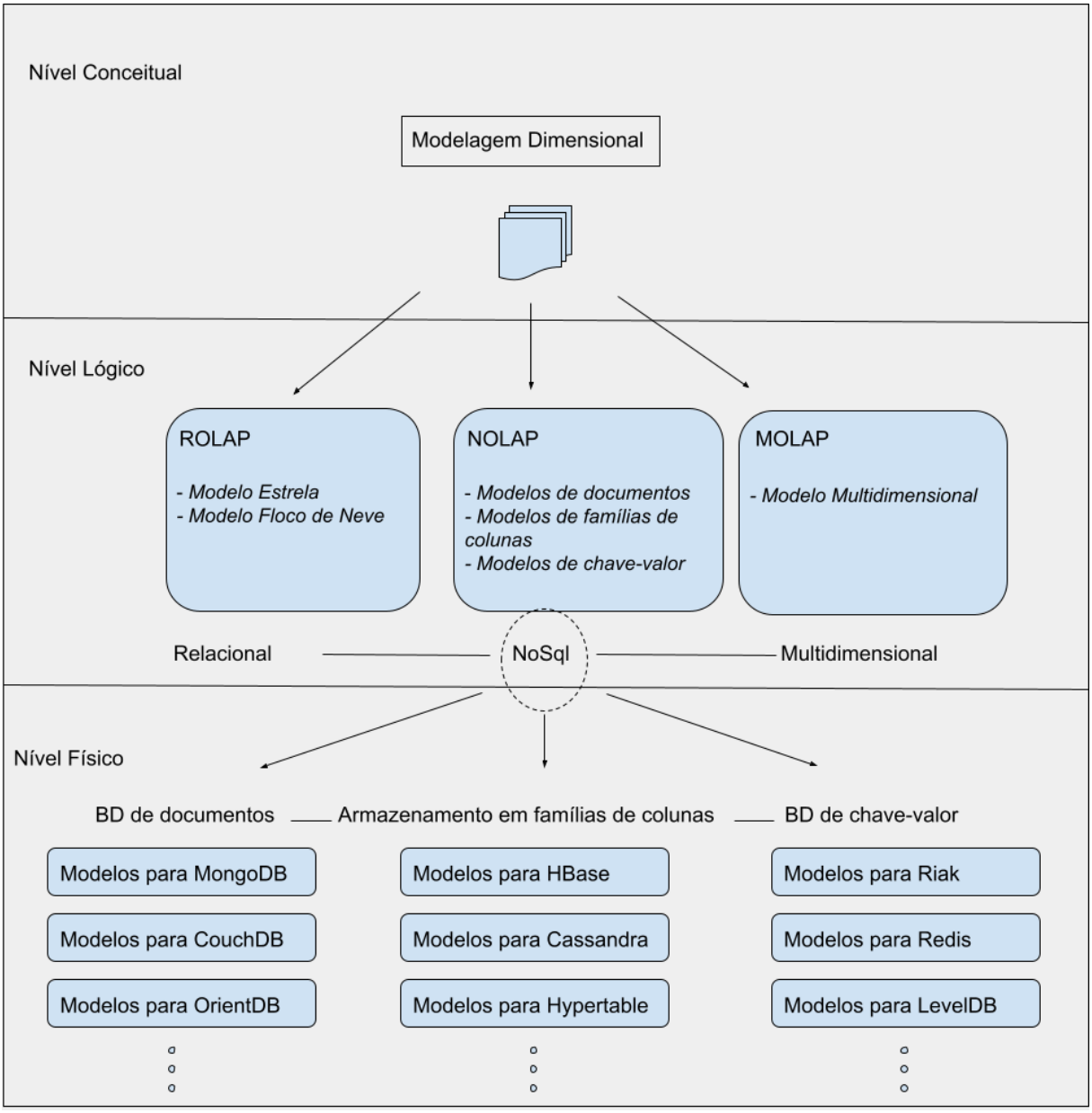

Figura 4.2: Abordagem dos níveis de abstração com foco no paradigma NoSQL.

4.2.1 Escolha do SGBD

Com a finalidade de tornar essa pesquisa mais diversificada, decidiu-se por escolher um SGBD para a categoria de BD orientados a documentos e outro para BD orientados a

colunas. Dessa forma, conseguiremos observar o comportamento do DW em diferentes ambientes, fortalecendo assim o resultado final. Portanto, para a categoria de BD orientados a documentos, escolhe-se o MongoDB, por sua ampla utilização nos trabalhos relacionados ao tema, conforme verificou-se em [56], [6], [49], [13], [57], [59], [60], [61], [62], [64], [65], [67], [68], [69], [70], [71], [72], [73], [87], [51], [54], [28] e [29]. Já para a categoria de BD orientados a colunas, escolhe-se o HBase devido à facilidade promovida pela integração com o *framework* Hadoop e todas as ferramentas associadas, principalmente o Hive, que possui uma linguagem semelhante à SQL, conforme mencionado anteriormente. A escolha do HBase também é embasada por sua ampla utilização, conforme pode-se averiguar em: [70], [9], [75], [12], [76], [77], [52], [80], [81], [82], [28], [29], [7] e [8].

Segundo [11], o MongoDB e o HBase são as tecnologias mais utilizadas por oferecerem algum apoio às junções e funções de agregação.

4.2.2 Modelagem Dimensional

Nesta etapa destaca-se o caráter conceitual da mesma. A modelagem dimensional tem a finalidade de representar o mundo natural, por meio de dimensões e fatos. As dimensões representam as características do negócio e as fatos, os indicadores. Esta etapa se caracteriza por ser independente de tecnologias, portanto, não existe uma diferença significativa entre suas atividades, independentemente do paradigma escolhido. Há apenas uma recomendação, que pode ser interpretada como uma melhor prática, que é definir as chaves das fato e das dimensões apenas na próxima etapa, a da modelagem lógica. Segundo Kimball [4], nesse estágio deve-se escolher o processo de negócio envolvido, declarar o grão, identificar as dimensões e os fatos ou métricas aplicáveis ao processo escolhido. Sendo assim, pode-se afirmar que é possível seguir a trilha de dados até a etapa da modelagem dimensional, sem depender de um SGBD. Portanto, nessa etapa não é necessária nenhuma adaptação.

4.2.3 Projeto Lógico NoSQL

Na etapa do projeto lógico NoSQL a escolha do SGBD é crucial para começar a construir o modelo lógico dos dados. É necessário observar que o modelo lógico, segundo Elmasri [88], em BDs relacionais não depende de SGBD. Contudo, de acordo com os artigos analisados [89], [6], [13], [12], [53] e [54], o modelo lógico pode variar significativamente, dependendo do banco escolhido. Como o NoSQL é um novo paradigma, foram propostas diferentes linhas de ação para cada categoria de BD NoSQL escolhida. O projeto lógico NoSQL pode ser considerado uma extensão da modelagem dimensional, agregando algumas características do SGBD NoSQL escolhido. Assim, no projeto lógico, o modelo a ser utilizado

depende do SGBD escolhido, pois, por exemplo, se o banco escolhido for o HBase, há uma limitação com relação a chaves compostas, caso a forma de importação seja com o comando ImportTSV (comando para importar arquivos CSV).

Kimball em [4], não especificou uma etapa separada para tratar do Projeto Lógico, citando-o na etapa do Projeto Físico. Para o BD relacional o modelo lógico de dados normalmente se resume a duas alternativas de estruturas: o esquema estrela e o esquema de floco de neve. Porém, para um BD não relacional, a modelagem irá variar consideravelmente a depender do banco NoSQL escolhido para armazenar esses dados. Por esse motivo, para o paradigma NoSQL, sugere-se a criação de uma etapa dedicada exclusivamente ao modelo lógico, tendo em vista que esse exige uma configuração mais complexa.

A inserção da etapa do Projeto Lógico NoSQL surge também da necessidade de encontrar um modelo que forneça o melhor desempenho às consultas envolvidas no projeto do DW, tornando-o performático ou viável. Trabalhos como [89],[6],[13],[12],[53] e [54] mostram como o desempenho das consultas pode variar de acordo com o modelo lógico escolhido. Essa variação no desempenho das consultas destaca a característica empírica do processo de escolha do modelo lógico e sua relação com as consultas analisadas e o desempenho do DW construído. A adaptação sugerida ao ciclo convencional de Kimball se baseou nessa relação supracitada, pois fornece, neste momento, um grau de liberdade para ajustar o modelo lógico até que o desempenho das consultas alcance um valor determinado pela equipe responsável.

A Figura 4.3 explica de uma forma didática, como é realizada uma transformação do nível conceitual multidimensional para o nível lógico sob o paradigma NoSQL.

De uma maneira geral, nessa etapa que trata do nível lógico, deve-se escolher o modelo de dados a ser utilizado no DW e descrever o relacionamento das chaves utilizadas. Por se tratar de SGBDs NoSQL, a escolha do modelo deve estar diretamente relacionada às consultas que serão mais utilizadas, o que significa escolher um modelo que minimize a quantidade de junções de dados utilizados para as consultas em questão. Essa observação se faz necessária devido à tendência de deterioração do desempenho das consultas em BD NoSQL, à medida que a quantidade de junções de dados é incrementada. Essa é uma das razões do Modelo Totalmente Desnormalizado (MTD) ser bem aceito para o paradigma não-relacional.

Outro aspecto a ser pontuado é quando se detecta um desempenho aquém do planejado na realização das consultas. Nesse momento, o projeto encontra-se na etapa do projeto físico e isso provoca um retorno à etapa do projeto lógico, para modificar as características do modelo escolhido, trocar de modelo ou até mesmo de SGBD. Na prática, essa iteração pode ocorrer diversas vezes até que o desempenho das consultas alcance um determinado valor pré-estabelecido, conforme mencionado anteriormente.

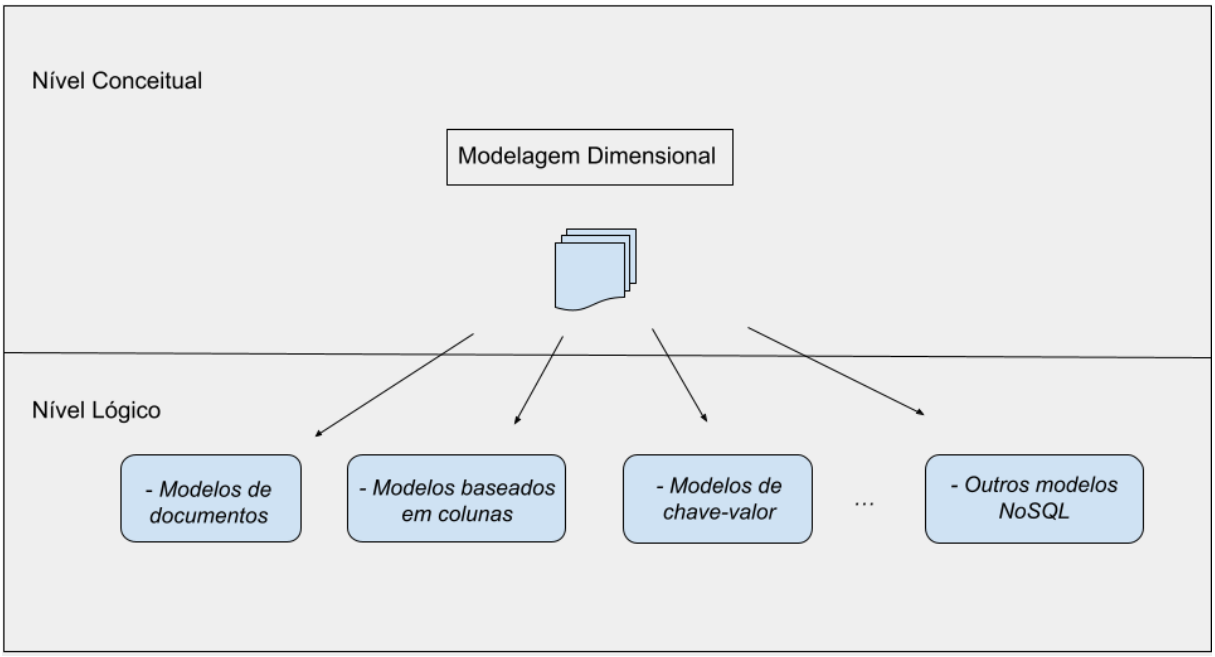

Figura 4.3: Transformação do nível conceitual multidimensional para o nível lógico.

Em resumo, devido à importância do projeto lógico e de sua influência no desempenho final do DW, sugeriu-se a criação de uma etapa separada para o projeto lógico, onde deverão ser feitas todas as escolhas referentes ao modelo e SGBD a serem utilizados no projeto. Para permitir uma maior flexibilidade ao projetista, será possível retornar do projeto físico de volta à etapa lógica para fazer ajustes necessários, caso o desempenho da configuração anterior esteja aquém do pré-estabelecido pela equipe de projeto.

4.2.4 Projeto Físico NoSQL

A adaptação da etapa do projeto físico ocorre em função do BD NoSQL escolhido. A abordagem mais normalizada como Floco de Neve e Estrela, por exemplo, usa diferentes tabelas para armazenar as fatos e as dimensões no nível físico, o que requer que se realize operações de junções de dados entre as tabelas para performar funções de agregação. Já a abordagem totalmente desnormalizada armazena as fatos e dimensões em um única tabela, evitando assim o uso de junções. Ao implementar o modelo escolhido na etapa do projeto lógico, o projeto físico deve especificar detalhes relacionados ao BD escolhido como, por exemplo, a definição de chaves primárias, do formato dos arquivos armazenados nas tabelas, dos índices, do particionamento etc. Nesse trabalho os BDs NoSQL escolhidos foram o MongoDB e o HBase, como mencionado em 4.2.1. No que tange à questão da relação do modelo escolhido com o desempenho de seu projeto físico, para se conseguir obter um bom desempenho será implementado o projeto lógico e físico de uma maneira

iterativa, conforme mencionado na etapa do projeto lógico NoSQL. Portanto, caso seja necessário melhorar o desempenho do projeto, deverão ser tomadas medidas como: alterar o formato dos arquivos das tabelas, modificar particionamentos ou até voltar ao projeto lógico e modificar o modelo para se buscar o desempenho desejado. Esse processo deve ser realizado até obter-se o resultado esperado.

Para respaldar as decisões realizadas nesta etapa do projeto, é necessário analisar algumas métricas e avaliar a relevância de cada uma para o seu projeto. Existe uma relação custo/benefício que varia de caso a caso.

4.3 Considerações e Critérios para Escolha de um BD NoSQL

Ao se trabalhar com bancos NoSQL importa analisar aspectos que não se aplicam ao paradigma relacional e que influenciam diretamente no desempenho do DW NoSQL, como formato dos dados, consultas e infraestrutura.

- Formato de dados: Em NoSQL há diferentes formatos de dados, como CSV, Parquet, ORC e Avro. Cada um possui uma forma de armazenamento distinta, que pode alterar completamente o desempenho de uma consulta;
- Consultas: A escolha das consultas também é outro fator que influencia muito na viabilidade do DW. Consultas com muitas junções e filtros deterioram o desempenho do DW; e
- Infraestrutura física: A possibilidade de usar *cluster* para o processamento distribuído é uma característica dos bancos NoSQL em geral, porém, a escolha do tipo de cluster, da quantidade de nós e da arquitetura de hardware também tem grande influência no resultado do desempenho do DW. Outro fator que influencia é se o DW será na nuvem ou local.

Observa-se que é sim possível utilizar o ciclo de Kimball para o desenvolvimento de um DW NoSQL, porém, não da mesma forma que o relacional. A seguir são apresentadas as métricas de avaliação mais utilizadas na literatura.

4.3.1 Métricas de Avaliação

As métricas de avaliação comumente utilizadas na literatura são:

1. Espaço utilizado em disco;
2. Tempo de carga das tabelas; e

3. Tempo de resposta para diferentes tipos de consultas, por exemplo, uma consulta com apenas um agregado (*select* e *count*), outra com muitas junções, outra com junções e ordenação etc.

Como mencionado anteriormente, a relevância de cada uma dessas métricas definidas, varia de projeto a projeto.

4.3.2 Critérios para Escolha do BD NoSQL

Nessa Subseção são listados alguns parâmetros que devem ser considerados no momento em que se escolhe o BD NoSQL que será utilizado como DW. Esses parâmetros foram retirados de diferentes artigos estudados no capítulo 3, como preparação para a presente pesquisa.

1^o Passo: Escolha da Categoria NoSQL

Assim, após definir o modelo dimensional para a escolha da categoria NoSQL (baseado em documento, colunar, baseado em grafo etc) mais adequada aos dados, o projetista deve analisar os seguintes aspectos:

1. Tempo de carga disponível/desejado;
2. Espaço em disco disponível;
3. Complexidade das consultas, se envolvem muitas junções, se são pouco ou muito seletivas; e
4. Variedade de esquema.

Categorias NoSQL Mais Utilizadas

Por meio do mapeamento da literatura realizado no tema de DWs baseados em BD NoSQL, registrou-se que as duas categorias mais utilizadas para o projeto de um DW NoSQL são os BDs de famílias de colunas e os BDs orientado a documentos. Isso se justifica por fatores como: a tecnologia é conhecida, são mais utilizados nas organizações e, no caso de família de colunas, são os que mais se assemelham aos DWs relacionais. Verificou-se também que não há indicações de melhores cenários para a utilização de cada BD NoSQL para aplicação em DW. Assim, a partir da leitura de diferentes artigos na área, foi possível sumarizar os principais benefícios de escolha das duas principais categorias supracitadas. Abaixo são apresentadas algumas das características que devem ser consideradas e que favorecem a escolha das categorias de BDs orientados a documentos e de família de colunas. O escopo do trabalho será limitado a essas duas categorias, por

serem as mais citadas na literatura. Os trabalhos referenciados que justificam a redução do escopo encontram-se na Seção 4.2.1.

Categoria de Banco de Dados de Documentos

Categoria bastante utilizada para aplicação em DW, principalmente utilizando-se o SGBD MongoDB, por sua popularidade. As características relatadas na literatura são [54] e [6]:

1. Melhora o desempenho com o aumento do volume de dados e com a distribuição dos mesmos;
2. Execução mais rápida para o modelo desnormalizado devido ao baixo suporte a junções;
3. Ocupa mais espaço de armazenamento devido à sua estrutura em disco e também ao fato dos sistemas orientados a documento repetirem o nome dos campos em todo o documento;
4. Tempo de carga baixo para o modelo estrela/floco de neve e bem alto para o modelo desnormalizado;
5. Tamanho do documento limita-se a 16 MB;
6. Favorável a consultas que utilizam múltiplos atributos;
7. Geralmente possui bom desempenho de leitura;
8. Flexibilidade e escalabilidade horizontal; e
9. Documento tem capacidade de organizar dados hierarquicamente, permitindo seu armazenamento ou recuperação sem o uso de junções.

Categoria de Banco de Dados de Famílias de Colunas

Essa categoria é uma das mais utilizadas na área de DW NoSQL e possui um bom desempenho no que se refere ao tempo de carga e espaço de disco utilizado. Em contrapartida, o desempenho de suas consultas depende de fatores como o caminho do armazenamento para acessar diferentes atributos de uma consulta. Dessa forma, pode-se resumir suas características mais favoráveis como [28] e [12]:

1. Tempo de carga mais competitivo que o de BD voltado a documentos;
2. Utiliza menor espaço de disco e é conhecido por sua eficiência em compressão de dados;
3. Desempenho melhor com consultas que utilizam menos argumentos; e

Dessa forma, ao se decidir por uma categoria, é necessário averiguar se os dados e consultas em questão se adequam bem às suas características mais relevantes.

2º Passo: Escolha do modelo de dados

Após a escolha da categoria de BD a ser utilizada, o projetista terá que decidir qual será utilizado com os dados em questão, pois os modelos diferem de acordo com a categoria selecionada, à exceção do MTD, constituído de apenas uma única tabela com todas as fatos e dimensões fundidas. Por exemplo, a Figura 4.4 mostra como um modelo conceitual pode gerar três estratégias distintas no modelo lógico.

3º Passo: Escolha do BD NoSQL

No projeto físico o projetista precisa saber o SGBD a ser utilizado para definir os detalhes de implementação, como relacionamento das chaves, formato dos dados das tabelas, assim como detalhes relativos ao cluster, se houver.

Neste trabalho as consultas serão realizadas em dois modelos distintos, tanto para o MongoDB quanto para o HBase, seus resultados serão comparados ao DW relacional, que será implementado no MySQL no mesmo ambiente que os SGBDs NoSQL.

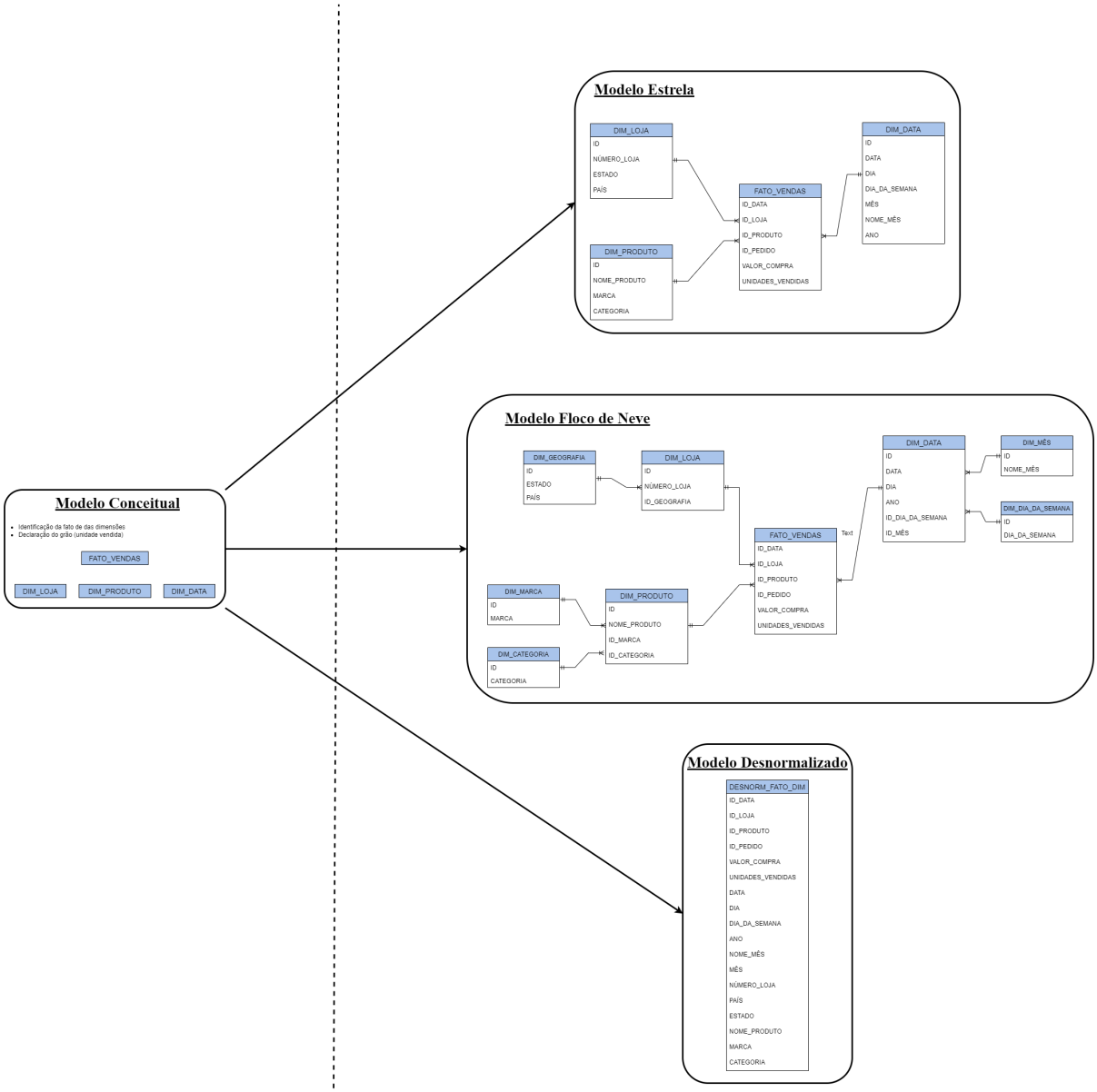

Figura 4.4: Exemplo de 3 possibilidades de modelos gerados a partir de um modelo conceitual.

Capítulo 5

Estudo de Caso

O objetivo do estudo de caso é validar o ciclo de vida proposto e verificar a viabilidade de se trabalhar com essa tecnologia em DW. Para isso, é implementado um DW utilizando-se o MongoDB e o HBase avaliando-se os seus desempenhos. Os melhores resultados de cada banco são então comparados com o DW relacional (utilizando o MySQL). Na implementação dos DW NoSQL são comparados os desempenhos das consultas, variando-se a forma como os dados são armazenados, Parquet e CSV, assim como o processamento de dados entre Hadoop e Spark, no caso do HBase. Este capítulo apresenta na Seção 5.1, o conjunto de dados, na Seção 5.2 as características das consultas e os cenários de teste e na Seção 5.3 a definição da arquitetura de hardware e o método aplicado para obter os resultados. Já a seção 5.4 apresenta as especificidades do MongoDB e do HBase.

5.1 Conjunto de Dados

O DW a ser analisado no estudo de caso contem informações referentes ao serviço militar brasileiro. Nele estão contidos dados dos cidadãos brasileiros residentes no Brasil e no exterior que se alistaram no Serviço Militar Obrigatório de 2011 a 2018. O Serviço Militar consiste no exercício de atividades específicas desempenhadas pelas Forças Armadas (Marinha, Exército e Aeronáutica) e compreende, na mobilização de pessoal, todos os encargos com a Defesa Nacional. Ele é obrigatório a todos os brasileiros, tem duração normal de 12 (doze) meses, porém se o soldado desejar e for selecionado, o mesmo pode continuar como soldado/cabo engajado por mais 7 anos.

Os dados contidos no DW são referentes a todos os brasileiros alistados e também aos cidadãos selecionados, por todo o tempo que permaneceram servindo. Esses dados são relevantes tanto para o Exército como para a sociedade como um todo, pois permite extrair informações relacionadas ao perfil do jovem brasileiro alistado. Dessa forma, por

sua dual importância e dimensão, foi escolhido o conjunto de dados do serviço militar para ser utilizado como estudo de caso desta pesquisa.

Para esta pesquisa serão utilizadas uma tabela fato FATO_EVENTO_CIDADAO e 23 dimensões ligadas a ela. O diagrama apresentado na Figura 5.1 apresenta o esquema do DW em questão. A quantidade de colunas das tabelas varia de 4 a 86.

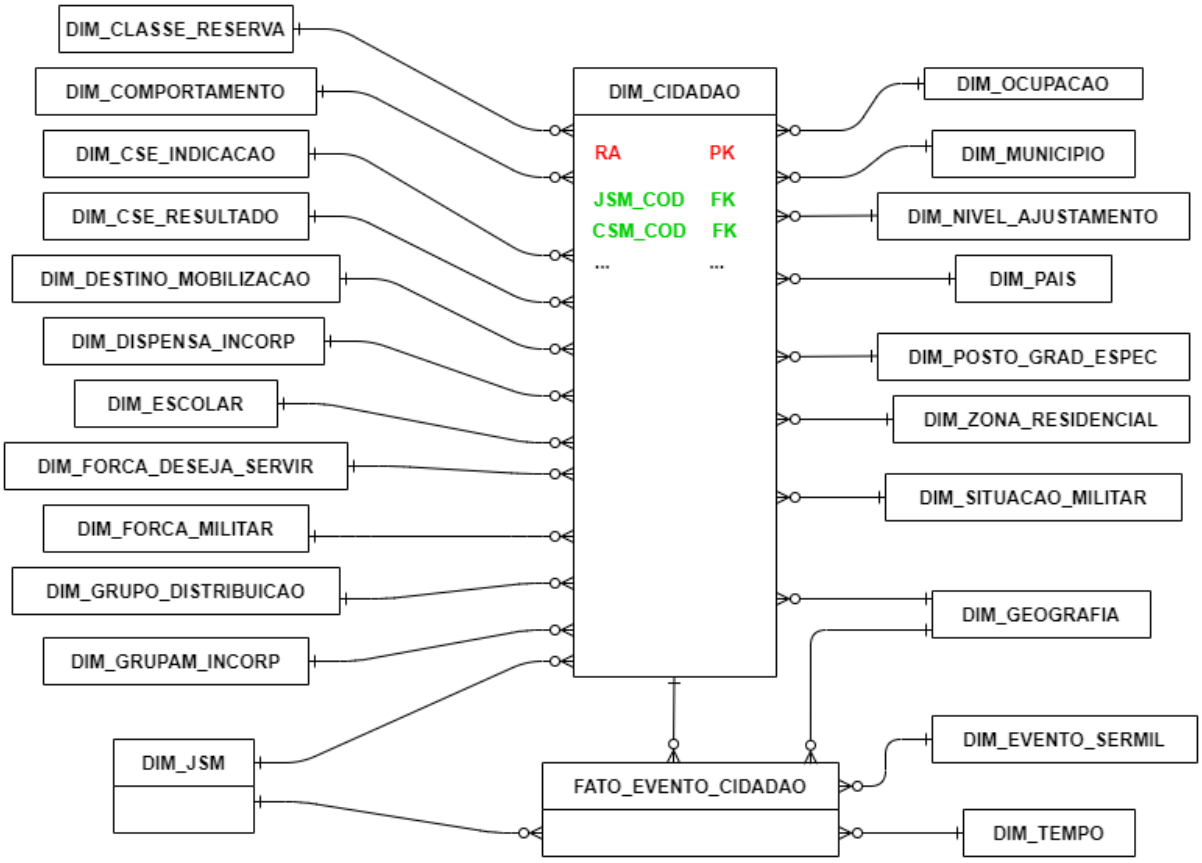

Figura 5.1: Esquema do DW a ser usado no estudo de caso.

A tabela de fatos une informações referentes ao alistado, como sua situação, altura e idade, a Circunscrição de Serviço Militar (CSM) e a Junta de Serviço Militar (JSM) em que se alistou e o número do documento de publicação. Algumas tabelas serão explicadas brevemente abaixo, porém, de todas as tabelas de dimensões, a mais relevante é a DIM_CIDADAO. Nela estão contidos os dados pessoais do cidadão como: registro de alistamento (RA), estado civil, endereço, dados médicos, medidas corporais, entre outros; também contém as chaves primárias de conexão com todas as demais dimensões, tornando o modelo deste DM, um modelo *SNOWFLAKE*.

O nome e a quantidade de colunas das tabelas acessadas pelas consultas deste trabalho são apresentadas na Tabela 5.1.

Tabela 5.1: Características das tabelas processadas.

Nome da Tabela	Número de Colunas
FATO_EVENTO_CIDADAO	11
DIM_GEOGRAFIA	86
DIM_TEMPO	23
DIM_FORCA_MILITAR	4
DIM_FORCA_DESEJA_SERVIR	4
DIM_SITUACAO_MOBILIZACAO	4
DIM_ZONA_RESIDENCIAL	4
DIM_CLASSE_RESERVA	4
DIM_COMPORTAMENTO	4
DIM_SITUACAO_MILITAR	4
DIM_OCUPACAO	4
DIM_JSM	22
DIM_POSTO_GRAD_ESPEC	11
DIM_CIDADAO	48
DIM_EVENTO_SERMIL	4
MTD	268

Além disso, há o Modelo Totalmente Desnormalizado (MTD) com 268 colunas, construído unindo-se todas as colunas de todas as tabelas do DW. A tabela Fato é FATO_EVENTO_CIDAD Possui 11 colunas e armazena dados como número de alistamento, datas, códigos geográficos e militares, idade, entre outros. Assim, as principais dimensões analisadas são:

- DIM_GEOGRAFIA, dimensão geográfica que armazena dados geográficos relacionados ao cidadão, como cidade, estado, país de nascimento e residência e dados geográficos da organização de alistamento militar, entre outros;
- DIM_TEMPO, dimensão de tempo armazena dados relacionados a medidas de tempo e suas variações como ANO, SEMANA, DATA e DIA;
- DIM_FORCA_MILITAR descreve a Força Armada a qual os dados estão relacionados;
- DIM_FORCA_DESEJA_SERVIR descreve em qual Força Armada o cidadão deseja servir ou se não deseja servir;
- DIM_DESTINO_MOBILIZACAO, esta dimensão apresenta as cidades para as quais os cidadãos incorporados serão mobilizados;
- DIM_ZONA_RESIDENCIAL descreve o tipo de zona residencial onde mora o cidadão alistado;
- DIM_CLASSE_RESERVA descreve o tipo de classe de reserva que o cidadão será classificado;
- DIM_COMPORTAMENTO descreve a classificação do comportamento do soldado;
- DIM_SITUACAO_MILITAR descreve a situação militar do cidadão, por exemplo, se foi excluído, alistado, capaz, incapaz, entre outros;

- Ocupação_Dim descreve todas as ocupações/empregos que o cidadão pode ter ao se alistar;
- DIM_JSM descreve todas as informações relacionadas às unidades de serviço militar;
- DIM_POSTO_GRAD_SPEC descreve todas as informações relacionadas ao posto ou graduação do militar;
- DIM_CIDADAO, uma das dimensões mais importantes, pois contém todos os dados pessoais do cidadão inscrito; e
- DIM_EVENTO_SERMIL, tabela que contém o código e a descrição de todos os eventos relacionados ao militar.

5.2 Características das Consultas e Cenários de Teste

Esta Seção apresenta as características das consultas e dos cenários de teste simulados no MongoDB, no HBase e no MySQL. Este trabalho compreende dez consultas. Cada consulta possui um grau de complexidade, que depende da quantidade de junções, de filtros, de agrupamento e de ordenamento. De uma maneira simplificada, a seguir serão apresentadas um resumo de cada consulta.

1. A consulta Q1 realiza uma contagem de todas linhas da tabela FATO_EVENTO_CIDADAO.
2. Q2 conta e agrupa por idade e situação de adiamento, os cidadãos que pediram adiamento no ano, a partir de 2011, ordenando por idade.
3. Q3 conta e agrupa a quantidade de soldados desertores por região militar e ano.
4. Q4 seleciona os atributos relacionados à unidade, região militar e evento de todos os soldados que optaram por uma situação específica descrita no código 7, a partir do ano de 2011 e ordenando por 3 atributos.
5. Q5 seleciona alguns atributos e conta os cidadãos que se enquadram no evento de código 8, agrupando por 8 atributos e ordenando por região militar, código da organização miliar e data do evento.
6. Q6 conta e agrupa os cidadãos que se alistaram em Santa Catarina, na 5^a região militar e que não desejam servir, a partir do ano de 2011, ordenando pela sigla da Circunscrição de Serviço Militar e a descrição de que não deseja servir.
7. Q7 seleciona atributos e conta os cidadãos que possuem um dos código de qualificação especificados na 11^a região militar, a partir de 2011, agrupando pelos atributos.

8. Q8 seleciona diversos atributos dos soldados que cometeram o crime de deserção, a partir do ano de 2011, nas 1^a, 3^a, 5^a e 12^a regiões militares.

9. Q9 seleciona a escolaridade, o município, UF, dentre outros atributos e conta os cidadãos que se alistaram no ano, a partir do ano de 2011, agrupando-os pelos atributos selecionados e ordenando por ano e grau de escolaridade.

10. Q10 realiza uma sub-consulta em que seleciona a região militar, o ano e o índice de massa corpórea (IMC) e conta os cidadãos alistados que possuem IMC maior que 25, a partir do ano de 2011, agrupando por região militar, ano e IMC.

A Tabela 5.2 mostra a complexidade de cada consulta analisada.

Tabela 5.2: Complexidade das consultas.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Junções	0	0	1	3	3	4	7	12	5	2
Filtros	0	2	2	2	1	5	3	3	2	3
Agrupamento	0	2	3	0	8	2	13	0	7	3
Ordenação	0	3	0	3	3	2	0	0	2	0
Atributos	1	5	5	7	9	8	14	15	8	10

Os dados utilizados referem-se aos anos de 2011 a 2018. Simulou-se a construção do DW inserindo dados na tabela de fatos ordenados por data em lotes anuais, aumentando a cardinalidade da tabela. Usamos nomes de referência A1 a A8 para identificar a tabela FATO com cardinalidade crescente, à medida que os dados de cada ano são adicionados. A Tabela 5.3 mostra o intervalo de tempo em anos dos lotes de dados carregados, seus nomes de referência, o respectivo número de linhas (totalizando 35.073.178) e o número de linhas retornadas para as consultas de Q1 a Q8. As consultas Q9 e Q10 foram testadas apenas com o HBase e MongoDB, mas, como referência, as linhas retornadas de Q9 e Q10 para o ano de 2011 foram 82.432 e 36, respectivamente. As Figuras 5.2 e 5.3 mostram os detalhes de cada consulta específica.

Tabela 5.3: Linhas analisadas por período.

Anos	**Referência**	**Número de Linhas da Tabela FATO**	**Número de Linhas Retornadas**							
			Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**
2011	A1	3,645,403	1	16	13	127	28	1	24	195
2011 to 2012	A2	9,458,748	1	16	26	308	80	1	43	380
2011 to 2013	A3	14,621,794	1	27	39	532	138	1	69	626
2011 to 2014	A4	18,206,124	1	16	52	899	185	1	133	847
2011 to 2015	A5	22,191,384	1	31	65	1468	226	1	162	1040
2011 to 2016	A6	26,059,430	1	32	77	4475	277	1	200	1262
2011 to 2017	A7	30,515,025	1	48	90	4766	2660	1	231	1504
2011 to 2018	A8	35,073,178	1	57	103	5296	7666	1	254	1619

Q1	Q2
SELECT COUNT(*) FROM Fato_Evento_Cidadao;	SELECT a11.Cidadao_Idade, a11.St_Adiamento, COUNT(DISTINCT a11.Cidadao_RA) QTD FROM Fato_Evento_Cidadao a11 WHERE Ano_Referencia >= 2011 AND Evento_Sermil_Cod = 2 GROUP BY a11.Cidadao_Idade, a11.St_Adiamento ORDER BY a11.Cidadao_Idade;
Q3	**Q4**
SELECT a13.OM_RM Ordem, a13.RM_Sigla, a11.Ano_Referencia, COUNT(DISTINCT a11.Cidadao_RA) QTD FROM Fato_Evento_Cidadao a11 JOIN Dim_Geografia a13 ON (a11.Geo_Cod_Seq = a13.Geo_Cod_Seq) WHERE (a11.Ano_Referencia >= 2011 AND a11.Evento_Sermil_Cod = 42) GROUP BY a13.OM_RM_Ordem, a13.RM_Sigla, a11.Ano_Referencia;	SELECT a13.Codom, a13.OM_Nome, a13.OM_RM_Ordem, a13.RM_Sigla, a12.Dtem_Ano, a11.Evento_Sermil_Cod, a14.Evento_Sermil_Desc FROM Fato_Evento_Cidadao a11 JOIN Dim_Tempo a12 ON (a11.Tempo_Cod_Seq = a12.Tempo_Cod_Seq) JOIN Dim_Geografia a13 ON (a11.Geo_Cod_Seq = a13.Geo_Cod_Seq) JOIN Dim_Evento_Sermil a14 ON (a11.Evento_Sermil_Cod = a14.Evento_Sermil_Cod) WHERE (a11.Evento_Sermil_Cod = 7 AND a12.Dtem_Ano >= 2011) ORDER BY a14.Evento_Sermil_Desc, a13.RM_Sigla, a13.OM_Nome;
Q5	**Q6**
SELECT a11.Cidadao_RA, a11.Dt_Evento, a12.OM_RM_Ordem, a12.RM_Sigla, a12.Codom, a12.OM_Nome, a11.Evento_Sermil_Cod, a14.Evento_Sermil_Desc, COUNT(DISTINCT a11.Cidadao_RA) QTD FROM Fato_Evento_Cidadao a11 JOIN Dim_Geografia a12 ON (a11.Geo_Cod_Seq = a12.Geo_Cod_Seq) JOIN Dim_Cidadao a13 ON (a11.Cidadao_RA = a13.RA) JOIN Dim_Evento_Sermil a14 ON (a11.Evento_Sermil_Cod = a14.Evento_Sermil_Cod) WHERE a11.Evento_Sermil_Cod = 8 GROUP BY a11.Cidadao_RA, a11.Dt_Evento, a12.OM_RM_Ordem, a12.RM_Sigla, a12.Codom, a12.OM_Nome, a11.Evento_Sermil_Cod, a14.Evento_Sermil_Desc ORDER BY OM_RM_Ordem, Codom, Dt_Evento;	SELECT a13.CSM_Sigla, a15.Forca_Deseja_Servir_Desc, COUNT(DISTINCT a11.Cidadao_RA) QTD FROM Fato_Evento_Cidadao a11 JOIN Dim_Tempo a12 ON (a11.Tempo_Cod_Seq = a12.Tempo_Cod_Seq) JOIN Dim_JSM a13 ON (a11.CSM_Cod = a13.CSM_Cod and a11.JSM_Cod = a13.JSM_Cod) JOIN Dim_Cidadao a14 ON a11.Cidadao_RA = a14.RA JOIN Dim_Forca_Deseja_Servir a15 ON (a14.Forca_Dsja_Serv_Cod = a15.Forca_Dsja_Serv_Cod) WHERE a12.Dtem_Ano >= 2011 AND a11.Evento_Sermil_Cod = 1 AND a13.UF_Sigla ='SC' AND a13.RM_Ordem = 5 AND a14.Forca_Dsja_Serv_Cod = 5 GROUP BY a13.CSM_Sigla, a15.Forca_Deseja_Servir_Desc ORDER BY a13.CSM_Sigla, a15.Forca_Deseja_Servir_Desc;

Figura 5.2: Detalhamento das consultas - Q1 a Q6.

Q7

```
SELECT  a13.Dest_Mob_Cod,
a17.Dest_Mob_Desc, a13.Qualif_Cod,
a14.Codom, a14.OM_Nome,
a13.Classe_Reserva_Cod, a16.Classe_Reserva_Desc,
a15.Pst_Grad_Cod, a15.Pst_Grad_Desc,
a12.Dtem_Ano, a14.OM_RM_Ordem,
a14.RM_Sigla, a18.JSM_Desc,
COUNT(DISTINCT a11.Cidadao_RA)  QTD
FROM    Fato_Evento_Cidadao a11
JOIN Dim_Tempo a12  ON
(a11.Tempo_Cod_Seq = a12.Tempo_Cod_Seq)
JOIN Dim_Cidadao a13 ON (a11.Cidadao_RA = a13.RA)
JOIN  Dim_Geografia  a14   ON
(a11.Geo_Cod_Seq = a14.Geo_Cod_Seq)
JOIN    Dim_Pst_Grad_Espec a15   ON
(a13.Pst_Grad_Cod_Seq = a15.Pst_Grad_Cod_Seq)
JOIN    Dim_Classe_Reserva  a16   ON
(a13.Classe_Reserva_Cod = a16.Classe_Reserva_Cod)
JOIN    Dim_Destino_Mobilizacao a17  ON
(a13.Dest_Mob_Cod = a17.Dest_Mob_Cod)
JOIN Dim_JSM a18   ON
(a18.CSM_Cod = a11.CSM_Cod
AND a18.JSM_Cod = a11.JSM_Cod
WHERE   (a12.Dtem_Ano >=  2011
AND a13.Qualif_Cod IN ('T22J', 'T23J', 'T24J', 'T25J',
  'T26J', 'T27J', 'T28J', 'T29J', 'T30J', 'T31J', 'T32J',
  'T33J', 'T34J', 'T35J', 'T36J', 'T37J', 'T38J', 'T39J',
  'T40J', 'T41J', 'T42J', 'T43J', 'T44J', 'T45J', 'T46J',
  'T47J', 'T48J', 'T49J', 'T50J', 'T51J', 'T52J', 'T53J',
  'T54J', 'T55J', 'T56J', 'T57J', 'T58J',  'T59J', 'T60J',
  'T61J', 'T62J', 'T63J', 'T64J', 'T65J', 'T66J', 'T67J',
  'T68J', 'T69J', 'T70J', 'T71J', 'T72J', 'T73J', 'T74J',
  'T75J', 'T76J', 'T77J', 'T78J', 'T79J')
AND OM_RM_Ordem = 11 )
GROUP BY    a13.Dest_Mob_Cod,
a17.Dest_Mob_Desc, a13.Qualif_Cod,
a14.Codom, a14.OM_Nome,
a13.Classe_Reserva_Cod, a16.Classe_Reserva_Desc,
a15.Pst_Grad_Cod, a15.Pst_Grad_Desc,
a14.RM_Sigla, a12.Dtem_Ano,
a14.OM_RM_Ordem, a18.JSM_Desc;
```

Q8

```
SELECT  a12.Dtem_Ano, a14.OM_Sigla,
a14.OM_RM_Ordem, a14.RM_Sigla,
a15.Situacao_Militar_Desc, a16.Forca_Militar_Desc,
a17.Forca_Deseja_Servir_Desc, a20.Ocupacao_Desc,
a19.Comportamento_Desc, a13.Ano_Vinculacao ,
a21.Classe_Reserva_Desc, a22.Dest_Mob_Desc,
a23.Evento_Sermil_Desc, a25.Zona_Residencial_Desc
FROM    Fato_Evento_Cidadao a11
JOIN    Dim_Tempo  a12        ON
(a11.Tempo_Cod_Seq = a12.Tempo_Cod_Seq)
JOIN    Dim_Cidadao a13       ON
(a11.Cidadao_RA = a13.RA)
JOIN    Dim_Geografia   a14     ON
(a11.Geo_Cod_Seq = a14.Geo_Cod_Seq)
JOIN    Dim_Forca_Militar  a16       ON
(a13.Forca_Militar_Cod = a16.Forca_Militar_Cod)
JOIN    Dim_Forca_Deseja_Servir  a17 ON
(a13.Forca_Dsja_Serv_Cod = a17.Forca_Dsja_Serv_Cod)
JOIN    Dim_Destino_Mobilizacao a22  ON
 (a13.Dest_Mob_Cod = a22.Dest_Mob_Cod)
JOIN    Dim_Zona_Residencial a25  ON
 a13.Zona_Residencial_Cod = a25.Zona_Residencial_Cod
JOIN    Dim_Classe_Reserva a21    ON
 a13.Classe_Reserva_Cod = a21.Classe_Reserva_Cod
JOIN    Dim_Comportamento a19    ON
a13.Comportamento_Cod = a19.Comportamento_Cod
JOIN    Dim_Evento_Sermil a23   ON
a11.Evento_Sermil_Cod =a23.Evento_Sermil_Cod
JOIN    Dim_Situacao_Militar a15 ON
a13.Situacao_Militar_Cod = a15.Situacao_Militar_Cod
JOIN    Dim_Ocupacao a20  ON
a13.Ocupacao_Cod = a20.Ocupacao_Cod
WHERE   a12.Dtem_Ano >= 2011
AND a11.Evento_Sermil_Cod = 42
AND a14.OM_RM_Ordem IN (1,3,5,12) ;
```

Q9

```
SELECT a14.Desc_Escolaridade, a13.Municipio_Nome,
a13.UF_Sigla, a15.RM_Sigla, a11.Ano_Referencia,
a11.Evento_Sermil_Cod, a16.Evento_Sermil_Desc,
COUNT(DISTINCT a11.Cidadao_RA)  QTD
FROM Fato_Evento_Cidadao a11
JOIN Dim_Cidadao a12 ON a11.Cidadao_RA= a12.RA
JOIN Dim_Municipio a13
ON  a12.Municipio_Resid_Cod=a13.Municipio_Cod
JOIN DIM_ESCOLAR a14
ON a12.Escolar_Cod_Seq = a14.Escolar_Cod_Seq
JOIN Dim_Geografia a15
ON a12.Geografia_Cod_Seq =a15.Geografia_Cod_Seq
JOIN Dim_Evento_Sermil a16
ON a11.Evento_Sermil_Cod=a16.Evento_Sermil_Cod
WHERE   a11.Evento_Sermil_Cod=1
AND a11.Ano_Referencia>= 2011
GROUP BY a11.Ano_Referencia, a13.UF_Sigla,
a14.Desc_Escolaridade, a13.Municipio_Nome,
a15.RM_Sigla, a11.Evento_Sermil_Cod,
a16.Evento_Sermil_Desc
ORDER BY a11.Ano_Referencia,
a14.Desc_Escolaridade;
```

Q10

```
SELECT  Teste.RM_Sigla, Teste.Ano, Teste.IMC,
COUNT(DISTINCT Teste.Cidadao_RA) QTD
FROM (
SELECT Ft.CIDADAO_RA   CIDADAO_RA,
Geo.RM_Sigla RM_Sigla, Ft.Ano_Referencia Ano,
FLOOR((Cid.Peso*10000)/(Cid.Altura*Cid.Altura)) AS IMC
FROM  Fato_Evento_Cidadao Ft
JOIN Dim_Cidadao Cid ON Ft.Cidadao_RA=Cid.RA
JOIN Dim_Geografia Geo
ON Ft.Geografia_Cod_Seq=Geo.Geografia_Cod_Seq
WHERE (IMC >= 25)  AND Ft.Ano_Referencia>=2011
AND Ft.Evento_Sermil_Cod= 1
            ) Teste
GROUP BY   Teste.RM_Sigla, Teste.Ano, Teste.IMC;
```

Figura 5.3: Detalhamento das consultas - Q7 a Q10.

5.3 Definição da Arquitetura de Hardware e Método Aplicado nas Simulações

Foram feitas simulações no Atlas MongoDB, utilizando-se um *sharded cluster* com 3 *shards*, conforme pode-se verificar nas figuras 5.4 e 5.5.

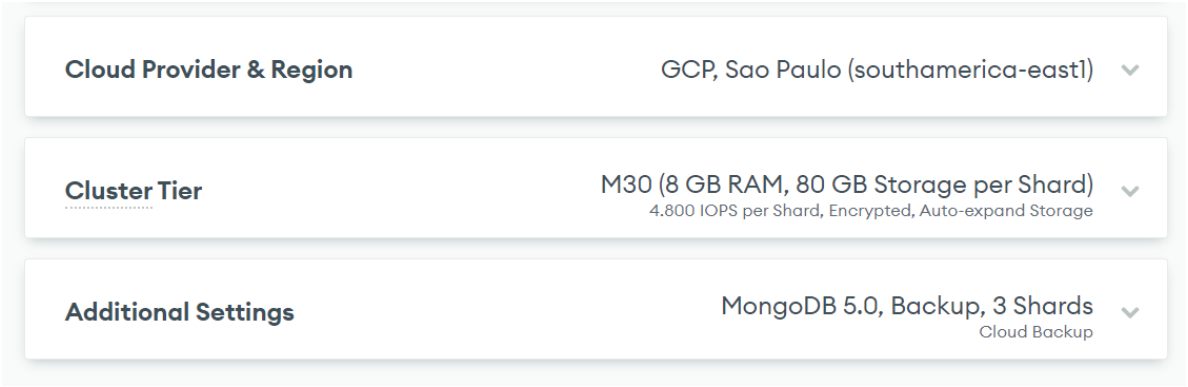

Figura 5.4: Configuração ambiente computacional MongoDB - parte 1.

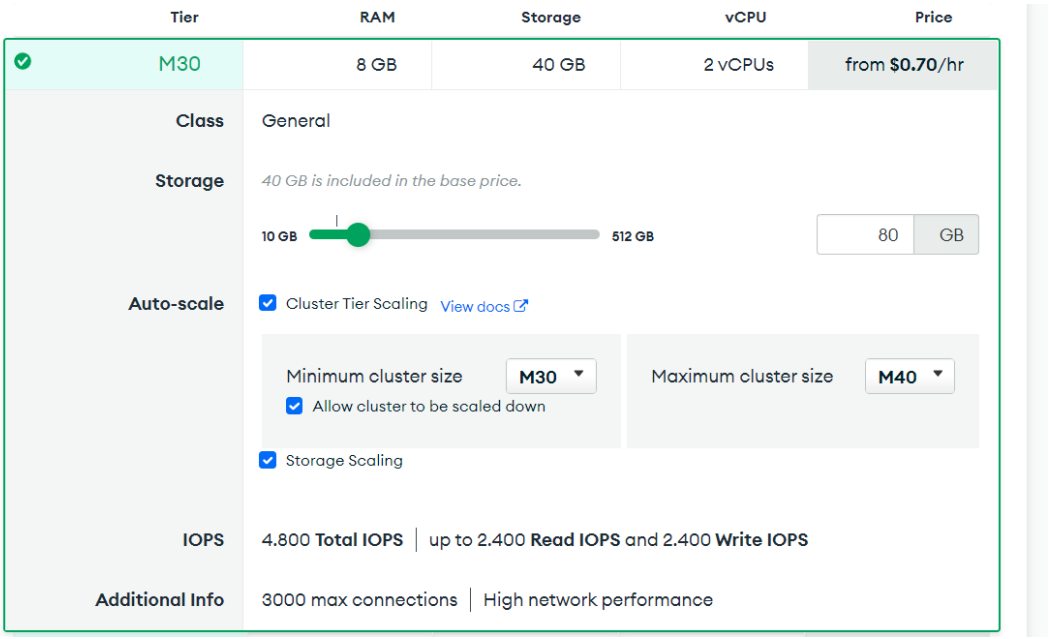

Figura 5.5: Configuração do ambiente computacional MongoDB - parte 2.

No ambiente do Google Cloud foi construído um *cluster* Hadoop com cinco nós, configurados como um mestre (YARN *Resource Manager*) e quatro trabalhadores (YARN *Node Managers*). A configuração do nó mestre escolhida foi a *Optimized for Compute C2 Standard*, com quatro CPUs com 16 GB de memória e um disco primário de 700 GB do tipo HDD. Por fim, cada nó de trabalho neste cluster possui duas CPUs com 7,5 GB

de memória e um disco primário com 700 GB de tipo de disco HDD. No Hadoop foram feitas simulações de Q1 a Q10 utilizando-se o HBase, juntamente com o Hive, com o MapReduce e Tez como motor de execução (*execution engine*). Esses resultados foram comparados também com os resultados obtidos em [55], em que foram simuladas Q1 a Q8, com armazenamento no HDFS e utilizando o Hive como DW com três processadores distintos: MR, Tez e Spark.

A configuração escolhida para o ambiente do MySQL está descrita na Figura 5.6.

Figura 5.6: Configuração do ambiente computacional MySQL.

A métrica selecionada para medir o desempenho foi o tempo de execução da consulta em segundos(s). As consultas foram executadas dez vezes e, em seguida, foi calculada a média das medições.

Devido a restrições financeiras não foi possível definir configurações equivalentes para os três SGBDs analisados, verificando-se por exemplo que a máquina do MySQL é bem

superior às demais. Portanto, no momento da análise dos resultados, é importante verificar nos resultados a possibilidade de implementação, mais do que avaliar os números dos tempos das consultas em si.

Neste trabalho serão realizadas as seguintes comparações.

1. comparação entre o Modelo Dimensional e o Modelo Totalmente Desnormalizado para o MongoDB e o HBase;
2. comparação entre os melhores resultados do MongoDB e HBase com o modelo do BD relacional MySQL; e
3. comparação entre os processadores e os formatos de arquivos utilizados no HBase

Além da comparação dos resultados, serão relacionados os desafios surgidos em cada BD específico e a análise da sugestão de adaptação do Ciclo de Vida para o os BD NoSQL em questão.

5.4 Especificidades dos Bancos Escolhidos

5.4.1 Especificidades do MongoDB

O MongoDB é conhecido por sua simplicidade de manuseio e configuração. Assim, a criação de uma coleção ("tabela") e inserção dos dados na mesma é uma tarefa simples e pode ser vista como uma vantagem desse BD. No que se refere ao formato dos dados, as coleções os armazenam no formato JSON. A conversão de JSON para Parquet não é nativa do MongoDB e sua implementação ocorre de maneira separada, mapeando cada coleção do Mongo em múltiplos arquivos Parquet. Existem soluções pagas que realizam essa integração. Atualmente, a única maneira oferecida pela própria empresa do MongoDB seria criando uma instância de BD federada no Atlas e conectando-a ao bucket S3 da Amazon AWS. A seguir a Figura **??** mostra detalhes da coleção $DIM\_CLASSE\_RESERVA$ definida no MOngoDB. Essa coleção foi utlizada nas simulações do modelo dimensional no MOngoDB, onde cada tabela do DW original foi convertida em uma coleção no MongoDB. Em seguida a Figura **??** mostra um exemplo de um documento da coleção única do MTD no MongoDB. Os dados utilizados para os exemplos são os mesmos do estudo de caso desta pesquisa.

O processamento das consultas no MongoDB é feito em sequência, chamado de pipeline. O objetivo do pipeline criado para as consultas do DW é aplicar primeiramente os comandos que irão reduzir a quantidade de dados a ser processada, para assim agilizar o tempo total da consulta. As Figuras 5.9 e 5.10 mostram o pipeline para a consulta Q6. Em ambas as figuras os dados na coluna à esquerda do código representam os dados de

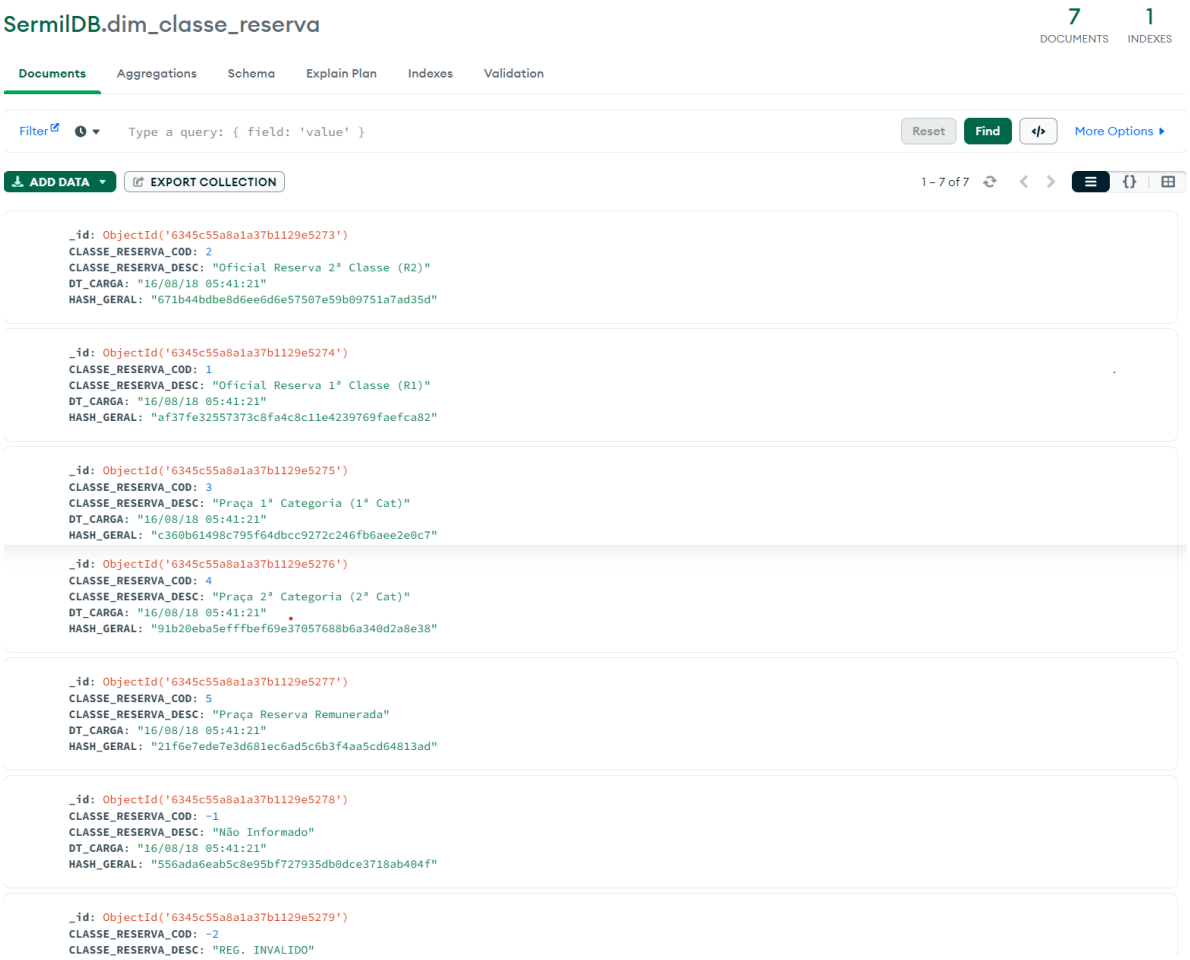

Figura 5.7: Detalhamento da coleção DIM_CLASSE_RESERVA definida no MongoDB.

entrada e os da coluna à direita são os dados de saída, após o código. Os dados à direita do código na parte 2 do pipeline é a saída da consulta Q6. Ressaltando que esses dados da figura são uma amostra do conjunto de dados total, a título de ilustração.

5.4.2 Especificidades do HBase

A configuração no HBase não é tão simples como no MongoDB. Para realizar consultas no HBase utilizando o Hive é necessário criar a tabela no HBase, inserir os dados do arquivo CSV e, então, criar uma tabela no Hive que seja mapeada exatamente junto à tabela desejada no HBase. Uma desvantagem do HBase é que ele não permite associar chaves compostas em conjunto com o comando ImportTsv, logo, como esse comando foi o utilizado para inserir os dados nas tabelas, para alguns casos foi necessário inserir uma coluna com uma chave artificial para contornar a situação.

```
{
  "_id": {
    "$oid": "634f02516d0a0ee3b259efa9"
  },
  "CIDADAO_RA": "40453400083",
  "EVENTO_SERMIL_COD": "49",
  "DT_EVENTO": "18/02/11",
  "CSM_COD": "4",
  "JSM_COD": "45",
  "GEOGRAFIA_COD_SEQ": "12185",
  "CIDADAO_IDADE": "20",
  "ALTURA": "176",
  "ST_ADIAMENTO": "N",
  "TEMPO_COD_SEQ": "22329",
  "ANO_REFERENCIA": "2011",
  "RA": "40453400083",
  "MUNICIPIO_NASCIMENTO_COD": "12280000",
  "PAIS_NASCIMENTO_COD": "1",
  "ESTADO_CIVIL_COD": "1",
  "ESTADO_CIVIL_DESC": "Solteiro",
  "SEXO_SG": "M",
  "ESCOLAR_COD_SEQ": "47",
  "OCUPACAO_COD": "010310",
  "SITUACAO_MILITAR_COD": "49",
  "VINCULACAO_ANO": "2009",
  "ATUALIZACAO_DATA": "",
  "ZONA_RESIDENCIAL_COD": "2",
  "MUNICIPIO_RESIDENCIAL_COD": "12280000",
  "PAIS_RESIDENCIA_COD": "1",
  "CSM_COD1": "4",
  "JSM_COD1": "45",
  "DISPENSA_INCORP_COD": "-1",
  "TIPO_SANGUINEO": "A",
  "FATOR_RH": "Positivo",
  "PESO": "66",
  "ALTURA1": "176",
  "FORCA_MUSCULAR": "110",
  "CABECA": "55",
  "CALCADO_NR": "40",
  "CINTURA_NR": "77",
  "UNIFORME_NR": "G",
  "ACUIDADE_VISUAL_DESC": "Perfeita",
  "ACUIDADE_AUDITIVA_DESC": "Perfeita",
  "FORCA_DESEJA_SERVIR_COD": "3",
  "SABE_NADAR_ST": "N",
  "CSE_INDICACAO_COD": "-1",
  "CSE_RESULTADO_COD": "-1",
  "GEOGRAFIA_COD_SEQ1": "12185",
  "GRUPAM_INCORP_COD": "A",
  "PADRAO_CODIGO": "C03",
  "TIPO_DISTRIBUICAO_COD": "1",
  "NIVEL_AJUSTAMENTO_COD": "7",
  "GRUPO_DISTRIB_COD": "2",
+ 235 campos
}
```


Figura 5.8: Exemplo de documento do MTD no MongoDB.

Figura 5.9: *Pipeline* consulta Q6 - parte 1.

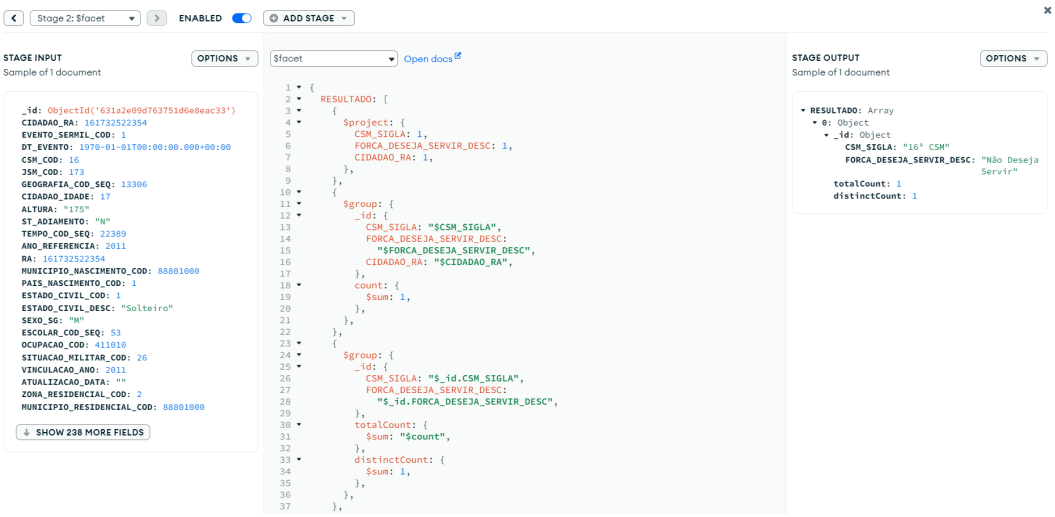

Figura 5.10: *Pipeline* consulta Q6 - parte 2.

Em contrapartida, a partir do momento em que as tabelas do Hive e HBase estão mapeadas, a consulta utilizando a linguagem HQL é algo bem simples e a conversão ao formato Parquet não é uma tarefa complexa. A grande vantagem do HBase é estar inserido no ecossistema Hadoop, o que lhe confere uma grande flexibilidade e diferentes possibilidades de integração com as ferramentas existentes no ecossistema Hadoop. As Figuras 5.11 e 5.12 mostram a estrutura de duas tabelas do DW no HBase.

A seguir, na Figura 5.13 são listados os comandos executados para criação das tabelas no HBase e no Hive.

Row Key	Dados								
RA	ESTADO_CIVIL_COD	ESTADO_CIVIL_DESC	ESCOLAR_COD_SEQ	SITUACAO_MILITAR_COD	VINCULACAO_ANO	ZONA_RESIDENCIAL_COD	MUNICIPIO_RESIDENCIAL_COD	PAIS_RESIDENCIA_COD	...
1593949059	1	Solteiro	50	17	2011	1	62823000	1	...
1593948663	1	Solteiro	65	17	2011	1	62823000	1	...
199091162	1	Solteiro	52	17	2011	2	78850000	1	...
...	...	...	...	...	...	...	...	...	...

Column Family ou Família de Colunas

Figura 5.11: Extrato da tabela DIM_CIDADAO no HBase.

Row Key	DESC	INFO	
CLASSE_RESERVA_COD	CLASSE_RESERVA_DESC	DT_CARGA	HASH_GERAL
4	Praça 2ª Categoria (2ª Cat)	18-APR-22	074658b4a971a27564385c9b3478ed105cf267a7
1	Oficial Reserva 1ª Classe (R1)	18-APR-22	12356c78649357090f434f3f9f674728d2b44394
3	Praça 1ª Categoria (1ª Cat)	18-APR-22	5b79f7b9299904892dfd7dc0b4191d551e5ec6b5
2	Oficial Reserva 2ª Classe (R2)	18-APR-22	817e5753986b5eb0d589368a84da8d3c00b3cb4b
5	Praça Reserva Remunerada	18-APR-22	c49ca11fac82963c0281edd5da3c4f59ddf96075
-1	Não Informado	18-APR-22	8eca77236585c58b2f74c17e3271dde9cfd783cf
-2	REG. INVALIDO		

Column Families ou Famílias de Colunas

Figura 5.12: Tabela HBase - DIM_CLASSE_RESERVA.

Criação da tabela no HBase

```
create 'comportamento', 'comport_desc', 'dt', 'hash'
```

Importação dos dados para o HBase

```
hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=, -Dimporttsv.columns=HBASE_ROW_KEY,comport_desc:comport_desc,
dt:dt,hash:hash comportamento gs://sermil/comportamento_sp_sh.csv
```

Criação da tabela no Hive vinculada ao HBase

```
CREATE EXTERNAL TABLE DIM_COMPORTAMENTO (
  COMPORTAMENTO_COD Double,
  COMPORTAMENTO_DESC Varchar(30),
  DT_CARGA VARCHAR(10),
  HASH_GERAL Varchar(42)
) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,
comport_desc:comport_desc,dt:dt,hash:hash") TBLPROPERTIES("hbase.table.name" = "comportamento");
```

Criação da tabela no Hive armazenada como formato Parquet

```
create table pq_dim_comportamento stored as parquet as select * from DIM_COMPORTAMENTO;
```

Figura 5.13: Comandos de criação das tabelas no HBase e Hive.

Capítulo 6

Resultados

Este Capítulo apresentará os resultados da pesquisa. A Seção 6.1 apresenta como primeiro resultado a proposta de ciclo de DW adaptado do ciclo de Kimball. Para validar esse ciclo proposto foram desenvolvidos DWs utilizando os SGBDs MongoDB e HBase, ambos pertencentes ao paradigma NoSQL. As próximas Seções 6.2 e 6.3 apresentam o desempenho de cada um deles e os compara ao desempenho da abordagem sob o paradigma relacional, utilizando-se o SGBD MySQL, na Seção 6.4. Essa comparação visa identificar a viabilidade de se trabalhar com a tecnologia NoSQL em DW. A Seção 6.5 faz uma discussão dos resultados, alinhada com os objetivos apresentados.

6.1 Ciclo de Vida Adaptado de Kimball

Conforme apresentado no Capítulo 4, na Figura 4.1,a adaptação sugerida localiza-se originalmente na etapa do projeto físico de Kimball. A etapa proposta permanece com as mesmas características e os mesmos entregáveis que a original, porém é subdividida em duas etapas, o projeto lógico NoSQL e o projeto físico NoSQL. A justificativa para essa subdivisão é separar o nível de abstração lógico do físico e mostrar que o relacionamento entre as duas etapas no paradigma NoSQL ocorre de maneira distinta ao relacional. Como o projeto físico não é tão dependente do lógico em NoSQL, as escolhas dos parâmetros específicos do banco escolhido podem acarretar em um mau desempenho do DW, provocando um retorno à etapa lógica para a construção de um novo modelo. Esse caminho de ida e volta, chamado de iteração, pode ocorrer diversas vezes até que se obtenha o resultado planejado. Dessa forma, após atender o primeiro objetivo específico, o estudo de caso visa validar o ciclo proposto, apresentar o desempenho de cada DW NoSQL para identificar a validade da aplicação da tecnologia NoSQL em DW.

6.2 Desempenho do DW Utilizando o MongoDB

Esta Seção apresenta o estudo de caso utilizando-se o BD MongoDB. São testados dois modelos diferentes e será escolhido o modelo que apresentar o melhor desempenho entre os dois, para então compará-lo ao modelo relacional e ao modelo que apresenta melhor desempenho no HBase. A etapa da projeto lógico NoSQL permite a formação de diferentes modelos, de acordo com a característica do banco escolhido.

As simulações no MongoDB desta pesquisa utilizaram um Modelo Totalmente Desnormalizado (MTD) e um Modelo Dimensional (MD). O MTD constitui-se de uma única coleção, onde cada coluna do modelo relacional equivale a um documento nesta grande coleção. Já o MD relaciona uma tabela do DW relacional a uma coleção no Mongo. Sendo assim, o MD é constituído de 24 coleções.

A Figura 6.1 mostra os resultados das consultas Q1 a Q10, relativos ao MTD, em escala logarítmica. A Tabela 6.1 apresenta o resultado geral das consultas no MOngoDB.

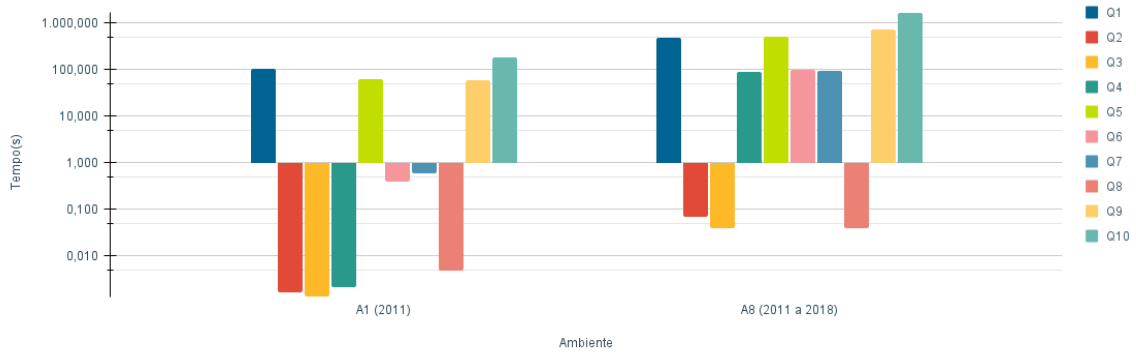

Figura 6.1: Consultas Q1 a Q10 - MTD - MongoDB - ano de 2011 e de 2011 a 2018.

Anos\Consulta	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
A1 (2011)	102,470	0,002	0,001	0,002	62,350	0,387	0,580	0,005	57,450	180,250
A2 (2011 a 2012)	142,291	0,002	0,008	0,005	145,224	0,920	0,936	0,009	178,031	442,373
A3 (2011 a 2013)	175,156	0,003	0,013	0,008	227,762	2,553	2,377	0,014	268,522	662,462
A4 (2011 a 2014)	253,179	0,004	0,018	0,013	288,628	3,292	3,565	0,019	388,688	893,167
A5 (2011 a 2015)	304,960	0,004	0,019	0,020	361,727	5,788	4,709	0,023	447,785	871,705
A6 (2011 a 2016)	352,234	0,006	0,027	0,058	405,140	7,690	8,032	0, 28	536,875	1.270,423
A7 (2011 a 2017)	412,235	0,019	0,031	24,267	444,360	10,244	10,836	0,033	635,705	1.455,131
A8 (2011 a 2018)	481,359	0,069	0,039	89,615	505,039	95,297	94,469	0,038	725,205	1.663,704

Tabela 6.1: Dados das consultas Q1 a Q10 - MTD - MongoDB.

Já a figura 6.2 compara os resultados do MD e MTD para o ambiente A1 (ano de 2011), que contém em torno de 3,6 milhões de registros. Ao realizar-se o projeto do DW com o MD, foi observada uma inviabilidade em se realizar todas as consultas com as coleções em questão. Devido ao tamanho dos dados e à configuração escolhida, o tempo

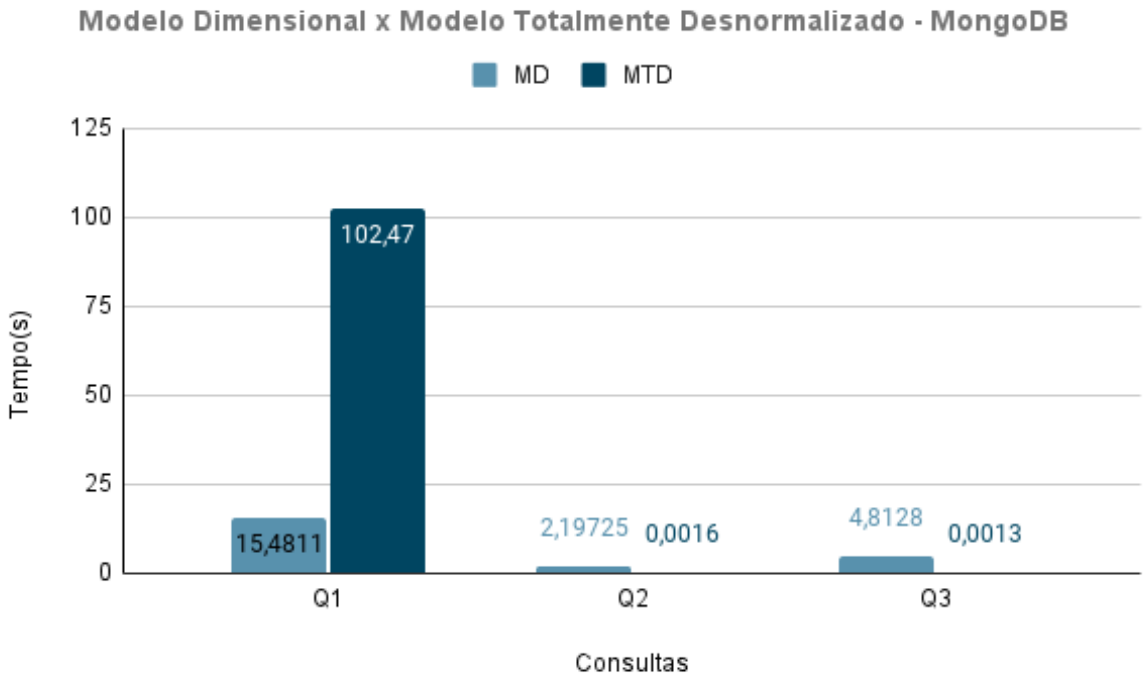

Figura 6.2: Comparação MD x MTD.

para realizar o "*join*"entre as tabelas, a partir de Q4, tornou o seu processamento inviável. Foram testadas diferentes *pipelines*, isto é, com comandos em ordens diferentes ou que fossem semelhantes, configurações que aumentam o limite de memória entre os passos do *pipeline*, porém nenhuma dessas configurações obtiveram sucesso, resultando no abandono dessa linha de ação. Sobre a carga dos dados, utilizando-se o MongoDB Atlas na nuvem, o processo de carga dos dados ocorreu de maneira incremental, onde cada ano adicionado foi carregado em torno de 90 minutos.

6.3 Desempenho do DW Utilizando o HBase

Esta Seção apresenta o estudo de caso utilizando-se o BD HBase. São testados dois modelos diferentes, com dois formatos de arquivos distintos e três motores de execução. Será escolhida a configuração que apresentar o melhor desempenho entre todos, para então compará-la ao modelo relacional e ao MTD do MOngoDB. O trabalho [55] apresenta simulações com os mesmos dados e parte das consultas aqui apresentadas, utilizando-se os motores de execução Tez e Spark. Já neste trabalho, os processamentos analisados foram o MR e o Tez. A Figura 6.3 destaca a diferença de tempo das consultas ao se variar o formato dos dados entre CSV e Parquet para o MD. O gráfico da Figura 6.4 compara

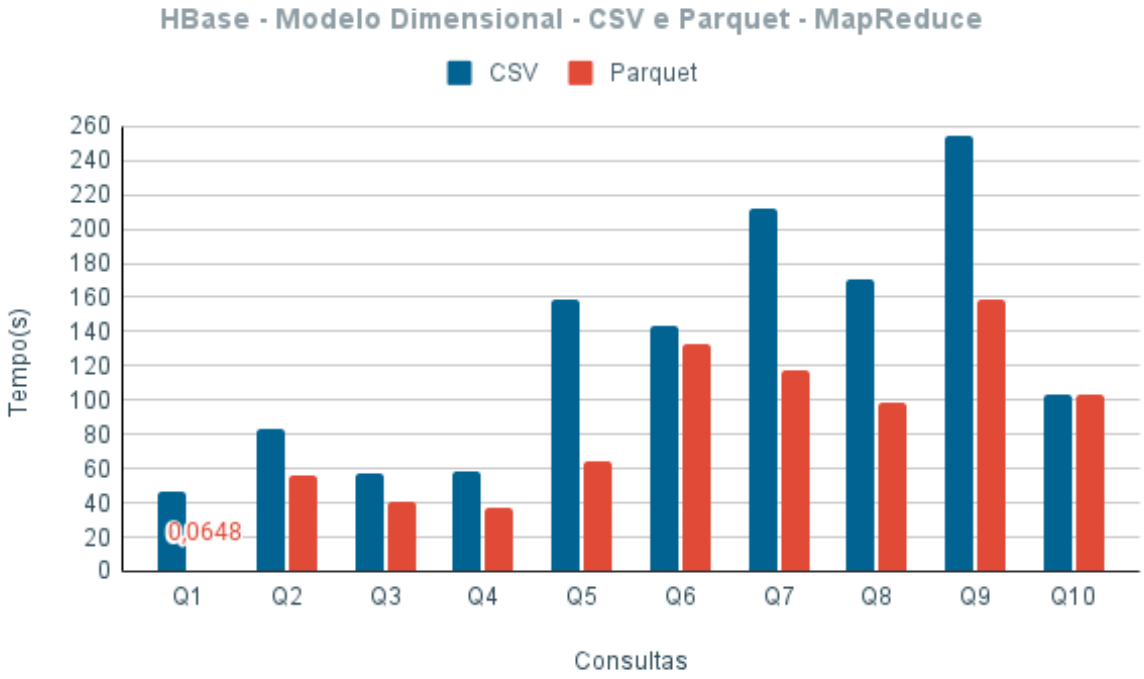

Figura 6.3: HBase - MD - MapReduce - CSV x Parquet.

o MD e o MTD, cada um com os formatos de dados CSV e Parquet. Vale destacar o melhor desempenho para os modelos com os dados das tabelas no formato Parquet.

A Figura 6.5 visa destacar a diferença de tempo do processamento ao se utilizar o motor de execução MR ou Tez. Os resultados do gráfico são do MTD. A Figura 6.6 mostra a comparação dos tempos das consultas para o ano de 2011, entre o MTD no MongoDB, o MD e o MTD no HBase, ambos utilizando Tez e dados em Parquet.

Inicialmente, o formato dos dados carregados nas tabelas era CSV, porém, no intuito de melhorar o desempenho das consultas, as tabelas foram criadas no formato Parquet. Essa modificação, assim como a escolha do Tez como motor de execução, melhoraram substancialmente os resultados. Essa abordagem sinaliza, por exemplo, a grande relevância do projeto físico quando se utiliza o HBase.

Com relação à carga, a mesma foi realizada de maneira incremental e o tempo de carga dos dados foi menor que no MongoDB, em torno de 10 minutos para cada ano incrementado.

6.4 Comparação com o Paradigma Relacional

As dez consultas foram executadas nos anos de 2011 a 2018, conforme mostrado na Tabela 5.3. Cada cenário anual apresenta uma quantidade de linhas distintas, que foram

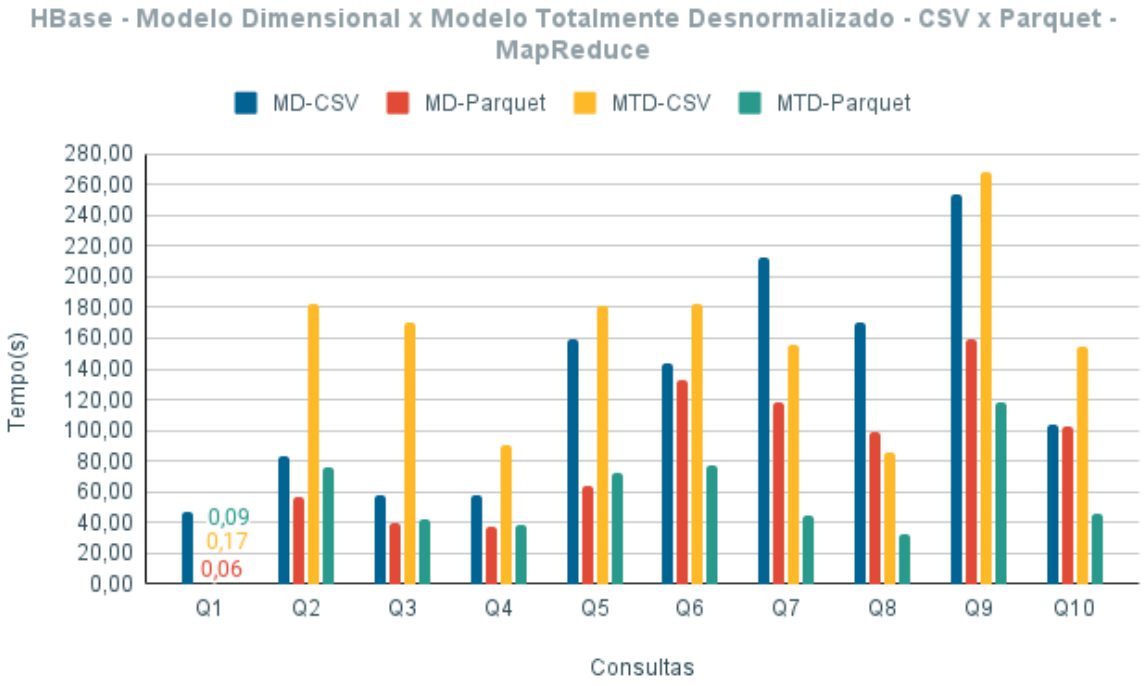

Figura 6.4: HBase - MapReduce - MD x MTD e CSV x Parquet.

incrementadas anualmente. Os desempenhos das simulações em relação às dez consultas estão representados pelas tabelas 6.2 e 6.3. As tabelas apresentam os resultados das simulações no DW MySQL, MongoDB com MTD, Tez com MTD e Spark com MD, este último publicado no artigo [55].

6.5 Discussão dos Resultados

No que diz respeito ao ciclo de vida NoSQL, percebe-se que há uma menor dependência entre o modelo físico e o lógico. Também observa-se que o tipo de formato de dados e o motor de execução impactam no resultado quando usa-se o ecossistema Hadoop. De acordo com os objetivos apresentados, através dos resultados pode-se verificar que o uso de Parquet favorece o tempo de todas as consultas com relação ao formato CSV. Outra comparação evidente foi entre MR e Tez. O uso do Tez como motor de execução favoreceu o tempo de consulta de todas as consultas também. Além disso, utilizando-se a configuração $Tez + Parquet$ e comparando-se o MTD e o MD, observa-se um melhor desempenho do MTD. O MD apresentou um melhor desempenho ao ser usado com *Spark*, conforme se verfica em [55]. Pode-se observar que os maiores impactos ocorrem no MongoDB. Tanto o MD, que não sustentou as simulações por completo, como o MTD apresentaram tempos de consulta elevados. Ainda é possível observar no MongoDB três consultas que

Tabela 6.2: Tempo em segundos (s) das simulações para 3,6 milhões de registros.

Consulta	MySQL	MongoDB - MTD	Spark - MD	Tez - MTD
Q1	0,243	102,470	0,149	0,073
Q2	0,002	0,002	3,780	3,240
Q3	0,007	0,001	5,030	16,905
Q4	0,004	0,002	4,910	4,480
Q5	0,002	62,350	6,527	5,845
Q6	0,220	0,387	4,770	15,750
Q7	0,150	0,580	3,920	17,971
Q8	0,006	0,005	17,720	2,650
Q9	17,057	57,450	-	60,347
Q10	6,859	180,250	-	11,853

Tabela 6.3: Tempo em segundos (s) das simulações para 35 milhões de registros.

Consulta	MySQL	MongoDB - MTD	Spark - MD	Tez - MTD
Q1	2,358	481,359	0,122	0,066
Q2	0,077	0,069	9,770	19,770
Q3	0,033	0,039	3,730	15,822
Q4	17,217	89,615	6,740	16,450
Q5	0,118	505,039	6,062	28,688
Q6	1,693	95,297	9,080	22,520
Q7	2,175	94,469	21,200	27,576
Q8	0,032	0,038	17,790	26,770
Q9	135,341	725,205	-	75,673
Q10	2,850	1663,704	-	19,120

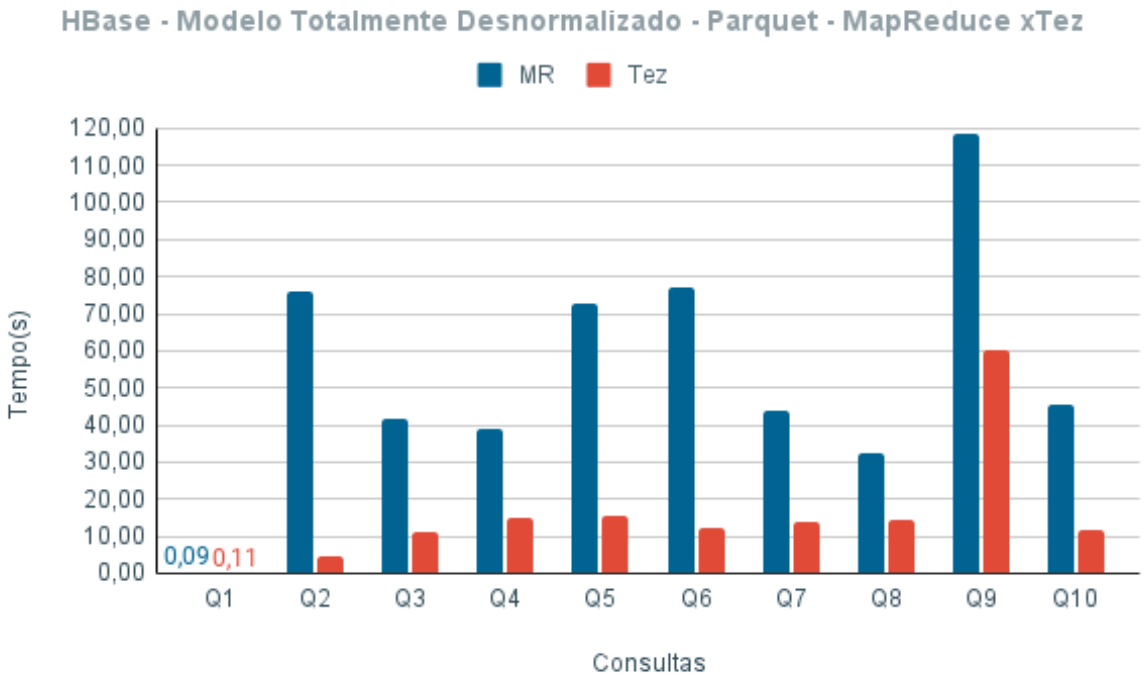

Figura 6.5: HBase - MTD - Parquet - MR x Tez.

apresentaram resultados competitivos. Talvez com um melhor *tuning* do banco, consiga-se melhorar os resultados de mais consultas, porém, com os resultados obtidos, o pior desempenho foi o do MongoDB.

De qualquer forma, é importante ressaltar que nao é possivel fazer uma comparação direta entre os tempos de cada DW, devido às diferenças nas configurações das máquinas. Contudo, o objetivo é apresentar que com NoSQL, o projeto físico e lógico pode causar muita diferença no resultado.

A ideia de comparação dos ambientes é mostrar a possibilidade de fazer as consultas, pois como não houve ambiente computacional disponível, as configurações dos ambientes foram diferentes e dessa forma, o valor do tempo das consultas demonstra uma viabilidade ou possibilidade e não que uma configuração é melhor que a outra de forma absoluta.

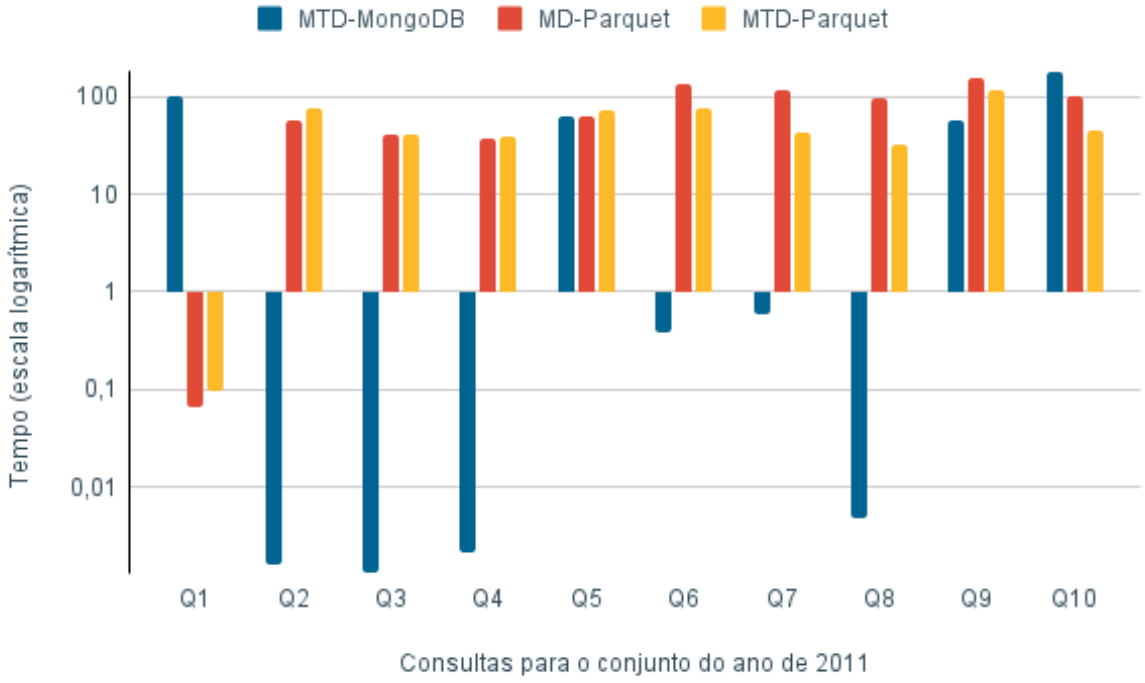

Figura 6.6: MTD MongoDB x MTD Parquet/Tez X MD Parquet/Tez HBase - ano de 2011.

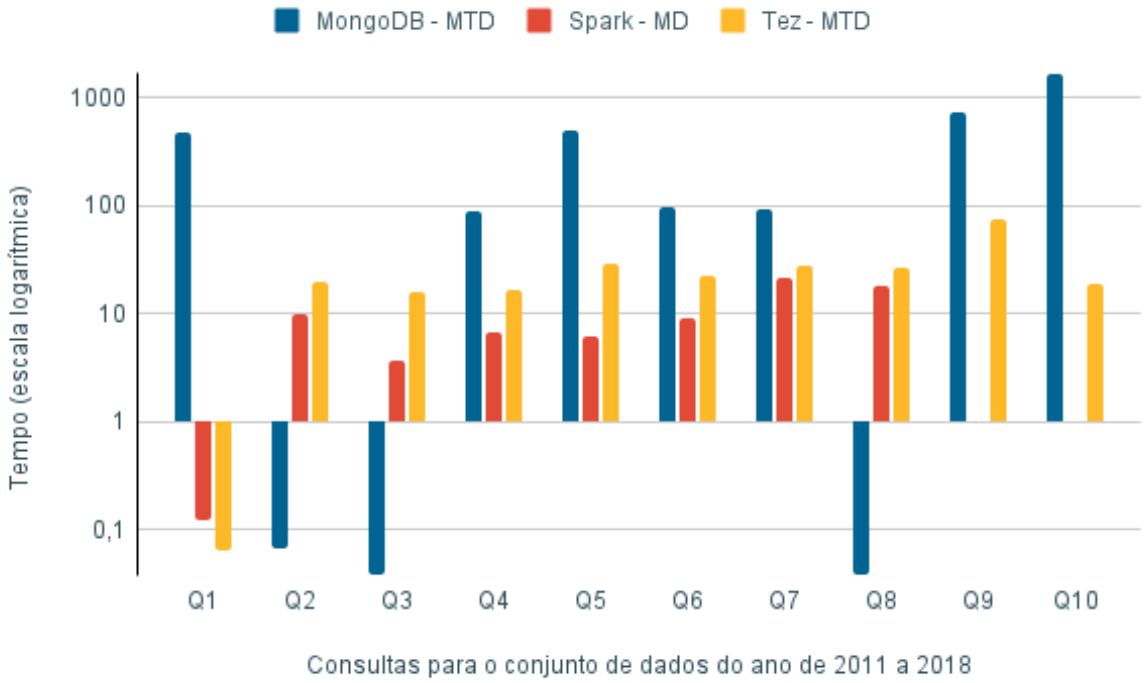

Figura 6.7: Comparação entre todos os resultados para o conjunto de dados de 2011 a 2018.

Capítulo 7

Conclusão

Nesse trabalho apresentou-se um estudo sobre o ciclo de vida do DW sob o paradigma NoSQL. De acordo com os objetivos, propôs-se uma adaptação ao ciclo de vida de DW, definido por Kimball, para o paradigma NoSQL. Dessa forma, cabe ressaltar que o objetivo não é criar um DW novo e sim adaptar um DW existente com novas arquiteturas.

Assim, buscou-se identificar as fases do ciclo de vida do DW que são mais impactadas pela migração de paradigma. As etapas escolhidas para análise foram a modelagem dimensional e o projeto físico, incluindo as atividades e passos relacionadas a elas. A partir da leitura dos artigos selecionados no Capítulo 3 definiram-se as adaptações necessárias.

Em seguida, foram especificados os bancos NoSQL (MongoDb e HBase) a serem utilizados no estudo de caso e os esquemas de interesse. O banco relacional escolhido foi o MySQL, que foi usado na comparação de desempenho do estudo de caso. Após esse procedimento, os dados foram editados, escolheram-se as consultas a serem realizadas e construíram-se os modelos a serem utilizados.

Como detalhamento do estudo foram definidos 10 consultas com perfis distintos e computados os tempos de resposta de cada uma para 8 diferentes cenários (anos), sendo que a cada novo cenário, mais registros eram incluídos. O intervalo de registros analisados variou de 3,6 a 35 milhões. Todos os ambientes foram configurados na nuvem e, ao final, os resultados obtidos foram comparados aos do modelo relacional, representado pelo BD MySQL.

A implementação de diferentes ambientes com bancos tão distintos fortalece algumas conclusões obtidas. Inicialmente, o formato dos dados carregados nas tabelas do HBase era CSV, porém, no intuito de melhorar o desempenho das consultas, as tabelas foram criadas no formato Parquet. Essa modificação, assim como a escolha do Tez como motor de execução, melhoraram substancialmente os resultados para os dois modelos, porém, o MTD apresentou um melhor desempenho em relação ao MD. Outra configuração que apresentou bons resultados foi o uso do Spark junto com o modelo dimensional e com

os dados no formato Parquet. É notório que no paradigma relacional o modelo lógico é totalmente dependente do modelo físico. Já no paradigma NoSQL esse relacionamento não é tão dependente, pois, conforme verificado, é possível escolher o modelo da tabela de acordo com o desempenho apresentado no modelo físico, ocasionando um ciclo iterativo, que é finalizado quando os resultados são otimizados ou alcançam um determinado desempenho. Assim, a escolha entre diferentes parâmetros físicos, como modelo e formato de dados e "motor de execução"podem trazer diferentes resultados para o desempenho final do DW. Dessa forma, no paradigma NoSQL, é necessário acrescentar essa possibilidade de retorno à etapa do modelo lógico caso se deseje melhorar o desempenho das consultas. O aumento da relevância do nível físico não diminui a importância do processo de modelagem conceitual e lógica, mas ressalta uma maior influência do projeto lógico e físico nos resultados e nas tabelas escolhidas.

Foi possível verificar em ambos BDs como o modelo físico, assim como o formato das tabelas, influenciam nos resultados das consultas. Nas simulações com MongoDB ficou evidente o desempenho superior do MTD com relação ao MD. De fato, o MD não se mostrou um modelo viável para se trabalhar com as tabelas em questão, devido ao seu tamanho e à grande quantidade de junções utilizadas nas consultas desta pesquisa. Nas simulações com o HBase destaca-se a influência do modelo das tabelas, formato dos dados e motor de execução no desempenho final das consultas. Apesar dos resultados apontarem para uma configuração específica que favoreça o desempenho das consultas em análise, tanto no MongoDB como no HBase, a flexibilidade de escolha destes parâmetros físicos, oferecida por ambos SGBDs, permitirá sempre que seja feito um estudo da relação custo x benefício adaptado aos dados a serem analisados. Portanto, mesmo não tendo havido ambiente computacional disponível para a comparação dos tempos dos três SGBDs, mais importante que isso, é mostrar que com NoSQL os projetos lógico e físico têm grande influência no resultado final do DW e que além disso é possível sim fazer as consultas aos DWs.

Portanto, o estudo de caso mostrou que, dentro do ciclo de vida proposto, as arquiteturas NoSQL apresentadas como resultado apresentam bom desempenho com relação à relacional, mesmo com diferença dos ambientes computacionais, mostrando que é viável utilizar-se a tecnologia NoSQL em DW. Como trabalhos futuros, pode-se analisar outros SGBDs e até mesmo encontrar outros modelos que se adaptem bem a aplicação do MongoDB a DW.

Referências

[1] Song, Il Yeol: *Data Warehousing Systems: Foundations and Architectures*, páginas 684–692. Springer US, 2009. 1

[2] Vaisman, Alejandro e Esteban Zimányi: *Data Warehouse Systems: Design and Implementation*. Springer, 2016. 1, 8, 9, 10

[3] Golfarelli, Matteo: *Data Warehouse Life-Cycle and Design*, páginas 658–664. Springer US, 2009. 2

[4] Kimball, Ralph, Margy Ross, Warren Thornthwaite, Joy Mundy e Bob Becker: *The Data Warehouse Lifecycle Toolkit*. Wiley, 2nd edição, 2008. 2, 4, 11, 12, 14, 15, 38, 39, 42, 43

[5] Krishnan, K.: *Data Warehousing in the Age of Big Data*. Elsevier, 1st edição, 2013. 2, 7, 20

[6] Chevalier, Max, Mohammed El Malki, Arlind Kopliku, Olivier Teste e Ronan Tournier: *Document-oriented data warehouses: Models and extended cuboids*. Proceedings - International Conference on Research Challenges in Information Science, 2016-August, 2016. 3, 28, 30, 31, 36, 42, 43, 47

[7] Dehdouh, Khaled, Fadila Bentayeb, Omar Boussaid e Nadia Kabachi: *Using the column oriented NoSQL model for implementing big data warehouses*. International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'15, páginas 469–475, 2015. 3, 17, 29, 42

[8] Dehdouh, Khaled: *Building olap cubes from columnar nosql data warehouses*. Model and Data Engineering, páginas 166–179, 2016. 3, 17, 34, 42

[9] Dehdouh, Khaled, Omar Boussaid e Fadila Bentayeb: *Big Data warehouse: Building columnar NoSQL OLAP cubes*. International Journal of Decision Support System Technology, 12(1):1–24, 2020. 3, 17, 33, 42

[10] Davardoost, Farnaz, Amin Babazadeh Sangar e Kambiz Majidzadeh: *Extracting OLAP Cubes from Document-Oriented NoSQL Database Based on Parallel Similarity Algorithms*. Canadian Journal of Electrical and Computer Engineering, 43(2):111–118, 2020. 3, 4, 31

[11] Petricioli, Lucija, Luka Humski e Boris Vrdoljak: *The Challenges of NoSQL Data Warehousing*. E-business technologies conference proceedings, 1(1):44–48, 2021. 3, 4, 42

[12] Scabora, Lucas C., Jaqueline J. Brito, Ricardo Rodrigues Ciferri e Cristina Dutra De Aguiar Ciferri: *Physical data warehouse design on NoSQL databases: OLAP query processing over HBase.* ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems, 1:111–118, 2016. 4, 29, 30, 33, 36, 42, 43, 47

[13] Chevalier, Max, Mohammed El Malki, Arlind Kopliku, Olivier Teste e Ronan Tournier: *Document-oriented models for data warehouses: NoSQL document-oriented for Data warehouses.* ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems, 1:142–149, 2016. 4, 30, 31, 36, 42, 43

[14] Sellami, Amal, Ahlem Nabli e Faiez Gargouri: *Transformation of Data Warehouse Schema to NoSQL Graph Data Base.* Advances in Intelligent Systems and Computing, 941:410–420, 2020. 4, 35

[15] Laudon, K.C. e J.P. Laudon: *Sistemas de Informação Gerenciais.* Pearson, 11ª edição, 2014. 6, 7

[16] Inmon, William H.: *Building the Data Warehouse.* Wiley, 4th edição, 2005. 8, 10

[17] Kimball, Ralph e Margy Ross: *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling.* John Wiley amp; Sons, Inc., 3rd edição, 2013. 8, 9, 10, 14

[18] P., GRAY e WATSON H.J: *Decision Support in the Data Warehouse.* Pearson, 1st edição, 1998. 8

[19] Sen, Arun e Varghese S. Jacob: *Industrial-strength data warehousing.* Commun. ACM, 41(9):28–31, 1998. 8

[20] Poe, Vidette, Patricia Klauer e Stephen Brobst: *Building A Data Warehouse for Decision Support.* Pearson, 2nd edição, 1998. 8

[21] Rainardi, Vincent: *Building A Data Warehouse With Examples In Sql Server.* Apress, 2008. 8, 10

[22] Sadalage, Pramod J. e Martin Fowler: *NoSQL essencial: um guia conciso para o mundo emergente da persistência poliglota.* Novatec Editora, 2013. 15, 16, 17

[23] Corbellini, Alejandro, Cristian Mateos, Alejandro Zunino, Daniela Godoy e Silvia Schiaffino: *Persisting big-data: The nosql landscape.* Information Systems, 63:1–23, janeiro 2017. 15

[24] Davoudian, Ali, Liu Chen e Mengchi Liu: *A survey on nosql stores.* ACM Comput. Surv., 51(2), apr 2018. 16

[25] Diogo, Miguel, Bruno Cabral e Jorge Bernardino: *Consistency Models of NoSQL Databases.* Future Internet, 11(2), 2019. 16

[26] Moniruzzaman, A B M e Syed Akhter Hossain: *Nosql database: New era of databases for big data analytics-classification, characteristics and comparison*, 2013. 16

[27] Khalil, Abdelhak e Mustapha Belaissaoui: *Key-value data warehouse: Models and OLAP analysis*. 2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science, ICECOCS 2020, 2020. 16, 35

[28] Chevalier, Max, Mohammed El Malki, Arlind Kopliku, Olivier Teste e Ronan Tournier: *How Can We Implement a Multidimensional Data Warehouse Using NoSQL ?* 2015 International Conference on Enterprise Information Systems, ICEIS 2015, 2:108–130, 2015. 17, 28, 30, 42, 47

[29] Chevalier, Max, Mohammed El Malki, Arlind Kopliku, Olivier Teste e Ronan Tournier: *Implementing multidimensional data warehouses into NoSQL*. ICEIS 2015 - 17th International Conference on Enterprise Information Systems, Proceedings, 1:172–183, 2015. 17, 28, 42

[30] *Db-engines*. `https://db-engines.com/en/ranking`. Accessed: 2022-08-29. 17

[31] Chodorow, Kristina: *MongoDB: The Definitive Guide*. O'Reilly, 2013. 18

[32] George, Lars: *Hbase: The definitive guide*. O'Reilly, 2011. 19, 20

[33] De Mauro, Andrea, Marco Greco e Michele Grimaldi: *What is big data? a consensual definition and a review of key research topics*. páginas 97–104, 2014. 20

[34] Zikopoulos, P. e C. Eaton: *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 1st edição, 2011. 20

[35] Khan, M. A., M. F. Uddin e N. Gupta: *Seven v's of big data: Understanding big data to extract value*. American Society for Engineering Education (ASEE Zone 1), página 1–5, 2014. 20

[36] Khan, N., I. Yaqoob, I. A. T. Hashem, W. K. Inayat, Z.and Mahmoud Ali, M. Alam e A. Gani: *Big data: Survey, technologies, opportunities, and challenges*. The Scientific World Journal, 2014. 20

[37] Ranjan, J.: *The 10 vs of big data framework in the context of 5 industry verticals*. Productivity, 2019. 21

[38] Katal, Avita, Mohammad Wazid e R. H. Goudar: *Big data: Issues, challenges, tools and good practices*. Em *2013 Sixth International Conference on Contemporary Computing (IC3)*, páginas 404–409, 2013. 21

[39] Schroeder, Ralph: *Big data business models: Challenges and opportunities*. Cogent Social Sciences, 2(1):11–15, 2016. 21

[40] Al-Mekhlal, Monerah e Amir Ali Khwaja: *A synthesis of big data definition and characteristics*. Em *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, páginas 314–322, 2019. 21

[41] White, Tom: *Hadoop: The Definitive Guide*. O'Reilly, 2012. 21, 22

[42] Dean, Jeffrey e Sanjay Ghemawat: *Mapreduce: A flexible data processing tool*. Communications of the ACM, 53(1):72–77, 2010. 21

[43] Costa, Eduarda A. P. da: *Organização e processamento de dados em big data warehouses baseados em hive*. Tese de Mestrado, Universidade do Minho, 2017. 22, 23

[44] Cassavia, Nunziato, Pietro Dicosta, Elio Masciari e Domenico Saccà: *Data preparation for tourist data big data warehousing*. Em *Proceedings of 3rd International Conference on Data Management Technologies and Applications*, página 419–426. SCITEPRESS - Science and Technology Publications, Lda, 2014. 22

[45] Sandoval, L. J.: *Design of business intelligence applications using big data technology*. Em *2015 IEEE Thirty Fifth Central American and Panama Convention (CONCAPAN XXXV)*, páginas 1–6, 2015. 22

[46] Birjali, Marouane, Abderrahim Beni-Hssane e Mohammed Erritali: *Evaluation of high-level query languages based on MapReduce in Big Data*. Journal of Big Data, 5, 2018. 23

[47] Oliveira, Beatriz Fragnan P. de, Marcio de Carvalho Victorino e Maristela Holanda: *Data warehouse based on nosql: a literature mapping*. Em *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–6, 2021. 24

[48] Kitchenham, Barbara, O Pearl Brereton, David Budgen, Mark Turner, John Bailey e Stephen Linkman: *Systematic literature reviews in software engineering – A systematic literature review*. Information and Software Technology, 51(1):7–15, 2009. 24

[49] Oditis, Ivo, Zane Bicevska, Janis Bicevskis e Girts Karnitis: *Implementation of NoSQL-based Data Warehouses*. Baltic Journal of Modern Computing, 6(1):45–55, 2018. 27, 31, 42

[50] Murazza, Muh Rafif e Arif Nurwidyantoro: *Cassandra and SQL database comparison for near real-time Twitter data warehouse*. Proceeding - 2016 International Seminar on Intelligent Technology and Its Application, ISITIA 2016: Recent Trends in Intelligent Computational Technologies for Sustainable Energy, páginas 195–200, 2016. 30, 34

[51] Khabibullina, Elena e Anatoly Pogodaev: *Data Warehouse Model of Regional Intelligent Transportation System in the Concept of ITS-Russia*. Proceedings - 2019 1st International Conference on Control Systems, Mathematical Modelling, Automation and Energy Efficiency, SUMMA 2019, páginas 599–603, 2019. 30, 35, 42

[52] Dehdouh, Khaled: *Building olap cubes from columnar nosql data warehouses*. Emerging Perspectives in Big Data Warehousing, páginas 129–157, 2019. 30, 42

[53] Costa, Eduarda, Carlos Costa e Maribel Yasmina Santos: *Efficient big data modelling and organization for hadoop hive-based data warehouses*. Lecture Notes in Business Information Processing, 299:3–16, 2017. 30, 35, 36, 42, 43

[54] Yangui, Rania, Ahlem Nabli e Faiez Gargouri: *Automatic Transformation of Data Warehouse Schema to NoSQL Data Base: Comparative Study*. Procedia Computer Science, 96(September):255–264, 2016. 30, 33, 36, 42, 43, 47

[55] Oliveira, Beatriz Fragnan P. de, Aline S. Oliveira Valente, Márcio Victorino, Edward Ribeiro e Maristela Holanda: *Análise da influência da modelagem e formato de dados no desempenho de data warehouse baseado em hadoop-hive*. Em *SBBD*, páginas 271–276. SBC, 2021. 30, 36, 58, 66, 68

[56] Bicevska, Zane, Andrejs Neimanis e Ivo Oditis: *NoSQL-based Data Warehouse Solutions: Sense, Benefits and Prerequisites*. Baltic Journal of Modern Computing, 4(3):597, 2016. 31, 42

[57] Bouaziz, Senda, Ahlem Nabli e Faiez Gargouri: *Design a data warehouse schema from document-oriented database*. Procedia Computer Science, 159:221–230, 2019. 31, 42

[58] Bouaziz, Senda, Ahlem Nabli e Faiez Gargouri: *Nosql big data warehouse: Review and comparison*. Em Abraham, Ajith, Vincenzo Piuri, Niketa Gandhi, Patrick Siarry, Arturas Kaklauskas e Ana Madureira (editores): *Intelligent Systems Design and Applications*, páginas 392–401. Springer International Publishing, 2021. 31

[59] Yangui, Rania, Ahlem Nabli e Faïez Gargouri: *ETL based framework for NoSQL warehousing*. Lecture Notes in Business Information Processing, 299(August):40–53, 2017. 31, 42

[60] Gallinucci, Enrico, Matteo Golfarelli e Stefano Rizzi: *Approximate OLAP of document-oriented databases: A variety-aware approach*. Information Systems, 85:114–130, 2019. 31, 42

[61] Challal, Zakia, Wafaa Bala, Hanifa Mokeddem e Kamel Boukhalfa: *Document-oriented versus Column-oriented Data Storage for Social Graph Data Warehouse*. 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), páginas 242–247, 2019. 32, 42

[62] Gallinucci, Enrico, Matteo Golfarelli e Stefano Rizzi: *Variety-aware OLAP of document-oriented databases*. CEUR Workshop Proceedings, 2062, 2018. 32, 42

[63] Hubail, M.A., A. Alsuliman, M. Blow, M. Careyl, D. Lychagin, I. Maxon e T. Westmannl: *Couchbase analytics: NoETL for scalable NoSQL data analysis*. Em *Proceedings of the VLDB Endowment*, volume 12, páginas 2275–2286, 2018. 32

[64] Chevalier, Max, Mohammed El Malki, Arlind Kopliku, Olivier Teste e Ronan Tournier: *Document-Oriented Data Warehouses: Complex Hierarchies and Summarizability*. Em *Advances in Ubiquitous Networking 2*, páginas 671–683. Springer, 2017. 32, 42

[65] Bicevska, Z. e I. Oditis: *Towards NoSQL-based Data Warehouse Solutions*. Em *Procedia Computer Science*, volume 104, páginas 104–111, 2016. 32, 42

[66] Souibgui, Manel, Faten Atigui, Sadok Ben Yahia e Samira Si-Said Cherfi: *Business Intelligence and Analytics: On-demand ETL over Document Stores*. Em *Research Challenges in Information Science*, páginas 556–561. Springer, 2020. 32

[67] Francia, M., E. Gallinucci, M. Golfarelli e S. Rizzi: *OLAP Querying of Document Stores in the Presence of Schema Variety*. Em *CEUR Workshop Proceedings*, volume 2646, páginas 128–135, 2020. 32, 42

[68] Ptiček, M., B. Vrdoljak e M. Gulić: *The potential of semantic paradigm in warehousing of big data*. Automatika, 60(4):386–396, 2019. 32, 42

[69] Ferro, M., Rogerio Fragoso e Robson Fidalgo: *Document-oriented geospatial data warehouse: An experimental evaluation of SOLAP queries*. Proceedings - 21st IEEE Conference on Business Informatics, CBI 2019, 1:47–56, 2019. 32, 42

[70] Alami, L., I. Hafidi e A. Metrane: *Entity resolution in NoSQL data warehouse*, volume 640. 2018. 33, 42

[71] Aluvalu, Rajanikanth e M. A. Jabbar: *Handling data analytics on unstructured data using MongoDB*. IET Conference Publications, 2018, 2018. 33, 42

[72] El Malki, Mohammed, Arlind Kopliku, Essaid Sabir e Olivier Teste: *Benchmarking big data OLAP NoSQL databases*, volume 11277 LNCS. Springer International Publishing, 2018. 33, 42

[73] Chouder, Mohamed L., Stefano Rizzi e Rachid Chalal: *Enabling self-service BI on document stores*. CEUR Workshop Proceedings, 1810, 2017. 33, 42

[74] Prakash, D.: *NosoLAP: Moving from data warehouse requirements to NoSQL databases*. Em *ENASE 2019 - Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering*, páginas 452–458, 2019. 33

[75] Mallek, Hana e Sakiet Ezzit: *Towards Extract-Transform-Load Operations in a Big Data context*. 12(2):77–95, 2020. 33, 42

[76] Jianmin, Wang, Zhao Wenbin, Fan Tongrang, Yang Shilong e Lv Hongwei: *An improved join-free snowflake schema for ETL and OLAP of data warehouse*. Concurrency Computation, 32(23), 2020. 34, 42

[77] Boumlik, Abdeljalil, Nassima Soussi e Mohamed Bahaj: *SMART-ETL-MR: NOVEL ETL FRAMEWORK for BUILDING DATA WAREHOUSE from BIG DATA SOURCE USING MAPREDUCE*. Journal of Theoretical and Applied Information Technology, 98(17):3449–3460, 2020. 34, 42

[78] Correia, J., M. Y. Santos, C. Costa e C. Andrade: *Fast Online Analytical Processing for Big Data Warehousing*. 2:435–442, 2018. 34

[79] Torres, Letícia, Gustavo Rezende Kröger, Marcio Seiji Oyamada, Clodis Boscarioli e Orbit Sistemas: *Evaluating the impact of data modeling on olap applications using relacional and columnar DBMS*. Proceedings - 2018 44th Latin American Computing Conference, CLEI 2018, páginas 464–471, 2018. 34

[80] Boussahoua, M., O. Boussaid e F. Bentayeb: *Logical schema for data warehouse on column-oriented NoSQL databases*. 2017. 34, 42

[81] Yang, Qishan e Markus Helfert: *Revisiting Arguments for a three layered data warehousing architecture in the context of the hadoop platform*. CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science, 2:329–334, 2016. 34, 42

[82] Pereira, J.L. e M. Costa: *Decision support in big data contexts: A business intelligence solution*, volume 444. 2016. 34, 42

[83] Vaisman, Alejandro, Florencia Besteiro e Maximiliano Valverde: *Modelling and Querying Star and Snowflake Warehouses Using Graph Databases*. Em *New Trends in Databases and Information Systems*, páginas 144–152. Springer International Publishing, 2019. 35

[84] Guminska, Ewa e Teresa Zawadzka: *EvOLAP graph – Evolution and OLAP-aware Graph data model*. Communications in Computer and Information Science, 928:75–89, 2018. 35

[85] Gómez, Leticia, Bart Kuijpers e Alejandro Vaisman: *Performing OLAP over graph data: Query language, implementation, and a case study*. ACM International Conference Proceeding Series, 2017. 35

[86] Akid, Hajer e Mounir Ben Ayed: *Towards NoSQL graph data warehouse for big social data analysis*. Advances in Intelligent Systems and Computing, 557:965–973, 2017. 35

[87] Ngo, Vuong M., Nhien An Le-Khac e M. Tahar Kechadi: *Designing and implementing data warehouse for agricultural big data*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), páginas 1–17, 2019. 35, 42

[88] Elmasri, R. e S. B. Navathe: *Sistemas de Banco de Dados*. Pearson-Addison-Wesley, 7th edição, 2019. 42

[89] Li, Xiang, Zhiyi Ma e Hongjie Chen: *QODM: A query-oriented data modeling approach for NoSQL databases*. Proceedings - 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications, WARTIA 2014, páginas 338–345, 2014. 42, 43