LEANDRO MENDES FERREIRA

STARTABLE - MAPEAMENTO MULTIDIMENSIONAL PARA NoSQL E SISTEMA BIG DATA

SÃO PAULO

2023

LEANDRO MENDES FERREIRA

STARTABLE - MAPEAMENTO MULTIDIMENSIONAL PARA NoSQL E SISTEMA BIG DATA

Versão Corrigida

Dissertação de Mestrado apresentada a Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências.

Área de Concentração:
Engenharia da Computação

Orientador:

Profa. Dra. Solange Nice Alves de Souza

SÃO PAULO

2023

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 01 de Fevereiro de 2024

Autor: Leandro Mendes Ferreira

Orientador(a): Solange Nice Alves de Souza

Catalogação-na-publicação

Ferreira, Leandro Mendes
STARTABLE - Mapeamento Multidimensional para NoSQL e Sistema Big Data / L. M. Ferreira -- São Paulo, 2023. 86 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.NoSQL 2. Modelagem Multidimensional 3. Big Data 4. Sistemas Distribuídos de Dados I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Nome: Ferreira, Leandro Mendes

Título: STARTABLE - Mapeamento Multidimensional para NoSQL e Sistema Big Data

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Aprovado em:

Banca Examinadora

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

RESUMO

Banco de dados analíticas como data warehouse permitem consultas a informações agregadas e sumarizadas, sendo voltadas para as necessidades do negócio e tomadas de decisão. Tais bancos são construídos a partir de modelos multidimensionais, representados por modelos conceituais como o de Entidade-Relacionamento e facilmente mapeado para um modelo de implementação, como o Relacional, com bom desempenho para as operações de consulta. Por outro lado, as atuais necessidades demandam por maior velocidade de processamento com enormes e cada vez maiores volumes de dados. Neste contexto, sistemas distribuídos de persistência de dados como banco de dados NoSQL e ferramentas de *Big Data* são alternativas importantes para desenvolvimento de bases analíticas. Entretanto, ainda é um desafio encontrar uma forma de mapeamento de modelos multidimensionais que se adeque a diferentes tipos de NoSQL e outros sistemas de persistência não relacional e que seja flexível para atender a diferentes tipos ou necessidades de negócio. Desta forma, essa pesquisa propõe a Startable, uma forma de mapeamento para NoSQL orientado a documentos, NoSQL orientado a colunas, e para um sistema de *Big Data*, que apresenta desempenho aceitável de leitura e que pode ser adotado por diferentes aplicações analíticas com diferentes necessidades de negócio. Para validar a proposta, utilizou-se um *benchmark* padrão, cujos testes de desempenho mostraram resultados superiores quando comparados com tradicionais implementações de sistemas analíticos com base em modelo multidimensional.

Palavras-chave: NoSQL, Modelagem Multidimensional, Big Data, Sistemas Distribuídos de Dados

ABSTRACT

Analytical databases such as data warehouses allow queries of aggregated and summarised information focused on business needs and decision-making. Such databases are built from multidimensional models, represented by conceptual models such as Entity-Relationship and easily mapped to an implementation model, such as Relational, with good performance for query operations. On the other hand, current needs demand greater processing speed with huge and ever-increasing volumes of data. In this context, distributed data persistence systems such as NoSQL Databases and Big Data tools are essential alternatives for developing analytical bases. However, it is still a challenge to find a way to map multidimensional models that suit different types of NoSQL and other non-relational persistence systems and are flexible to meet various business needs. Thus, this research proposes Startable, a form of mapping for document-oriented NoSQL, column-oriented NoSQL, and a Big Data system, which presents acceptable reading performance and can be adopted by different analytical applications with varying needs of business. A standard benchmark was used to validate the proposal, whose performance tests showed superior results compared to traditional implementations of analytical systems based on a multidimensional model.

Keywords: NoSQL, Multidimensional Modelling, Big Data, Data Distributed Systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

3FN - Terceira Forma Normal

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

ACM - *Association for Computing Machinery*

AWS - *Amazon Web Services*

BDR - Banco de Dados Relacional

BI - *Business Intelligence*

CQL - *Cassandra Query Language*

DM - *Data Mart*

DW - *Data Warehouse*

ELT - *Extract, Load and Transform*

ETL - *Extract, Transform and Load*

FD - Fontes de Dados

GCP - *Google Cloud Platform*

GFS - *Google File System*

HDFS - *Hadoop Distributed File System*

IEEE - *Institute of Electrical and Electronic Engineers*

JCR - *Journal Citation Report*

JSON - *JavaScript Option Notation*

MOLAP - *Multidimensional On Line Analytical Processing*

NoSQL - *Not Only SQL*

ODS - *Operational Data Store*

OLAP - *On Line Analytical Processing*

OLTP - *On Line Transactions Processing*

RSL - Revisão Sistemática de Literatura

SQL - *Structured Query Language*

SSB - *Star Schema Benchmark*

TPC-H - *Transaction Processing Performance Council Benchmark™ H*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Por diversos anos, os bancos de dados relacionais (BDRs) têm sido utilizados como parte principal na persistência de dados para sistemas analíticos e de tomada de decisão. Neste sentido o conceito de "*Data Warehouse*" (DW), tornou-se um dos principais componentes destes sistemas analíticos (Yessad e Labiod, 2016).

Sendo proposto no final dos anos 1980, DW possui como principal característica uma base de dados relacional, na qual os dados são dispostos em uma estrutura específica e especializada para aplicação analítica (Inmon e Hackathorn, 1994). Se destacam como estrutura para DW os modelos multidimensionais estrela ("*Star Schema*") e floco de neve ("*Snow Flake*"), sendo este último uma variação do "*Star Schema".*

As estruturas principais do modelo multidimensional são as dimensões e os fatos. As dimensões são tabelas que contém informações de segmentação, dimensionalidade e análise granular. Fatos são tabelas que armazenam dados quantitativos associados a cada tabela dimensão. O modelo estrela é formado por uma tabela fato e diversas tabelas dimensões (Kimball e Ross, 2002).

De acordo com Bilal *et al.* (2016), por muitos anos os BDRs foram as principais fontes de persistência de dados para sistemas analíticos. Entretanto, com o aumento do volume, complexidade dos formatos e a necessidade de aumento da velocidade do processamento de dados dos últimos anos, outras alternativas de persistência tornaram-se necessárias, além da necessidade de novas estruturas ou adaptações para implementação do modelo multidimensional em tais sistemas de persistência.

Nesse contexto, tem-se as tecnologias de *Big Data* e Bancos de Dados NoSQL como novas formas de persistência de dados aptas a suprir demandas de persistência e processamento até então não atendidas pelos BDRs (Madden, 2012; Prando *et al.*, 2017; Romero *et al.*, 2015; Sh *et al.*, 2015; Thusoo, Shao, *et al.*, 2010; Wang *et al.*, 2014).

O ecossistema *Hadoop* é um dos pioneiros no âmbito das tecnologias de *Big Data*. Suas principais tecnologias abrangem um sistema de distribuição de arquivos conhecidos como *Hadoop Distributed File System* (HDFS), um sistema para processamento, por exemplo *MapReduce*, além de outras ferramentas para distribuição e gestão dos dados.

Os bancos de dados NoSQL em geral são classificados em quatro grupos de tecnologia, sendo elas: NoSQL orientado a colunas, NoSQL orientados a documentos, NoSQL orientados a grafos e NoSQL orientados a chave-valor (Moniruzzaman e Hossain, 2013).

Tanto sistemas de *Big Data* como o *Hadoop* quanto NoSQL têm diferenças significativas em relação aos BDRs. Por exemplo, NoSQL utilizam estruturas de dados não relacionais, não possuem uma linguagem de consulta padronizada, como a *Structured Query Language* (SQL), bem como não priorizam o atendimento às propriedades ACID (acrônimo para Atomicidade, Consistência, Isolamento e Durabilidade), além de permitirem de maneira simplificada a persistência de dados de forma distribuída (Moniruzzaman e Hossain, 2013).

O mapeamento de elementos do Modelo Entidade-Relacionamento e do Modelo Multidimensional para BDR é simples e fácil (Chaudhuri e Dayal, 1997a; Kimball e Ross, 2002; Silberschatz, Sundarshan e Korth, 2016), porém não se aplicam facilmente as novas tecnologias de *Big Data* como *Hadoop* e NoSQL.

1.1 Problema de Pesquisa

O desenvolvimento desta pesquisa tem como foco principal sistemas analíticos para apoio à tomada de decisão implementados sobre banco de dados NoSQL e sistemas de *Big Data* como o *Hadoop*. O problema pode ser expresso na seguinte pergunta: **Qual mapeamento de dados é apropriado para sistemas analíticos sobre NoSQL e sobre sistema *Big Data*?**

1.2 Objetivo

O objetivo deste trabalho é propor uma forma de mapear o modelo multidimensional proposto por Kimball e Ross (2002) e Inmon (2002) para sistemas de banco de dados NoSQL e sistemas de *Big Data* (*Hadoop*).

A proposta deste mapeamento está baseada em três requisitos:

1. Simplicidade, baseada em conceitos conhecidos;
2. Ser aplicável a diferentes estruturas de NoSQL e a tecnologia de *Big Data*;
3. Resulte em maior eficiência para consultas ao banco de dados quando comparado com implementação tradicional sobre banco de dados NoSQL.

Assim, propõe-se a *Startable*, um mapeamento/implementação, com definições e regras, para representar o modelo multidimensional proposto por Kimball e Ross (2002) e Inmon (2002) em modelos físicos de NoSQL tanto orientado a documentos quanto orientado a colunas e tecnologia de *Big Data*.

1.3 Metodologia

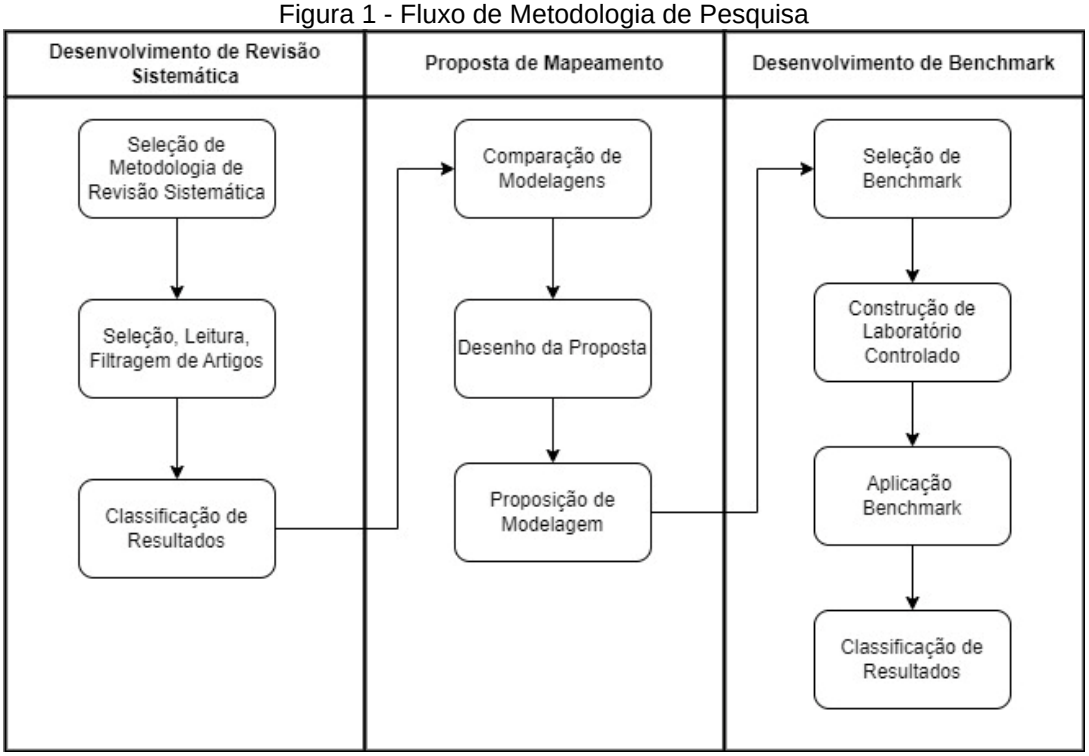
Inicialmente foi realizada uma revisão sistemática de literatura para identificar propostas prévias de mapeamento de modelo multidimensional sobre banco de dados NoSQL. A revisão sistemática é feita com base nas informações extraídas da literatura consultada. Assim, a revisão sistemática discute e analisa tendências, técnicas, modelos e processos levantados a partir dos artigos considerados na revisão.

Para comprovação da eficácia da proposta mapeada foi utilizado um *benchmark*. Para isto, foi selecionado o *Star Schema Benchmark* (SSB) pela simplicidade de aplicação e por ser um *benchmark* específico para modelo multidimensional (O'neil, O'neil e Chen, 2009). Para desenvolvimento e comparação, o *benchmark* foi realizado em ambiente distribuído e controlado, possibilitando a comparação com o desempenho da *Startable* com a

implementação tradicional do Modelo Estrela apresentado por Inmon (2014). Os testes desenvolvidos foram realizados para NoSQL Orientado a Colunas, NoSQL Orientado a Documentos e sistema distribuído de arquivos HDFS (*Hadoop*). Utilizou-se também a ferramenta de "*SQL Engine*" para realizar consultas com linguagem SQL sobre banco de dados NoSQL.

Na Figura 1 apresenta-se o fluxo de como a metodologia foi aplicada para o desenvolvimento desta pesquisa. Vale destacar que duas atividades foram desenvolvidas, uma revisão sistemática e a proposta da *Startable*. Para o desenvolvimento da *Startable* foi fundamental a comparação das propostas encontradas na literatura (primeira atividade). Por fim, para testar o desempenho da proposta foi utilizado um *benchmark*. Acrescenta-se a publicação de dois artigos como parte dos resultados da pesquisa (Ferreira, Alves-Souza e Da Silva, 2023, 2022).

Figura 1 - Fluxo de Metodologia de Pesquisa



Fonte: Elaborado pelo Autor

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução

Neste capítulo são apresentados os principais conceitos vistos neste trabalho, sendo eles, sistemas analíticos, DW, modelagem multidimensional e o modelo estrela com sua principal vertente, Sistemas Distribuídos de Dados na sua vertente de *Big Data*, banco de dados NoSQL e pôr fim a apresentação de conceitos de indexação, agrupamento de dados, particionamento e *bucking*.

2.2 Sistemas Analíticos

Quando tratamos de processamento, armazenamento e análise de dados, temos na literatura uma divisão clássica entre sistemas transacionais e sistemas analíticos. Esses sistemas são também denominados como Sistemas OLTP (*On Line Transactions Processing*) e Sistemas OLAP (*On Line Analitical Processing*).

Os Sistemas OLTP geralmente automatizam tarefas administrativas de processamento de dados que requerem transações únicas ou em séries, como por exemplo, a entrada de transações bancárias, pedidos de compras de uma empresa e outras operações comuns de uma organização. Essas operações trabalham com mais frequência com dados estruturados, com certo grau de repetição e consistem em transações curtas, atômicas e isoladas. Essas transações exigem dados detalhados e constantemente atualizados com operações constantes de leitura e escrita em sistemas de persistência de dados, em geral BDR (Gray *et al.*, 1997). Como alternativas aos relacionais, muitos sistemas OLTP atualmente utilizam banco de dados NoSQL.

Sistemas OLAP permitem análises, consulta de dados históricos e são utilizados para apoiar as tomadas de decisões. Os sistemas analíticos são especialmente úteis para responderem perguntas como: "O que aconteceu com meu negócio?" ou "Como isso aconteceu, quando e por quê? (Buse e Zimmermann, 2012). Com os sistemas OLAP é possível reunir informações mais completas de fontes diversas, permitindo gerar percepções por camadas de

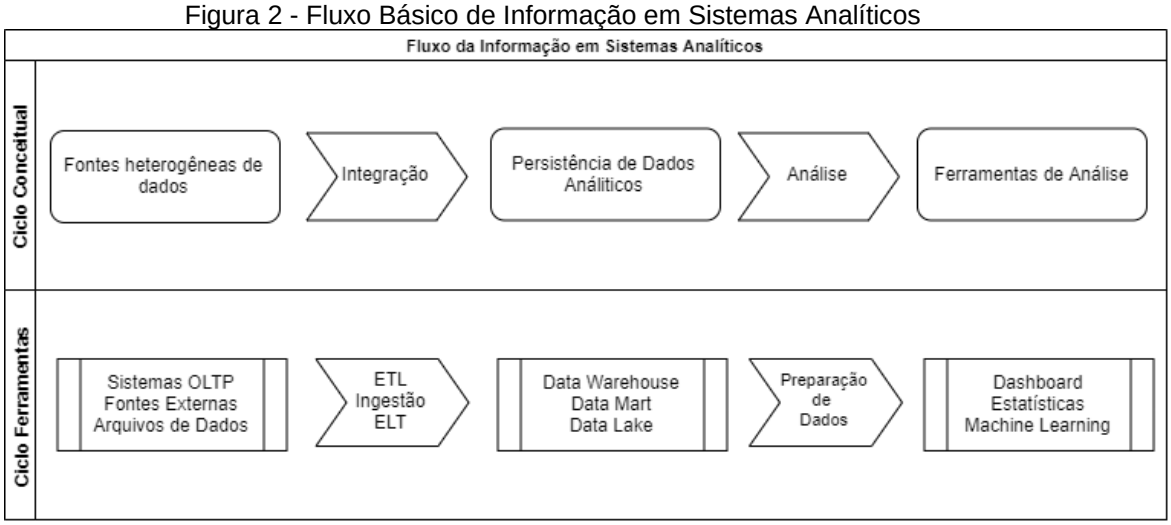
diferentes tipos de análises, sendo possível também resumir, filtrar, modelar e experimentar diversos tipos de visões (Buse e Zimmermann, 2012).

Segundo Chaudhuri e Dayal (1997a), tentar executar consultas OLAP complexas nos bancos de dados operacionais pode resultar em um desempenho inaceitável nesses sistemas. Além disso, para o suporte à decisão, umas das principais funcionalidades dos sistemas analíticos, requer dados que podem estar ausentes nos bancos de dados transacionais. No entanto, as análises complexas como entender tendências ou fazer previsões requerem dados históricos. Porém, bancos de dados operacionais em geral armazenam apenas dados atuais. Além disso, sistemas analíticos geralmente suportam a consolidação de dados de muitas fontes externas e heterogêneas. As diferentes fontes podem conter dados de qualidade variável ou usar representações, códigos e formatos inconsistentes, que precisam ser reconciliados. Por esses motivos que os sistemas OLAP são implementados separadamente dos bancos de dados operacionais e seus sistemas OLTP (Chaudhuri e Dayal, 1997b).

Os sistemas OLAP comumente persistem seus dados em BDRs na forma de DW centralizados e integrados. Esses DW são frequentemente acessados usando ferramentas de mineração de dados, visualização, processamento analítico on-line, análise estatística quantitativa e auxiliam também na criação de modelos estatísticos explicativos e preditivos (Sharma *et al.*, 2010).

O fluxo básico das informações em um sistema analítico é expresso na Figura 2 em duas visões: (1) a conceitual, que é independente das tecnologias utilizadas, e (2) a de implementação, com exemplos de possíveis ferramentas que podem ser utilizadas. Inicialmente tem-se a integração de dados, na qual os dados captados de sistemas transacionais são integrados e persistidos em sistemas analíticos. Posteriormente esses dados são analisados por ferramentas de análise. Na camada de tecnologias, a integração é feita via extração, transformação e carga (*extract, transform and load* - ETL), Ingestão (*extract, load and transform* – ELT) e persistidos em *Data Warehouse*, *Data Mart* ou *Data Lake*. A análise pode ser feita utilizando diferentes tecnologias como ferramentas de *Dashboard*, Estatísticas ou *Machine Learning*.

Figura 2 - Fluxo Básico de Informação em Sistemas Analíticos



Fonte: Elaborado pelo Autor

Dentre os elementos dos sistemas analíticos temos a persistência de dados em DW, onde o modelo multidimensional é adotado para a criação de DW sobre BDRs. Esta pesquisa foca nos sistemas de persistência de dados, pois são nestes em que a modelagem de dados é necessária.

2.3 *Data Warehouse* e Modelagem Multidimensional

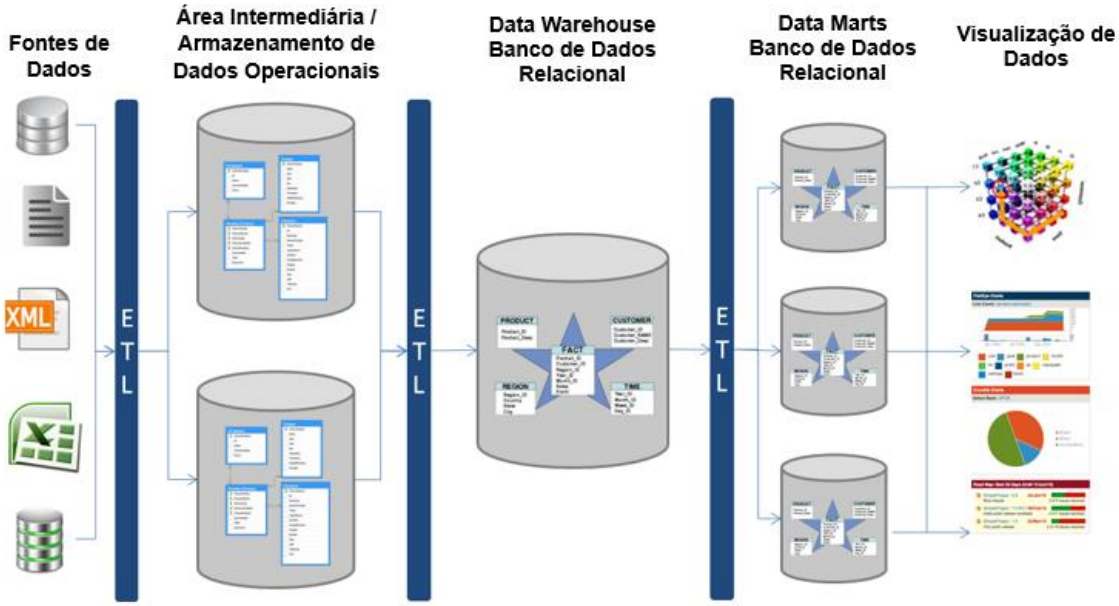
Para Mousa e Shiratuddin (2015) a primeira discussão sobre DW foi publicada em 1988 por Devlin e Murphy (1988). Nesta foram apresentadas definições como o tipo de banco de dados usado e com característica apenas de leitura, operações de integração de dados oriundas de fontes distintas, bem como ferramentas para auxiliar os usuários a interagirem com os dados armazenados (Devlin e Murphy, 1988; Mousa e Shiratuddin, 2015).

Mousa e Shiratuddin (2015) afirmam que mesmo sendo Devlin e Murphy os mais antigos a definiram o conceito de DW, a definição proposta por William Inmon recebeu maior atenção ao longo dos anos. Segundo Inmon e Hackathorn (1994), “DW” é uma coleção de dados orientada a assunto, integrada, não-volátil, e variante no tempo para suporte a decisões gerenciais. Uma proposta muito aceita também para DW é a de Kimball e Ross (2002) em que definem DW como um conjunto de Data Marts (DM) consistentes com base em dimensões compartilhadas. Essas duas abordagens sobre DW foram por muitos anos as principais aceitas pela comunidade científica e empresarial (Yessad e Labiod,

2016). Neste trabalho utiliza-se a abordagem proposta por Inmon e Hackathorn (1994).

Na Figura 3 apresenta-se as principais estruturas e processos que compõem um sistema analítico de tomada de decisão conhecido como *Business Intelligence* (BI) de acordo com Kimball e Ross (2002) e Inmon (2002).

Figura 3 - Itens que compõem um Ambiente de BI



Fonte: Elaborado pelo Autor

As estruturas apresentadas na Figura 3 podem ser descritas da seguinte forma:

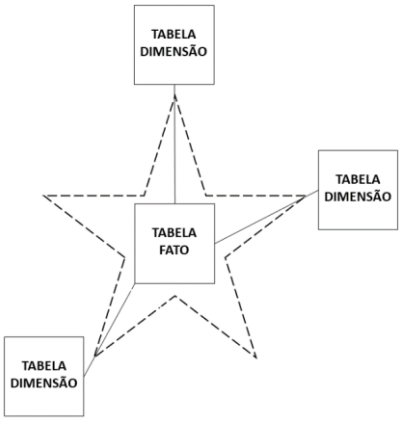
1. **Fontes de Dados (FD - *Data Source*):** Repositórios com dados digitais. podendo envolver arquivos sequenciais, BDR, sistemas transacionais e quaisquer outras fontes de dados que possam ser processadas e incrementadas ao *DW/DM*.
2. **Extração, Transformação e Carga (ETL):** Processo que envolve extrair os dados das FD. Aplicar um processo transformação que envolve limpeza, adequação e agregação dos dados para que estes possam ser carregados no DW.
3. **Área Intermediária (*Staging Area*) \ *Armazenamento de dados Operacionais (Operational Data Store - ODS)*:** Área temporária na qual

são armazenados os dados antes da transformação. A área intermediária geralmente é utilizada para carga dos dados brutos dos FD para posterior transformação, sendo acessada por desenvolvedores e responsáveis de ETL. Já a ODS é utilizada como área de integração de dados pelos usuários e/ou sistemas transacionais.

4. ***Data Warehouse*** **(DW)**: Corresponde ao armazém central de dados. Neste são disponibilizados de forma corporativa todos os dados integrados pelo processo de ETL. São bases de dados históricas não voláteis, ou seja, possuem dados gravados fisicamente que não sofrem alterações e representam a informação em diferentes instantes.
5. ***Data Mart*** **(DMs)**: Pequenos armazéns de dados com uma parte da totalidade de dados armazenados no DW e atendem necessidades específicas de áreas de negócio. É sobre a camada de DM que se empregam as ferramentas de visualização de dados.
6. **Ferramentas de Visualização de Dados:** Utilizadas para visualização dos dados contidos nos DMs. São expoentes as ferramentas OLAP que apresentam os dados de forma multidimensional. Destacam-se também ferramentas de relatórios e as de apresentação em painéis de controle (*DashBoard*), que apresentam as informações em forma de gráficos, indicadores e tabelas. Além das ferramentas para Mineração de Dados e *Machine Learning*.

Kimball e Ross (2013) e Inmon (2002) orientam a modelagem do *DW e os DMs,* como esquema estrela ou modelo estrela (*Star Schema*), apresentado graficamente na Figura 4 a seguir. O modelo estrela baseia sua estrutura em dois tipos de tabelas: (1) é a tabela FATO, o centro do modelo estrela, composta por atributos que podem ser mensurados e/ou calculados e atributos de chaves estrangeiras que ligam às tabelas DIMENSÃO; (2) as tabelas DIMENSÃO que contém as informações para segmentação e análise dos dados. Estas por sua vez, são relacionadas a uma tabela FATO, formando assim um desenho que visualmente se assemelha a uma estrela.

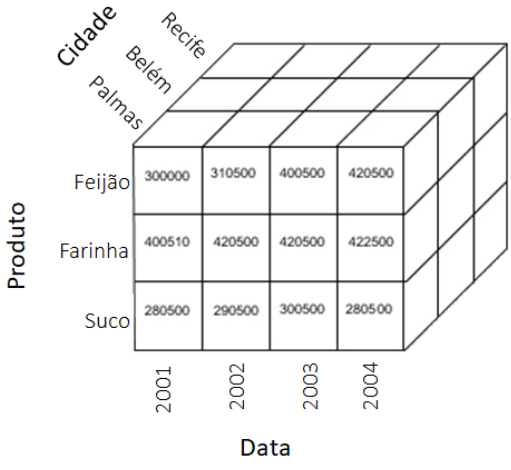
Figura 4 - Representação do Modelo Estrela de Kimball e Ross (2013)



Fonte: Elaborado pelo Autor

O modelo multidimensional é um modelo de dados conceitual que representa um conjunto de medidas numéricas que são os objetos de análise (os fatos), como pode ser visto na Figura 5. Cada uma das medidas numéricas depende do conjunto de dimensões. Presume-se que um conjunto de dimensões determinem uma medida. Desta forma, os dados multidimensionais têm uma medida como um valor único no espaço multidimensional de suas dimensões (Chaudhuri e Dayal, 1997b). As dimensões geralmente têm associação hierárquica. Essas hierarquias especificam níveis de agregação e granularidade dos dados. Uma dimensão comum com níveis de hierarquias é a “Data”. Nesta tem-se, por exemplo, os atributos: Dia, Mês, Trimestre, Ano. Estes atributos representam uma hierarquia data, pois cada um específica vários níveis de agregação (Agrawal, Gupta e Sarawagi, 1997).

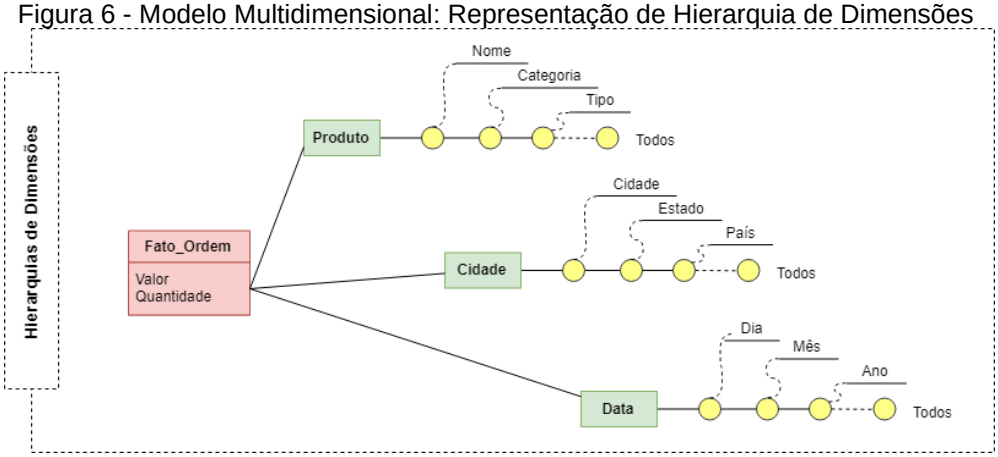
Figura 5 - Modelo Multidimensional



Fonte: Elaborado pelo Autor

A Figura 5 exibe um modelo multidimensional composto por três dimensões (“*Produto*”, “*Cidade*” e “*Data*”) e uma métrica. Consegue-se representar essas dimensões como um cubo. Uma representação gráfica das hierarquias representada pelos atributos de cada uma das dimensões, representados com graus de hierarquia pode ser vista na Figura 6.

Figura 6 - Modelo Multidimensional: Representação de Hierarquia de Dimensões



Fonte: Elaborado pelo Autor

É importante ressaltar que a modelagem multidimensional foi desenvolvida com o intuito de facilitar análises e visualizações complexas. As dimensões e suas hierarquias facilitam operações comuns em sistemas OLAP com as de “*Rollup*” (aumentando o nível de agregação) e “*Drill Down*” (diminuição de nível de agregação ou aumentando o nível de detalhes) que são executadas ao longo de uma ou mais hierarquias de dimensão. Outras operações comuns são “*Slice and Dice*” (seleção e projeção de dados) e “*Pivot*” (reorientação da visão multidimensional de dados). Mais detalhes podem ser vistos em Chaudhuri e Dayal (1997b).

2.4 Sistemas de *Big Data*

Big Data é o termo usado para especificar dados, tecnologia empregada para processamento, armazenamento e análise de dados, como também para aplicações (Gandomi e Haider, 2015). Laney (2001) foi um dos pioneiros na definição do termo. O autor define *Big Data* como “modelo dos 3Vs”, no qual cada “V” é representado por:

- **Volume**: Refere-se ao grande volume de dados.
- **Velocidade**: Refere-se à velocidade de geração de dados e à velocidade de processamento e disponibilização para uso destes dados.
- **Variedade**: Refere-se aos diferentes formatos de dados. Comumente representado por dados estruturados, semiestruturados e não estruturados.

Desta forma, todo sistema capaz de lidar com essas três características pode ser considerado um sistema de *Big Data* (Laney, 2001), que comumente envolve sistemas distribuídos e computação em cluster. Esses sistemas de *Big Data* conseguem tratar questões complexas de sistemas distribuídos como: processo de leitura e escrita distribuída, combinação distribuída de dados, heterogeneidade de *hardware*, tolerância a falhas e coordenação de tarefas distribuídas (Katal, Wazid e Goudar, 2013).

Google, LinkedIn, Facebook, Amazon foram as pioneiras em empregar setores de pesquisas e desenvolvimento para criarem sistemas distribuídos capazes de lidar com *Big Data*. Assim, os primeiros artigos relacionados a *Big Data* estão relacionados à Google, como os referente ao *Google File System* (GFS), sistemas distribuídos de gerenciamento de arquivos (Ghemawat, Gobioff e Leung, 2003); *MapReduce*, técnica de processamento distribuída desenvolvida pelo Google (Dean e Ghemawat, 2008) e também sobre o *Big Table*, banco de dados distribuído e NoSQL (Chang *et al.*, 2008).

Em 2005, Doog Cutting inspirado nos artigos do Google, criou uma versão do *MapReduce* para o projeto *Nutch*, sua versão rodou em um *cluster* distribuído com 20 nós de processamento e armazenamento. Este projeto foi apoiado pela empresa *Yahoo Inc*. com a contribuição de Tom White, que deu início ao projeto *Hadoop*. Posteriormente, o *Hadoop* se tornou um projeto de código aberto, mantido pela Fundação *Apache* (White, 2012). Então, o *Hadoop* se popularizou e é um dos principais destaque das tecnologias propulsoras do *Big Data*.

A arquitetura básica do *Hadoop* é baseada em dois componentes principais: o HDFS e o *MapReduce*. O HDFS é um sistema de arquivos distribuído baseado no GFS do Google e foi projetado para executar sobre os sistemas de arquivos locais dos nós em cluster. Outra função é o armazenamento de arquivos de grandes volumes. Além disso, o HDFS é um sistema tolerante a falhas e pode escalar de um único servidor para milhares de nós em cluster, cada nó oferece computação e armazenamento local. O *MapReduce* é um sistema de processamento simplificado, desenhado para processar grandes conjuntos de dados. De acordo com Hashem *et al.* (2015), o recurso HDFS e o *MapReduce* estão intimamente relacionados no *Hadoop*, no qual o sistema de armazenamento é fisicamente acoplado ao sistema de processamento.

Atualmente o *Hadoop* é visto como um ecossistema, composto por várias ferramentas. Desta forma, diversas tecnologias e programas são relacionados ao *Hadoop*, como: *Hive*, *Hbase*, *Mahout*, *Pig*, *Zookeeper*, *Spark*, *Avro,* entre outros (Hashem *et al.*, 2015). Atualmente outras tecnologias como os Banco de Dados NoSQL também estão relacionadas a *Big Data*.

Vale destacar que existe uma grande intersecção entre *Cloud Computing* e o *Big Data*. Hashem *et al*. (2015) afirma que “a computação em nuvem é uma tecnologia poderosa para executar computação em escala massiva e complexa”. Chen *et al*. (2014) ressaltam que *Cloud Computing* e de *Big Data* “estão combinados” e que existe uma “pressão para que as organizações adotem e implementem rapidamente tecnologias, como a computação em nuvem, para enfrentar o desafio das demandas [...] *Big Data*”. Desta forma, diversos provedores de *Cloud Computing* oferecem nativamente serviços de *Big Data*, desde serviços baseados em recursos e tecnologias próprias, até serviços baseados no ecossistema *Hadoop*. Dentre esses provedores pode-se destacar a *Amazon Web Services* (AWS), a *Google Cloud Platform* (GCP) e a *Azure Microsoft* (Chen *et al.*, 2014; Hashem *et al*., 2015).

2.5 Banco de Dados NoSQL

NoSQL (“*Not only SQL”)* é um termo que agrupa banco de dados não relacionais, que atendem a grandes volumes de dados com alto desempenho e alta disponibilidade. NoSQL apresentam as seguintes características:

- **Ausência da linguagem padrão de consulta SQL-ANSI:** Em geral disponibiliza interfaces de programação de aplicações (APIs) para consulta através de comandos ou uma linguagem própria de consulta como o caso do *Cassandra Query Language* (CQL) ou *MongoQuery* do *MongoDB*.
- **Escalabilidade horizontal**: Capacidade de processamento em diversos nós de um *cluster* ou diversos *clusters*. Permitem que a carga de processamento seja dividida em diversos agentes, aumentando significativamente a capacidade de processamento do banco de dados. É possível incluir no mesmo *cluster*, *hardwares* heterogêneos, ou *hardwares* de arquiteturas e configurações distintas, possibilitando a diminuição de custo da estrutura física necessária para sua implementação.
- ***Schemaless ou Schema Free***: Esquema de dados flexíveis ou totalmente sem esquemas, favorecendo o trabalho com diversos tipos de dados e aplicações de alta mutabilidade de formatos.
- ***Sharding* e Replicação**: Replicação de dados em vários nós do banco de dados e compartilhamento (*sharding*) de dados. Estas características são nativas de banco de dados NoSQL, favorecendo a disponibilidade do sistema.
- **Consistência Eventual**: Paradigma otimista de consistência de dados, ou seja, com grandes volumes de dados, inconsistências eventuais e temporárias devem ser aceitas e previstas para priorizar a disponibilidade do sistema. A consistência nesse paradigma é efetuada em um momento posterior (Gilbert e Lynch, 2002).

Segundo Lóscio, Oliveira e Pontes (2011) banco de dados NoSQL são relevantes para aplicações que “necessitam de uma tecnologia que ofereça suporte ao gerenciamento e escalabilidade de grandes volumes de dados, de

maneira simples e eficiente”. Por exemplo, aplicações web, em geral, produzem volumes de dados imensos, empresas focadas em aplicações web como o Google, Facebook, Twitter, Amazon, LinkedIn desenvolverem e/ou adotarem tecnologias de banco de dados NoSQL.

Banco de Dados NoSQL são classificados em quatro tipos básicos de modelo de dados: Chave-Valor, Orientado a Documentos, Orientado a Coluna e Orientado a Grafos, o qual pode possuir uma ou mais características desses modelos (Moniruzzaman e Hossain, 2013).

Chave-Valor: São normalmente orientados a uma chave alfanumérica e valores associados, sendo esses valores simples como texto ou complexos como: listas, tabelas (referenciadas com *hash tables*), imagens entre outros. Exemplos desses bancos de dados NoSQL orientados a Chave-Valor são o *Dynamo da Amazon*, *Voldemort do LinkedIn*, *o Riak e o Redis*. Na Figura 7 apresenta-se uma representação gráfica da estrutura Chave-Valor.

Figura 7 - Representação Gráfica do Modelo de Dados NoSQL Orientado a Chave-valor

Table: PersonalData	
Key: 010101	Values: {Name: Bob, Age: 22, Gender: Male}
Key: 010102	Values: {Name: Alex, Age: 55, Gender: Male}
Key: 010103	Values: {Name: Mary, Age: 18, Gender: Female}

Fonte: Elaborado pelo Autor

Orientados a Documentos: São banco de dados construídos para armazenar documentos. Esses documentos são representados internamente de uma forma padrão de troca de documentos como *eXtensible Markup Language* (XML) e *Javascript Option Notation* (JSON), como apresentado na Figura 8. Exemplos desse NoSQL orientados a documentos: *MongoDB, Apache CouchDB e o RavenDB*.

Orientado a Colunas: São orientados por múltiplos atributos para cada chave, portanto os atributos são representados por colunas, como pode ser visto na Figura 9. Cada coluna da tabela é gravada separadamente e armazenada de

forma contínua para cada valor de chave. Exemplos de NoQL orientado a colunas: *Apache Cassandra*, *Big Table da Google e Apache HBase*.

Figura 8 - Representação Gráfica de Modelo de Dados NoSQL Orientado a Documentos


```
{
   "Document":"PersonalData",
   "Fields":[
      {
         "ID":"010101",
         "Name":"Bob",
         "Age":22,
         "Gender":"Male"
      },
      {
         "ID":"010102",
         "Name":"Alex",
         "Age":55,
         "Gender":"Male"
      },
      {
         "ID":"010102",
         "Name":"Mary",
         "Age":22,
         "Gender":"Female"
      }
   ]
}
```


Fonte: Elaborado pelo Autor

Figura 9 - Representação Gráfica de Modelo de Dados NoSQL Orientado a Colunas

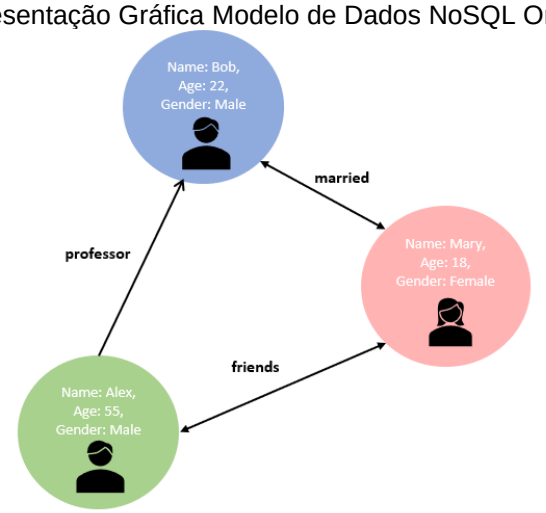
Table: PersonalData		
ID	**Identification**	**Dates**
10101	**Name:** Bob **Midle Name:** Smith **Gender:** Male	**Born:** 1986-01-01
10102	**Name:** Bob **Gender:** Male	**Born:** 1965-10-28
10101	**Name:** Julia **Midle Name:** Silva	**Born:** 1990-06-09

Fonte: Elaborado pelo Autor

Orientado a Grafos: São desenvolvidos sobre os conceitos da teoria dos grafos, na qual as “tabelas” são representadas como uma rede orientada a objetos de nós (objetos conceituais), relações entre nós (arestas) e propriedades (atributos de objetos expressos como pares de valores-chave), como pode ser visto na Figura 10. Geralmente são utilizadas em aplicações em que existe alta complexidade de relacionamentos, como redes sociais e sistemas de

recomendações. Exemplos de banco de dados NoSQL orientados a grafos: *Titan, Apache Giraph* e *Neo4j*.

Figura 10 - Representação Gráfica Modelo de Dados NoSQL Orientado a Grafos



Fonte: Elaborado pelo Autor

2.6 Indexação, Agrupamento de Dados, Particionamento e *Bucketing*

A indexação é uma técnica fundamental para melhorar o desempenho de consultas em bancos de dados. Ela cria estruturas de dados especiais, chamadas de índices, que permitem que o banco de dados encontre registros específicos mais rapidamente. Os índices funcionam como um "mapa" do banco de dados, direcionando as consultas para os locais exatos onde os dados estão armazenados. Isso evita que o banco de dados precise realizar uma busca completa em todo o conjunto de dados, o que pode ser muito lento para grandes conjuntos de dados (Lightstone, Teorey e Nadeau, 2007).

Em contextos específicos, como bancos de dados de texto, técnicas de indexação avançadas são necessárias para melhorar o desempenho das consultas. Por exemplo, a indexação de texto cria índices que mapeiam palavras ou frases para suas localizações em documentos ou registros. Esses índices são frequentemente construídos usando estruturas como árvores invertidas ou tabelas *hash*, permitindo consultas de texto eficientes, mesmo em grandes conjuntos de dados (De Moura, 2009).

Além dessas técnicas de indexação, é essencial considerar a relação entre particionamento e agrupamento, como técnicas especiais para otimização de banco de dados. Samet (2004) destacou a importância de desacoplar esses dois conceitos, argumentando que tal abordagem pode superar certas limitações associadas à indexação espacial, especialmente quando combinada com técnicas de "*bucketing"*.

No entanto, o agrupamento de dados é uma técnica essencial em sistemas de banco de dados, especialmente em contextos de *Big Data* e NoSQL. Essa técnica envolve a organização de dados em partições (*partitions*) ou baldes (*buckets*), de modo que os dados dentro de cada grupo sejam mais semelhantes entre si do que com dados de outros grupos.

Também, vale mencionar que o particionamento envolve a divisão de um conjunto de dados em partes menores, chamadas partições, que podem ser gerenciadas e acessadas independentemente. Isso permite que as operações de leitura e gravação sejam distribuídas de maneira mais eficiente, especialmente em ambientes distribuídos. Em contextos de alta dimensionalidade, técnicas como a superposição de esquemas de particionamento espacial têm sido propostas para melhorar a eficiência da indexação (Lukaszuk e Orlandic, 2004).

Em sistemas de *Big Data* e bancos de dados NoSQL o particionamento é utilizado como uma técnica para gerenciar e acessar eficientemente grandes volumes de dados distribuídos. Em um ambiente de *Big Data*, onde os volumes de dados excedem a capacidade de um único servidor ou sistema, o particionamento divide o conjunto de dados em segmentos menores, chamados partições, que são distribuídos entre vários nós ou servidores (Sakr, 2019). Esta abordagem permite que os sistemas de *Big Data* alcancem escalabilidade horizontal, adicionando mais nós ao sistema conforme a necessidade, sem comprometer o desempenho.

Os sistemas de bancos de dados NoSQL, como Cassandra, *HBase* e *MongoDB*, empregam técnicas de particionamento para distribuir os dados entre

vários servidores e garantir alta disponibilidade e tolerância a falhas. Por exemplo, o Cassandra utiliza uma chave de partição para determinar a distribuição dos dados entre os nós, garantindo que todos os dados associados a uma chave específica residam no mesmo nó, otimizando assim as consultas (Shamsi e Khojaye, 2021). Além disso, o particionamento em sistemas NoSQL também suporta a replicação, onde cada partição pode ter várias réplicas em diferentes nós, garantindo a recuperação de dados em caso de falhas de nó.

É importante notar que a estratégia de particionamento escolhida pode ter um impacto significativo no desempenho do sistema. Uma escolha inadequada pode levar a "*nós sobrecarregados*", onde um nó pode ter uma carga desproporcionalmente alta em comparação com outros nós. Portanto, a seleção e implementação de uma estratégia de particionamento eficaz é necessária para otimizar o desempenho e garantir uma distribuição equilibrada da carga entre os nós (Sakr, 2018).

Salienta-se que o *bucketing* é uma forma especializada de agrupamento de dados e é frequentemente utilizado em bancos de dados NoSQL e sistemas de *Big Data*, como Hadoop e Hive. Esta técnica divide os dados em baldes (*buckets*) com base em uma função *hash* ou algum critério de particionamento, aplicando também a ordenação dos dados nesses *buckets*, facilitando assim a gestão e a recuperação eficiente dos dados. O *bucketing* é particularmente eficaz para otimizar consultas, pois reduz o espaço de busca, concentrando a procura de dados dentro de *buckets* específicos ao invés de vasculhar todo o conjunto de dados (Thusoo, Shao, *et al.*, 2010). Além disso, agrupar e ordenar dados que são frequentemente acessados juntos em um mesmo *bucket*, pode-se reduzir significativamente o custo computacional de operações de junção (Thusoo, Sarma, *et al.*, 2010).

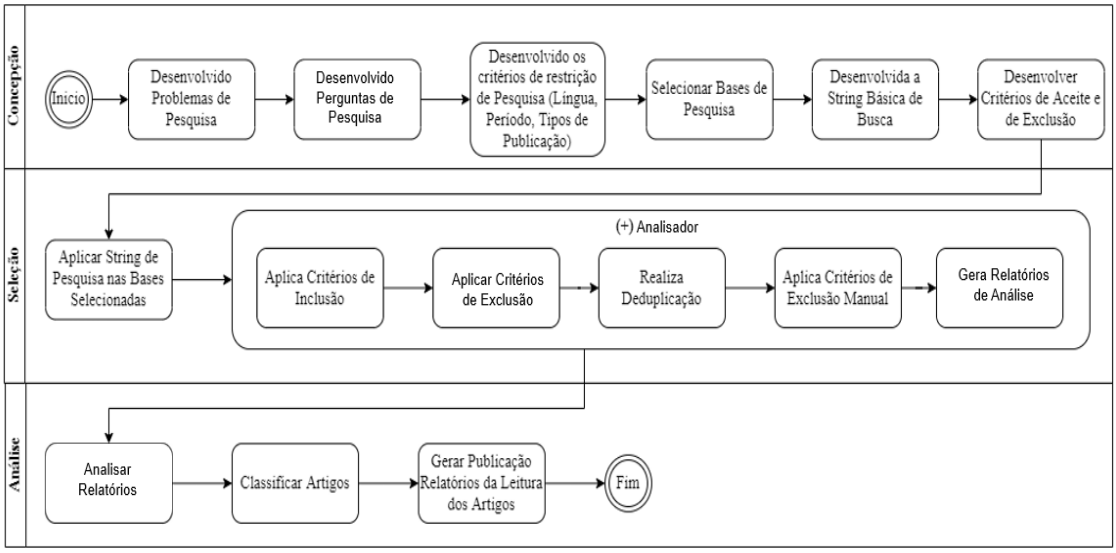
A aplicação do particionamento e técnicas de “*bucketing*” em sistemas de *Big Data* como *Hadoop* e *Hive* bem como em banco de dados NoSQL é detalhada na sessão 4.

3 REVISÃO SISTEMATICA DE LITERATUA

A revisão sistemática de literatura (RSL) é uma forma de identificar um conjunto de pesquisas relevantes de uma determinada área de pesquisa. Segundo Kitchenham e Charters (2007), o diferencial da RSL é que segue uma metodologia rigorosa que pode ser replicada.

Kitchenham e Charters (2007) definem duas principais etapas na construção de uma RSL: planejamento (*planning)* e condução (*conducting)*. Na primeira definem-se as questões de pesquisas, o protocolo da revisão sistemática, que é composto pelos critérios de aceitação e rejeição dos artigos levantados, a seleção das bases de dados de pesquisa e das palavras chaves utilizadas para buscas dos artigos. Nesta fase são definidos também os critérios de avaliação dos artigos. Na fase de condução faz-se a seleção, extração de dados e avaliação dos artigos. Para a RSL adaptou-se a proposta de Kitchenham e Charters (2007), como pode ser visto na Figura 11.

Figura 11 - Fluxo da Revisão Sistemática de Literatura



Fonte: Elaborado pelo Autor

Conforme apresentado na Figura 11, uma vez definidas as questões de pesquisa, procedeu-se aos demais pontos detalhados a seguir:

- **Desenvolvimento dos critérios de Pesquisa:**

- **Idioma em que os artigos devem estar escritos:** somente artigos escritos em inglês foram selecionados.
- **Período para a seleção dos artigos**: para ser obter as publicações recentes, definiu-se o período de 10 anos para a seleção dos artigos. Assim, a busca incluiu o intervalo de janeiro de 2012 até 2022.
- **Tipo de publicação**: inicialmente pesquisou-se apenas artigos publicados em revistas com fator de impacto igual ou maior que 1 (fator de impacto escolhido: *Journal Citation Report* - JCR). Entretanto, o número de artigos obtidos foi pequeno e incluiu-se também artigos de congressos científicos.

- **Seleção das bases de pesquisa**: devido a já reconhecida qualidade para a área de computação foram selecionadas *Association for Computing Machinery* (ACM) *Digital Library*, *Institute of Electrical and Electronic Engineers* (IEEE) *Explorer*, *Scopus* (*Elsevier*), *Springer Link* e *Web of Knowledge* (*Web of Science*).
- ***String* de busca**: uma palavra de busca (*tag)* foi definida de forma que fosse possível ser executada em qualquer uma das bases citadas, sendo as *tag*: ***"'data warehouse' OR 'olap' OR 'star schema' OR 'multidimensional' AND 'NoSQL'"***.

Os resultados foram armazenados em arquivos no formato *bibtex*. Foram criados vários módulos de programas na linguagem *Python* para auxiliar na manipulação dos arquivos *bibtex*. Esses programas também auxiliaram na seleção dos artigos.

Na sequência foi realizada uma segunda filtragem usando o "*abstract*" e "*keywords*" dos arquivos bibtex, aplicando a "*string*" exclusivamente a esses campos. Os artigos pré-selecionados passaram pela etapa de seleção manual. Nesta, fez-se a leitura do título e do resumo dos artigos pré-selecionados. Os critérios de exclusão previamente definidos, foram:

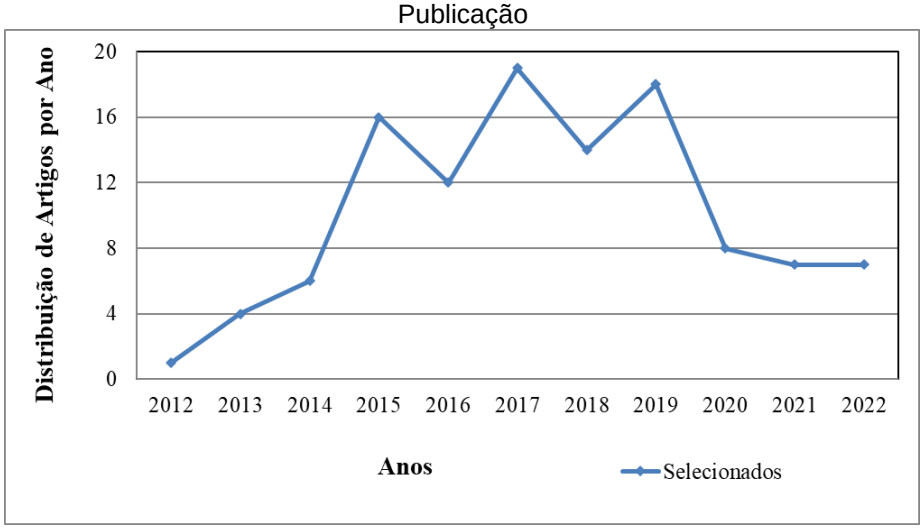
- **Artigo não revisado por pares:** Verificou-se que existiam entre os artigos pré-selecionados artigos publicados em congressos e revistas de teor não científico. Esses artigos foram excluídos**.**
- **O artigo citava NoSQL, porém não apresentava conceitos aplicáveis à NoSQL:** Verificou-se que existiam artigos com citações de NoSQL nas palavras chaves e/ou no resumo, porém com a leitura mais detalhada verificou-se que o artigo não abordava o assunto. Esses artigos foram excluídos.
- **O artigo não tratava nenhum aspecto de modelagem lógica para DW sobre NoSQL:** Verificou-se que existiam artigos que não abordavam diretamente modelagem lógica para DW ou ainda modelagem multidimensional aplicada a NoSQL. Esses artigos foram excluídos.

3.1 Resultados do Processo de Seleção

Inicialmente foram retornados 8.271 artigos. Destes, houve problemas para recuperação de 46 artigos, resultando em 8.225 artigos. Destes 8.225 artigos, após filtragem e seleção manual, foram selecionados 112 artigos para leitura completa, aproximadamente 1,36% do número inicial de artigos. A partir dos 112 artigos selecionados, cerca de 51% dos artigos pré-selecionados foram encontrados em várias bases de dados, e 34,75% foram excluídos do processo de seleção manual. Ainda notamos que dos 112 artigos selecionados, 75 foram publicados em congressos e conferências. Isso mostra a relevância atual do tema na pesquisa, uma vez que o processo de publicação em periódicos geralmente leva mais tempo do que em conferências. Mais detalhes da seleção dos trabalhos podem ser visto em (Ferreira, Alves-Souza e Da Silva, 2023).

Na Figura 12 apresenta-se a distribuição de artigos de 2012 a 2022; temas relacionados a NoSQL e modelagem multidimensional aumentaram durante este período. De 2020 a 2022, houve uma grande redução na publicação sobre este tema, ressaltando que este foi o período da ocorrência da pandemia da COVID-19, o que pode ter impactado no número de publicações.

Figura 12 - Distribuição das Publicações Selecionadas para Leitura Completa por Ano de Publicação

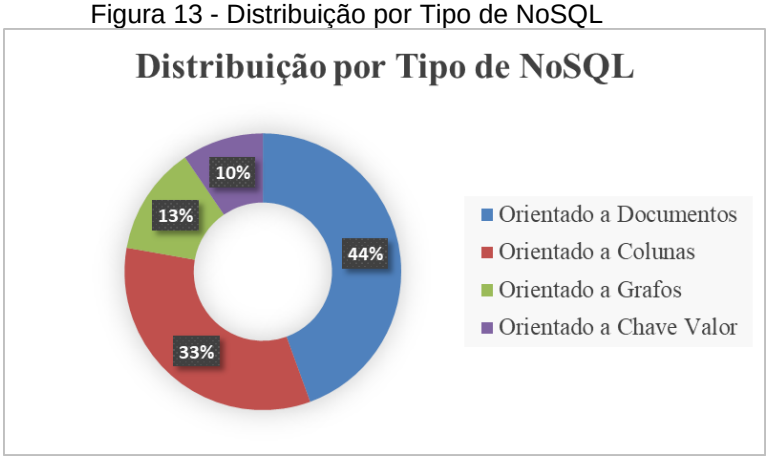


Fonte: Elaborado pelo Autor

3.2 Resultados Após Leitura dos Artigos

Apenas os artigos que trataram da modelagem multidimensional para NoSQL foram considerados relevantes. Assim, após a leitura de 112 artigos, 67 (~60%) foram classificados como adequados. De acordo com a Figura 13, os NoSQL Orientados a Colunas e Orientados a documentos são os mais utilizados (aproximadamente 77% dos casos). NoSQL orientado a documentos é o mais popular, alcançando 44% dos artigos citados; dentre este tipo de NoSQL, o *MongoDB* é o mais utilizado. NoSQL orientado a colunas aparece em segundo lugar para a maioria dos artigos publicados, e o *Apache Cassandra* e *Apache HBase* são os principais mencionados nesta classificação de NoSQL.

Figura 13 - Distribuição por Tipo de NoSQL

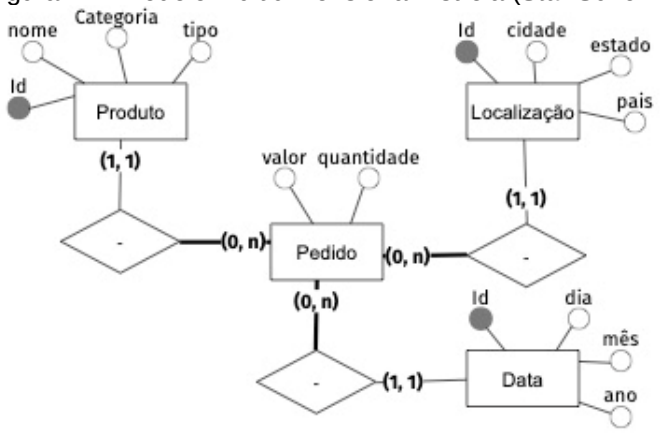


Fonte: Elaborado pelo Autor

Vários trabalhos empregam mais de um tipo de NoSQL, além de diferentes implementações NoSQL. A maioria dos artigos selecionados para esta RSL apresentaram implementações de modelos analíticos baseados no modelo multidimensional estrela (Kimball e Ross, 2002). A seção 3.2 apresenta mais detalhes sobre o modelo de dados. Como resultado, esta RSL evidenciou as diferenças entre a implementação do modelo multidimensional em BDR e em diferentes estruturas de NoSQL. Aproximadamente 45% dos artigos utilizaram algum tipo de ferramenta, tecnologia auxiliar, para realizar operações junto aos NoSQL. Isso pode ser explicado porque tais ferramentas permitem executar operações complementares sob NoSQL como junções de tabelas, agregações ou operações analíticas mais complexas.

A maioria dos artigos apresentaram diferenças entre as implementações do modelo estrela para NoSQL devido as diferenças da estrutura física de cada NoSQL. Segundo Pereira, Oliveira e Rodrigues (2015a), nos BDRs são criadas consultas em conformidade com o esquema e modelo de dados definidos, já no caso do NoSQL, a implementação, ou o mapeamento do modelo conceitual, é feita com base nas consultas definidas para o sistema analítico. Assim, identificou-se três grandes linhas de implementação do modelo multidimensional para NoSQL, nomeadas aqui como: (1) Modelo_Estrela (*Star Schema*), (2) Modelo_Plano (*Flat* Model), (3) Modelo_Plano_Agregado, ou Tabela Agregada (*Agreggated Flat Table*). Para facilitar a compreensão a Figura 14 ilustra o Modelo Multidimensional Estrela. Com base neste mostra-se o seu mapeamento para BDR e usando a álgebra relacional (Elmasri e Navathe, 2016; Silberschatz, Korth e Sudarshan, 2011), o modelo multidimensional estrela (Modelo_Estrela) e suas derivações: o Modelo_Plano e o Modelo_Plano_Agregado.

Figura 14 - Modelo Multidimensional Estrela (*Star Schema*)



Fonte: Elaborado pelo Autor usando Modelo Entidade-Relacionamento (Heuser, 2009)

No modelo conceitual multidimensional apresentado na Figura 14 existem quatro entidades: Produto, Pedido, Localização e Data. A forma como essas entidades são mapeadas para o banco de dados depende de sua estrutura. Assim, para um BDR corresponde as tabelas: Fato_Pedido, Dim_Produto, Dim_Localizacao e Dim_Data apresentadas a seguir. Os campos em negrito representam a chave-primária da tabela.

Fato_Pedido (valor, quantidade**, id_produto, id_localizacao, id_data**)
Dim_Produto (**id_produto**, nome, categoria, tipo)
Dim_Localizacao (**id_localizacao**, cidade, estado, pais)
Dim_Data (**id_data**, ano, mes, ano)

Para NoSQL, com exceção do orientado a grafos, como detalhado a seguir, as quatro tabelas são mapeadas para uma única estrutura (documento, tabela, nós, objeto), com três diferentes possibilidades, apresentadas nas equações 1, 2 e 3, usando o formalismo da álgebra relacional.

$$MODELO\_ESTRELA = (((Fato\_Pedido \bowtie \text{Dim\_Data}) \bowtie \text{Dim\_Localizacao}) \bowtie \text{Dim\_Produto}) \quad (1)$$

O Modelo_Plano é derivado a partir do Modelo_Estrela e apresentado na equação 2:

$$MODELO\_PLANO = \Pi_{Valor,\ Quantidade,\ Ano,\ Mes,\ Dia,\ Cidade,\ Estado,\ Pais,\ Nome,\ Categoria,\ Tipo}(MODELO\_ESTRELA) \quad (2)$$

O Modelo_Plano_Agregado é derivado do Modelo_Plano, a partir da agregação dos dados desejados (equação 3):

$$MODELO\_PLANO\_AGREGADO = \Pi_{Ano,\ Tipo,\ Pais,\ \boldsymbol{sum}(Valor),\ \boldsymbol{sum}(Quantidade)}\left(G_{Ano,\ Tipo,\ País}(MODELO\_PLANO)\right) \quad (3)$$

Onde G representa o agrupamento sobre os campos: ano, tipo e país.

3.3 Modelo_Estrela (*Star Schema*)

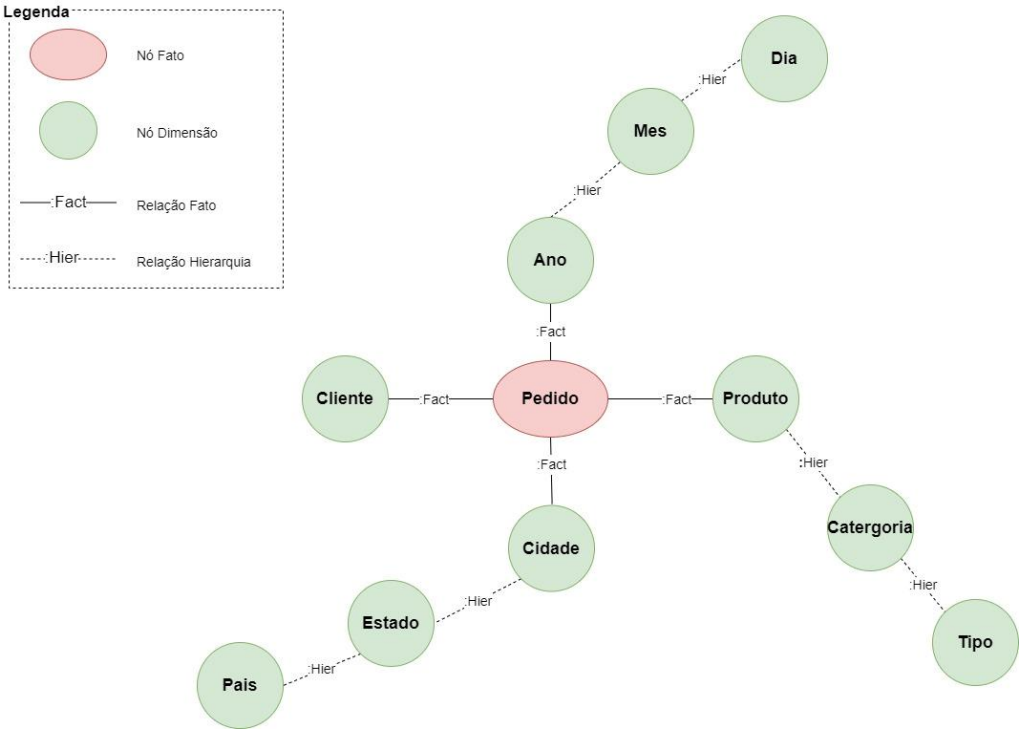
Modelo_Estrela é o mapeamento do modelo estrela (*Star Schema*) clássico em banco de dados NoSQL. Desta forma, fatos e dimensões tornam-se elementos mantidos fisicamente separados e com um relacionamento lógico. Modelo_Estrela foi principalmente observado nesta RSL em NoSQL orientado a grafos, devido a facilidade de representação de relacionamentos.

A representação mais interessante de Modelo_Estrela sobre NoSQL é vista em Castelltort e Laurent (2014a) com o cubo gráfico (*Graph Cube*) apresentado na Figura 15. O cubo gráfico foi desenvolvido originalmente por Zhao *et al.* (2011). Fatos e dimensões são mapeados para nós *(Fact Node* e *Dimension Node*) e interligados por arestas (*Fact Relation*) (Castelltort e Laurent, 2014a; Jianmin *et al.*, 2011). Os atributos que compõem cada dimensão são associados hierarquicamente e ligados por arestas (*hier*). Em NoSQL orientado a grafo, fatos e dimensões (nós) são mantidos fisicamente separados com relacionamento lógico entre eles como observado na Figura 15.

Liu e Vitolo (2013) apresentam um modelo similar ao de Castelltort e Laurent (2014a), apresentado na Figura 15, porém adicionaram a informação de dados sumarizados ou pré-computados em nós de hierarquias de dimensões. Apenas Dehdouh (2016), dentre todos os artigos analisados na revisão, propõe o uso da Modelo_Estrela para outros NoSQL que não sejam orientados a grafos. Entretanto, a junção de tabelas e cálculos agregados tornam-se um problema para NoSQL orientados a colunas, documentos e chave-valor visto que esses sistemas em geral não possuem uma linguagem com comandos de agregação

ou não possuem estrutura de processamento própria para agregação (Carniel *et al.*, 2012a; Chevalier *et al.*, 2015a; Yangui; Nabli; Gargouri, 2016a).

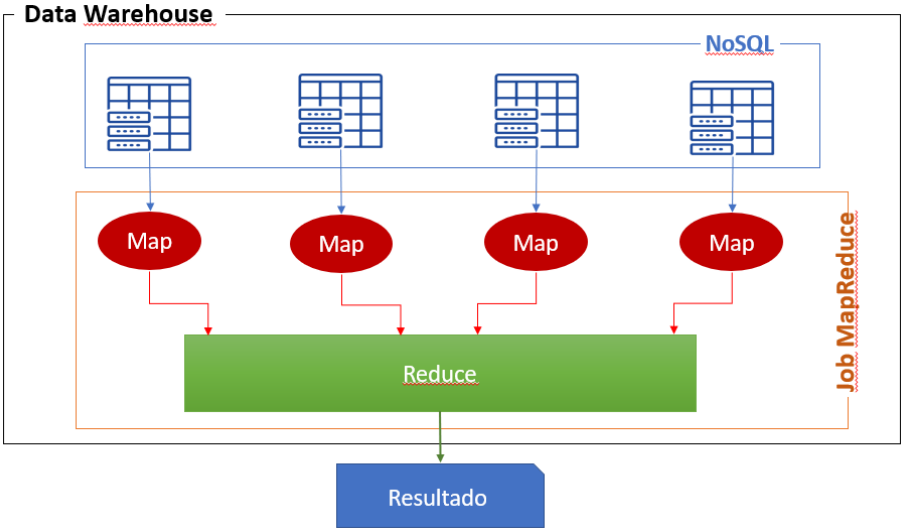
Figura 15 - Modelagem Multidimensional NoSQL Orientado a Grafos



Fonte: Adaptado de Castelltort e Laurent (2014a) pelo Autor

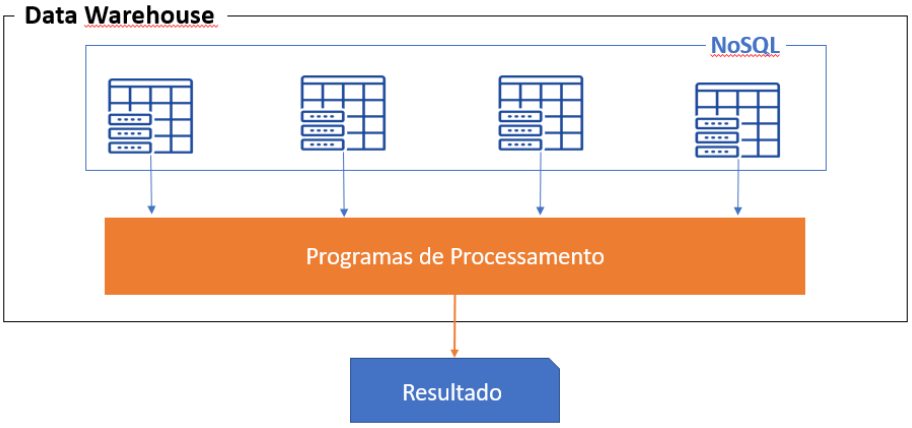
Assim, para aplicar Modelo_Estrela em NoSQL não orientado a grafos, é preciso utilizar soluções complementares como *MapReduce* para realizar operações de junção e agregações. Outra solução comumente utilizada é o emprego de alguma ferramenta que permita fazer tais operações. Cugnasco *et al.*(2016) empregam *Apache Hive*, Dehdouh *et al.* (2014a) utilizaram *Apache Phoenix*, além de ferramentas desenvolvidas pelos próprios autores Zhao e Ye (2014a). O processamento com *MapReduce* ou com auxílio de ferramentas de *software* para computação de junções e agregações são representadas nas Figuras 16 e 17.

Figura 16 - *Job* de Processamento de DW com *MapReduce*



Fonte: Elaborado pelo Autor

Figura 17 - *Job* de Processamento de DW com Ferramenta de *Software* para Execução de Junções e Agregações



Fonte: Elaborado pelo Autor

Os principais artigos que propões implementação do modelo analítico sobre NoSQL com base no Modelo_Estrela são: (Carniel *et al.*, 2012b; Castelltort e Laurent, 2014b; Challal *et al.*, 2019; Chevalier *et al.*, 2015b; Colella, 2019; Liu e Vitolo, 2013; Maghfiroh e Nugraha, 2018; Ravat *et al.*, 2019; Robinson, Webber e Eifrem, 2015; Zhao e Ye, 2014b).

3.4 Modelo_Plano (*Flat Model*)

Nesta forma de representação do modelo conceitual estrela em NoSQL, fatos e dimensões são representados na mesma entidade de dados (tabela,

documento), como foi apresentado na Figura 14. A junção da Fato com as Dimensões resulta em duplicação dos dados (equação 2). Entretanto, esta abordagem apresenta melhoria significativa para a operação de consulta a dados. O Modelo_Plano foi o mais citado nos artigos selecionados nesta RSL, porém com diferentes nomenclaturas, como está apresentada a sumarização na Tabela 1.

Tabela 1 - Nomenclaturas Encontradas para Modelo Plano

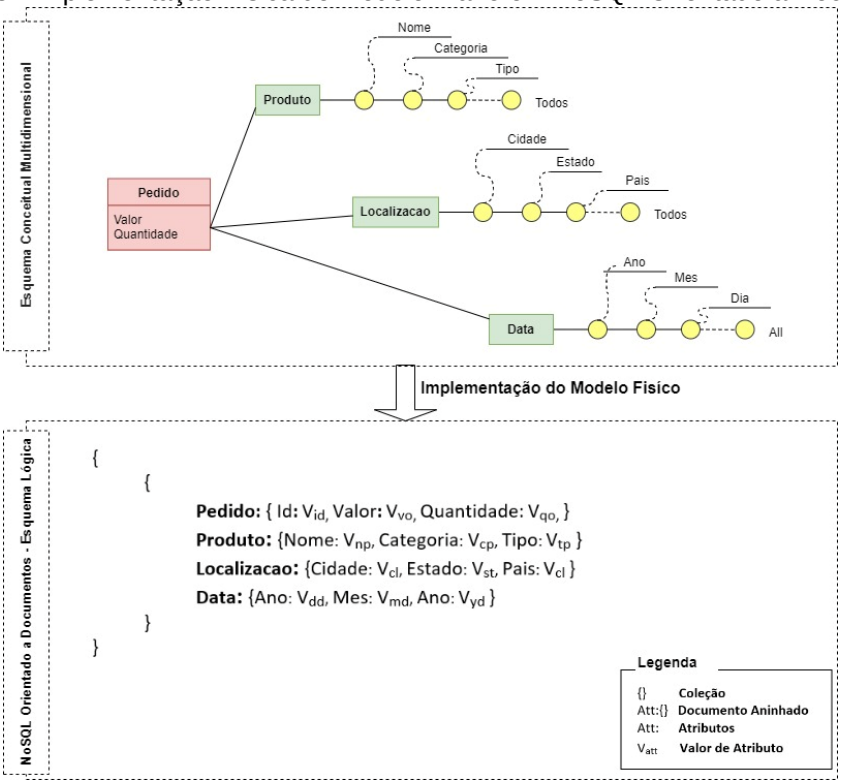
Nomenclatura	Acrônimo	Referência
Columnar NoSQL CUBE	CN-CUBE	Dehdouh *et al.* (2014a)
Columnar NoSQL Star Schema Benchmark	CNSSB	Scabora *et al.* (2016a)
Composite Object Model	COM	Pasqualin *et al.* (2016a)
Document Flat Model	DFM	Chevalier *et al.* (2016)
Document Warehouse	DocW	Messaoud, Ali e Feki (2017)
Falling Star	FS	Ferrahi *et al.* (2017)
Flat Model	FM	Murazza e Nurwidyantoro (2017)
Hierarchical Transformation	HT	Yangui, Nabli e Gargouri (2016b)
OLAP Dimension	OLAP-D	Chevalier *et al.* (2015b)
Same Column Family	SameCF	Scabora *et al.* (2016b)
Vertical Fragmentation View	VFV	Carniel *et al.* (2012b)
WideTable	WT	Dai (2019)

A utilização do Modelo Plano é em geral apropriada para NoSQL Orientado a Documentos e Orientado a Colunas. Em Chevalier *et al.* (2017) os documentos possuem uma estrutura que unifica todas as informações em um mesmo documento. Pereira, Oliveira e Rodrigues (2015a) propõem a completa desnormalização do esquema dentro dos documentos, sendo cada fato representado como um documento específico, com suas dimensões representadas como subdocumentos (*nested documents*). Esta representação do modelo é chamada de implementação física aninhada.

Outra proposta consiste na representação linear dos documentos, em que fatos e dimensões fazem parte de uma mesma coleção (*collection*). Nesta, os documentos que representam diferentes dimensões são organizados de forma linear, sem aninhamento. A Figura 18 mostra uma representação Modelo_Plano para NoSQL orientado a documentos não aninhada.

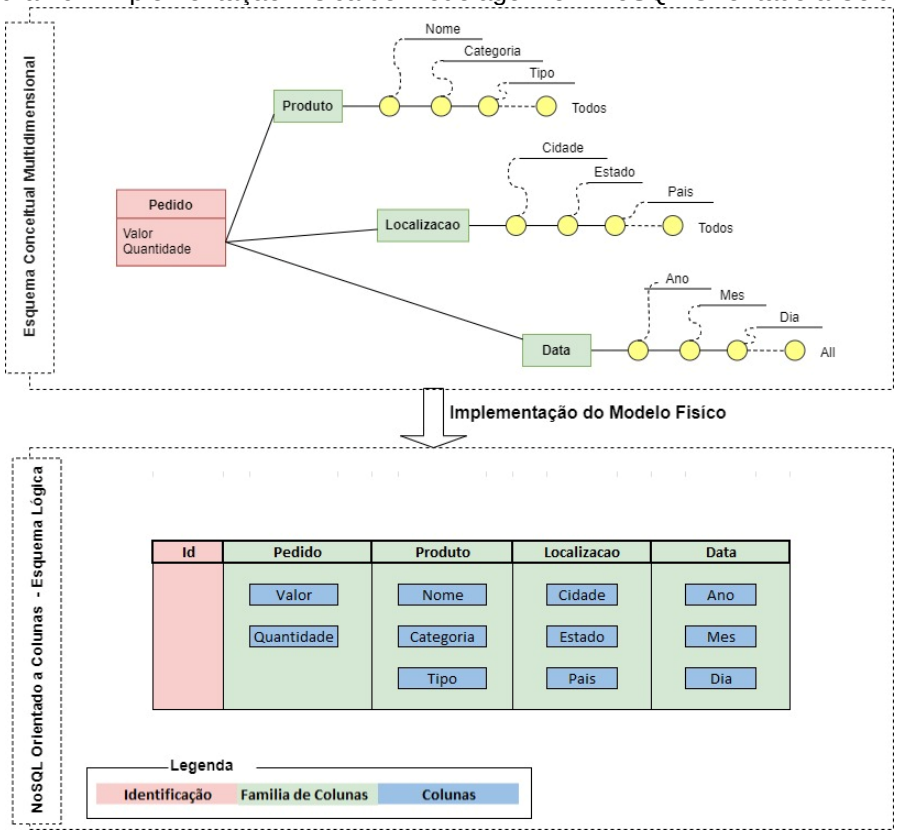
A implementação física do Modelo_Plano sobre NoSQL orientado a colunas é feita pela junção de fatos e dimensões na mesma tabela, utilizando uma única família de coluna como visto em Dehdouh *et al.* (2014a). Esta implementação física é semelhante à NoSQL orientado a documentos, como pode ser visto na Figura 18. Outra opção para NoSQL orientado a colunas é visto em Chevalier *et al.* (2015c), em que fatos e dimensões pertencem a mesma tabela, mas os fatos são apresentados em uma família de coluna e as dimensões são dispostas cada uma em uma família de coluna distinta. Na Figura 19 há uma representação gráfica da implementação física em que a tabela final é única, mas uma implementação fato é representada pela família de coluna “Pedido” e cada dimensão é representada na mesma tabela por uma família de coluna (“Produto”, “Localização”, “Data”).

Figura 18 - Implementação Física do Modelo Plano em NoSQL Orientado a Documentos



Fonte: Adaptado de Chevalier *et al.* (2015b) pelo Autor

Figura 19 - Implementação Física de Modelagem em NoSQL Orientado a Colunas



Fonte: Adaptado de (Chevalier *et al.*, 2015b) pelo Autor

Existem ainda outras vertentes para NoSQL orientados a colunas, pois existem implementações físicas diferentes para cada NoSQL. Por exemplo, a implementação do *HBase* é diferente do Cassandra, exigindo assim uma implementação física do modelo colunar (orientado a coluna) adequada a essa diferença (Chevalier *et al.*, 2015c; Pasqualin *et al.*, 2016a). Chevalier *et al.* (2015d) fez um *benchmark* para NoSQL orientado a documentos para avaliar o desempenho para consultas considerando as implementações físicas do Modelo_Plano aninhado e linear, e verificou que existe pequena diferença de desempenho entre as duas abordagens.

Segundo Pasqualin *et al.* (2016a), documentos com diversos níveis de aninhamento podem afetar o desempenho de consultas. O mesmo *benchmark* sobre NoSQL orientado a colunas para as duas abordagens comentadas anteriormente foi feito em Chevalier *et al.* (2015e), que também verificaram pouca diferença de desempenho. Utilizando um *benchmark* semelhante sobre

NoSQL orientado a colunas, Scabora *et al.* (2016a) também não observaram diferença significativa de desempenho entre os dois tipos de implementações.

Outro ponto importante em relação a Modelo_Plano é a organização de hierarquias complexas, ou seja, organização e granularidade dos atributos de uma mesma dimensão. Chevalier *et al.* (2017) e Bonnet *et al.* (2011a) propõem a utilização de "*arrays*" para construção de hierarquias complexas para NoSQL orientados a documentos. Pasqualin *et al*. (2016a) sugerem o emprego de documentos aninhados para o tratamento de hierarquias complexas, salientando que hierarquias de dimensões muitos profundas podem degradar o desempenho das consultas. Para NoSQL orientado a colunas, Chevalier *et al.* (2015f) propuseram que cada hierarquia de dimensões deve ser apresentada como uma coluna para cada dimensão, ou seja, colunas físicas nas famílias de colunas que representam cada dimensão.

Os principais artigos que sugerem a implementação do Modelo_Plano podem ser encontrados em: Akid *et al.* (2022); Akoka *et al.* (2021); Bonnet *et al.* (2011a); Bouaziz, Nabli e Gargouri (2019); Carniel *et al.* (2012b); Chevalier *et al.* (2016); Chevalier *et al.* (2015a); Dai (2019); Dehdouh *et al.* (2014b); Dehdouh, Boussaid e Bentayeb (2020); El Moukhi, El Azami e Hajbi (2022); Ferrahi *et al.* (2017); Ferreira, Alves-Souza e Da Silva, (2022); Khalil e Belaissaoui (2020), Khalil e Belaissaoui (2022); Khalil *et al.* (2020); Messaoud, Ali e Feki (2017); Murazza e Nurwidyantoro (2017); Oliveira, Victorino e Holanda (2021); Pasqualin *et al.* (2016b); Pereira, Oliveira e Rodrigues (2015b); Rizzi (2022); Sarr, Bame e Boly (2022); Scabora *et al.* (2016b); Sellami, Amal, Nabli e Gargouri (2020a) ; Sellami, Amal, Nabli e Gargouri (2020b); Tian, Carey e Maxon (2020); Tournier (2017); Yangui, Nabli e Gargouri (2016b).

3.5 Modelo_Plano_Agregado

Modelo_Plano_Agregado (equação 3) é uma abordagem que aplica o cálculo prévio de dados que necessitam ser sumarizados, ou seja, adicionam campos pré-computados de visões agregadas. Esta abordagem tem origem no conceito de *Multidimensional On Line Analytical Processing* (MOLAP), na qual os dados

são pré-processados e as agregações de dados são calculadas de forma prévia (Zhao e Ye, 2014a). O Modelo_Plano_Agregado foi proposto em diversos artigos selecionados nesta RSL, porém com diferentes nomenclaturas, apresentadas na Tabela 2.

Tabela 2 – Nomenclaturas Encontradas para Modelo Plano Agregado

Nomenclatura	Acrônimo	Referência
Exploratory OLAP *over Document Stores*	EXODuS	Chouder, Rizzi e Chalal (2019)
Data Cube	*DataCube*	Stantic (2018)
Materialized View	MV	Carniel *et al.* (2012b)
Cubo ID	cuboid	Chevalier *et al*. (2016)
NoSQL OLAP	NOSOLAP	Chevalier *et al*. (2016)

Carniel *et al.* (2012b), Chevalier *et al.* (2016, 2017), Stantic (2017a) e Zhao e Ye (2014a) abordaram sumarização, ordenação das hierarquias e indexação para NoSQL orientado a documentos. Pasqualin *et al.*(2016a) destacaram que esta abordagem favorece o desempenho de operações de consulta de valores agregados como "*drill-down*", "*roll-up*", dentre outras operações clássicas de sistemas OLAP.

Bonnet *et al.* (2011b) trabalharam com *MongoDB* e enfatizaram o uso de *MapReduce* para realização das agregações prévias de dados. Cugnasco *et al.* (2016) utilizaram o programa *Apache Spark* para realização da agregação prévia dos dados. Dehdouh *et al.* (2014a) utilizaram *MapReduce* no *Hadoop* com *Hive* e o *Apache Phoenix* com o mesmo propósito. Castelltort e Laurent (2014a); Chevalier *et al.* (2015f); Zhao e Ye (2014a) utilizaram programas, ou ferramentas, construídos especialmente para realização da agregação prévia.

A abordagem de agregação prévia é utilizada frequentemente em sistemas analíticos sobre BDR e assim pode ser facilmente aplicada sobre NoSQL orientado a grafos (Chevalier *et al.*, 2015e).

Os principais artigos que consideram o Modelo_Plano_Agregado podem ser visto em: Banerjee *et al*. (2021); Carniel *et al*. (2012b); Chevalier *et al*. (2016); Chouder, Rizzi e Chalal (2019); Davardoost, Sangar e Majidzadeh, (2020); Davardoost, Sangar e Majidzadeh, (2022); Fonseca, De Carvalho e Holanda

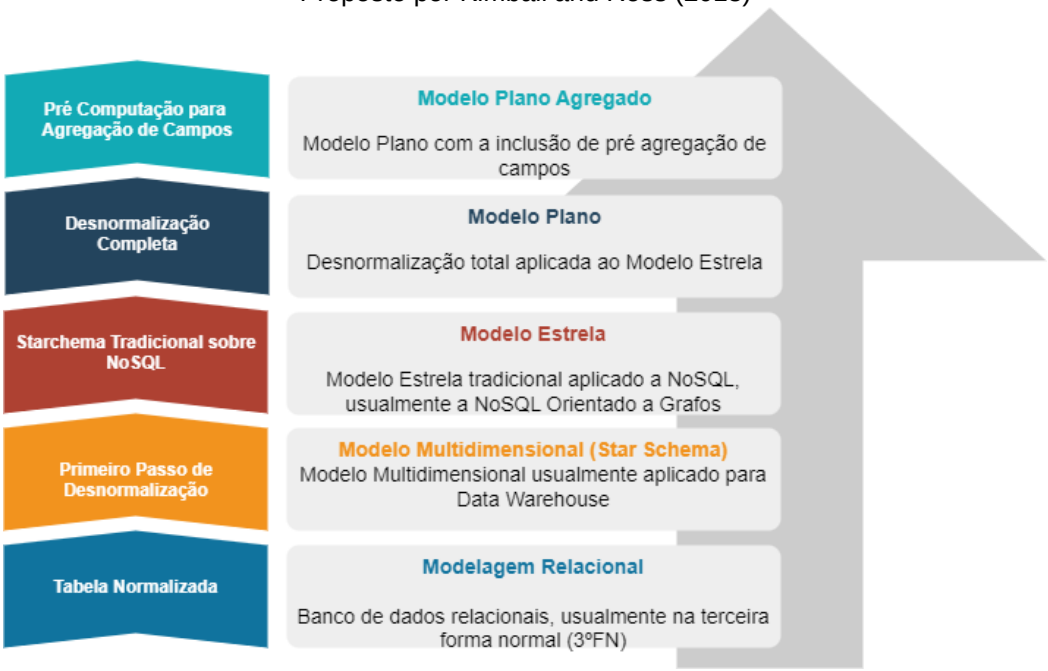
(2020); Pasqualin *et al*. (2016b); Prakash (2019); Stantic (2017b), Stantic (2018); Tournier (2017); Zhao e Ye (2014b).

3.6 Comparação entre os Mapeamentos do Modelo Estrela para NoSQL.

Modelo_Estrela, Modelo_Plano e Modelo_Plano_Agregado definidos, respectivamente, pelas equações 1, 2 e 3, são cada um, uma sobreposição do anterior com extensões em pontos específicos para melhor adaptação aos diferentes NoSQL. A Figura 20 esquematiza essa sobreposição e extensões feitas.

O Modelo_Estrela é a aplicação de uma desnormalização (3FN) ao modelo relacional, porém, mantendo tabelas fatos e dimensões separadamente para evitar duplicação de dados. As tuplas (objetos) são relacionadas por meio de chaves de relacionamento em tempo de consulta (Kimball e Ross, 2002). O Modelo_Plano é a adaptação do Modelo_Estrela para realizar a junção de Fatos e Dimensões. Por conseguinte, corresponde a total desnormalização, que é realizada quando todas as tuplas (objetos), representadas por fatos e dimensões, são relacionadas em um único documento ou em uma única tabela. Esta junção em um único objeto de dados pode gerar duplicação de dados. Similarmente, Modelo_Plano_Agregado estende o Modelo_Plano, acrescentado dados pré-computados de campos agregados (ou agrupados). Desta forma, tem-se dados totalmente desnormalizados em um único objeto de dados com a adição de valores pré-computados. Modelo_Plano_Agregado é muito semelhante aos modelos de MOLAP para BDR (Zhao e Ye, 2014a).

Figura 20 - Evolução das Representações com Base no Modelo Multidimensional Estrela Proposto por Kimball and Ross (2013)



Fonte: Elaborado pelo Autor

Nesta RSL não se identificou nenhuma pesquisa que apontasse alguma das abordagens apresentadas como a mais empregado em sistemas analíticos utilizando NoSQL como sistema de persistência de dados. Entretanto, pela experiência do autor pode-se averiguar que o Modelo_Plano é o mais utilizado nas organizações, tanto em corporações privadas como em órgãos científicos e de pesquisa.

Esta abordagem também apresenta uma flexibilidade, pois não é preciso conhecer previamente quais as agregações serão necessárias para o modelo de negócio. Porém, pode resultar na diminuição do desempenho para consultas que precisam de resultados sobre agregações. Contudo, isso não justifica o grau de complexidade ao adotar a abordagem Modelo_Plano_Agregado. Esta deve ser a razão em sua maioria dos artigos analisados sugerirem o emprego do Modelo_Plano.

4 *STARTABLE*: MAPEAMENTO DE MODELO MULTIDIMENSIONAL EM *BIG DATA* E NOSQL

Distintas pesquisas têm proposto uma forma de implementar sistemas analíticos utilizando NoSQL e sistemas de *Big Data* (sistemas de arquivos distribuídos) junto com ferramenta de *software* para consulta aos dados (ex. arquivos HDFS e *Apache Hive*), como visto no capítulo 3.

Entender as estruturas internas dos repositórios permite avaliar e escolher formas melhores para o armazenamento e distribuição dos dados visando a otimização das consultas. Assim, para a proposição da *Startable* procurou-se explorar propostas na literatura que propusessem a implementação de modelo analítico utilizando NoSQL orientado a documentos e a colunas, e ambientes *Big Data* além de entender as arquiteturas internas de produtos representativos dessas tecnologias.

Os produtos escolhidos foram o *MongoDB* e *Cassandra* como NoSQL orientado, respectivamente, a documentos e a colunas; como ambiente *Big Data* escolheu-se o *Hadoop*, usando HDFS como estrutura de arquivos com *Apache Hive*. Esses produtos, além de serem os mais utilizados na literatura, pelo menos até onde a RSL alcançou, também são bem conhecidos por esse autor. Apesar do foco em produtos específicos, que é importante pois se visa a implementação com otimização para consultas, procurou-se explorar estruturas que existam, ou que tenham similaridades, em outros produtos.

Hive apresenta um conjunto de características que facilita seu uso para a implementação de banco de dados analíticos (Thusoo *et al*., 2009), também possui catálogo de metadados que auxilia no mapeamento dos dados distribuídos no HDFS; permite simular diversas características/funções de BDR sobre os dados distribuídos no HDFS, por exemplo, manipular arquivos como tabelas, fazer o particionamento de dados, esquemas e consultar dados usando uma linguagem similar a SQL.

O uso adequado da estrutura de particionamento na distribuição de dados permite evitar varreduras de tabelas completas, reduzindo o tempo de execução de consultas, e assim, é um ótimo caminho para a otimização destas. A estrutura de particionamento é presente nas diferentes arquiteturas de banco de dados, desde relacionais a NoSQL e no *Hadoop*.

Vários autores como Nebot e Berlanga (2012); Amirthalingam e Rais, (2018); Santos e Costa (2016) exploraram a estrutura de particionamento para implementar o Modelo_Plano (apresentado na seção 3.3) sobre HDFS/*Hive*. De acordo com Santos e Costa (2016) o particionamento pode ajudar a reduzir os dados usados em uma cláusula "*where*", enquanto a estrutura de *buckets* (disponibilizada pelo *Hive*) funciona bem quando o atributo (ou campo) tem alta cardinalidade. Na implementação feita por esses autores há um conjunto de regras para a junção de algumas tabelas dimensão, além de tabelas com agregações pré-computadas, pois não é feita a junção de todas as tabelas dimensão e fato. Na manipulação de partições e *buckets* os autores levam em consideração exclusivamente a cardinalidade dos dados, não considerando as regras de negócio.

Chebotko *et al* (2015) propuseram uma implementação para o NoSQL Cassandra fortemente baseada nas regras de negócio e consultas que serão realizadas pelas aplicações. São aplicados conceitos de representação de chaves sobre o NoSQL envolvendo técnicas e conceitos de particionamento e clusterização. As escolhas das chaves e das partições são baseadas nas regras de negócios para melhorar o desempenho das consultas a serem realizadas sobre o banco de dados NoSQL. Desta forma Chebotko *et al* (2015) definiram princípios importantes de representação de dados, regras de mapeamento e padrões de mapeamento para orientar a modelagem lógica de dados sobre NoSQL. Apesar da proposta dos autores ser eficiente e atender os requisitos de negócios de diversos sistemas, esta foi desenvolvida exclusivamente para sistemas transacionais, não atendendo assim as necessidades para sistemas analíticos sobre NoSQL.

4.1 Particionamento e Clusterização de Dados em NoSQL e *HadooplHive*

O particionamento dos dados e a escolha das chaves são os principais mecanismos utilizados nos bancos de dados NoSQL e nos sistemas de *Big Data* para distribuição balanceada dos dados no cluster. Desta forma, esses aspectos têm importância fundamental no desempenho destas tecnologias.

Nas Figuras 21, 22 e 23 apresenta-se um esquema da estrutura de particionamento de dados em um *cluster MongoDB*, cluster *Cassandra* e *Cluster Hadoop/Hive* respectivamente. Vale destacar que a nomenclatura de técnicas de particionamento e *bucketing* são diferentes para Cassandra e *Hive*. Neste trabalho padronizamos a nomenclatura conforme demostrado na Tabela 3.

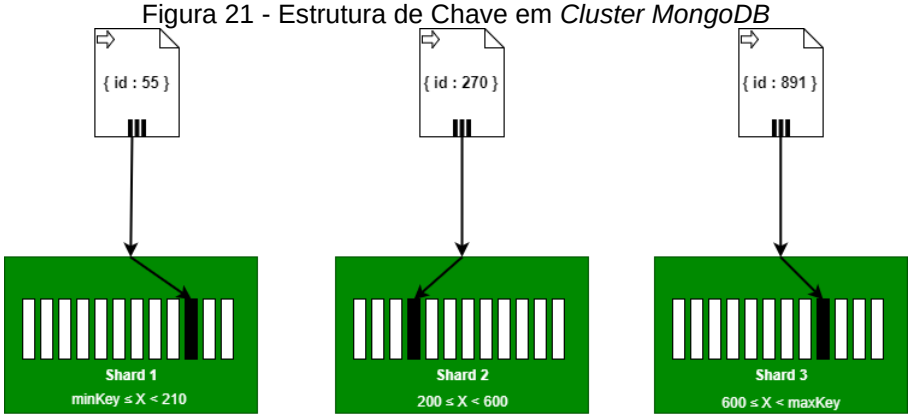
Tabela 3 - Padronização de Nomenclatura para Particionamento e *Bucketing*

Neste Trabalho	Cassandra	Hive	MongoDB	Descrição
Partição	Cluster	Partition	*	Separação de dados em grandes agrupamentos
Bucket	Partition	Bucket	*	Agrupamento de dados em pequenas partições com ordenação

* MongoDB disponibiliza apenas criação de índices, fazendo o particionamento e *bucketing* de forma automática conforme os índices criados.

Fonte: Elaborado pelo Autor

Figura 21 - Estrutura de Chave em *Cluster MongoDB*

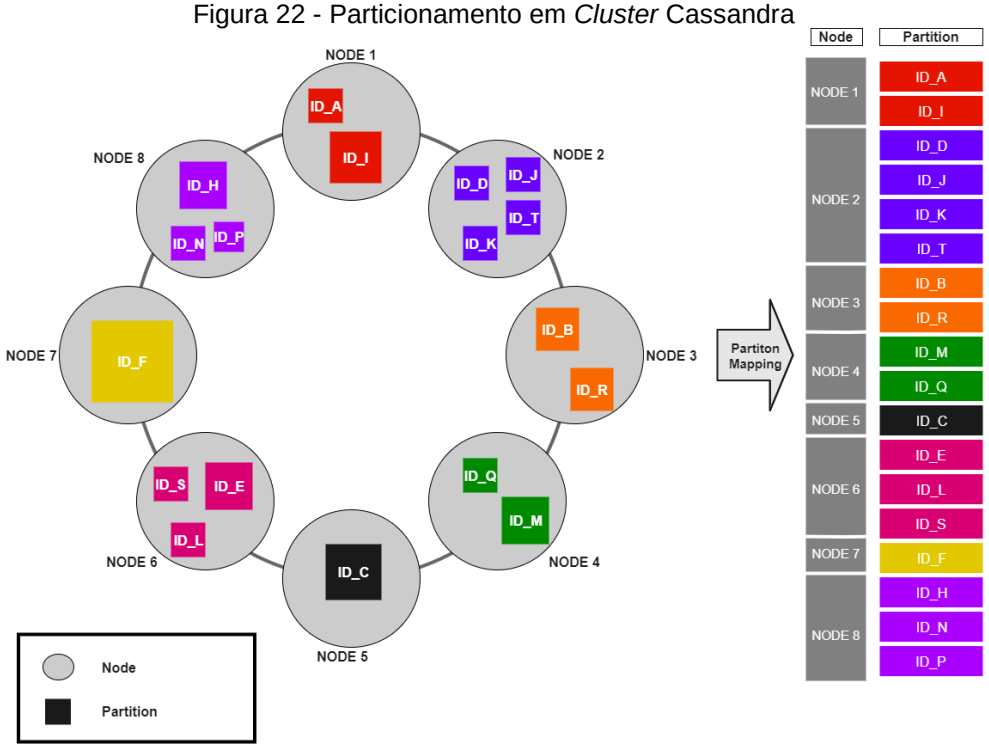


Fonte: Elaborado pelo Autor

Na Figura 21, no *MongoDB* os dados são distribuídos de acordo com os valores da chave e das faixas especificadas para o estrutura de *index* denominado *Shared Key* (semelhante à partição do *Hive* e MySQL) (MongoDB,

2023). O *cluster Cassandra* é representado como um anel e os dados são distribuídos pelo *cluster* de acordo com os campos escolhidos para serem parte da estrutura de particionamento denominada *Cluster Key* (semelhante a estrutura de *partição* no *Hive* e MySQL) (Cassandra, 2023), como pode ser visto na Figura 22.

Figura 22 - Particionamento em *Cluster* Cassandra



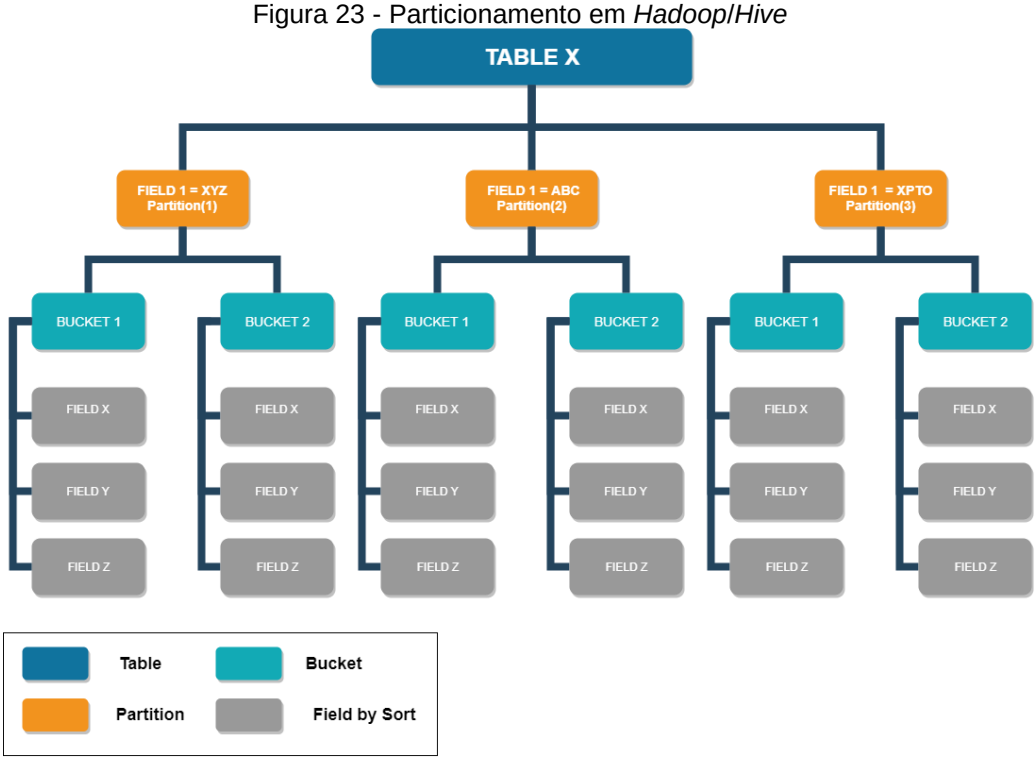
Fonte: Elaborado pelo Autor

Na Figura 23, tem-se o esquema da estrutura do *Hive* que distribui os arquivos físicos pelos nós de processamento para cada arquivo, desta forma os dados são distribuídos por partições e *buckets* de acordo com os valores do campo chave escolhido semelhante ao processo realizado pelo Cassandra (Hive, 2023).

No *Hive* e no *Cassandra*, os dados (elementos ou registros ou objetos) são armazenados de forma ordenada. Além disso, nestes e em outros NoSQL como o *MongoDB* e sistemas de *Big Data* são utilizados algoritmos de indexação e *hash* baseados em partições e *buckets* para auxiliar nas operações de inserção e de consulta a dados. A escolha equivocada de campos e da distribuição dos valores das chaves de busca pelas partições e *buckets* pode ocasionar desbalanceamento dos dados no cluster, resultando em partições com excesso

de dados, ou com poucos dados ou vazias, em um nó. Em outras palavras, a distribuição dos valores de chave por partições e *buckets* afeta a eficiência de consultas. Mais detalhes sobre as estruturas internas para indexação e *hash* podem ser encontradas em Cassandra (2023); Hive (2023); MongoDB (2023).

Figura 23 - Particionamento em *Hadoop*/*Hive*



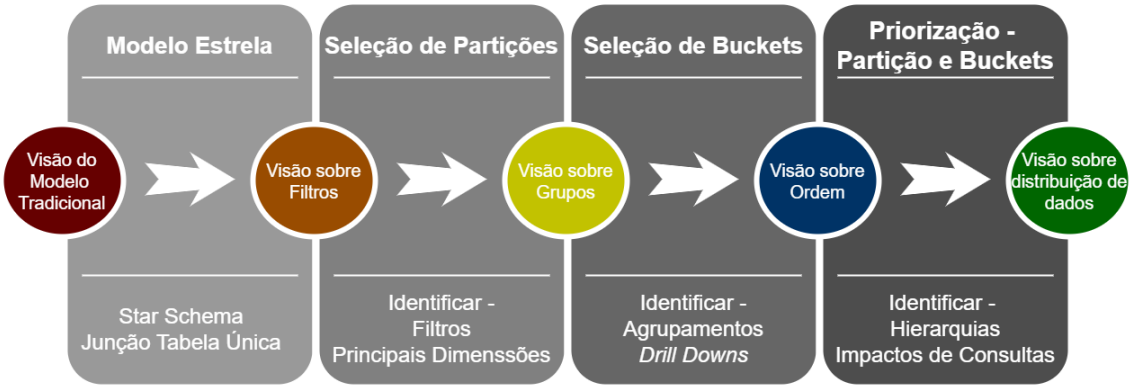
Fonte: Elaborado pelo Autor

4.2 Startable

Startable é proposto para a implementação de sistemas analíticos de auxílio à tomada de decisão sob tecnologias de persistência de dados NoSQL Orientado a Colunas (Cassandra), NoSQL Orientado a Documentos (*MongoDB*) e Sistemas de *Big Data* como *Hadoop* (HDFS/*Hive*). Como já dito anteriormente, apesar de ter-se avaliado as estruturas internas para três produtos representantes de cada uma dessas tecnologias, entende-se que a proposta da *Startable* pode ser também aplicável a outros produtos como S3 da *Amazon Web Service* (AWS), *Blob Storage/ Azure Data Lake* da *Microsoft Azure*, assim como outros sistemas distribuídos de dados baseados no conceito de *Object Storage*.

Considerando a proposta de Chebotko *et al.* (2015), propõe-se um conjunto de princípios e um fluxo de atividades que orientam a utilização da *Startable*. A Figura 24 esquematiza o fluxo de atividades proposto, que foi testado em ambiente controlado para verificar sua aplicabilidade e eficiência.

Figura 24 - Fluxo de Atividades para a Execução do *Startable*



Fonte: Elaborado pelo Autor

O Modelo_Plano apresenta eficiência em ambientes distribuídos e clusterizados. Além disso, a junção de dados pré-computados e materializados evitam a latência de rede e processamento adicional para realização de junções (*joins)* dos dados. Desta forma, existe um ganho de desempenho em comparação ao modelo estrela implementado. Vale destacar que há ganhos significativos quando se avalia os valores da chave de busca e os distribui adequadamente por partições e *buckets* (Nebot e Berlanga, 2012; Vasilakis *et al*., 2017; Boussahoua *et al*., 2017; Cugnasco *et al*., 2017).

Em Cugnasco *et al.* (2017) e em Boussahoua *et al.* (2017) foi visto que a implementação de algoritmos estatísticos é usada para identificar melhores formas de agrupamento e indexação de dados na implementação do Modelo_Plano em NoSQL orientado a colunas. Cugnasco *et al* (2017) utilizaram algoritmos de regressão linear para identificar campos mais apropriados para serem usados como *index*. Já Boussahoua *et al.* (2017) aplicaram técnicas de *K-means* para identificar melhores campos para serem usados nas chaves de agrupamento. Vasilakis *et al.* (2017) apresentaram uma técnica baseada em algoritmos de *Hash* de atributos, juntamente com algoritmo de compressão de dados para formação de chaves eficientes para particionamento e consultas.

Nesses três casos os autores utilizaram *benchmarks* obtendo ganhos reais de desempenho em consultas utilizando as técnicas apresentadas. Entretanto, as abordagens propostas por esses autores não abrangem problemas de negócios sofridos no mundo real, que é de enorme valia para organizações construírem sistemas analíticos para tomada de decisão.

Desta forma, o diferencial da proposta da *Startable* é a aplicação das regras de negócio que direcionam a distribuição dos dados por partições e *buckets*, além da apresentação detalhada do processo para a implementação do modelo estrela sob NoSQL e sistemas de *Big Data* como *Hadoop*. O fluxo com detalhamento das atividades para execução da *Startable* foi visto na Figura 24.

Na Figura 24 foram destacadas as 4 principais etapas para criação do *Startable*, detalhando-se os pontos principais envolvidos nas tarefas envolvidas em cada uma dessas etapas.

1. **Modelo Estrela:**
    - Desenvolver o processo padrão de modelagem multidimensional visto em Kimball e Ross (2013) e Inmon *et al*. (2019);
    - Realizar a desnormalização completa transformando o Modelo_*Estrela* em uma tabela única composta por fato e todas as dimensões, implementando o Modelo_Plano visto no ponto 3.4 e proposto em Dehdouh *et al*. (2014a), Ferreira (2015), e Chevalier *et al*. (2015d).
2. **Seleção de Partição:**
    - Identificar as dimensões que são mais utilizadas como filtros nas consultas, como por exemplo: Datas, Geolocalização (país, estados, cidades), Dados Organizacionais (nome de empresas, nome de filial);
    - Identificar os filtros necessários e os que se aplicam em proporção de Pareto na maioria dos casos, como por exemplo: 80% dos casos usam 20% das dimensões como filtros, de acordo com o conceito

hipotético de Pareto. Mais detalhes de como é realizada a proporção de Pareto pode ser visto em Aalst (2020);

- Identificar as dimensões que possuem uma distribuição mais uniforme dos dados no *cluster* - uma dimensão de dados mais granular tem uma distribuição mais uniforme. Sendo assim, entre as dimensões que identificam dados de geolocalização, por exemplo, a dimensão Cidade possui melhor característica para distribuição no *cluster* do que as dimensões País ou Estado;
- Minimizar a quantidade de campos para partições (sugerido de 1 a 5 campos de partições);
- Selecionar como campo de partição preferencialmente campos que são utilizados como filtros obrigatórios nas consultas para não gerar a varredura completa (*Full Scan*) da base de dados.

3. **Seleção de *Buckets*:**
    - Identificar a frequência de uso de filtros que não são necessários;
    - Identificar a frequência do uso dos campos em ordenações (*order by*);
    - Identificar a frequência do uso dos campos em agrupamentos (*group by*) e *drill down*;
    - Identificar a ordem da granularidade das dimensões e inserir de forma inversa (menos granular para o mais granular), por exemplo: para uma dimensão de tempo podemos ter a seguinte hierarquia: Ano, Mês, Dia. O campo Ano é o menos granular da dimensão. Caso necessitemos dos três campos nos *buckets* devemos inserir primeiro o campo ano, seguido por mês e dia, pois isto facilita a ordenação e o agrupamento dos dados, bem como no uso de *Drill Downs*.
4. **Priorização de Partição e *Buckets*:**
    - Selecionar como Partição e *Buckets* campos que aparecem em grande parte das consultas que respondem as perguntas de negócio.

- Caso haja a possibilidade de teste de desempenho de consultas, deve-se selecionar como Partição e *Buckets* campos que são usados nas consultas com maior impacto na carga de trabalho do sistema analítico, por exemplo, campos que são usados em filtros para diminuição do tempo de consulta complexas (campos em funções de igualdade em declarações "*where"*), ou campos utilizados em agrupamentos, funções de agrupamentos e filtros de agrupamentos (*group by, having, max, min, mean etc)*;
- Em caso de hierarquias de dimensões utilizar o campo mais expressivo (ou composto) como *bucket*, caso não exista ou haja necessidade de utilizar mais de um campo de hierarquia, utilizar primeiro os campos com menor granularidade. Exemplo: Hierarquia de datas. Caso haja um campo Data (mais expressivo) (YYYYMMDD) utilizar este campo como *bucket*. Caso contrário, utilizar os campos em ordem de menor granularidade no *bucket* (Ano, Mês, Dia);
- Todos os campos utilizados como *partição* devem ser utilizados sempre que possível como filtros obrigatórios nas consultas para evitar *Full Scan*.

Além das tarefas detalhadas, cujas principais foram destacadas na Figura 24, para cada uma das etapas, há ainda as visões que permeiam cada etapa do processo. Uma visão representa o foco nos dados e no modelo de negócio em cada uma das transições entre as etapas. Tem-se 5 visões:

- **Visão do Modelo Tradicional** – Desenvolver o modelo estrela tradicional para os dados existentes;
- **Visão sobre Filtros** - Estabelecer de forma clara os principais filtros utilizados, considerando as características do negócio para a maior parte das questões que serão respondidas para a execução da *Startable*;
- **Visão de Grupo** – Estabelecer os principais agrupamentos utilizados de acordo com as regras do negócio;

- **Visão de Ordem** – Estabelecer a ordem dos dados e as prioridades que serão apresentadas pelas consultas;
- **Visão de Distribuição dos Dados** – Focar na distribuição dos dados no cluster. É necessário ponderar sobre a melhor forma de distribuição dos dados pelos clusters, contribuindo com a eficiência do *cluster* no processamento das consultas.

Seguindo cada uma das etapas e a sequência das tarefas propostas, considerando a visão que se deve atingir em cada fase, cumpre-se todos os requisitos para a criação da *Startable*. Na próxima seção mostra-se a execução do fluxo proposto utilizando o *benchmark Star Schema Benchmark* (SSB).

4.3 *Benchmark* – Modelagem da Aplicação

Para testar o desempenho foi utilizado um *benchmark* analítico sobre ambiente controlado. Sim, Easterbrook e Holt (2003); Wang *et al.* (1999) destacam o uso de *benchmarks* como importantes instrumentos para comparar o desempenho de sistemas de computadores, algoritmos e a recuperação de informações em bancos de dados e muitas outras tecnologias.

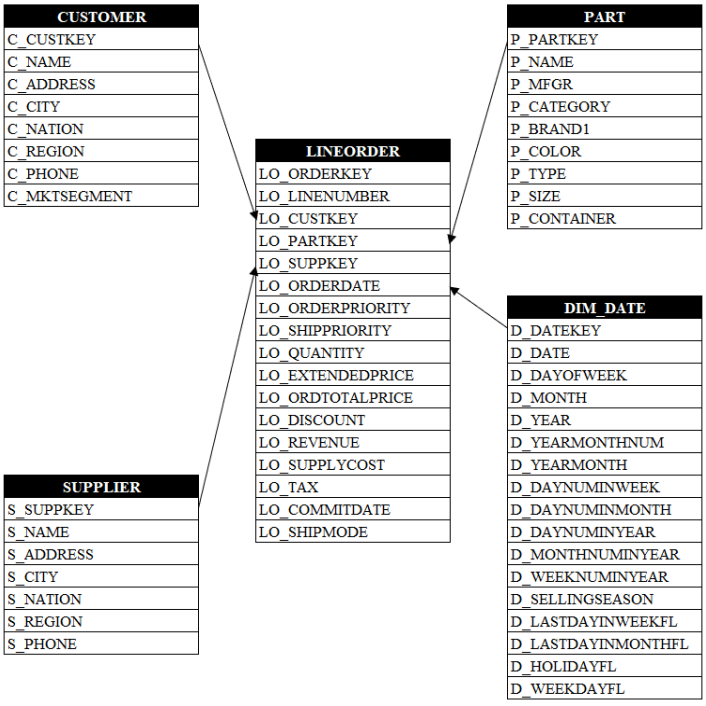
O modelo escolhido como *benchmark* foi o SSB, proposto em O'Neil *et al*. (2009). O SSB é conhecido pela simplicidade de aplicação e por ser um *benchmark* específico para modelagem multidimensional. Além disto, é uma simplificação do *benchmark Transaction Processing Performance Council Benchmark™ H – (TPC-H)*. O SSB transforma o modelo *Snow Flake* proposto no TPC-H no modelo multidimensional estrela (*Star Schema)*.

O SSB dispõe de um gerador de dados e de um conjunto de 13 consultas prontas, que permitem medir o desempenho sobre diversos aspectos como filtros, igualdades, desigualdades, agrupamentos e ordenação. As 13 consultas aplicadas sobre este modelo podem ser vistas de forma detalhada em (O'Neil *et al*., 2009) e no Apêndice. O modelo de dados do SSB é composto por uma tabela fato e 4 dimensões como pode ser visto na Figura 25. Ao modelo da Figura 25

foi aplicado ao fluxo de atividades proposto para a execução da *Startable*. O resultado das etapas dessa execução é apresentado na Figura 26.

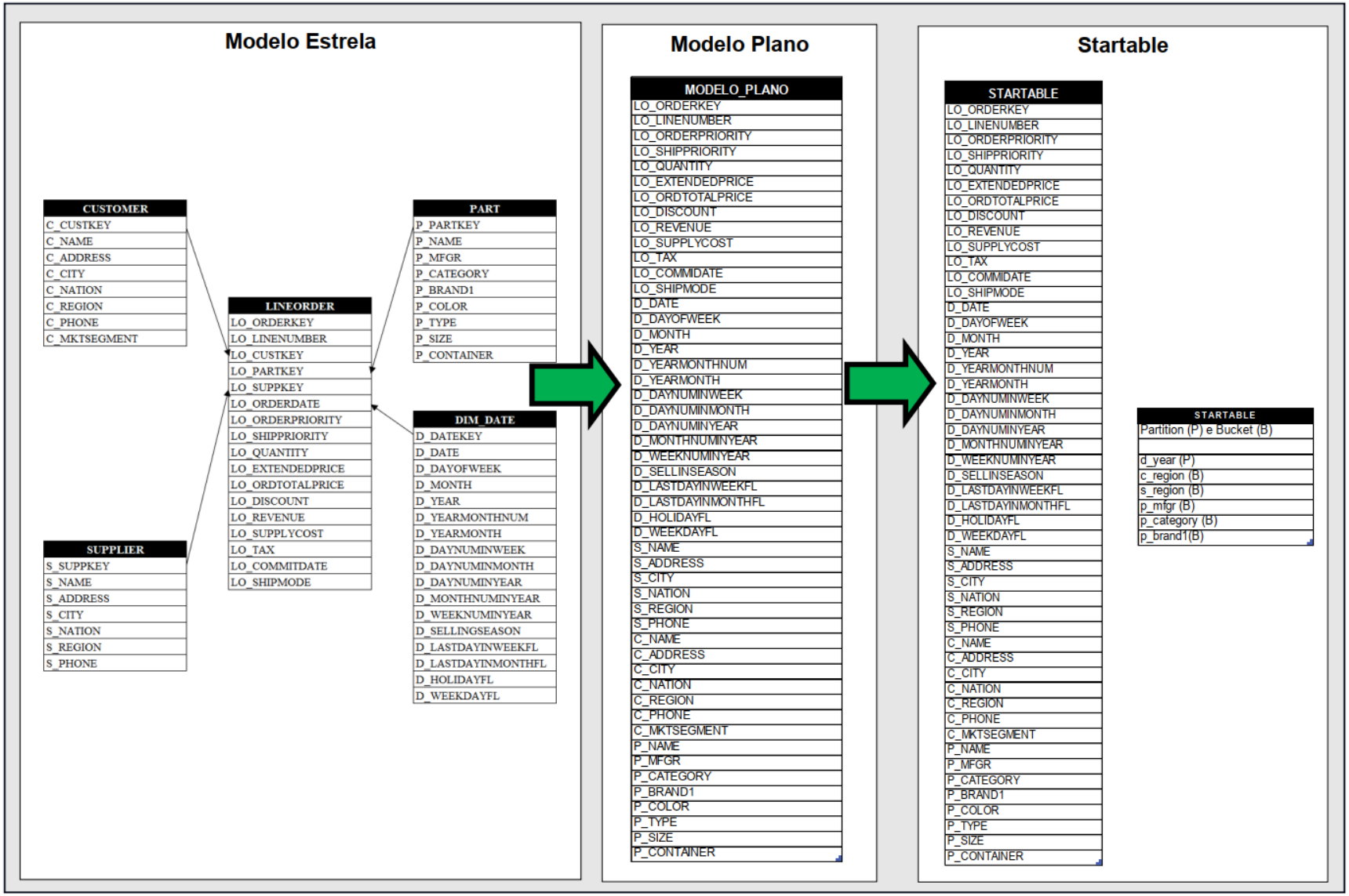
Na Figura 26 é observado o processo de transformação do modelo de dados do SSB com o Modelo_Estrela, seguindo para o Modelo_Plano, sendo finalizado no modelo *Startable*. O modelo de dados gerado como *Startable* foi replicado fisicamente sobre um *cluster* de banco de dados NoSQL *Cassandra*, um *cluster* de banco de dados NoSQL *MongoDB* e um *cluster Hadoop* com *Hive*. Foi medido também o desempenho da modelagem *Startable* em contraposição ao modelo_estrela, bem como uma comparação com a versão modelo_plano, modelo este proposto por diversos autores (Carniel *et al*., 2012a; Dehdouh *et al*., 2014a; Ferreira, 2015; Chevalier *et al*., 2015e).

Figura 25 - Modelo de Dados SSB (Modelo Estrela)



Fonte: Elaborado pelo Autor

Figura 26 - Resultado de Algumas Etapas do Fluxo de Execução da *Startable*



Fonte: Elaborado pelo Autor

Como não se pode descrever o modelo de negócio do SSB, definindo por exemplo, um campo que funciona obrigatoriamente como filtro em todas as consultas, fez-se um estudo dos dados e tabelas envolvidas para o entendimento do modelo de negócio implícito no SSB. Posterior a isto, aplicou-se as 4 etapas da *Startable*, tendo em mente as visões que permeavam cada uma dessas etapas. Desta forma chegou-se o modelo final *Startable*. Para facilitar o entendimento das consultas e do modelo de negócio original do SSB realizamos uma contagem de todos os campos que são utilizados nas 13 consultas do modelo. A Tabela 4 apresenta a contagem dos campos em consultas do modelo SSB.

Tabela 4 - Contagem de Campos em Consultas do Modelo SSB

Rótulos de Linha	Contagem de Campos	Soma de Where	Soma de Group By	Soma de Order By
c_city	3	0	3	0
c_nation	3	1	2	1
c_region	4	4	0	0
d_year	12	7	10	10
d_yearmonth	1	1	0	0
d_yearmonthnum	2	2	0	0
lo_discount	3	3	0	0
lo_quantity	3	3	0	0
p_brand1	4	2	4	4
p_category	3	2	1	1
p_mfgr	2	2	0	0
s_city	4	1	4	1
s_nation	3	1	2	1
s_region	7	7	0	0
Total Geral	**54**	**36**	**26**	**18**

Fonte: Elaborado pelo Autor

Na Tabela 4 pode-se destacar a quantidade de vezes que o campo “d_year” é utilizado nas consultas, principalmente nas condições “*where*”. Além disto, o campo “d_year” aparece de alguma forma como argumento em 12 das

13 consultas que temos para o SSB. Com estas informações descritas e analisando os dados do modelo SSB, a *Startable* foi aplicada e analisada com base nas 4 principais etapas apresentadas na Figura 24:

1ª Etapa: **Modelo Estrela:**

- Não foi necessária ação para modelar os dados segundo o modelo estrela já que o modelo de dados SSB já estava nesta etapa.
- Realizou-se apenas a junção dos fatos e dimensões alterando o modelo_*estrela* para um modelo_plano e suprimindo as colunas de chaves primárias das tabelas dimensões.

2ª Etapa: **Seleção de Partição**:

- Identificou-se que o campo “d_year” era o campo mais utilizado como filtro, seguindo a proporção de Pareto. O campo “d_year” também possuía distribuição uniforme dos dados qualificando-o ainda mais como candidato a **partição**.
- Outro campo que poderia ser utilizado como **partição** era o campo “s_region”, pois era muito utilizado como filtro e poderia distribuir as **partições** em segmentos menores. Porém este poderia também fragmentar as partições de maneira expressiva. Assim o único campo utilizado como **partição** foi apenas o campo “d_year”.

3ª Etapa: **Seleção de *Buckets***:

- Os campos “s_region” e “c_region” foram identificados como campos usados com muita frequência em filtros.
- Os campos “p_mfgr”, “p_category”, “p_brand1” foram identificados como campos com muita frequência em clausulas “*ordem by*” e “*group by*”.
- Outra informação importante a se destacar é que os campos da dimensão “*part*” são campos organizados de forma hierárquica, sendo o campo “p_mfgr” o menos granular e o campo “p_brand1”, o

mais granular. A Tabela 5 apresenta as informações contidas em um campo mais granular e em um campo menos granular.

Tabela 5 - Exemplo de Valores para os Campos da Dimensão "part"

Campo	Valor
p_mfgr	MFGR#1
p_category	MFGR#12
p_brand1	MFGR#123

Fonte: Elaborado pelo Autor

4ª Etapa: **Priorização de Partição e *Buckets***:

- Pela frequência que era utilizada nas consultas em cláusulas de filtros e pela sua distribuição mais homogênea de dados o campo “d_year” foi utilizado como único campo para partição.
- Pela frequência que era utilizado nas consultas em cláusulas de filtros os campos “s_region” e “c_region” foram escolhidos para serem os principais campos de *bucket*. Tanto “s_region” como “c_region” são utilizados em 7 consultas como filtros, não sendo utilizados nenhuma única vez para agrupamento ou ordenação. Ainda assim, o campo “s_region” possui uma distribuição menor. Desta forma, este campo foi escolhido para ser utilizado como campo principal de *bucket* seguido por “c_region”.
- Pela frequência do uso em cláusulas de agrupamento e ordenação, utilizou-se os campos “p_mfgr”, “p_category”, “p_brand1” como campos de *buckets* seguindo a hierarquia da dimensão para este uso.

Desta forma, a estrutura final das partições e *buckets* ficou definida como pode ser visto na Tabela 6. A ordem dos campos, conforme aparece na Tabela 6, foi a ordem para ordenação física dos dados nas partições.

Tabela 6 - Estrutura de Partições e *Buckets* aplicado ao SSB

Startable
Partições (*P) e Buckets (B)*
d_year (P)
s_region (B)
c_region (B)
p_mfgr (B)
p_category (B)
p_brand1 (B)

Fonte: Elaborado pelo Autor

4.4 *Benchmark* – Ambiente e Implementação

O *benchmark* foi implementado em um *cluster* distribuído. O *cluster* utilizado possuía 4 máquinas. Para construção deste *cluster* foi utilizado o *Google Cloud Platform* como a plataforma de serviço (*Platform as a Service* - PaaS). Todas as máquinas utilizadas tinham as seguintes configurações: 200 GB de Disco Rígido (não SSD); 8 CPUS (Intel Skylake); 48 GB de Memória; e Sistema Operacional Cento OS7.

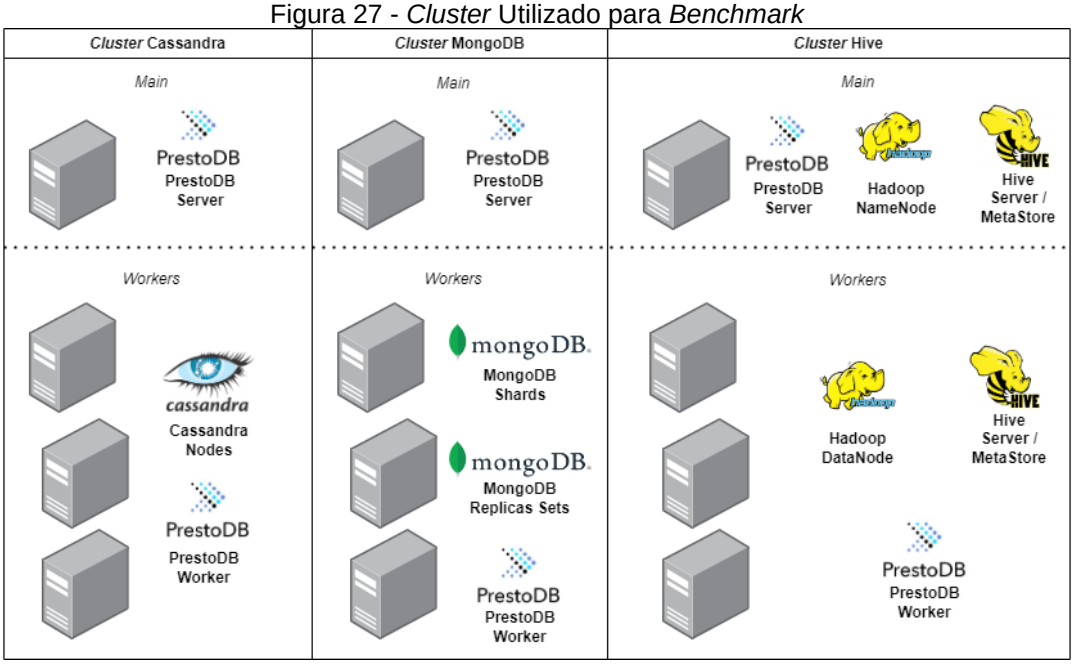
O *benchmark* foi implementado em três sistemas de persistência diferentes, conforme já falado: o primeiro foi HDFS; o segundo foi um *cluster* de NoSQL no *Apache Cassandra*; e o terceiro foi um *cluster* de NoSQL no *MongoDB*. Junto ao HDFS foi utilizado o *Hive* como repositório de metadados.

O *cluster Hadoop* com HDFS e *Hive* foi configurado com um nó principal (*main*) com três nós trabalhadores (*workes*). Já o *cluster* de *Cassandra* e *MongoDB* não tiveram nós principais (*main*), porque a arquitetura do *Cassandra* não requer nenhum tipo de nó principal (*main*), enquanto, o *MongoDB* utiliza uma estrutura chamada *Sharding* para trabalhar em cluster. A estrutura do *MongoDB* necessita de serviços de *Shards* e de *Replica Set* designados para distribuir e acessar os dados. Estes serviços poderiam estar em nós diferentes dos nós de serviços de armazenamento de dados, entretanto, para padronização, cada nó

de armazenamento de dados no *MongoDB* foi instalado um serviço de *Shards* e de *Replica Set.*

Entre os diversos sistemas de consultas disponíveis do mercado foi escolhido o *PrestoDB* pela sua capacidade de ser utilizado com o HDFS, *MongoDB* e *Cassandra*, facilitando a implementação do *benchmark*. Pode-se visualizar uma representação em alto nível da arquitetura final do *cluster* utilizado no *benchmark* na Figura 27:

Figura 27 - *Cluster* Utilizado para *Benchmark*



Fonte: Elaborado pelo Autor

Pode-se ver na Figura 27 que 1 nó *cluster* é utilizado com principal (*main*). Neste nó foi instalado os serviços de *Hadoop NameNode*, *Hive Serve*, *Hive MetaStore* e *PrestoDB Server*. Os outros 3 nós do *cluster* foram utilizados como trabalhadores (*workers*) e nestes foram instalados o *Cassandra*, *MongoDB Shard*, *MongoDB Replica Set*, *Hadoop* DataNode além dos serviços de *Hive* e do *PrestoDB Worker*.

As versões utilizadas dos *softwares* descritos foram: *MongoDB Community Server* 4.0.6; *Apache Cassandra* 3.11.4; *PrestoDB* 0.214; e *Horton Works Data Platform* 2.6.5 (Distribuição de *Hadoop* e *Hive* como pacote). O

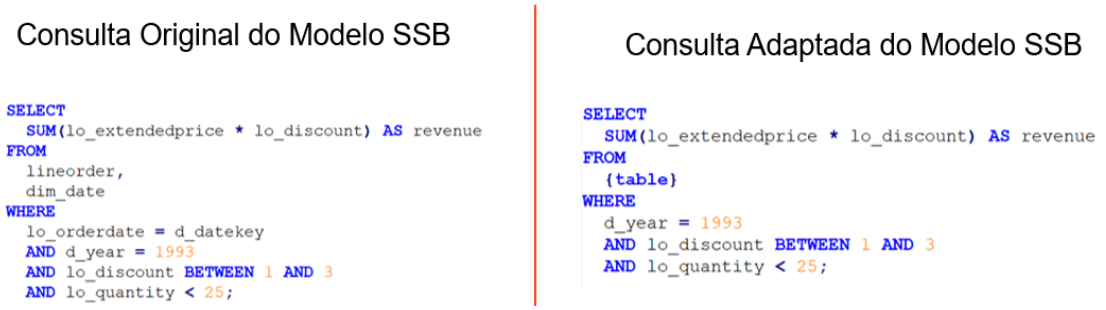
modelo de dados SSB convertido em *Startable* foi implementado em cada uma das estruturas de persistência de dados.

Para os dados no *Hadoop*/*Hive* tivemos a seguinte estrutura de *Partições* e *Buckets*: PARTITIONED BY (d_year); and CLUSTERED BY (s_region, c_region, p_mfgr, p_category, p_brand1) SORTED BY (s_region, c_region, p_mfgr, p_category, p_brand1) INTO 64 BUCKETS. Para o NoSQL *Apache Cassandra* tivemos a seguinte estrutura de Clusters (corresponde a partições no *Startable*) e Partições (corresponde aos *buckets* no *Startable*): PRIMARY KEY ((D_YEAR), S_REGION, C_REGION, P_MFGR, P_CATEGORY, P_BRAND1)). Destaca-se que para o *Apache Cassandra* os parênteses mais internos referem-se aos campos definidos como partições (*Clusters*) e os campos nos parentes mais externos referem-se aos campos definidos como *buckets* (*Partitions*).

NoSQL *MongoDB* não possui uma estrutura explicita de partições e *buckets*. O NoSQL *MongoDB* usa uma estrutura de indexação e faz o particionamento dos dados de acordo com a escolha da chave de indexação e da chave de distribuição (*Shard Key*). Desta forma, os campos colocados como primeiros nas chaves são utilizados principalmente em cláusulas de filtragem e os campos mais distantes são usados para ordenação e agrupamento. Desta forma, o modelo definido de partição e *buckets* no *Startable* foi implementado no *MongoDB* como chave única de indexação da seguinte maneira: db.table.createIndex({d_year: 1, s_region: 1, c_region: 1, p_mfgr: 1, p_category: 1, p_brand1: 1}). Além disso, nenhum campo no documento do *MongoDB* foi aninhado, ou seja, todos os campos foram definidos no primeiro nível do documento.

Para executar as consultas para o Modelo_Plano e *Startable* as consultas padrões do modelo SSB precisaram ser alteradas. Todas as cláusulas de “*join”* foram removidas e a declaração de “*from*” foi substituída para realizar as consultas em uma única tabela. A cláusulas “*select*”, “*where*”, “*group by*” e “*order by*” não foram alteradas. A Figura 28 apresenta um exemplo das mudanças realizadas.

Figura 28 - Adaptação de Consultas do Modelo SSB



Fonte: Elaborado pelo Autor

Como pode-se ver na Figura 28 o exemplo exposto precisou de adaptação simples, basicamente alterando a cláusula "*from*" por {table}. O argumento {table} recebe o nome do modelo que originou a tabela, podendo ser por exemplo *Startable*.

4.5 Resultados de Desempenho da Implementação *Startable*

Nesta seção serão apresentados os resultados em relação a metodologia descrita acima. Para se analisar a implementação da *Startable* foram realizados dois testes com cargas de dados de 1GB e 10 GB de dados. Foram testadas: modelo_estrela, modelo_plano e a *Startable*. Cada um destes foi testado com cargas de 1GB e 10GB. A Tabela 7 mostra o volume total de dados e a quantidade de registros gerados para cada caso. Cada um destes casos foi implementado no *Hadoop/Hive*, *MongoDB* e *Apache Cassandra.*

Os testes foram baseados em tempo de resposta de cada consulta do SSB sobre cada um dos sistemas de persistência. Destaca-se que todas as consultas foram submetidas no *PrestoDB*, que as executava no *Hadoop/Hive*, *MongoDB* ou *Cassandra*. As consultas eram disparadas para o *PrestoDB* a partir de um *script* desenvolvido na linguagem *Python*, o qual realizava a devida mensuração dos tempos gastos em cada consulta. Após os testes os dados foram compilados e os resultados dos tempos de respostas das consultas, em segundos, para os três modelos testados (modelo_estrela, modelo_plano e *Startable*) podem ser observados nas Figuras de 29 a 34. Todos os resultados

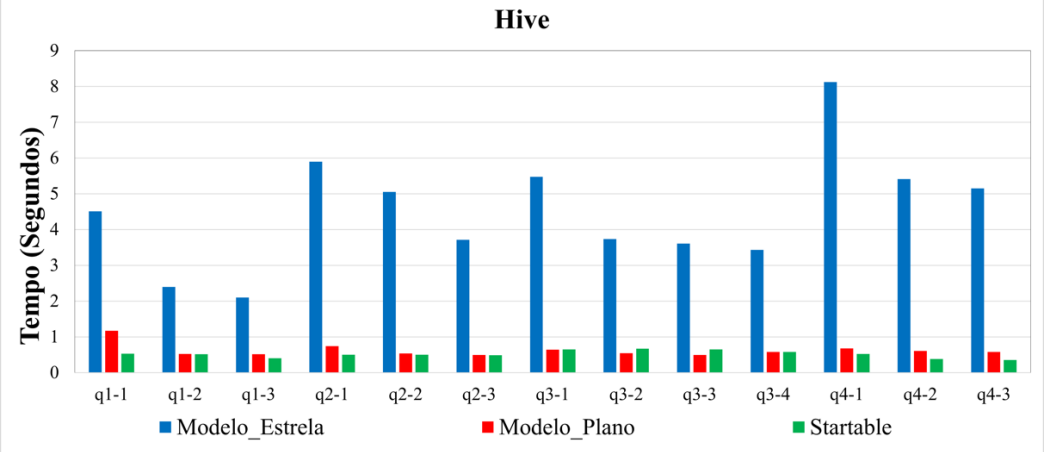
são apresentados de acordo com o tempo de retorno, em segundos, de cada uma das 13 consultas propostas pelo SSB.

Tabela 7 - Volumetria de Dados do *Benchmark*

Volume	Tabela	Espaço	Quantidade
1G	customer	5.5M	60000
1G	supplier	328K	4000
1G	dim_date	228K	2556
1G	part	33M	400000
1G	lineorder	1.2G	11998051
1G	modelo_plano	1.2G	11998051
1G	*Startable*	1.2G	11998051
10G	customer	47M	510000
10G	lineorder	9.9G	101987916
10G	part	83M	1000000
10G	supplier	2.8M	34000
10G	dim_date	228K	2556
10G	modelo_plano	10G	101987916
10G	*Startable*	10G	101987916

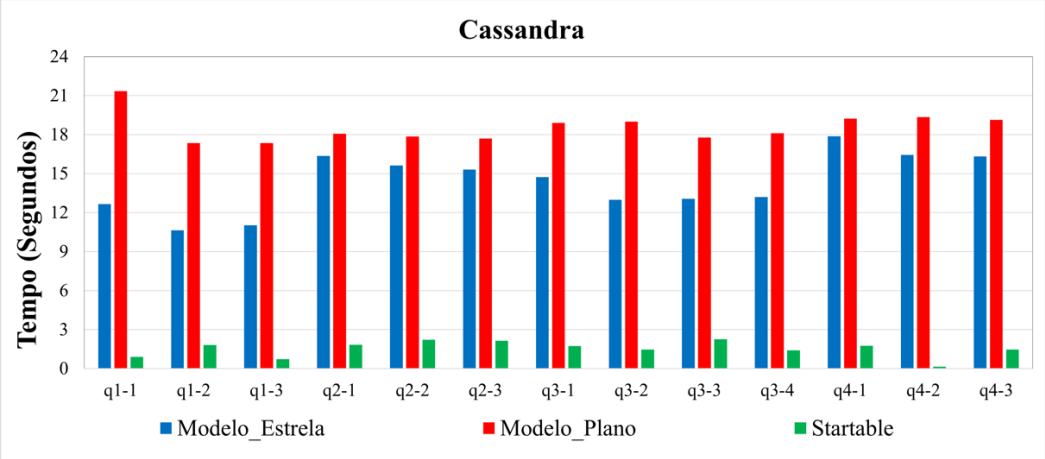
Fonte: Elaborado pelo Autor

Figura 29 - *Benchmark* de Consultas SSB para a *Startable* sobre a Carga de 1GB - *Hive*



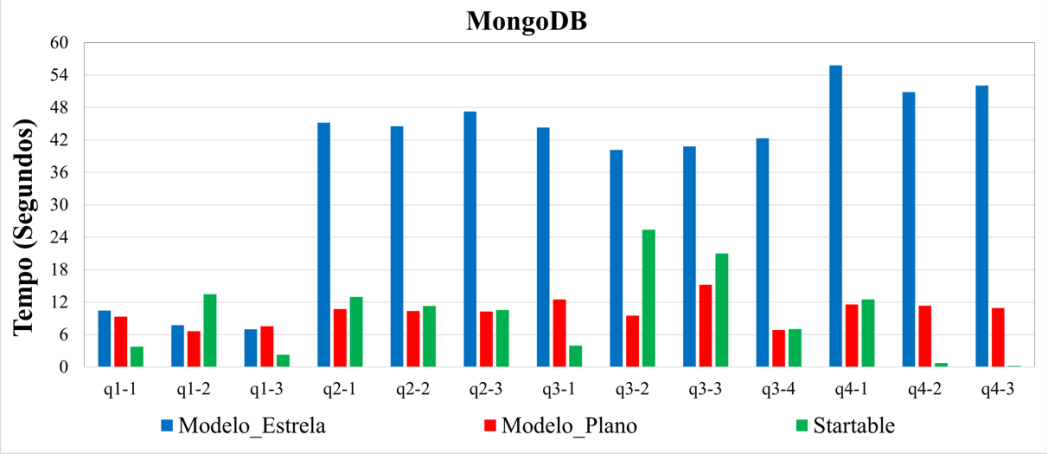
Fonte: Elaborado pelo Autor

Figura 30 - *Benchmark* de Consultas SSB para a *Startable* sobre a Carga de 1GB - Cassandra



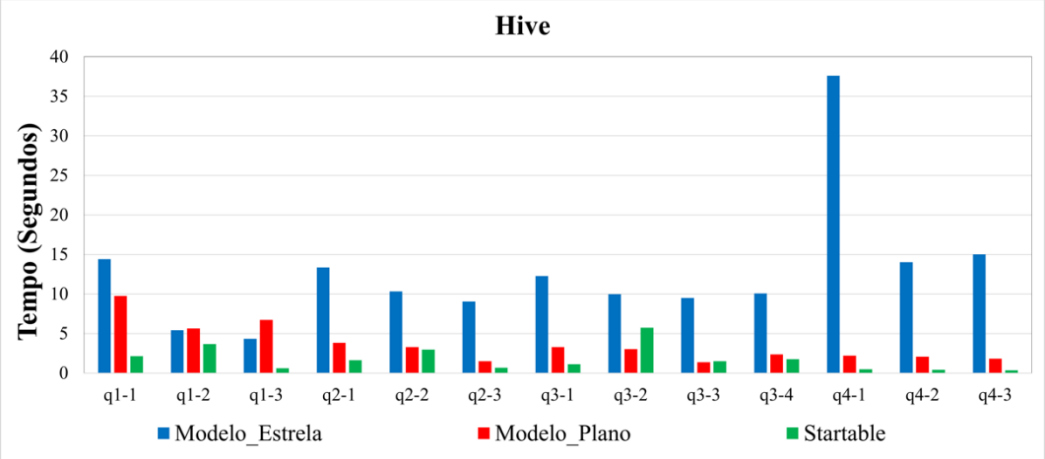
Fonte: Elaborado pelo Autor

Figura 31 - *Benchmark* de Consultas SSB para a *Startable* sobre a Carga de 1GB - *MongoDB*



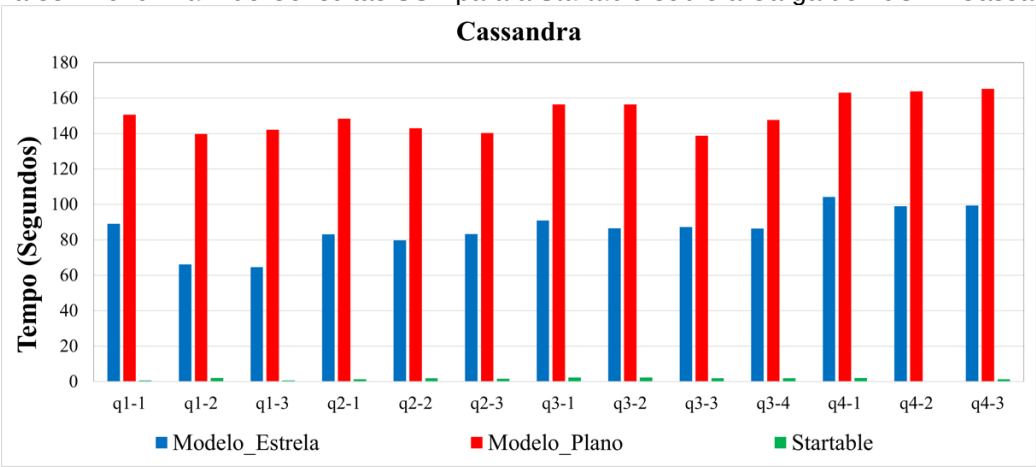
Fonte: Elaborado pelo Autor

Figura 32 - *Benchmark* de Consultas SSB para a *Startable* sobre a Carga de 10GB - *Hive*



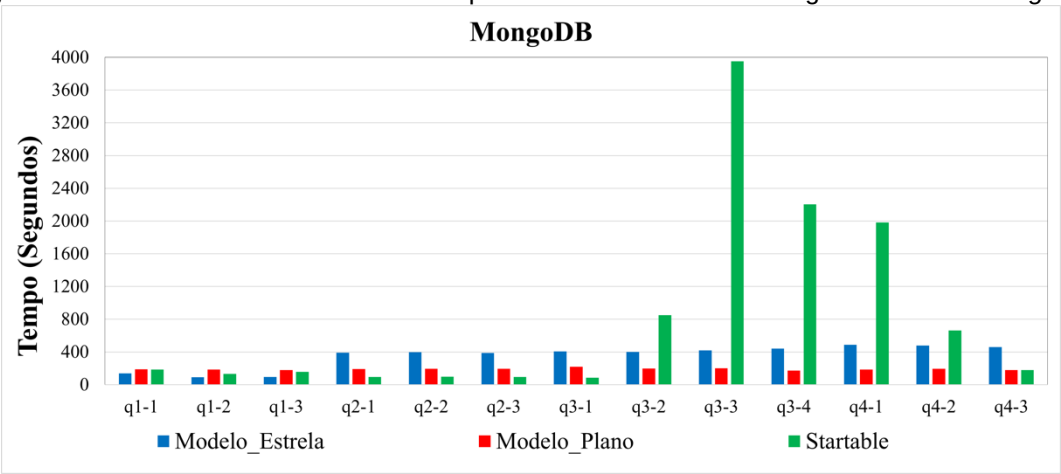
Fonte: Elaborado pelo Autor

Figura 33 - *Benchmark* de Consultas SSB para a *Startable* sobre a Carga de 10GB - Cassandra



Fonte: Elaborado pelo Autor

Figura 34 - *Benchmark* de Consultas SSB para a *Startable* sobre a Carga de 10GB – *MongoDB*



Fonte: Elaborado pelo Autor

Sobre o NoSQL *MongoDB* a *Startable* apresentou ganhos significativos de desempenho já que 6 das 13 consultas (q1-1, q1-3, q3-1, q3-4, q4-2, q4-3) das cargas de 1GB apresentaram desempenho superior ou igual as duas outras implementações. Em 3 consultas (q2-1, q2-2, q4-1) tiveram desempenho superior ao modelo_estrela e desempenho muito próximo ao modelo_plano. Já para a carga de 10GB sobre *MongoDB* vimos que a modelagem *Startable* desempenhou melhor em 5 consultas (q2-1, q2-2, q2-3, q3-1, q4-3) e demonstrou desempenho levemente inferior em 3 consultas (q1-1, q1-2, q1-3). Entretanto, tivemos um desempenho muito inferior em 4 consultas (q3-2, q3-3, q3-4, q4-1), com o desempenho chegando a ser 10 vezes mais lenta.

Considerando as duas cargas, observa-se que a proposta apresenta melhora significativa de desempenho no Cassandra e, aparentemente, para maiores cargas o desempenho tende a melhorar mais. Para ter certeza seria preciso testes com cargas maiores, mas os custos impediram a realização de maior número de teste, pois foram custeados pelo autor sem suporte de projetos de fomento ou de empresas. A implementação no *Hive* também mostrou melhora de desempenho com a *Startable*, também com aparente melhora para cargas maiores. No *MongoDB* a melhora não é tão evidente para a comparação entre o Modelo_Plano e *Startable*, havendo perda significativa de desempenho com a *Startable* para determinadas consultas (q3-2, q3-3, q3-4, q4-1, q4-2) para cargas maiores.

Há 4 possíveis fatores que podem explicar esse baixo desempenho para estas consultas: (i) essas consultas possuem filtros que em grande parte não se enquadram com a chave definida; (ii) na estrutura de *Shards* do *MongoDB* que a distribuição dos dados é automática, podendo não ter sido realizada da melhor forma possível, (iii) a forma de concepção do *MongoDB* está direcionada para retorno de poucos documentos por vez, sendo indicado para uso transacional, e assim há degradação de desempenho para consultas que exigem retorno de uma quantidade massiva de documentos (Mongo, 2023). Assim a adaptação para o uso analítico com uma grande massa de dados pode ter causado efeitos inversos ao se executar tais consultas; (iv) restrição da ferramenta auxiliar (PrestoDB), utilizada executar consultas SQL, quando há retorno de grandes volumes de dados.

O *PrestoDB* recupera apenas 4MB por cursor de *bach size* no *MongoDB* (PrestoDB, 2023), ou seja, para cada lote de leitura de dados junto ao *MongoDB*, o *PrestoDB* consegue recuperar apenas 4MB por vez, forçando assim a realizar diversos lotes de leitura, sobrecarregando a quantidade de I/O junto ao NoSQL, o que pode fazer com que as consultas que possuam filtros muito abrangentes tenham o desempenho depreciado. Isto não acontece no *drive* do *PrestoDB* para *Cassandra* e *HDFS*/*Hive*.

Mesmo assim, é possível afirmar que a proposta *Startable* atingiu resultados significativos e apresenta ser apropriada para os dois tipos de NoSQL testados e também para o ambiente *big data HDFS*/*Hive*.

5 CONCLUSÃO

NoSQL são amplamente utilizados em sistemas transacionais para persistência de dados devido à alta escalabilidade, disponibilidade e poder de processamento. Entretanto, ainda não é comum utilizar tais sistemas como base de dados para sistemas analíticos. Já sistemas de *Big Data* como o *Hadoop* e as ferramentas que compõe seu ecossistema, como HDFS e *Apache Hive*, são amplamente utilizados para a persistência de dados para sistemas analíticos.

Nesta pesquisa primeiramente foi realizada uma revisão sistemática de literatura para identificar propostas prévias de mapeamento e implementação de modelo multidimensional sobre NoSQL, tendo sido identificados 3 formas de mapeamento: (i) o modelo_estrela, que consiste implementação das tabelas fato e dimensões sobre NoSQL, principalmente sobre NoSQL orientados a grafos; (ii) o modelo_plano, que consiste na junção das tabelas fatos e dimensões (i) formando uma única Tabela ou Documento; (iii) o modelo_plano_agreagado, que consiste na projeção e agregação (ou agrupamentos) e sumarizações de dados pré-calculados sobre o resultado de (ii).

A partir de tais resultados foi possível desenhar a proposta *Startable*, que implementa o modelo_plano, porém considerando particionamento, *bucket* e ordenação para a distribuição dos dados. No contexto de banco de dados, regras e requisitos de negócio definem a estrutura da base e consultas esperadas, assim com base nessas regras, detalhou-se como partições e *buckets* devem ser utilizadas para a distribuição dos dados. Isto não foi observado em nenhum trabalho prévio, pelo menos até onde a RSL alcançou. Assim, *Startable* tem como fator diferencial ser baseado nas regras de negócio do sistema analítico a ser implementado.

A *Startable* atendeu aos requisitos iniciais estabelecidos de simplicidade, aplicabilidade a diferentes estruturas de NoSQL e a tecnologia de *Big Data*; além de apresentar maior eficiência para consultas em comparação à implementação tradicional (Modelo_Estrela, seção 3.2).

A proposta da *Startable*, um mapeamento focado na simplicidade, aplicável a diferentes estruturas de NoSQL e eficiente em consultas de dados, foi submetida a testes em diferentes ambientes de banco de dados e sob variadas cargas de dados. Um *benchmark* padrão foi utilizado para testar a proposta, sendo esse o único modelo de negócio testado. Não foi possível testar para diferentes modelos por razões econômicas. Porém, o detalhamento apresentado para a execução da *Startable* facilita sua replicabilidade para outros domínios de dados e negócios.

Os resultados dos testes no *benchmark* SSB demonstraram que a *Startable* atendeu eficazmente aos objetivos propostos. Em ambientes NoSQL como *MongoDB* e *Cassandra*, assim como no contexto do *Big Data* com *Hadoop/Hive*. Também, a Startable se mostrou mais eficiente em diversas consultas em comparação aos modelos tradicionais, especialmente sob cargas maiores de dados. Esses resultados indicam uma adaptação bem-sucedida do mapeamento do modelo multidimensional para sistemas NoSQL e Big Data, conforme as demandas modernas de análise de dados e toma de decisão.

Esta pesquisa atingiu seu objetivo principal, propondo e validando um mapeamento de dados eficiente para sistemas analíticos em ambientes NoSQL e *Big Data*. A *Startable* provou ser uma abordagem promissora, adaptável a diferentes estruturas de dados e eficiente em ambientes de dados de grande escala.

Como trabalho futuro sugere-se a execução de novos testes com a *Startable* sobre outras plataformas e NoSQL bem como outros volumes de dados. Nesse sentido, é importante que plataformas de persistência distribuídas de dados em *Cloud Computing*, sistemas de persistência de dados, NoSQL com *Azure CosmoDB* ou NoSQL *AWS DynamoDB*, os quais possuem estruturas internas híbridas que podem funcionar tanto como orientado a documentos como orientado a colunas sejam testados. Além disso, devem ser empregados outros *benchmarks* sejam realizados, principalmente os propostos para avaliação de banco de dados distribuídos e de DW.

6 REFERÊNCIAS

AALST, W. M. VAN DER. **On the Pareto Principle in Process Mining, Task Mining, and Robotic Process Automation.**DATA. **Anais**, 2020.

AGRAWAL, R.; GUPTA, A.; SARAWAGI, S. **Modeling multidimensional databases.** Proceedings 13th International Conference on Data Engineering. **Anais**...Birmingham, UK: IEEE Comput. Soc. Press, 1997. Disponível em: <http://ieeexplore.ieee.org/document/581777/>.

AKID, H.; FREY, G.; AYED, M. B.; LACHICHE, N. **Performance of NoSQL Graph Implementations of Star vs. Snowflake Schemas**. v. 10, p. 48603–48614, 2022.

AKOKA, J.; COMYN-WATTIAU, I.; MOUZA, C. DU; PRAT, N. Mapping Multidimensional Schemas to Property Graph Models. v. 13012 LNCS, p. 3–14, 2021.

AMIRTHALINGAM, T.; RAIS. H. **Automated Table Partitioner (ATAP) in Apache Hive**, 2018 4th International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 2018, pp. 1-6, doi: 10.1109/ICCOINS.2018.8510580.

BANERJEE, S.; BHASKAR, S.; SARKAR, A.; DEBNATH, N. C. A unified conceptual model for data warehouses. v. 5, n. Special issue 5, p. 162–169, 2021.

BILAL, M.; OYEDELE, L. O.; QADIR, J.; MUNIR, K.; AJAYI, S. O.; AKINADE, O. O.; OWOLABI, H. A.; ALAKA, H. A.; PASHA, M. Big Data in the construction industry: A review of present status, opportunities, and future trends. **Advanced Engineering Informatics**, v. 30, n. 3, p. 500–521, ago. 2016.

BONNET, L.; LAURENT, A.; SALA, M.; LAURENT, B.; SICARD, N. **Reduce, you say: What NoSQL can do for data aggregation and BI in large repositories,** 2011a.

___. **Reduce, you say: What NoSQL can do for data aggregation and BI in large repositories,** 2011b.

BOUAZIZ, S.; NABLI, A.; GARGOURI, F. **Design a data warehouse schema from document-oriented database,** 2019.

BOUSSAHOUA, M.; BOUSSAID, O.; BENTAYEB, F. **Logical schema for data warehouse on column-oriented NoSQL databases.** In Database and Expert Systems Applications: 28th International Conference, DEXA 2017, Lyon, France, August 28-31, 2017, Proceedings, Part II 28, pp. 247-256. Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-64471-4_20.

BUSE, R. P. L.; ZIMMERMANN, T. **Information needs for software development analytics,** 2012. 34th International Conference on Software Engineering (ICSE). **Anais**: Zurich: IEEE, jun. 2012. Disponível em: <http://ieeexplore.ieee.org/document/6227122/>. Acesso em: 7 nov. 2021.

CARNIEL, A. C.; AGUIAR SA, A. DE; BRISIGHELLO, V. H. P.; RIBEIRO, M. X.; BUENO, R.; CIFERRI, R. R.; AGUIAR CIFERRI, C. D. DE. **Query processing over data warehouse using relational databases and NoSQL**2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI). **Anais**: IEEE, out. 2012a. Disponível em: <http://ieeexplore.ieee.org/document/6427228/>.

___. **Query processing over data warehouse using relational databases and NoSQL**2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI). **Anais**... *Em*: 2012 XXXVIII CONFERENCIA LATINOAMERICANA EN INFORMATICA (CLEI). Medellin, Colombia: IEEE, out. 2012b. Disponível em: <http://ieeexplore.ieee.org/document/6427228/>. Acesso em: 11 maio. 2020.

CASSANDRA APACHE, 2023. Cassandra Documentation. Disponível em: https://cassandra.apache.org/doc/3.11/index.html [acessado em 31 janeiro 2023].

CASTELLTORT, A.; LAURENT, A. **NoSQL Graph-based OLAP Analysis.** Proceedings of the International Conference on Knowledge Discovery and Information Retrieval. **Anais:** SCITEPRESS - Science and and Technology Publications, 2014a. Disponível em: <http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005072902170224>.

___. **NoSQL Graph-based OLAP Analysis.** Proceedings of the International Conference on Knowledge Discovery and Information Retrieval. **Anais**... *Em*: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND INFORMATION RETRIEVAL. Rome, Italy: SCITEPRESS - Science and and Technology Publications, 2014b. Disponível em: <http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005072902170224>. Acesso em: 7 maio. 2020.

CHALLAL, Z.; BALA, W.; MOKEDDEM, H.; BOUKHALFA, K.; BOUSSAID, O.; BENKHELIFA, E. **Document-oriented versus Column-oriented Data Storage for Social Graph Data Warehouse**. [s.l: s.n.], 2019.

CHANG, F.; DEAN, J.; GHEMAWAT, S.; HSIEH, W. C.; WALLACH, D. A.; BURROWS, M.; CHANDRA, T.; FIKES, A.; GRUBER, R. E. Bigtable: A Distributed Storage System for Structured Data. **ACM Transactions on Computer Systems**, v. 26, n. 2, p. 1–26, jun. 2008.

CHAUDHURI, S.; DAYAL, U. An overview of data warehousing and OLAP technology. **ACM SIGMOD Record**, v. 26, n. 1, p. 65–74, mar. 1997a.

___. An overview of data warehousing and OLAP technology. **ACM SIGMOD Record**, v. 26, n. 1, p. 65–74, mar. 1997b.

CHEBOTKO, A.; KASHLEV, A.; LU, S. **A Big Data Modeling Methodology for Apache Cassandra,** 2015 IEEE International Congress on Big Data. **Anais**. jun. 2015.

CHEN, M.; MAO, S.; LIU, Y. Big Data: A Survey. **Mobile Networks and Applications**, v. 19, n. 2, p. 171–209, abr. 2014.

CHEVALIER, MAX; EL MALKI, M.; KOPLIKU, A.; TESTE, O.; TOURNIER, R. **How Can We Implement a Multidimensional Data Warehouse Using NoSQL?** (S. Hammoudi, L. Maciaszek, E. Teniente, O. Camp, & J. Cordeiro, Eds.): Lecture Notes in Business Information Processing.Inst Syst & Technol Informat Control & Commun; Assoc Advancement Artificial Intelligence; IEICE Special Interest Grp Software Enterprise Modelling; ACM Special Interest Grp Management Informat Syst; ACM Special Interest Grp Artificial Intelligence; Asociac Espanola Inteligencia Artificial; Informat Res Ctr, 2015a.

CHEVALIER, M.; EL MALKI, M.; KOPLIKU, A.; TESTE, O.; TOURNIER, R. **Implementation of Multidimensional Databases with Document-Oriented NoSQL** (S. Madria & T. Hara, Eds.): Lecture Notes in Computer Science.2015b.

CHEVALIER, MAX; EL MALKI, M.; KOPLIKU, A.; TESTE, O.; TOURNIER, R. **Benchmark for OLAP on NoSQL Technologies Comparing NoSQL Multidimensional Data Warehousing Solutions** (C. Rolland, D. Anagnostopoulos, P. Loucopoulos, & C. GonzalezPerez, Eds.): International Conference on Research Challenges in Information Science.2015c. Disponível em: <https://ieeexplore.ieee.org/document/7128909>.

CHEVALIER, M.; EL MALKI, M.; KOPLIKU, A.; TESTE, O.; TOURNIER, R. Implementation of Multidimensional Databases with Document-Oriented NoSQL. *Em*: MADRIA, S.; HARA, T. (Eds.). **Big Data Analytics and Knowledge Discovery**. Lecture Notes in Computer Science. [s.l.] Springer International Publishing, 2015d. v. 9263p. 379–390.

___. **Implementation of Multidimensional Databases with Document-Oriented NoSQL** (S. Madria & T. Hara, Eds.): Lecture Notes in Computer Science, 2015e.

___. **Implementing multidimensional data warehouses into NoSQL** (T. E. Maciaszek L. Hammoudi S. ,. Maciaszek L., Ed.), 2015f. Disponível em: <https://www.scitepress.org/Link.aspx?doi=10.5220/0005379801720183>.

CHEVALIER, M.; EL MALKI, M.; KOPLIKU, A.; TESTE, O.; TOURNIER, R. **Document-oriented data warehouses: Models and extended cuboids, extended cuboids in oriented document,** 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS). **Anais**: IEEE, jun. 2016. Disponível em: <http://ieeexplore.ieee.org/document/7549351/>.

___. Document-Oriented Data Warehouses: Complex Hierarchies and Summarizability. *Em*: EL-AZOUZI, R.; MENASCHE, D. S.; SABIR, E.; DE PELLEGRINI, F.; BENJILLALI, M. (Eds.). . **Advances in Ubiquitous Networking 2**. Lecture Notes in Electrical Engineering. Singapore: Springer Singapore, 2017. v. 397p. 671–683.

CHOUDER, M. L.; RIZZI, S.; CHALAL, R. EXODuS: Exploratory OLAP over Document Stores. v. 79, n. SI, p. 44–57, jan. 2019.

COLELLA, A. A. C. **Performances of OLAP Operations in Graph and Relational Databases**. [s.l: s.n.], 2019.

CUGNASCO, C.; BECERRA, Y.; TORRES, J.; AYGUADÉ, E. **D8-tree: a de-normalized approach for multidimensional data analysis on key-value databases**Proceedings of the 17th International Conference on Distributed Computing and Networking - ICDCN '16. **Anais**: ACM Press, 2016. Disponível em: <http://dl.acm.org/citation.cfm?doid=2833312.2833314>.

CUGNASCO, C.; BECERRA, Y.; TORRES, J.; AYGUADÉ, E. E**xploiting key-value data stores scalability for HPC**. In 2017 46th International Conference on Parallel Processing Workshops (ICPPW) 2017 Aug 14 (pp. 85-94). IEEE.

DAI, J. **SQL to NoSQL : What to do and How,** 2019.

DAVARDOOST, F.; BABAZADEH SANGAR, A.; MAJIDZADEH, K. **Extracting OLAP Cubes from Document-Oriented NoSQL Database Based on Parallel Similarity Algorithms**. v. 43, n. 2, p. 111–118, 2020.

DAVARDOOST, F.; SANGAR, A. B.; MAJIDZADEH, K. **An Innovative Model for Extracting OLAP Cubes from NOSQL Database Based on Scalable Naive Bayes Classifier**ADAM HOUSE, 3RD FLR, 1 FITZROY SQ, LONDON, W1T 5HF, ENGLANDHINDAWI LTD, , 11 abr. 2022.

DE MOURA, E. S. Text Indexing Techniques. *Em*: LIU, L.; ÖZSU, M. T. (Eds.). **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 3058–3061.

DEAN, J.; GHEMAWAT, S. MapReduce: simplified data processing on large clusters. **Communications of the ACM**, v. 51, n. 1, p. 107–113, jan. 2008.

DEHDOUH, K. **Building OLAP cubes from columnar NoSQL data warehouses.** In Model and Data Engineering: 6th International Conference, MEDI 2016, Almería, Spain, September 21-23, 2016, Proceedings 6, pp. 166-179. Springer International Publishing, 2016.

DEHDOUH, K.; BENTAYEB, F.; BOUSSAID, O.; KABACHI, N. **Columnar NoSQL CUBE: Agregation operator for columnar NoSQL data warehouse,** 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC). **Anais:** IEEE, out. 2014a. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6974527>.

___. **Columnar NoSQL CUBE: Agregation operator for columnar NoSQL data warehouse**2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC). **Anais**: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS - SMC. San Diego, CA, USA: IEEE, out. 2014b. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6974527>. Acesso em: 11 maio. 2020.

DEHDOUH, K.; BOUSSAID, O.; BENTAYEB, F. Big Data warehouse: Building columnar NoSQL OLAP cubes. v. 12, n. 1, p. 1–24, 2020.

DEVLIN, B. A.; MURPHY, P. T. An architecture for a business and information system. **IBM Systems Journal**, v. 27, n. 1, p. 60–80, 1988.

EL MOUKHI, N.; EL AZAMI, I.; HAJBI, S. Towards a new hybrid approach for building document-oriented data warehouses. v. 12, n. 6, p. 6423–6431, 2022.

ELMASRI, R.; NAVATHE, S. **Fundamentals of database systems**. Seventh edition ed. Hoboken, NJ: Pearson, 2016.

FERRAHI, I.; BIMONTE, S.; KANG, M.-A.; BOUKHALFA, K. **Design and implementation of falling star a non-redudant spatio-multidimensional logical model for document stores,** 2017. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85023196546&partnerID=40&md5=beda87991d007daaaf0e3daa03cae41a>.

FERREIRA, L. M. Modelo de Processo para Criação de BI em Banco de Dados NoSQL Orientado a Colunas. In Conferencia Ibero Americana WWW/Internet-CIAWI, vol. 1. 2015.

FERREIRA, L. M.; ALVES-SOUZA, S. N.; DA SILVA, L. M. **Multidimensional Modelling in NoSQL Database: A Systematic Review.** 2023 18th Iberian Conference on Information Systems and Technologies (CISTI), Aveiro, Portugal, 2023, pp. 1-6, doi: 10.23919/CISTI58278.2023.10211592.

FERREIRA, L. M.; ALVES-SOUZA, S. N.; DA SILVA, L. M. DA. **Startable: Multidimensional Modelling for Column-Oriented NoSQL. In** Proceedings of the 11th International Conference on Data Science, Technology and Applications - DATA; ISBN 978-989-758-583-8; ISSN 2184-285X, SciTePress, pages 21-30. DOI: 10.5220/0011142100003269, 2022.

FONSECA, R.; DE CARVALHO VICTORINO, M.; HOLANDA, M. **ROLAP DW transformation proposal for OLAP architecture in NoSQL database**Association for Computing Machinery, 2020.

GANDOMI, A.; HAIDER, M. Beyond the hype: Big data concepts, methods, and analytics. **International Journal of Information Management**, v. 35, n. 2, p. 137–144, abr. 2015.

GHEMAWAT, S.; GOBIOFF, H.; LEUNG, S.-T. **The Google file system.** Proceedings of the nineteenth ACM symposium on Operating systems principles - SOSP '03. **Anais:** Bolton Landing, NY, USA: ACM Press, 2003. Disponível em: <http://portal.acm.org/citation.cfm?doid=945445.945450>. Acesso em: 7 nov. 2021.

GILBERT, S. AND LYNCH, N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. **ACM SIGACT News**, 33, 51-59, 2002.

GRAY, J.; CHAUDHURI, S.; BOSWORTH, A.; LAYMAN, A.; REICHART, D.; VENKATRAO, M.; PELLOW, F.; PIRAHESH, H. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data mining and knowledge discovery. **Data Mining and Knowledge Discovery**, v. 1, n. 1, p. 29–53, 1997.

HASHEM, I. A. T.; YAQOOB, I.; ANUAR, N. B.; MOKHTAR, S.; GANI, A.; ULLAH KHAN, S. The rise of "big data" on cloud computing: Review and open research issues. **Information Systems**, v. 47, p. 98–115, jan. 2015.

HEUSER, C. A. **Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS**. [s.l.] Bookman Editora, 2009.

HIVE, APACHE, 2023. Apache Hive Documentation. Disponível em: https://cwiki.apache.org/confluence/display/HIVE [acessado em 31 janeiro 2023].

INMON, W. H. **Building the Data Warehouse,3rd Edition**. 3rd. ed. [s.l.] John Wiley & Sons, Inc., 2002. INMON, W. H. **Data architecture: a primer for the data scientist**. 1st edition ed. [s.l.] Elsevier, 2014.

INMON, W. H.; LINSTEDT, D.; LEVINS, M. **Data Architecture: A Primer for the Data Scientist: A Primer for the Data Scientist**. Academic Press, 2019. ISBN: 0128169168.

INMON, W. H.; HACKATHORN, R. D. **Using the Data Warehouse**. USA: Wiley-QED Publishing, 1994.

JIANMIN, W.; WENBIN, Z.; TONGRANG, F.; SHILONG, Y.; HONGWEI, L. An improved join-free snowflake schema for ETL and OLAP of data warehouse. [s.d.], 2011.

KATAL, A.; WAZID, M.; GOUDAR, R. H. **Big data: Issues, challenges, tools and Good practices,** 2013. Sixth International Conference on Contemporary Computing (IC3). **Anais**: Noida, India: IEEE, ago. 2013. Disponível em: <http://ieeexplore.ieee.org/document/6612229/>. Acesso em: 7 nov. 2021.

KHALIL, A.; BELAISSAOUI, M. **New approach for implementing big datamart using NoSQL key-value stores** (O. A. Essaaidi M. Zbakh M., Ed.)Institute of Electrical and Electronics Engineers Inc., 2020.

KHALIL, A.; BELAISSAOUI, M. **A Graph-oriented Framework for Online Analytical Processing**19 BOLLING RD, BRADFORD, WEST YORKSHIRE, 00000, ENGLANDSCIENCE & INFORMATION SAI ORGANIZATION LTD, maio 2022.

KHALIL, ABDELHAK; BELAISSAOUI, MUSTAPHA; KHALIL, A.; BELAISSAOUI, M. Key-value data warehouse: Models and OLAP analysis. *Em*: HASSANIEN, A. E.; DARWISH, A.; ABD EL-KADER, S. M.; ALBOANEEN, D. A. (Eds.). . Singapore: Institute of Electrical and Electronics Engineers Inc., 2020. p. 179–191.

KIMBALL, R.; ROSS, M. **The data warehouse toolkit: the complete guide to dimensional modeling**. 2nd ed ed. [s.l.] Wiley, 2002.

KIMBALL, R.; ROSS, M. **The data warehouse toolkit: the definitive guide to dimensional modeling.** 3rd ed. John Wiley & Sons, 2013.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [s.l.] Keele University and Durham University Joint Report, 2007.

LANEY, D.; OTHERS. 3D data management: Controlling data volume, velocity and variety. **META group research note**, v. 6, n. 70, p. 1, 2001.

LIGHTSTONE, S.; TEOREY, T.; NADEAU, T. Basic indexing methods. *Em*: **Physical Database Design**. [s.l.] Elsevier, 2007. p. 15–29.

LIU, Y.; VITOLO, T. M. **Graph Data Warehouse: Steps to Integrating Graph Databases into the Traditional Conceptual Structure of a Data Warehouse**: IEEE International Congress on Big Data.IEEE; IEEE Comp Soc, 2013.

LÓSCIO, B. F.; OLIVEIRA, H. DE; PONTES, J. DE S. NoSQL no desenvolvimento de aplicações Web colaborativas. **VIII Simpósio Brasileiro de Sistemas Colaborativos**, v. 10, n. 1, p. 11, 2011.

LUKASZUK, J.; ORLANDIC, R. **Efficient high-dimensional indexing by superimposing space-partitioning schemes.** Proceedings. International Database Engineering and Applications Symposium, 2004. IDEAS '04, 2004.

MADDEN, S. From Databases to Big Data. **IEEE Internet Computing**, v. 16, n. 3, p. 4–6, may 2012.

MAGHFIROH, L. R.; NUGRAHA, I. G. B. B. **Survey data and metadata modelling using document-oriented NoSQL**. [s.l.] Telkom Univ, Sch Comp; Telkom Univ, Sch Comp, Fackultas Informatika; Indonesian Assoc Computat Linguist; Indonesia Data Scientist Soc; Telkomsigma; Yayasan Pendidikan Telkom, 2018. v. 971.

MESSAOUD, I. B.; ALI, R. B.; FEKI, J. **From document warehouse to column-oriented NoSQL document warehouse,** 2017. Disponível em:

<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85029302678&partnerID=40&md5=f0d8397880de4f016d1cc698e3353f14>.

MONGODB, ATLAS, 2023. MongoDB Database Manual. Disponível em: https://www.mongodb.com/docs/manual/ [acessado em 31 janeiro 2023].

MONIRUZZAMAN, A. B. M.; HOSSAIN, S. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. **Int J Database Theor Appl**, v. 6, 2013.

MOUSA, A. H.; SHIRATUDDIN, N. **Data Warehouse and Data Virtualization Comparative Study,** 2015. International Conference on Developments of E-Systems Engineering (DeSE). **Anais:** Duai, United Arab Emirates: IEEE, dez. 2015. Disponível em: <http://ieeexplore.ieee.org/document/7563665/>. Acesso em: 7 nov. 2021.

MURAZZA, M. R.; NURWIDYANTORO, A. **Cassandra and SQL database comparison for near real-time Twitter data warehouse,** 2017.

NEBOT, V.; BERLANGA, R. Building data warehouses with semantic web data. **Decision Support Systems**, 52, no. 4 (2012): 853-868. https://doi.org/10.1016/j.dss.2011.11.009.

OLIVEIRA, B. F. P. DE; VICTORINO, M. DE C.; HOLANDA, M. **Data Warehouse Based on NoSQL: a literature mapping**. [s.l: s.n.], 2021.

O'NEIL, P.; O'NEIL, B.; CHEN, X. The Star Schema Benchmark (SSB). Revision 3, June 5, 2009. Disponível em: < https://www.cs.umb.edu/~poneil/StarSchemaB.PDF>.

PASQUALIN, D.; SOUZA, G.; BURATTI, E. L.; DE ALMEIDA, E. C.; DEL FABRO, M. D.; WEINGAERTNER, D. **A case study of the aggregation query model in read-mostly NoSQL document stores** (D. E. Desai B.C., Ed.), 2016a.

___. **A case study of the aggregation query model in read-mostly NoSQL document stores** (D. E. Desai B.C., Ed.), 2016b.

PEREIRA, D.; OLIVEIRA, P.; RODRIGUES, F. **Data warehouses in MongoDB vs SQL Server: A comparative analysis of the querie performance,** 2015a.

___. **Data warehouses in MongoDB vs SQL Server: A comparative analysis of the querie performance,** 2015b.

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. 2019. Disponível em: <https://dl.acm.org/doi/10.5220/0007748304520458>.

PRANDO, A. V.; CONTRATRES, F. G.; SOUZA, S. N. A.; SOUZA, L. S. DE. **Content-based Recommender System using Social Networks for Cold-start Users.** Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management. **Anais** SCITEPRESS - Science and Technology Publications, 2017. Disponível em: <http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006496301810189>.

PRESTODB, FACEBOOK, 2023. PrestoDB Documentation. Disponível em: https://prestodb.io/docs/current [acessado em 31 Janeiro 2023].

RAVAT, F.; SONG, J.; TESTE, O.; TROJAHN, C. **Improving the performance of querying multidimensional RDF data using aggregates**. [s.l.] Assoc Comp Machinery Special Interest Grp Appl Comp, 2275-2284, 2019.

RIZZI, S. OLAP and NoSQL: Happily Ever After. v. 13389 LNCS, p. 35–44, 2022.

ROBINSON, I.; WEBBER, J.; EIFREM, E. **Graph databases**. Second edition ed. Beijing: O'Reilly, 2015.

ROMERO, O.; HERRERO, V.; ABELLÓ, A.; FERRARONS, J. Tuning small analytics on Big Data: Data partitioning and secondary indexes in the Hadoop ecosystem. **Information Systems**, v. 54, p. 336–356, dez. 2015.

SAKR, S. NoSQL Database Systems. *Em*: SAKR, S.; ZOMAYA, A. (Eds.). **Encyclopedia of Big Data Technologies**. Cham: Springer International Publishing, 2018. p. 1–6.

___. NoSQL Database Systems. *Em*: SAKR, S.; ZOMAYA, A. Y. (Eds.). **Encyclopedia of Big Data Technologies**. Cham: Springer International Publishing, 2019. p. 1193–1198.

SAMET, H. Decoupling partitioning and grouping: Overcoming shortcomings of spatial indexing with bucketing. **ACM Transactions on Database Systems**, v. 29, n. 4, p. 789–830, 12 dez. 2004.

SANTOS, M.; COSTA, C. **Data warehousing in big data: from multidimensional to tabular data models**. In Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering, pp. 51-60. 2016. https://doi.org/10.1145/2948992.2949024.

SARR, J. G.; BAME, N.; BOLY, A. **Generic model for multidimensional data stream summary**. [s.l: s.n.], 2022.

SCABORA, L. C.; BRITO, J. J.; CIFERRI, R. R.; AGUIAR CIFERRI, C. D. DE. **Physical Data Warehouse Design on NoSQL Databases OLAP Query Processing over HBase** (S. Hammoudi, L. Maciaszek, M. Missikoff, O. Camp, & J. Cordeiro, Eds.), 2016a.

___. **Physical Data Warehouse Design on NoSQL Databases OLAP Query Processing over HBase** (S. Hammoudi, L. Maciaszek, M. Missikoff, O. Camp, & J. Cordeiro, Eds.), 2016b.

SELLAMI, AMAL; NABLI, A.; GARGOURI, F. **Graph NoSQL Data Warehouse Creation**. New York, NY, USA: Association for Computing Machinery, 2020a.

SELLAMI, A.; NABLI, A.; GARGOURI, F. **Transformation of Data Warehouse Schema to NoSQL Graph Data Base**. v. 941, p. 410–420, 2020b.

SH, H.; AZEZ, A.; KHAFAGY, M. H.; OMARA, F. A. JOUM. An Indexing Methodology for Improving Join in Hive Star schema. **International journal of scientific and engineering research**, v. 6, 3, p. 111-119, 2015.

SHAMSI, J. A.; KHOJAYE, M. A. NoSQL Systems. *Em*: SHAMSI, J. A.; KHOJAYE, M. A. (Eds.). **Big Data Systems**. 1. ed. [s.l.] Chapman and Hall/CRC, 2021. p. 143–170.

SHARMA, R.; REYNOLDS, P.; SCHEEPERS, R.; SEDDON, P. B.; SHANKS, G. G. **Business Analytics and Competitive Advantage: A Review and a Research Agenda.** 2010.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database system concepts**. 6th ed. New York: McGraw-Hill, 2011.

SILBERSCHATZ, A.; SUNDARSHAN, S.; KORTH, H. F. **Sistema de Banco de Dados**. [s.l.] Elsevier, 2016.

SIM, S. E.; EASTERBROOK, S.; HOLT, R. C. **Using benchmarking to advance research: a challenge to software engineering.** 25th International Conference on Software Engineering, 2003. Proceedings. **Anais**...Portland, OR, USA: IEEE, 2003Disponível em: <http://ieeexplore.ieee.org/document/1201189/>. Acesso em: 7 nov. 2021.

STANTIC, N. F. R. **Answering Temporal Analytic Queries over Big Data Based on Precomputing Architecture**. [s.l: s.n.], 2017a.

___. **Answering Temporal Analytic Queries over Big Data Based on Precomputing Architecture**. [s.l: s.n.], 2017b.

___. Precomputing architecture for flexible and efficient big data analytics. 2018.

THUSOO, A.; SARMA, J. S.; JAIN, N.; SHAO, Z.; CHAKKA, P.; ZHANG, N.; ANTONY, S.; LIU, H.; Wyckoff, P.; MURTHY, R., 2009. **Hive: a warehousing solution over a map-reduce framework.** Proc. VLDB Endow. 2, 2 (August 2009), 1626–1629, 2009. https://doi.org/10.14778/1687553.1687609.

THUSOO, A.; SARMA, J. S.; JAIN, N.; SHAO, Z.; CHAKKA, P.; ZHANG, N.; ANTONY, S.; LIU, H.; MURTHY, R. **Hive - a petabyte scale data warehouse using Hadoop,** 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010). **Anais**: IEEE 26TH INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE 2010). Long Beach, CA, USA: IEEE, 2010Disponível em: <http://ieeexplore.ieee.org/document/5447738/>. Acesso em: 17 set. 2020.

THUSOO, A.; SHAO, Z.; ANTHONY, S.; BORTHAKUR, D.; JAIN, N.; SEN SARMA, J.; MURTHY, R.; LIU, H. **Data Warehousing and Analytics Infrastructure at Facebook.** Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. **Anais**: SIGMOD 10.ACM, 2010. Disponível em: <http://doi.acm.org/10.1145/1807167.1807278>.

TIAN, Y.; CAREY, M.; MAXON, I. **Benchmarking HOAP for Scalable Document Data Management: A First Step** (X. Wu et al., Eds.): IEEE International Conference on Big Data.345 E 47TH ST, NEW YORK, NY 10017 USA IEEE , 2020.

TOURNIER, M. C. E. M. K. T. **Document-Oriented Data Warehouses: Complex Hierarchies and Summarizability**. [s.l.] Springer Singapore, 2017.

WANG, S.; PANDIS, I.; WU, C.; HE, S.; JOHNSON, D.; EMAM, I.; GUITTON, F.; GUO, Y. High dimensional biological data retrieval optimization with NoSQL technology. **BMC Genomics**, v. 15, n. Suppl 8, p. S3, 2014.

VASILAKIS, N.; YASH, P.; SMITH, J.M. **Query-efficient partitions for dynamic data.** In Proceedings of the 8th Asia-Pacific Workshop on Systems, pp. 1-8. 2017. https://doi.org/10.1145/3124680.3124744.

WANG, Y.; WICKBERG, H.; DORLING, A.; KAARTINEN, M. **Establishment of a national benchmark of software engineering practices**Proceedings 4th IEEE International Software Engineering Standards Symposium and Forum (ISESS'99). "Best Software Practices for the Internet Age". **Anais**...Curitiba, Brazil: IEEE Comput. Soc, 1999. Disponível em: <http://ieeexplore.ieee.org/document/766574/>. Acesso em: 7 nov. 2021.

WHITE, T. **Hadoop: the definitive guide**. Third edition ed. Beijing: O'Reilly, 2012.

YANGUI, R.; NABLI, A.; GARGOURI, F. **Automatic Transformation of Data Warehouse Schema To NoSQL Data Base: Comparative Study** (R. Howlett, L. Jain, B. Gabrys, C. Toro, & C. Lim, Eds.): Procedia Computer Science.KES Int, 2016a.

___. **Automatic Transformation of Data Warehouse Schema To NoSQL Data Base: Comparative Study** (R. Howlett, L. Jain, B. Gabrys, C. Toro, & C. Lim, Eds.): Procedia Computer Science.KES Int, 2016b.

YESSAD, L.; LABIOD, A. **Comparative study of data warehouses modeling approaches: Inmon, Kimball and Data Vault,** 2016 International Conference on System Reliability and Science (ICSRS), Paris, France, 2016, pp. 95-99. **Anais** IEEE, nov. 2016. Disponível em: <http://ieeexplore.ieee.org/document/7815845/>, doi: 10.1109/ICSRS.2016.7815845.

ZHAO, Y., SANDARA, M., HUANG, S., SADEK, A.W., GEORGE, T. AND HUTCHINS, A.M., 2011. **Intelligent Transportation Systems Data Warehouses and Their Applications**. In ICEIS (1) (pp. 343-347), 2011.

ZHAO, H.; YE, X. A Practice of TPC-DS Multidimensional Implementation on NoSQL Database Systems. *Em*: NAMBIAR, R.; POESS, M. (Eds.). **Performance Characterization and Benchmarking**. Lecture Notes in Computer Science. [s.l.] Springer International Publishing, 2014a. v. 8391p. 93–108.

___. A Practice of TPC-DS Multidimensional Implementation on NoSQL Database Systems. *Em*: NAMBIAR, R.; POESS, M. (Eds.). **Performance Characterization and Benchmarking**. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014b. v. 8391p. 93–108.

7 APÊNDICE

7.1 Consultas Disponíveis no *Star Schema Benchmark* (SSB)

q1-1

```
select sum(v_revenue) as revenue
from p_lineorder
left join dates on lo_orderdate = d_datekey
where d_year = 1993
and lo_discount between 1 and 3
and lo_quantity < 25;
```

q1-2

```
select sum(v_revenue) as revenue
from p_lineorder
left join dates on lo_orderdate = d_datekey
where d_yearmonthnum = 199401
and lo_discount between 4 and 6
and lo_quantity between 26 and 35;
```

q1-3

```
select sum(v_revenue) as revenue
from p_lineorder
left join dates on lo_orderdate = d_datekey
where d_weeknuminyear = 6 and d_year = 1994
and lo_discount between 5 and 7
and lo_quantity between 26 and 35;
```

q2-1

```
select sum(lo_revenue) as lo_revenue, d_year, p_brand
from p_lineorder
left join dates on lo_orderdate = d_datekey
left join part on lo_partkey = p_partkey
```

```
left join supplier on lo_suppkey = s_suppkey
where p_category = 'MFGR#12' and s_region = 'AMERICA'
group by d_year, p_brand
order by d_year, p_brand;
```

q2-2

```
select sum(lo_revenue) as lo_revenue, d_year, p_brand
from p_lineorder
left join dates on lo_orderdate = d_datekey
left join part on lo_partkey = p_partkey
left join supplier on lo_suppkey = s_suppkey
where p_brand between 'MFGR#2221' and 'MFGR#2228' and s_region = 'ASIA'
group by d_year, p_brand
order by d_year, p_brand;
```

q2-3

```
select sum(lo_revenue) as lo_revenue, d_year, p_brand
from p_lineorder
left join dates on lo_orderdate = d_datekey
left join part on lo_partkey = p_partkey
left join supplier on lo_suppkey = s_suppkey
where p_brand = 'MFGR#2239' and s_region = 'EUROPE'
group by d_year, p_brand
order by d_year, p_brand;
```

q3-1

```
select c_nation, s_nation, d_year, sum(lo_revenue) as lo_revenue
from p_lineorder
left join dates on lo_orderdate = d_datekey
left join customer on lo_custkey = c_custkey
left join supplier on lo_suppkey = s_suppkey
where c_region = 'ASIA' and s_region = 'ASIA'and d_year >= 1992 and d_year
<= 1997
group by c_nation, s_nation, d_year
order by d_year asc, lo_revenue desc;
```

q3-2

```
select c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
from p_lineorder
left join dates on lo_orderdate = d_datekey
left join customer on lo_custkey = c_custkey
left join supplier on lo_suppkey = s_suppkey
where c_nation = 'UNITED STATES' and s_nation = 'UNITED STATES'
and d_year >= 1992 and d_year <= 1997
group by c_city, s_city, d_year
order by d_year asc, lo_revenue desc;
```

q3-3

```
select c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
from p_lineorder
left join dates on lo_orderdate = d_datekey
left join customer on lo_custkey = c_custkey
left join supplier on lo_suppkey = s_suppkey
where (c_city='UNITED KI1' or c_city='UNITED KI5')
and (s_city='UNITED KI1' or s_city='UNITED KI5')
and d_year >= 1992 and d_year <= 1997
group by c_city, s_city, d_year
order by d_year asc, lo_revenue desc;
```

q3-4

```
select c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
from p_lineorder
left join dates on lo_orderdate = d_datekey
left join customer on lo_custkey = c_custkey
left join supplier on lo_suppkey = s_suppkey
where (c_city='UNITED KI1' or c_city='UNITED KI5') and (s_city='UNITED KI1'
or s_city='UNITED KI5') and d_yearmonth = 'Dec1997'
group by c_city, s_city, d_year
order by d_year asc, lo_revenue desc;
```

q4-1

```
select d_year, c_nation, sum(lo_revenue) - sum(lo_supplycost) as profit
from p_lineorder
```

```
left join dates on lo_orderdate = d_datekey
left join customer on lo_custkey = c_custkey
left join supplier on lo_suppkey = s_suppkey
left join part on lo_partkey = p_partkey
where c_region = 'AMERICA' and s_region = 'AMERICA' and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by d_year, c_nation
order by d_year, c_nation;
```

q4-2

```
select d_year, s_nation, p_category, sum(lo_revenue) - sum(lo_supplycost) as profit
from p_lineorder
left join dates on lo_orderdate = d_datekey
left join customer on lo_custkey = c_custkey
left join supplier on lo_suppkey = s_suppkey
left join part on lo_partkey = p_partkey
where c_region = 'AMERICA'and s_region = 'AMERICA'
and (d_year = 1997 or d_year = 1998)
and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by d_year, s_nation, p_category
order by d_year, s_nation, p_category;
```

q4-3

```
select d_year, s_city, p_brand, sum(lo_revenue) - sum(lo_supplycost) as profit
from p_lineorder
left join dates on lo_orderdate = d_datekey
left join customer on lo_custkey = c_custkey
left join supplier on lo_suppkey = s_suppkey
left join part on lo_partkey = p_partkey
where c_region = 'AMERICA'and s_nation = 'UNITED STATES'
and (d_year = 1997 or d_year = 1998)
and p_category = 'MFGR#14'
group by d_year, s_city, p_brand
order by d_year, s_city, p_brand;
```