Avaliação do Star Schema Benchmark aplicado a
bancos de dados NoSQL distribuídos e orientados a
colunas

Lucas de Carvalho Scabora

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 30 / 05 / 2016

Assinatura: _____

Lucas de Carvalho Scabora

Avaliação do Star Schema Benchmark aplicado a bancos de dados NoSQL distribuídos e orientados a colunas

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Cristina Dutra de Aguiar Ciferri

USP – São Carlos
Maio de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

S277a Scabora, Lucas de Carvalho
Avaliação do Star Schema Benchmark aplicado a
bancos de dados NoSQL distribuídos e orientados a
colunas / Lucas de Carvalho Scabora; orientadora
Cristina Dutra de Aguiar Ciferri. - São Carlos - SP,
2016.
113 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática Computacional)
- Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2016.

1. *Data warehouse*. 2. banco de dados NoSQL.
3. HBase. 4. Hadoop MapReduce. 5. *Star Schema
Benchmark*. I. Ciferri, Cristina Dutra de Aguiar,
orient. II. Título.

Lucas de Carvalho Scabora

**Evaluation of the Star Schema Benchmark applied to
NoSQL column-oriented distributed database systems**

Master dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-
USP, in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Profa. Dra. Cristina Dutra de Aguiar Ciferri

USP – São Carlos
May 2016

AGRADECIMENTOS

A Deus pela força, e àquele que é mestre por excelência, Senhor Jesus Cristo.

Aos meus pais José e Maria e a minha irmã Thais, pelo apoio, amor, carinho, incentivo e por compreender minha ausência. Ao meu grande amigo Gabriel pelo apoio e auxílio. À minha família, em especial aos meus parentes por sempre me apoiarem e acreditarem em mim.

À minha orientadora, Profa. Cristina Dutra de Aguiar Ciferri, pela oportunidade e orientação neste trabalho. Aos docentes da USP e da UFSCar, em especial ao Prof. José Fernando Rodrigues Junior e Prof. Ricardo Rodrigues Ciferri pelas colaborações e sugestões.

Aos meus colegas e amigos do GBDI, os quais tiveram um papel essencial no desenvolvimento desta dissertação de mestrado, durante nossa convivência no laboratório e em reuniões de grupo.

Ao ICMC-USP, pelo apoio institucional e por ter sido minha segunda casa tanto durante a graduação quanto no mestrado.

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP, processo nº 2014/12233-2), CAPES e CNPQ pelo suporte financeiro que possibilitou o desenvolvimento deste trabalho.

A todos que de alguma maneira contribuíram para a realização deste trabalho.

*“In God we trust,
all others (must) bring data.”
(John Lynch and David Stuckler)*

RESUMO

SCABORA, L. C.. **Avaliação do Star Schema Benchmark aplicado a bancos de dados NoSQL distribuídos e orientados a colunas**. 2016. 113 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Com o crescimento do volume de dados manipulado por aplicações de *data warehousing*, soluções centralizadas tornam-se muito custosas e enfrentam dificuldades para tratar a escalabilidade do volume de dados. Nesse sentido, existe a necessidade tanto de se armazenar grandes volumes de dados quanto de se realizar consultas analíticas (ou seja, consultas OLAP) sobre esses dados volumosos de forma eficiente. Isso pode ser facilitado por cenários caracterizados pelo uso de bancos de dados NoSQL gerenciados em ambientes paralelos e distribuídos. Dentre os desafios relacionados a esses cenários, destaca-se a necessidade de se promover uma análise de desempenho de aplicações de *data warehousing* que armazenam os dados do *data warehouse* (DW) em bancos de dados NoSQL orientados a colunas. A análise experimental e padronizada de diferentes sistemas é realizada por meio de ferramentas denominadas *benchmarks*. Entretanto, *benchmarks* para DW foram desenvolvidos majoritariamente para bancos de dados relacionais e ambientes centralizados. Nesta pesquisa de mestrado são investigadas formas de se estender o *Star Schema Benchmark* (SSB), um *benchmark* de DW centralizado, para o banco de dados NoSQL distribuído e orientado a colunas HBase. São realizadas propostas e análises principalmente baseadas em testes de desempenho experimentais considerando cada uma das quatro etapas de um *benchmark*, ou seja, esquema e carga de trabalho, geração de dados, parâmetros e métricas, e validação. Os principais resultados obtidos pelo desenvolvimento do trabalho são: (i) proposta do esquema FactDate, o qual otimiza consultas que acessam poucas dimensões do DW; (ii) investigação da aplicabilidade de diferentes esquemas a cenários empresariais distintos; (iii) proposta de duas consultas adicionais à carga de trabalho do SSB; (iv) análise da distribuição dos dados gerados pelo SSB, verificando se os dados agregados pelas consultas OLAP estão balanceados entre os nós de um *cluster*; (v) investigação da influência de três importantes parâmetros do *framework* Hadoop MapReduce no processamento de consultas OLAP; (vi) avaliação da relação entre o desempenho de consultas OLAP e a quantidade de nós que compõem um *cluster*; e (vii) proposta do uso de visões materializadas hierárquicas, por meio do *framework* Spark, para otimizar o desempenho no processamento de consultas OLAP consecutivas que requerem a análise de dados em níveis progressivamente mais ou menos detalhados. Os resultados obtidos representam descobertas importantes que visam possibilitar a proposta futura de um *benchmark* para DWs armazenados em bancos de dados NoSQL dentro de ambientes paralelos e distribuídos.

Palavras-chave: *Data warehouse*, banco de dados NoSQL, HBase, Hadoop MapReduce, *Star Schema Benchmark*.

ABSTRACT

SCABORA, L. C.. **Evaluation of the Star Schema Benchmark applied to NoSQL column-oriented distributed database systems**. 2016. 113 f. Monograph (Master student – Program in Computer Science and Computational Mathematics) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Due to the explosive increase in data volume, centralized data warehousing applications become very costly and are facing several problems to deal with data scalability. This is related to the fact that these applications need to store huge volumes of data and to perform analytical queries (i.e., OLAP queries) against these voluminous data efficiently. One solution is to employ scenarios characterized by the use of NoSQL databases managed in parallel and distributed environments. Among the challenges related to these scenarios, there is a need to investigate the performance of data warehousing applications that store the data warehouse (DW) in column-oriented NoSQL databases. In this context, benchmarks are widely used to perform standard and experimental analysis of distinct systems. However, most of the benchmarks for DW focus on relational database systems and centralized environments. In this master's research, we investigate how to extend the Star Schema Benchmark (SSB), which was proposed for centralized DWs, to the distributed and column-oriented NoSQL database HBase. We introduce proposals and analysis mainly based on experimental performance tests considering each one of the four steps of a benchmark, i.e. schema and workload, data generation, parameters and metrics, and validation. The main results described in this master's research are described as follows: (i) proposal of the FactDate schema, which optimizes queries that access few dimensions of the DW; (ii) investigation of the applicability of different schemas for different business scenarios; (iii) proposal of two additional queries to the SSB workload; (iv) analysis of the data distribution generated by the SSB, verifying if the data aggregated by OLAP queries are balanced between the nodes of a cluster; (v) investigation of the influence caused by three important parameters of the Hadoop MapReduce framework in the OLAP query processing; (vi) evaluation of the relationship between the OLAP query performance and the number of nodes of a cluster; and (vii) employment of hierarchical materialized views using the Spark framework to optimize the processing performance of consecutive OLAP queries that require progressively more or less aggregated data. These results represent important findings that enable the future proposal of a benchmark for DWs stored in NoSQL databases and managed in parallel and distributed environments.

Key-words: *Data warehouse, NoSQL databases, HBase, Hadoop MapReduce, Star Schema Benchmark.*

LISTA DE ILUSTRAÇÕES

Figura 2.1 – Exemplo de modelagem de dados tridimensional.	29
Figura 2.2 – Exemplo do esquema estrela para o modelo multidimensional presente na Figura 2.1.	31
Figura 2.3 – Exemplo do esquema estrela após a normalização dos atributos <i>país</i> , <i>estado</i> e <i>cidade</i> . Este novo esquema é denominado esquema floco de neve.	32
Figura 2.4 – Representação do modelo multidimensional conceitual em pelo menos três esquemas relacionais lógicos: estrela, floco de neve e constelação de fatos. . .	33
Figura 2.5 – Exemplo de como uma tabela conceitual pode ser armazenada segundo o paradigma orientado a coluna, organizado em famílias de colunas.	36
Figura 2.6 – Exemplo de organização de uma tabela relacional em famílias de colunas no HBase, representando o armazenamento lógico e físico.	37
Figura 2.7 – Principais componentes da hierarquia de sistemas distribuídos utilizados durante este estudo.	38
Figura 2.8 – Exemplo de particionamento e armazenamento de um arquivo dentro do HDFS. .	39
Figura 2.9 – Execução do modelo MapReduce para contar palavras de um texto.	40
Figura 2.10 – Exemplo do processamento em MapReduce de uma consulta SQL envolvendo três tabelas.	42
Figura 2.11 – Adaptação de um DW para sistemas NoSQL distribuídos.	44
Figura 3.1 – Esquema do <i>benchmark</i> TPC-H.	48
Figura 3.2 – Esquema resumido do <i>benchmark</i> TPC-DS.	51
Figura 3.3 – Esquema do <i>benchmark</i> SSB.	52
Figura 3.4 – Modelo de dados do BigBench.	59
Figura 4.1 – Ilustração das quatro etapas de um <i>benchmark</i> de DW, relacionando cada uma delas com a seção em que elas são investigadas neste trabalho.	65
Figura 4.2 – Representação da organização física do esquema proposto, denominado FactDate, em contraste com os esquemas existentes na literatura.	66
Figura 4.3 – Comparação entre os três esquemas avaliados dentro da categoria de consultas de baixa dimensionalidade.	70
Figura 4.4 – Comparação entre os três esquemas avaliados dentro da categoria de consultas de alta dimensionalidade.	71
Figura 4.5 – Comparação do tempo médio decorrido das consultas, divididas pela quanti- dade de dimensões agregadas pelas consultas.	74
Figura 4.6 – Consulta Qnew1, responsável por agregar dados das dimensões <i>Date</i> e <i>Part</i> . .	75

Figura 4.7 – Consulta Qnew2, a qual consiste na operação de <i>drill-down</i> sobre a consulta Qnew1, tornando-se dez vezes mais seletiva.	75
Figura 4.8 – Quantidade percentual de registros que são agregados de cada <i>RegionServer</i> , agrupados por consulta, considerando SF = 10.	78
Figura 4.9 – Quantidade percentual de registros que são agregados de cada <i>RegionServer</i> , agrupados por consulta, considerando SF = 20.	79
Figura 4.10–Tempo médio de processamento de consultas que agregam dados de soma total.	81
Figura 4.11–Tempo médio de processamento de consultas que agregam dados divididos em categorias, ou seja, possuem a cláusula <i>group by</i>	82
Figura 4.12–Tempo médio de processamento de consultas Qnew1 e Qnew2, que realizam duas agregações dos dados divididos em longas categorias.	82
Figura 4.13–Tempo decorrido das consultas de classe 4 (Q4.1, Q4.2 e Q4.2) conforme varia-se a quantidade máxima de processos Reduce definidos.	84
Figura 4.14–Tempo decorrido de consultas OLAP sobre um arquivo de texto desnormalizado, correspondente ao valor 20 de SF.	86
Figura 4.15–Tempo decorrido de consultas OLAP sobre o esquema FactDate do HBase. .	86
Figura 4.16–Tempo decorrido de consultas OLAP sobre o esquema FactDate do HBase, variando o tamanho dos <i>RegionServers</i>	87
Figura 4.17–Teste de escalabilidade, considerando a comparação do tempo médio de processamento das consultas sobre uma mesma base de dados, variando a quantidade de nós utilizados no processamento das consultas.	89
Figura 4.18–Percentual de ganho de desempenho das quatro categorias de consultas, ao comparar o tempo de processamento de uma consulta em 4 nós com o tempo decorrido dessa mesma consulta em 2 nós.	90
Figura 4.19–Tempo de processamento das consultas OLAP iterativas, organizada por classe de consultas.	94
Figura 4.20–Análise do tempo das consultas iterativas, variando o volume de dados nos valores 10, 50 e 100 de SF.	95
Figura A.1 – Exemplo de uma instância do SSB, mostrando como essa tupla foi traduzida para o formato JSON, para que possa ser armazenada no MongoDB.	110
Figura A.2–Consulta Q1.1 da carga de trabalho do SSB, representada de duas formas para ser processada dentro do MongoDB.	111
Figura A.3–Consulta Q4.2 da carga de trabalho do SSB, traduzida para comandos MapReduce.	112
Figura A.4–Tempo de processamento das consultas no MongoDB, em contraste com o processamento dessas mesmas consultas no HBase, usando os esquemas SameCF e FactDate.	113

LISTA DE ALGORITMOS

Algoritmo 1 – mapQ1	67
Algoritmo 2 – reduceSOMA	68
Algoritmo 3 – mapQ2	68
Algoritmo 4 – reduceMaximoMinimo	77

LISTA DE TABELAS

Tabela 3.1 – Tamanho de domínio dos atributos do SSB, os quais possuem distribuição uniforme.	53
Tabela 3.2 – Resumo das cinco cargas de trabalho definidas pelo YCSB, ilustrando a quantidade relativa de operações em cada carga.	58
Tabela 4.1 – Volumes de dados utilizados durante os experimentos.	69
Tabela 4.2 – Resumo da seletividade das consultas OLAP definidas pelo SSB, juntamente com as duas consultas propostas Qnew1 e Qnew2.	76
Tabela 4.3 – Três configurações investigadas, referentes à quantidade de processos Map que podem executar simultaneamente em um mesmo nó, juntamente com a memória alocada para cada processo Map.	80
Tabela 4.4 – Número de categorias geradas por consulta, considerando o volume de dados correspondente ao valor 20 de SF.	83
Tabela 4.5 – Quantidade de blocos de dados distribuídos de acordo com o tamanho deles. Durante o processamento, para cada bloco é criado um processo Map. . . .	85
Tabela 4.6 – Atributos presentes nas Visões Materializadas (VMs) específicas para cada consulta, juntamente com os atributos armazenados nas VM propostas para otimização de operações OLAP iterativas.	92
Tabela 4.7 – Volume ocupado pelas visões materializadas (VMs), mostrando o espaço ocupado tanto em memória primária quanto em memória secundária.	93

SUMÁRIO

1	Introdução	21
1.1	Contextualização	21
1.2	Motivação	22
1.3	Objetivos e Contribuições	23
1.4	Organização do Trabalho	25
2	<i>Data Warehouse</i> e bancos de dados NoSQL	27
2.1	Considerações Iniciais	27
2.2	<i>Data Warehouse</i>	28
2.2.1	<i>Características dos Dados</i>	28
2.2.2	<i>Modelo Multidimensional</i>	29
2.2.3	<i>Níveis de Agregação</i>	30
2.2.4	<i>Esquema Estrela e Normalização dos Dados</i>	31
2.3	Bancos de Dados NoSQL	33
2.3.1	<i>Paradigmas de Armazenamento NoSQL</i>	34
2.3.2	<i>NoSQL Orientado a Coluna</i>	35
2.4	Processamento Distribuído	37
2.4.1	<i>Hadoop MapReduce</i>	38
2.4.2	<i>Spark</i>	42
2.5	Considerações Finais	43
3	<i>Benchmarks</i> de Bancos de Dados	45
3.1	Considerações Iniciais	45
3.2	Etapas de um <i>Benchmark</i> de <i>Data Warehouse</i>	46
3.3	<i>Benchmarks</i> para <i>Data Warehouse</i>	47
3.3.1	<i>TPC-H</i>	47
3.3.1.1	<i>Processamento das consultas do TPC-H usando Hadoop MapReduce</i>	49
3.3.1.2	<i>Proposta de métricas para avaliar ambientes distribuídos</i>	50
3.3.2	<i>TPC-DS</i>	50
3.4	<i>Star Schema Benchmark</i>	52
3.4.1	<i>Columnar NoSQL Star Schema Benchmark</i>	54
3.4.2	<i>Modelagem Multidimensional em bancos de dados NoSQL</i>	55
3.4.3	<i>Distribuição de atributos em famílias de colunas</i>	56
3.4.4	<i>SSB+</i>	57

3.5	Benchmarks para Sistemas Distribuídos e NoSQL	58
3.5.1	<i>YCSB</i>	58
3.5.2	<i>BigBench</i>	59
3.5.3	<i>TPCx-HS</i>	60
3.6	Considerações Finais	61
4	Investigação das Etapas de um Benchmark para Data Warehouse .	63
4.1	Considerações Iniciais	63
4.2	Esquema e da Carga de Trabalho	65
4.2.1	<i>Organização de um DW em Famílias de Colunas</i>	65
4.2.2	<i>Novas consultas para a Carga de Trabalho do SSB</i>	74
4.3	Distribuição dos dados gerados pelo SSB	77
4.4	Parâmetros do Processamento Distribuído	79
4.4.1	<i>Processos Map executados simultaneamente</i>	80
4.4.2	<i>Quantidade total de processos Reduce</i>	83
4.4.3	<i>Tamanho do bloco de dados distribuído</i>	85
4.5	Processamento e Validação	87
4.5.1	<i>Escalabilidade contra variação do volume de dados</i>	88
4.5.2	<i>Uso de Visões Materializadas em Consultas OLAP Iterativas</i>	90
4.6	Considerações Finais	96
5	Conclusões	99
5.1	Conclusão Geral	99
5.2	Contribuições	100
5.3	Divulgação de Resultados	101
5.4	Trabalhos Futuros	101
Referências		103
APÊNDICE A	Processamento de Consultas em Bancos de Dados NoSQL Orientado a Documentos	109

INTRODUÇÃO

Neste capítulo de introdução são detalhados a contextualização, a motivação e os objetivos de realização desta dissertação. Dentro da motivação são descritos os principais desafios de se adaptar um benchmark de DW. Já em objetivos, é explicada a principal pergunta que essa dissertação de mestrado visa responder, bem como são resumidas as principais contribuições geradas pela realização deste estudo.

1.1 Contextualização

Um ambiente de *data warehousing* serve como base para o ambiente informacional de uma empresa, sendo responsável por providenciar informações estratégicas e sumarizadas sobre o negócio e por facilitar o processo de tomada de decisão da empresa (CHAUDHURI; DAYAL, 1997). Nesse ambiente, a informação de provedores autônomos, heterogêneos e distribuídos é integrada, tornando-se disponível para consulta ou análise imediata de usuários de sistemas de suporte à decisão (GOLFARELLI; RIZZI; CELLA, 2004; GOLFARELLI; RIZZI, 2009; ARIYACHANDRA; WATSON, 2010; GONZALES *et al.*, 2011; CHAUDHURI; DAYAL; NARASAYYA, 2011). Usualmente, essas análises são conduzidas por meio de consultas denominadas *online analytical processing* (OLAP), as quais fornecem para as organizações um método de acessar e visualizar os dados corporativos com alta flexibilidade e desempenho.

Nos últimos anos, pesquisas relacionadas a *data warehousing* e OLAP têm produzido diversas tecnologias responsáveis pelo gerenciamento de sistemas de informação para oferecer suporte à tomada de decisão (SCHNEIDER; VOSSEN; ZIMÁNYI, 2011). Nesse mesmo período, o termo *data warehousing* evoluiu para o termo inteligência de negócio e diversos aspectos têm sido investigados, dentre os quais pode-se citar: (i) o armazenamento dos dados do *data warehouse* em diferentes tecnologias, como relacional e em estruturas de dados especializadas; (ii) o uso de visões materializadas para organizar os dados do *data warehouse* em um cubo de dados com diferentes níveis de agregação; (iii) o desenvolvimento de modelos de dados conceituais multidimensionais; (iv) o processamento e a otimização de consultas OLAP, incluindo a incorporação de conceitos de paralelismo e distribuição dos dados; (v) o desenvolvimento de

processos para subsidiar as etapas de extração, transformação e limpeza dos dados, bem como ferramentas de análise e consulta; e (vi) a validação da qualidade e do desempenho de diferentes implementações de um *data warehouse* em bancos de dados relacionais e em bancos de dados *Not Only SQL* (NoSQL).

Para analisar o desempenho das implementações referentes à etapa (vi), são necessários mecanismos para medir e comparar diferentes implementações de um *data warehouse* (DW) (CIFERRI, 1995). A análise de desempenho é uma técnica para medir a eficiência com a qual diferentes sistemas efetuam uma determinada tarefa. Essa eficiência é definida como uma medida escalar, tal como o tempo de processamento e a quantidade de recursos computacionais utilizados. Por meio dessa medida é possível comparar diferentes sistemas, de forma a ordená-los pelo desempenho. Uma das formas de se realizar uma análise de desempenho é por meio de testes experimentais, os quais são realizados utilizando-se um conjunto preestabelecido de operações. A técnica de *benchmark* consiste na padronização dessas operações, de forma a determinar o melhor sistema para um determinado domínio. Em relação ao domínio de ambientes de *data warehousing*, um exemplo típico de operações a serem analisadas são consultas OLAP.

1.2 Motivação

Com o grande crescimento do volume de dados manipulado por aplicações de *data warehousing*, as soluções tradicionais tornam-se muito custosas e exibem dificuldades para tratar a escalabilidade do volume de dados. Nesse sentido, existe a necessidade tanto de se armazenar grandes volumes de dados quanto de se realizar consultas OLAP sobre esses dados volumosos de forma eficiente, o que pode ser facilitado em ambientes paralelos e distribuídos. Os bancos de dados NoSQL representam uma alternativa para manipular grandes volumes de dados sem a necessidade de organizá-los seguindo o modelo relacional (MONIRUZZAMAN; HOSSAIN, 2013), além de possibilitar tanto o armazenamento distribuído desses dados quanto o processamento deles. Surge, então, a necessidade de se promover uma análise de desempenho nesses ambientes paralelos e distribuídos proporcionados pelo uso de bancos de dados NoSQL para armazenar os dados do *data warehouse*. Dentro do contexto deste estudo, investiga-se a análise de desempenho por meio da técnica de *benchmark*.

Na literatura existem tanto *benchmarks* para DW centralizados quanto *benchmarks* para sistemas distribuídos e bancos de dados NoSQL. Adicionalmente, já existem estudos voltados à proposta de *benchmarks* para DWs armazenados em bancos de dados NoSQL. Porém, como identificado e discutido no Capítulo 3, ainda existem diversos desafios em relação a adaptar um *benchmark* de DW aos bancos de dados NoSQL distribuídos, em especial aos bancos NoSQL orientados a colunas, que é o enfoque desta dissertação de mestrado. Este paradigma de armazenamento foi escolhido para ser investigado pois proporciona ganhos de desempenho de até uma ordem de magnitude, quando comparado ao paradigma tradicional orientado a instâncias,

para cargas de trabalho analíticas focadas em leitura (ABADI; MADDEN; HACHEM, 2008).

Existem diversos desafios de se adaptar um *benchmark* de DW a um banco de dados NoSQL distribuído e orientado a coluna. Dentre esses desafios, destacam-se:

- Como deve ser modelado o esquema de um DW em um banco de dados NoSQL orientado a colunas? De que forma diferentes esquemas impactam no desempenho do processamento de consultas OLAP?
- Quais parâmetros de processamento distribuído influenciam durante a execução de consultas OLAP sobre bancos de dados NoSQL?
- Como diferentes formas de processamento e validação de um *benchmark* podem influenciar no desempenho de operações OLAP?

Essas perguntas, juntamente a outros estudos realizados, são retomadas e analisadas no Capítulo 4, o qual refere-se às investigações, propostas e trabalhos desenvolvidos durante esta pesquisa de mestrado.

1.3 Objetivos e Contribuições

Esta pesquisa de mestrado visa a investigação sobre formas de se estender o *Star Schema Benchmark* (SSB) (O'NEIL *et al.*, 2009), um *benchmark* de DW centralizado, para bancos de dados NoSQL distribuídos e orientados a colunas. O SSB é um *benchmark* que define uma aplicação de DW voltada ao armazenamento de dados históricos referentes a pedidos e vendas de uma empresa de varejo, durante um certo período de tempo. Ele foi escolhido para ser usado como base para essa pesquisa de mestrado porque, diferentemente dos outros *benchmarks* para DW analisados nessa dissertação, o SSB armazena os dados seguindo o esquema estrela, que propicia simplicidade e facilidade para usuários realizarem operações sobre os dados do esquema, além de proporcionar melhor desempenho no processamento de consultas (KIMBALL; ROSS, 2002). O problema reside no fato de que o SSB é voltado para comparação de bancos de dados relacionais centralizados. Dessa forma, surge a necessidade de investigar o comportamento desse *benchmark* frente a bancos de dados NoSQL distribuídos e orientado a colunas.

Dentro deste contexto, a principal premissa a ser respondida nesta pesquisa de mestrado é: ***"O que é necessário adicionar ao SSB, ou a outro benchmark de DW, para validar efetivamente ambientes de data warehousing baseados em bancos de dados NoSQL distribuídos e orientados a colunas?"***. Para responder a esse questionamento, deve-se analisar quais as limitações deste *benchmark* quando aplicado nesses bancos NoSQL. A investigação dessas limitações consiste no objetivo desta pesquisa de mestrado.

Para se atingir o objetivo proposto, nesta pesquisa de mestrado são analisadas adaptações referentes a cada uma das etapas de um *benchmark* de DW, a saber: (i) esquema e carga de

trabalho, (ii) geração de dados, (iii) parâmetros e métricas, e (iv) validação. As análises realizadas usaram majoritariamente como base o *framework* Hadoop¹ MapReduce e o banco de dados NoSQL orientado a coluna HBase. O MapReduce é um *framework* para processamento de grandes quantidades de dados armazenados de forma distribuída, sendo responsável por prover principalmente simplicidade, elevada escalabilidade, flexibilidade e tolerância as falhas (DEAN; GHEMAWAT, 2004). Já o HBase é um banco de dados NoSQL, orientado a colunas e de código aberto, utilizado em aplicações que necessitam realizar o armazenamento e o processamento intensivo de dados (KONISHETTY *et al.*, 2012). Apenas uma das análises realizadas utilizou como base o *framework* Spark, desde que esse *framework* facilita o uso de visões materializadas por meio de *Resilient Distributed Datasets* (RDDs).

Com base nas etapas de um *benchmark* para DW, as principais contribuições dessa pesquisa de mestrado são:

- Na etapa (i), foram investigadas diferentes maneiras de se modelar o esquema de um DW armazenado no HBase, avaliando o impacto que cada uma delas tem no processamento de consultas OLAP;
- Ainda na etapa (i), realizou-se uma análise da completude da carga de trabalho do SSB, considerando a quantidade de tabelas agregadas durante o processamento das consultas. Neste sentido, foram propostas duas consultas adicionais à carga de trabalho original do SSB, para a avaliação de consultas envolvendo duas tabelas de dimensão do esquema;
- Na etapa (ii), avaliou-se a distribuição dos dados gerados pelo SSB, verificando se os dados agregados pelas consultas OLAP estão balanceados entre os nós de um *cluster*, ou seja, se o volume de dados recuperado de cada nó durante o processo de agregação é similar;
- Na etapa (iii), estudou-se os três principais parâmetros do processamento distribuído, usando o *framework* Hadoop MapReduce, e identificou-se a influência deles sobre a métrica de desempenho reportada pelo SSB. Esses parâmetros referem-se à quantidade de processos Map executados simultaneamente em cada nó do *cluster*, número total de processos Reduce atribuídos durante o processamento de uma consulta e tamanho do bloco de dados distribuído;
- Na etapa (iv), pensando na validação de um sistema distribuído, foi avaliada a relação entre o desempenho das consultas OLAP e a quantidade de nós que compõem um *cluster*. Essa relação também foi analisada considerando diferentes volumes de dados; e
- Ainda na etapa (iv), foi proposto o uso de visões materializadas, por meio do *framework* Spark², com o objetivo de otimizar o processamento de operações OLAP. Essas visões materializadas armazenam dados originados de uma consulta *ad-hoc*, porém também são

¹ <http://hadoop.apache.org/>

² <http://spark.apache.org/>

previamente armazenadas outras agregações dos dados pertencentes à mesma hierarquia dos atributos utilizados na consulta original. Assim, futuras consultas sobre esses dados, baseadas em operações OLAP (como *drill-down* e *roll-up*), não necessitam acessar os dados diretamente do DW.

Os resultados obtidos permitiram a identificação de limitações e problemas relacionados a aplicação do SSB para a validação de DWs armazenados no HBase. Ademais, também foram levantadas características importantes que devem ser consideradas durante a execução de um *benchmark* de DW em ambientes distribuídos, usando como processamento o *framework* Hadoop MapReduce. Por exemplo, foi estudada a influência no desempenho do processamento de consultas causada pelas diferentes divisões lógicas dos atributos em famílias de colunas promovidas pelo HBase. Essas descobertas representam um primeiro passo para a identificação das características necessárias para o posterior desenvolvimento de um *benchmark* para DWs armazenados no banco de dados NoSQL distribuído HBase, sendo este capaz de validar efetivamente bancos de dados NoSQL distribuídos e orientados a colunas. Note que o objetivo deste estudo não é propor um *benchmark* para DWs armazenados em bancos de dados NoSQL orientados a colunas, mas sim realizar um conjunto de propostas e análises majoritariamente baseadas em testes de desempenho experimentais que ofereçam fundamentos para a posterior proposta desse *benchmark*.

1.4 Organização do Trabalho

Além desse capítulo introdutório, esta dissertação possui mais quatro capítulos, estruturados da seguinte forma:

- No Capítulo 2 é descrita a fundamentação teórica necessária ao entendimento do trabalho. Primeiramente, são resumidos conceitos relacionados ao DW, incluindo as características dos dados, o modelo multidimensional, os níveis de agregação e formas de representação lógica dos dados, como o esquema estrela e suas variações, oriundas da normalização dos dados. Na sequência, são detalhadas as características de bancos de dados NoSQL distribuídos, com enfoque no paradigma de armazenamento orientado a colunas. Ademais, para processar os dados desses bancos NoSQL, são descritos dois principais *frameworks* de processamento distribuído denominados Hadoop MapReduce e Spark.
- No Capítulo 3 estão sumarizados conceitos relacionados a *benchmarks*. A partir desses conceitos, são descritos como correlatos os principais *benchmarks* de DW e de sistemas distribuídos ou bancos NoSQL, cada qual com a sua limitação para o estudo realizado. Com relação aos *benchmarks* de DW existentes, foram detalhadas adaptações de cada um deles aos bancos de dados NoSQL, sendo essas abordagens consideradas fortemente relacionadas aos experimentos realizados e apresentados nesta dissertação.

- No Capítulo 4 são resumidas as propostas e investigações realizadas neste estudo, referentes a cada uma das etapas de um *benchmark* de DW, a saber: diferentes maneiras de modelar o esquema de um DW em agrupamentos de famílias de colunas distintas; proposta de duas consultas bidimensionais à carga de trabalho do SSB; verificação do balanceamento dos dados gerados pelo SSB entre as máquinas do *cluster*; análise da influência de três parâmetros no processamento de consultas OLAP; testes relacionados a variação tanto do volume de dados quanto do tamanho do *cluster*; e proposta de visões materializadas para otimizar o processamento de operações OLAP.
- No Capítulo 5 conclui-se os resultados dessa dissertação, destacando-se as contribuições e os trabalhos futuros.

Além desses capítulos, esta dissertação de mestrado também incorpora um apêndice. No apêndice A são detalhados experimentos adicionais realizados sobre o banco de dados NoSQL MongoDB, comparando-o com os resultados obtidos no Capítulo 4, os quais basearam-se no banco de dados HBase.

DATA WAREHOUSE E BANCOS DE DADOS NOSQL

Neste capítulo são investigados os principais conceitos relacionados ao desenvolvimento desta dissertação. É descrito o ambiente de data warehousing, com respeito às características de seus dados, modelagem conceitual e lógica e organização dos dados em níveis de agregação. Também são detalhados os principais conceitos relacionados a bancos de dados NoSQL distribuídos, incluindo técnicas distribuídas para o processamento de dados.

2.1 Considerações Iniciais

Um ambiente de *data warehousing* abrange diversas arquiteturas, algoritmos e ferramentas que possibilitam unificar, em uma base de dados, as informações de diferentes provedores autônomos, heterogêneos e distribuídos (CIFERRI, 2002). O principal componente do ambiente de *data warehousing* é o *data warehouse* (DW), que possibilita aos usuários a realização de operações analíticas sobre esses dados, como a execução de consultas OLAP (*Online Analytical Processing*), a geração de relatórios e a mineração dos dados.

A seguir são resumidos os principais conceitos relacionados a *data warehousing*. O capítulo está organizado da seguinte forma. Na Seção 2.2 são descritos conceitos fundamentais sobre um DW, contendo detalhes sobre as características dos dados (Seção 2.2.1), a modelagem multidimensional (Seção 2.2.2), os níveis de agregação (Seção 2.2.3) e o esquema estrela, bem como a normalização dos dados (Seção 2.2.4). Na Seção 2.3 são investigados os conceitos sobre bancos de dados NoSQL, descrevendo os principais paradigmas de armazenamento (Seção 2.3.1), com foco no paradigma orientado a coluna (Seção 2.3.2), enquanto que na Seção 2.4 são detalhados dois principais *frameworks* para processamento distribuído, a saber: Hadoop MapReduce (Seção 2.4.1) e Spark (Seção 2.4.2). Por fim, na Seção 2.5 são descritas as considerações finais do capítulo.

2.2 Data Warehouse

Um DW é um banco de dados voltado ao suporte à tomada de decisão, sendo responsável por prover informações estratégicas e garantir eficiência e flexibilidade em sua obtenção por meio de consultas. Essas informações servem para auxiliar o processo de gerência, possibilitando melhores decisões com o objetivo de assegurar o sucesso de uma empresa.

O acesso à informação integrada dos dados do DW ocorre principalmente em duas etapas (CIFERRI, 2002). Na primeira etapa, os dados relevantes à aplicação são extraídos de cada um dos provedores para serem armazenados no DW. A fim de integrar e armazenar esses dados, os quais podem estar organizados em diferentes formatos e modelos, ocorre o tratamento desses dados. Esse tratamento é realizado por um processo denominado *extract, transform and load* (ETL), que consiste nas seguintes subetapas: (i) extração e tradução dos dados (*extract*), na qual ocorre a conversão dos dados oriundos de cada provedor para o formato e o modelo usado no DW; (ii) transformação, filtragem e limpeza dos dados (*transform*), a qual manipula dados duvidosos e/ou incorretos a fim de garantir confiabilidade nos dados mantidos no DW. As principais causas desses dados duvidosos e/ou incorretos decorrem da existência de campos contendo comprimentos inválidos, da violação de restrições de integridade, das abreviações não padronizadas de atributos, dentre outros problemas; (iii) armazenamento e integração dos dados (*load*), a qual consiste em eliminar replicações de dados a fim de integrá-los em uma única base, ou seja, no DW. Uma das causas dessas replicações consiste no fato de que os provedores foram modelados de forma independente, possibilitando a existência do mesmo dado em provedores distintos. Por fim, na segunda etapa do acesso à informação, são realizadas operações analíticas diretamente sobre o DW, como consultas OLAP, sem a necessidade dos provedores de informação.

Um exemplo de aplicação de DW de uma rede de supermercados é a integração dos dados relativos à quantidade vendida de cada produto, em diferentes lojas, ao longo dos anos. Assim, os usuários dessa aplicação podem realizar análises de tendências simples (qual a quantidade vendida, por semestre, de um determinado produto no ano de 2015?), análises comparativas (qual a quantidade vendida, por semana, dos produtos de uma determinada marca nos últimos três anos?) e análises de tendências múltiplas (qual a quantidade vendida, por dia, dos produtos de uma determinada categoria nos últimos cinco anos, de acordo com períodos sazonais, como a época de Natal?).

2.2.1 Características dos Dados

Os dados presentes em um DW são caracterizados por serem orientados ao assunto, integrados, não voláteis e históricos (INMON, 2002). Uma vez que um DW é voltado ao suporte à tomada de decisão, seus dados são orientados ao assunto pois armazenam dados relativos às operações e transações de maior interesse para a corporação, como produtos, clientes e

fornecedores. Já a característica de integração dos dados é proporcionada pela execução do processo ETL, explicada na Seção 2.2. Os dados de um DW são considerados não voláteis devido ao fato de que os valores são armazenados por longos períodos de tempo, permanecendo estáveis durante todo esse período.

Por fim, os dados são históricos porque estão sempre associados a um tempo específico que registra quando o fato ocorreu. Uma vez que os dados no DW permanecem válidos por longos períodos, cada um deles é associado a um componente “tempo”, diferentemente de banco de dados convencionais, em que geralmente armazena-se apenas as informações atuais. Considerando o exemplo anterior, de uma aplicação de DW responsável em armazenar dados referentes às transações realizadas por uma rede de supermercados, a quantidade vendida de cada produto, em cada loja, está associada ao tempo em que ela ocorreu.

2.2.2 Modelo Multidimensional

Os dados do DW são usualmente modelados multidimensionalmente, tendo como objetivo a visualização dos dados segundo diferentes perspectivas (conhecidas como dimensões). Uma visão multidimensional consiste um conjunto de medidas de interesse, denominadas fatos, as quais são os objetos de análise relevantes ao negócio, e um conjunto de dimensões, as quais contêm atributos que determinam o contexto para as medidas (CHAUDHURI; DAYAL, 1997; IN-MON, 2002). Esse contexto determinado pelas dimensões é baseado nas perguntas (KIMBALL; ROSS, 2002): quem, o que, quando, onde, como e porque. Além disso, o modelo multidimensional organiza os dados em um formato que não só facilita o entendimento dos usuários como também melhora o desempenho de consultas.

Figura 2.1 – Exemplo de modelagem de dados tridimensional, usada no exemplo da rede de supermercados.

A forma mais comum de se ilustrar visualmente a multidimensionalidade do DW é por meio do cubo de dados. Partindo do exemplo descrito na Seção 2.2, na Figura 2.1 são mostradas três dimensões, denominadas Produto, Loja e Tempo, modeladas na forma de um cubo. Cada uma dessas dimensões representa um eixo desse cubo, e cada célula, como exemplo (P_2, L_3, T_1) ,

representa a quantidade vendida por produto (P), por loja (L), por período de tempo (T, como dia, mês ou ano). Assim, é possível realizar consultas que promovam a agregação desses dados, por exemplo, calcular a quantidade vendida por produto por dia, em que se agregam (por meio da soma) a quantidade vendida de cada loja.

2.2.3 Níveis de Agregação

Os atributos de uma dimensão podem ter um relacionamento com outros atributos da mesma dimensão por meio de hierarquias, as quais determinam diferentes níveis de granularidade, relacionadas com o grau de agregação dos dados. Por exemplo, considere a aplicação de *data warehousing* explicada na Seção 2.2 e ilustrada na Figura 2.1, a qual armazena dados relativos à quantidade vendida segundo diferentes produtos e lojas ao longo dos anos. Neste exemplo, a medida numérica *quantidade vendida* é determinada pelas dimensões Produto, Loja e Tempo. Ademais, a dimensão Loja contém dados referentes à localização dela, os quais podem ser expressos pela hierarquia de atributos (*país*) $\preceq$ (*estado*) $\preceq$ (*cidade*) $\preceq$ (*endereço*). Neste contexto, o atributo *endereço* possui a menor granularidade, enquanto *país* é o atributo de maior granularidade. O operador utilizado para expressar essa hierarquia, $\preceq$, impõe uma relação de ordem parcial entre os atributos, especificando que uma agregação de maior granularidade pode ser determinada usando uma agregação, da mesma hierarquia, com menor granularidade. Um exemplo disso é o atributo *estado*, que pode ser determinado a partir do atributo *cidade*.

Esses níveis de agregação são representados por meio de uma visão multidimensional dos dados, que variam desde um nível inferior, possuindo dados detalhados, até um nível superior, com dados muito resumidos (LIMA; HIRATA, 2011). Nessa visão também são armazenados os níveis intermediários, que representam graus de agregação crescentes, os quais são gerados com base nas hierarquias de atributos de níveis inferiores. Nesse sentido, por meio do cubo de dados multidimensional, é possível visualizar não somente os valores de medidas do nível inferior, mas também identificar as várias agregações que podem ser originadas ao longo de todas as dimensões. Por exemplo, pode-se analisar: (i) *quantidade vendida* por *ano*, por *faixa de preço*, por *loja* (identificada pelo *endereço*, nível inferior); (ii) *quantidade vendida* por *ano*, por *faixa de preço* e por *estado* (nível intermediário); e (iii) *quantidade vendida* por *ano*, por *faixa de preço* e por *país* (nível superior).

Operações OLAP típicas incluem consultas *drill-down*, *roll-up*, *slice and dice*, *pivot* e *drill-across* (CHAUDHURI; DAYAL, 1997), as quais são detalhadas a seguir:

- *Drill-down*: consultas que analisam os dados em níveis progressivamente mais detalhados, ou seja, consultas que vão de um nível superior de agregação (ou seja, sumarizados) até um nível inferior de agregação (ou seja, detalhados).
- *Roll-up*: operação inversa à de *drill-down*, ou seja, consultas que investigam dados em níveis de agregação progressivamente menos detalhados.

- *Slice and dice*: consultas que permitem restringir os dados que estão sendo analisados a um subconjunto desses dados. Essa restrição pode ser realizada sobre apenas uma dimensão (operação *slice*) ou duas ou mais dimensões do cubo simultaneamente (operação *dice*).
- *Pivot*: consultas que reorientam a visão multidimensional dos dados, ou seja, alteram a ordem das dimensões, modificando a orientação sob a qual os dados são visualizados.
- *Drill-across*: consultas que comparam medidas de diferentes cubos de dados, as quais são relacionadas entre si por uma ou mais dimensões em comum.

2.2.4 Esquema Estrela e Normalização dos Dados

Em DWs implementados em bancos de dados relacionais, as dimensões e medidas são modeladas majoritariamente por meio do esquema estrela (KIMBALL; ROSS, 2002). A tabela de fatos contém as medidas de interesse (ou seja, os fatos), além de chaves estrangeiras (*foreign key*, FK) para as tabelas de dimensão, as quais são responsáveis pela ligação dos fatos às dimensões que os contextualizam. Cada tabela de dimensão, por sua vez, contém os atributos da dimensão e uma chave primária (*primary key*, PK) para cada uma de suas instâncias.

Figura 2.2 – Exemplo do esquema estrela para o modelo multidimensional presente na Figura 2.1.

Um exemplo de esquema estrela é ilustrado na Figura 2.2, o qual possui uma tabela de fatos localizada visualmente no centro da estrela e um conjunto de tabelas de dimensão localizadas nas extremidades do esquema. Nesse esquema a tabela de fatos, denominada Vendas, é responsável por armazenar a medida numérica de interesse *quantidade_vendida*, que é o número de produtos vendidos em uma loja em um determinado período. Já as tabelas de dimensão (Produto, Loja e Tempo) armazenam atributos que caracterizam essa medida numérica, além de suas respectivas PKs. Ademais, a tabela de fatos possui FKs para as tabelas de dimensão (por exemplo, *FK_tempo*, *FK_produto* e *FK_loja*), as quais são responsáveis pela ligação dos fatos às dimensões que os contextualizam. O conjunto dessas FKs também é utilizado pela tabela de fatos Vendas como chave primária dela.

A organização dos dados do DW na forma de um esquema estrela requer a realização de junções-estrela no processamento das consultas OLAP. Uma junção-estrela consiste em acessar todas as tabelas de fatos e de dimensão envolvidas na consulta OLAP, e realizar todas as junções e agregações necessárias, além de resolver operações de seleção definidas nos predicados convencionais. Além da junção-estrela, outro recurso comumente utilizado é o uso de visões materializadas (VM), responsáveis por minimizar a quantidade de junções-estrela realizadas ao armazenar previamente os atributos e os resultados de funções de agregações sobre as medidas de interesse.

A normalização de uma tabela do banco de dados refere-se ao controle de redundância dos dados (KIMBALL; ROSS, 2002). Seguindo o exemplo da Figura 2.2, os atributos da tabela de fatos são normalizados em diferentes dimensões, e cada dimensão possui seus atributos desnormalizados. Um exemplo disso consiste no fato de que várias lojas podem pertencer a um mesmo país, em que este atributo será repetido em várias as instâncias da dimensão Loja. A fim de minimizar essa redundância, esses dados podem ser normalizados, de forma a criar uma tabela denominada País, armazenando informações sobre países presentes no esquema e unindo essa nova tabela à dimensão Loja por meio de uma FK. Essa mesma abordagem é realizada sobre os atributos *estado* e *cidade*, dado que pode-se ter mais de uma loja em um mesmo estado ou até mesmo em uma mesma cidade. Sendo assim, essa normalização providencia duas vantagens ao esquema: para cada loja, o tamanho do atributo que representa o país foi minimizado (de campo texto para campo numérico ou *bytes*) e eventuais alterações nos dados do país ocorrem apenas em uma tupla da tabela País, ao invés de alterar diversas instâncias da dimensão Loja. Ao normalizar as dimensões do esquema estrela, tem-se uma variação dela denominada esquema floco de neve (CIFERRI, 2002), conforme ilustrado na Figura 2.3.

Figura 2.3 – Exemplo do esquema estrela presente na Figura 2.2 após a normalização dos atributos *país*, *estado* e *cidade*. Este novo esquema é denominado esquema floco de neve.

Ademais, quando se há a necessidade de se armazenar múltiplas tabelas de fatos em um mesmo esquema, outra possível implementação do DW consiste na constelação de fatos. Este esquema possui duas ou mais tabelas de fatos, cada qual armazenando diferentes medidas de interesse, que compartilham algumas tabelas de dimensão. A Figura 2.4 representa um resumo, em que a modelagem multidimensional conceitual pode ser traduzida em pelo menos três possíveis esquemas lógicos relacionais, estudados neste projeto: estrela, floco de neve e constelação de fatos (CIFERRI *et al.*, 2013).

Figura 2.4 – Resumo da representação do modelo multidimensional conceitual em pelo menos três esquemas relacionais lógicos: estrela, floco de neve e constelação de fatos (adaptado de CIFERRI *et al.*, 2013).

Essa representação se refere a bancos de dados relacionais e, uma vez que o enfoque do projeto envolve bancos de dados NoSQL, na Seção 2.3 são descritos os conceitos fundamentais sobre esses bancos NoSQL.

2.3 Bancos de Dados NoSQL

Um aspecto importante para a computação distribuída, dentro do contexto de bancos de dados descentralizados, refere-se à forma de armazenamento dos dados. Segundo o Teorema CAP (*Consistency, Availability and Partition Tolerance*), os sistemas computacionais descentralizados possuem três propriedades (GILBERT; LYNCH, 2002): (i) consistência: todas as réplicas dos dados devem possuir sempre o mesmo valor; (ii) disponibilidade: o sistema continua a operar

se pelo menos um nó do *cluster* estiver funcionando corretamente; e (iii) tolerância a partições: caso ocorra um erro em um dos nós do *cluster*, como a perda de mensagens entre diferentes nós, o correto funcionamento do sistema não deve ser afetado. Não é possível garantir essas três propriedades ao mesmo tempo (BREWER, 2000; GILBERT; LYNCH, 2002), porém é possível garantir quaisquer duas delas simultaneamente.

Sendo assim, torna-se inviável garantir as propriedades ACID (*Atomicity, Consistency, Isolation, Durability*), presente em sistemas de gerenciamento de banco de dados tradicionais. Isso ocorre pois, em geral, esses bancos tradicionais são desenvolvidos com o objetivo de garantir a consistência tanto dos dados quanto de transações realizadas neles. Para contornar esse problema, novas definições de consistência têm sido utilizadas, com foco na *consistência eventual*. Nesse sentido, as propriedades que têm se sobressaído na literatura são denominadas BASE (*Basically Available, Soft State, Eventually Consistent*) (PRITCHETT, 2008). Assim, considera-se a latência para sincronizar os dados recém-alterados, garantindo a consistência após um determinado intervalo de tempo.

Outra importante propriedade é denominada PACELC (*Partition, Availability and Consistency, else, Latency, Consistency*) (ABADI, 2012). Esse termo é dividido em duas partes. Na primeira parte considera-se a necessidade da tolerância a partições, em que deve-se ponderar se a aplicação priorizará a disponibilidade ou a consistência dos dados. Essa primeira parte está relacionada ao teorema CAP, fixando a existência da tolerância a partições. Já a segunda parte consiste na inexistência de partições e, conseqüentemente, da tolerância a essas partições. Nesse caso, para prover elevada disponibilidade da informação, os sistemas empregam a replicação dos dados. A atualização dos dados em todas as suas réplicas requer um determinado intervalo de tempo (ou seja, latência). Tendo em vista essa latência para que os dados estejam consistentes, deve-se ponderar entre a consistência efetiva de todas as réplicas e a latência para torná-las consistentes.

Com isso, surge o termo *Not Only SQL* (NoSQL), que refere-se a um grupo de sistemas gerenciadores de bancos de dados não relacionais (MONIRUZZAMAN; HOSSAIN, 2013), em que os dados não são exclusivamente armazenados em relações e a manipulação dos dados não é realizada apenas por meio da *Structured Query Language* (SQL). Sistemas NoSQL distribuídos são destinados ao armazenamento e processamento de grandes volumes de dados de forma massivamente paralela, promovendo a execução de consultas com baixa latência e o suporte a modelos flexíveis de armazenamento, replicação e distribuição dos dados (VIEIRA *et al.*, 2012).

Esta seção está subdividida em duas partes. Primeiramente, na Seção 2.3.1, descreve-se os quatro principais paradigmas de armazenamento em bancos de dados NoSQL. A partir disso, na Seção 2.3.2 detalha-se o paradigma de armazenamento orientado a coluna.

2.3.1 Paradigmas de Armazenamento NoSQL

Bancos de dados NoSQL são baseados em diferentes paradigmas de armazenamento de dados, em que os principais existentes são (SOUSA *et al.*, 2010; MONIRUZZAMAN; HOSSAIN, 2013; CHEN; MAO; LIU, 2014): chave-valor, baseado em grafos, orientado a documento e orientado a coluna. O paradigma chave-valor consiste no armazenamento dos dados por meio de uma estrutura conhecida como *Distributed Hash Table* (DHT). Nessa estrutura, os dados são representados por pares de chave e valor, em que o valor pode ser tanto dados simples, como um número ou um texto, quanto dados complexos, como imagens. Exemplos incluem Dynamo (SIVASUBRAMANIAN, 2012), Riak e Redis (REDMOND; WILSON, 2012). Já o paradigma baseado em grafos utiliza uma estrutura de dados de grafos para a representação dos dados. O principal exemplo deste paradigma é o Neo4j (REDMOND; WILSON, 2012).

O paradigma de armazenamento orientado a documento oferece um modelo de dados livre de esquemas, ou seja, permite que diferentes instâncias de uma entidade possuam estruturas de dados distintas. Os formatos dos documentos armazenados são geralmente XML, *JavaScript Option Notation* (JSON) e *Binary JSON* (BSON) (MONIRUZZAMAN; HOSSAIN, 2013). Exemplos incluem MongoDB e CouchDB (REDMOND; WILSON, 2012). Nesse paradigma, cada elemento da base de dados é representado por um documento, que pode ser estruturado ou semiestruturado. Os documentos estruturados são divididos em atributos, cada qual relacionado a um campo numérico, a um texto, a um vetor de outros atributos ou até mesmo a outro documento aninhado. Já nos documentos semiestruturados, os atributos podem variar entre cada documento presente na base, possibilitando até que alguns documentos sequer possuam atributos. Esse paradigma é considerado eficiente para armazenar dados em que cada instância possui atributos distintos, além de ser útil nos casos em que cada instância contém apenas textos e dados complexos ao invés de atributos simples. Dentro do contexto de ambientes de *data warehousing*, uma vez que um DW foi gerado a partir do processo de ETL (Seção 2.2), seus dados já estão estruturados, não aproveitando diretamente essa característica semiestruturada do paradigma de armazenamento orientado a documentos.

Na Seção 2.3.2 é detalhado o paradigma de armazenamento orientado a coluna, o qual é usado como base para o desenvolvimento desta dissertação de mestrado.

2.3.2 NoSQL Orientado a Coluna

O paradigma de armazenamento orientado a coluna consiste em particionar as tabelas verticalmente, em que cada coluna passa a representar uma coleção constituída pela PK da instância e o respectivo valor de um atributo (ou conjunto de atributos). Esse tipo de particionamento, ao ser comparado com o tradicional armazenamento orientado a linhas ou instâncias, pode promover elevados ganhos de desempenho (maiores do que uma ordem de magnitude) durante o processamento analítico dos dados (ABADI; MADDEN; HACHEM, 2008). Isto ocorre principalmente em operações de leitura, pois a maioria das consultas geralmente processa apenas

atributos específicos das tabelas. Logo, o número de operações de leitura em disco tende a ser reduzido. Considerando o contexto de consultas OLAP, composto majoritariamente por operações de leitura e pela agregação de alguns atributos de cada dimensão, este paradigma tende a ser uma abordagem interessante para armazenar um DW.

Os principais exemplos de bancos de dados NoSQL orientados a coluna são HBase (GEORGE, 2011; REDMOND; WILSON, 2012), Cassandra (LAKSHMAN; MALIK, 2010) e Accumulo (CORDOVA; RINALDI; WALL, 2015). Os três têm em comum um conceito denominado família de colunas, que consiste em uma divisão lógica dos atributos em uma tabela, geralmente traduzida para uma separação física dos dados em disco. A Figura 2.5 ilustra um exemplo do paradigma de armazenamento orientado a colunas, baseado em famílias de colunas. A tabela conceitual possui quatro atributos: PK, cliente, produto e unidades vendidas. Na Figura 2.5 é representado o armazenamento dessa tabela seguindo o paradigma orientado a coluna, em que os atributos estão agrupados em duas famílias de colunas, mantendo os atributos da mesma família próximas no disco. Essa separação em famílias de colunas beneficia consultas analíticas que agregam o total de unidades vendidas por produto, ao passo que para analisar o total de unidades vendidas por cliente é necessário realizar uma junção entre as duas famílias de colunas, o que pode ser custoso computacionalmente.

Figura 2.5 – Exemplo de como uma tabela conceitual pode ser armazenada segundo o paradigma orientado a coluna, organizado em famílias de colunas.

Dentre os três bancos de dados NoSQL investigados, o HBase se destaca por existirem trabalhos adaptando-o ao contexto de DW, como o CNSSB (DEHDOUH; BOUSSAID; BENTAYEB, 2014, que é descrito na Seção 3.4.1). HBase (GEORGE, 2011; REDMOND; WILSON, 2012) é um banco de dados NoSQL de código aberto, desenvolvido sob a *Apache Software Foundation* (GEORGE, 2011). Ele é utilizado em aplicações que necessitam armazenar e realizar um processamento intensivo de dados (KONISHETTY *et al.*, 2012). Cada um dos atributos de uma tabela é convertido para uma instância com o formato $\langle PK, família, coluna, timestamp, valor \rangle$, cujos campos são, respectivamente: (i) *PK*: número do registro; (ii) *família*: família de

coluna à qual o atributo pertence; (iii) *coluna*: nome da coluna; (iv) *timestamp*: data de inserção ou alteração do registro; e (iv) *valor*: valor do atributo.

Na Figura 2.6 é ilustrado o exemplo de uma tabela relacional sendo armazenada no HBase, representando cada atributo de acordo com o formato $\langle PK, família, coluna, timestamp, valor \rangle$. Nesse exemplo, as colunas *dia*, *mês* e *ano* podem ser agrupadas como membros da família *data*. Considerando que as famílias de colunas devem ser armazenadas próximas fisicamente, o HBase armazena cada família em um mesmo arquivo denominado *HFile*. As tabelas também podem ser particionadas horizontalmente em regiões, denominadas *RegionServers*, que armazenam faixas de PK da tabela (WHITE, 2012). Conforme novas instâncias são inseridas, elas são armazenadas dentro dos *RegionServers* existentes, porém essas regiões possuem um limite superior de tamanho, o qual é configurável no sistema. Caso um *RegionServer* exceda o limite máximo, ele é dividido em duas regiões, balanceando as instâncias entre essas regiões.

Figura 2.6 – Exemplo de organização de uma tabela relacional em famílias de colunas no HBase, representando o armazenamento lógico e físico da família de colunas “data” (inspirado em GEORGE, 2011).

2.4 Processamento Distribuído

Para manipular dados distribuídos em sistemas NoSQL, é necessário estudar como é realizado o processamento dos mesmos, a fim de realizar operações de agregação. Logo, nesta seção são detalhados dois principais *frameworks* para o processamento distribuído, que foram utilizados neste projeto: Hadoop MapReduce (Seção 2.4.1) e Spark (Seção 2.4.2).

A Figura 2.7 representa uma simplificação dos componentes da hierarquia de sistemas distribuídos utilizados durante este estudo. A base desses sistemas, localizada na parte inferior da Figura 2.7, é composta por um sistema de arquivos distribuído (por exemplo, o *Hadoop*

Distributed File System, conhecido por HDFS, detalhado na Seção 2.4.1). Acima disso estão localizados dois elementos: um banco de dados NoSQL, no caso o HBase, e um módulo de gerenciamento de recursos (denominado Yarn, existente desde a versão 2.0 do Hadoop). Uma aplicação pode interagir com os dados do HBase diretamente, por meio de operações simples como recuperação de uma tupla de uma tabela (ou um conjunto delas). Essa mesma aplicação também pode, por meio da integração do Yarn com o HBase, executar operações mais complexas por meio dos *frameworks* Hadoop MapReduce e Spark. Exemplos dessas operações complexas são a junção de tabelas e a agregação dos dados. Adicionalmente, a aplicação pode também interagir com ferramentas de interface SQL, responsáveis pela tradução de comandos SQL para o processamento distribuído.

Figura 2.7 – Principais componentes da hierarquia de sistemas distribuídos utilizados durante este estudo (adaptado de MOHAMMAD; BRESS; SCHALLEHN, 2012).

2.4.1 Hadoop MapReduce

O Hadoop é um *software* livre que visa confiabilidade, escalabilidade e computação distribuída. Em detalhes, o Hadoop enfoca o processamento distribuído de grandes conjuntos de dados em *clusters* de computadores usando modelos de programação simples. Seu projeto prevê o aumento de escalabilidade, desde um único servidor até milhares de servidores, cada um oferecendo recursos computacionais e de armazenamento. Adicionalmente, o Hadoop foi projetado para detectar e lidar com falhas na camada de aplicação, proporcionando um serviço de alta disponibilidade independente de recursos de *hardware*. Dentre os componentes do Hadoop (DOULKERIDIS; NØRVÅG, 2014), destacam-se o HDFS e o MapReduce, os quais são descritos a seguir.

Hadoop Distributed File System (HDFS)

O HDFS ([SHVACHKO et al., 2010](#)) é um sistema de arquivos distribuído capaz de armazenar grandes quantidades de dados, garantindo alta tolerância a falhas e podendo ser empregado em máquinas com *hardware* de baixo custo. Os arquivos armazenados são divididos em blocos de tamanho configurável e, por padrão, o HDFS mantém três réplicas de cada bloco. A divisão e a replicação do arquivo em blocos proveem um menor consumo de rede, além de aumentar a confiabilidade ([DOULKERIDIS; NØRVÅG, 2014](#)). O HDFS é baseado no sistema mestre-escravo, dividido entre *NameNode* e *DataNode*. O *NameNode*, nó mestre, tem o papel de manter metadados sobre os dados armazenados, enquanto os *DataNodes*, nós escravos, são responsáveis por armazenar os blocos de dados. Além disso, uma mesma máquina pode desempenhar os papéis de *NameNode* e *DataNode* simultaneamente.

Figura 2.8 – Exemplo de particionamento e armazenamento de um arquivo dentro do HDFS, considerando que ele foi dividido em quatro blocos de dados e que cada bloco foi replicado duas vezes (em respeito ao padrão de três réplicas por bloco) (adaptado de [ENGFER, 2011](#)).

Para acessar os dados armazenados no HDFS, primeiramente o cliente comunica-se com os *NameNodes* para obter os metadados sobre as localizações de blocos de interesse. A partir dessas localizações, são realizadas operações de escrita e leitura de dados diretamente nos *DataNodes*. A arquitetura do HDFS também inclui um *NameNode* secundário que funciona como um *backup* do *NameNode* primário, assumindo seu lugar quando necessário. A Figura 2.8 representa a estrutura do HDFS, exemplificando o armazenamento de um arquivo e detalhando sua organização em quatro *DataNodes*, que armazenam três réplicas de cada bloco que compõe o arquivo.

MapReduce

O MapReduce ([DEAN; GHEMAWAT, 2004](#)) é um *framework* de programação desenvolvido para processar grandes quantidades de dados de maneira paralela e distribuída, garantindo

simplicidade, alta escalabilidade, tolerância a falhas, flexibilidade e independência da técnica de armazenamento dos dados. O principal objetivo desse modelo consiste em esconder os detalhes de implementação e execução paralela, além de permitir aos programadores focar apenas nas estratégias de processamento de dados (LEE *et al.*, 2012). Assim, o modelo é considerado útil para processar dados de sistemas heterogêneos considerando diferentes ambientes de armazenamento de dados (SAKR *et al.*, 2011). O *framework* é baseado na definição de duas funções: *Map* e *Reduce*. A função *Map* processa pares chave-valor para gerar um conjunto intermediário de dados, enquanto a função *Reduce* combina os resultados intermediários associados a uma mesma chave para gerar o resultado. A Figura 2.9 exemplifica a execução do *framework* MapReduce para contar palavras de um texto, cuja sequência de ações é detalhada a seguir (SAKR *et al.*, 2011; LEE *et al.*, 2012; DOULKERIDIS; NØRVÅG, 2014):

Figura 2.9 – Execução do modelo MapReduce para contar palavras de um texto (adaptado de RAJARAMAN; ULLMAN, 2011).

1. *Split*: ao submeter o arquivo ao HDFS, ele é dividido em blocos. No caso do exemplo do contador de palavras, presente na Figura 2.9, cada bloco contém de três a cinco palavras presentes no documento original;
2. *Map*: para cada bloco, é gerado um processo Map para computar os dados e gerar pares chave-valor (LEE *et al.*, 2012). Ainda no exemplo do contador de palavras, a chave é a própria palavra e o valor é a quantidade de vezes que a palavra aparece dentro do bloco (ou seja, uma vez). Ao final deste processo, os dados de cada processo Map são armazenados temporariamente em disco;
3. *Shuffle*: esta etapa consiste na comunicação dos nós realizada por meio de protocolos HTTPS e funções *hash*, promovendo o agrupamento dos pares chave-valor, armazenados

temporariamente em disco, por meio da junção pela mesma chave. Ao final deste processo, cada palavra distinta (ou seja, chave) possui um vetor de valores;

4. *Reduce*: esses agrupamentos de pares chave-valor são atribuídos aos processos *Reduce*, que irão combinar esses pares, acrescentando os resultados de cada chave em um arquivo de resultados. Cada processo *Reduce* pode processar agrupamentos contendo chaves distintas, mas os dados de uma mesma chave não podem ser processados em diferentes processos *Reduce*. Ainda no exemplo do contador de palavras, o processo *Reduce* realiza a soma dos elementos do vetor de valores para emitir o resultado.

Tendo em vista este *framework* de processamento paralelo e distribuído, é necessário analisar como consultas SQL convencionais podem ser traduzidas para esses processos.

Tradução de consultas SQL em MapReduce

Uma ferramenta denominada MRBench (KIM *et al.*, 2008) detalha a adaptação de consultas SQL para o *framework* MapReduce, especificando a tradução das cláusulas *select*, *where*, *group by*, *having* e *join* em processos *Map* e *Reduce*. Essa adaptação ocorre da forma descrita a seguir. Durante a operação de projeção (cláusula *select*), os processos *Map* criam pares chave-valor, nos quais a chave corresponde à chave primária do registro e o valor possui os atributos selecionados (que podem estar estruturados na forma de um vetor, por exemplo). Enquanto os processos *Map* processam cada um dos registros da tabela para emitir um par chave-valor, é realizada a filtragem dos dados (cláusula *where* da consulta). Ou seja, se o registro não respeitar as restrições da consulta SQL, não será emitido um par chave-valor. Em ambas as operações descritas acima, não é necessário criar um processo *Reduce* para agrupar os dados.

Caso seja solicitado o agrupamento dos dados (cláusula *group by*), para cada registro da tabela, os processos *Map* geram pares chave-valor, em que a chave contém os atributos de agrupamento e o valor possui um ou mais atributos que são agregados. A partir disso, os processos *Reduce* calculam, para cada chave, a função de agregação sobre os respectivos valores, filtrando os dados caso necessário (cláusula *having*). Por exemplo, caso seja um agrupamento de soma, é realizada uma soma de todos os valores que possuem a mesma chave. A operação de junção (cláusula *join*) consiste em um processo *Map* gerando pares chave-valor, em que a chave contém os atributos de junção e o valor armazena os demais atributos necessários à consulta. Por fim, o processo *Reduce* combina os valores de mesma chave, gerando um par chave-valor, em que o valor contém os atributos das duas tabelas combinados.

Usando as adaptações destacadas anteriormente, é possível traduzir qualquer consulta SQL em processos *Map* e *Reduce*. Se uma mesma consulta processar todas as cláusulas descritas acima, é possível, na própria junção (cláusula *join*), realizar a filtragem (cláusulas *where* e *having*) e projeção (cláusula *select*) das tuplas, durante o processo *Map*. Caso sejam realizadas

outras junções ou operações de agrupamento, são criados múltiplos processos MapReduce, que são executados sequencialmente e salvam os resultados intermediários no HDFS. A Figura 2.10 ilustra o processamento de uma consulta SQL envolvendo três tabelas, realizando a junção entre elas e, ao mesmo tempo, aplicando as operações de filtragem, projeção e agrupamento.

Relacionadas a essas traduções descritas pela ferramenta MRBench, existem programas que realizam uma tradução genérica e automática de comandos SQL para o *framework* MapReduce, como o Hive¹. O sistema Hive é uma solução em código aberto que incorpora ao Hadoop funcionalidades de *data warehousing* (THUSOO *et al.*, 2009). Ele oferece suporte para consultas expressas em uma linguagem declarativa denominada *HiveQL*, a qual é responsável por compilar comandos similares ao SQL (*SQL-like*) em processos MapReduce. Esta linguagem permite operações de definição de dados (DDL), como exemplo criação e remoção de tabelas, além de operações de manipulação dos dados (DML), abrangendo seleção, projeção, agregação, união e subconsultas. Adicionalmente, o sistema possui um repositório de metadados que armazena esquemas e estatísticas. O sistema Hive pode ser integrado a outras soluções, como o HBase e o Spark, servindo ao propósito de compilar comandos em uma linguagem similar à SQL para processos MapReduce dentro desses sistemas.

Figura 2.10 – Exemplo do processamento em MapReduce de uma consulta SQL envolvendo três tabelas.

Outro *framework* para processamento paralelo e distribuído é denominado Spark, que é detalhado na Seção 2.4.2. Esse *framework* pode ser considerado um concorrente do Hadoop MapReduce, embora o Spark também possa utilizar como sistema de arquivos distribuídos o HDFS.

2.4.2 Spark

Apache Spark é um *framework* que acrescenta componentes em memória distribuídos entre os nós do cluster, os quais podem ser reutilizados pelas operações realizadas. Como consequência, este *framework* consegue otimizar a velocidade de leitura e processamento em

¹ <https://hive.apache.org/>

comparação ao Hadoop MapReduce em até dez vezes (ZAHARIA *et al.*, 2010), além de incluir algoritmos de aprendizado de máquina e ferramentas de análise de dados de forma interativa. Considerando a Figura 2.9, os resultados intermediários entre os processos Map e Reduce podem ser mantidos em memória, ao invés de serem escritos em disco ao final de um processo Map e lidos do disco no começo do próximo processo Reduce. Além disso, dados intermediários entre múltiplos processos MapReduce, executados sequencialmente, também podem ser mantidos em memória para otimizar o processamento. O Spark é composto por *Resilient Distributed Datasets* (RDDs) e operações paralelas nestes *datasets*.

Um RDD consiste em uma coleção particionada de objetos entre os nós do *cluster* (ZAHARIA *et al.*, 2012), cujos elementos não necessariamente estão fisicamente armazenados em disco. Os RDDs são obtidos a partir da leitura dos arquivos armazenados em um sistema distribuído e de transformações a partir de outros RDDs. Como exemplo de transformações, pode-se citar operações de mapeamento, junção e filtragem de dados. Já o processamento desses dados é composto por ações executadas sobre os RDDs. Essas ações podem ser, por exemplo, a contagem do número de elementos do RDD ou uma redução pela chave, procedimento similar ao processo Reduce ilustrado na Figura 2.9. Devido à característica deste *framework* de reaproveitar resultados intermediários para futuros processos, o Spark é indicado para aplicações iterativas.

O Spark possui uma interface, denominada *DataFrame* (ARMBRUST *et al.*, 2015), que possibilita a integração de programas desenvolvidos usando o *framework* Spark com a linguagem SQL. *Dataframes* são coleções estruturadas de registros que podem ser manipuladas de forma análoga às tabelas de bancos de dados relacionais. Esses *Dataframes* podem ser formados a partir de arquivos armazenados em um sistema distribuído ou a partir de um RDD. Essa interface tem o intuito de facilitar o desenvolvimento de aplicações e de possibilitar a otimização de operações analíticas sobre os dados, uma vez que ela provê a manipulação dos dados sendo processados pelo Spark por meio de comandos usando a linguagem SQL.

Por fim, o Spark também fornece uma implementação SQL chamada Shark, a qual realiza operações similares ao Hive, porém incorpora duas otimizações principais (XIN *et al.*, 2013): organização dos RDDs em armazenamento baseado em coluna; e utilização de estatísticas dos dados para otimizar a consulta dinamicamente durante a execução. No processo de análise e compilação de um comando SQL, enquanto o Hive gera processos MapReduce, o Shark monta um plano de acesso baseado em transformações e ações em RDDs. Com isso, essa implementação SQL possibilita a execução de consultas analíticas complexas de maneira rápida, mantendo tolerância a falhas.

2.5 Considerações Finais

Neste capítulo foram investigados os conceitos fundamentais relacionados à proposta desta dissertação de mestrado. Foram descritos com mais detalhes conceitos sobre DW, bancos

de dados NoSQL, com destaque ao paradigma de armazenamento orientado a colunas, e dois principais *frameworks* de processamento distribuídos. Também foram explicados conceitos referentes ao esquema estrela e à normalização dos dados da tabela. Uma vez que, normalmente, os sistemas de arquivos distribuídos empregam a replicação de dados para prover disponibilidade, uma abordagem interessante consiste em desnormalizar o esquema do DW, conforme discorrido nos trabalhos correlatos da Seção 3.4.

Nos principais componentes do Hadoop, o HDFS e o MapReduce, existem alguns parâmetros configuráveis que podem influenciar o desempenho do processamento dos dados. Exemplos desses parâmetros consistem na quantidade de processos Map e Reduce executando simultaneamente sobre um arquivo do HDFS (KAMBATLA; PATHAK; PUCHA, 2009) e configurações de memória secundária que afetam e otimizam a etapa de *Shuffle* (DESANI *et al.*, 2016). Alguns desses parâmetros, relacionados ao armazenamento (tanto no HDFS quanto no HBase) e processamento dos dados, foram explorados durante as análises presentes realizadas neste estudo.

Em resumo, conforme ilustrado na Figura 2.11, o DW tem sido predominantemente implementado em ambientes centralizados, sendo que adaptações estão sendo propostas na literatura para permitir que DWs sejam armazenados e manipulados usando bancos de dados NoSQL distribuídos, conforme detalhado no Capítulo 3. Adicionalmente, considerando que o foco desta dissertação é investigar a aplicação do *benchmark* SSB para avaliar um banco de dados NoSQL orientado a coluna, no Capítulo 3 são explicados os conceitos relacionados à técnica de *benchmarking*. Além disso, são descritos os principais *benchmarks* existentes na literatura, juntamente com os trabalhos relacionados a esta pesquisa, envolvendo adaptações dos *benchmarks* para DW em bancos de dados NoSQL.

Figura 2.11 – Ilustração do processo de adaptação de um DW, geralmente armazenado em bancos de dados relacionais centralizados, para sistemas NoSQL distribuídos.

BENCHMARKS DE BANCOS DE DADOS

Neste capítulo são investigados os principais benchmarks existentes, incluindo os trabalhos relacionados e correlatos a este estudo. Primeiramente é descrita a técnica de benchmark de DW e, a partir dela, são detalhados os principais benchmarks tanto para DWs quanto para sistemas distribuídos e bancos de dados NoSQL. Esses benchmarks podem ser considerados como trabalhos relacionados a esta dissertação, logo, é descrita a limitação deles para o estudo realizado. Em especial, também são detalhadas extensões dos benchmarks de DW, adaptando-os majoritariamente para bancos de dados NoSQL orientados a colunas. Cada uma dessas adaptações são consideradas correlatas a este trabalho e, por conta disso, são descritas as limitações de cada uma delas, relacionando-as com as seções do Capítulo 4, o qual contém as investigações e as propostas realizadas nesta dissertação.

3.1 Considerações Iniciais

Benchmarks são ferramentas para responder perguntas como: “Qual é o melhor sistema em um determinado domínio?” ou, considerando como domínio o ambiente de *data warehousing*, “Qual é o melhor sistema de bancos de dados para operações OLAP?” (FOLKERTS *et al.*, 2012). Essas perguntas são respondidas por meio de abordagens sistemáticas, baseadas nas propriedades e restrições de cada um dos sistemas sendo avaliados. Assim, o objetivo de um *benchmark* é reportar comparativamente a qualidade e o desempenho de diferentes sistemas, levando em consideração as características e limitações de cada um. Isso serve como base para auxiliar, por exemplo, uma empresa na escolha de um sistema para atender a suas necessidades em um determinado domínio. Na prática, os *benchmarks* são utilizados para auxiliar decisões referentes à estratégia de aquisição de uma infraestrutura mais econômica e que, ao mesmo tempo, proporciona maiores ganhos de desempenho.

Um *benchmark* deve cumprir quatro requisitos (BOG, 2013): (i) relevância: as métricas coletadas devem auxiliar na comparação entre sistemas dentro de um domínio específico; (ii) portabilidade: o *benchmark* deve ser facilmente adaptável a diferentes sistemas e arquiteturas; (iii) escalabilidade: o *benchmark* deve ser capaz de medir o comportamento do sistema considerando diferentes volumes de dados; e (iv) simplicidade: os resultados do *benchmark*

devem ser facilmente interpretáveis. Além desses requisitos, existem também outros aspectos que são importantes para um *benchmark*: (v) repetibilidade: o *benchmark* deve sempre gerar os mesmos resultados toda vez que for executado para a mesma base de dados; (vi) justiça: todos os sistemas comparados devem ser capazes de participar igualmente dos experimentos; (vii) verificabilidade: o resultado do *benchmark* é interpretado como real; e (viii) econômico: a execução deve ser financeiramente acessível.

Note que o objetivo desta dissertação não é propor um *benchmark*, mas sim investigar diferentes aspectos de interesse que podem ser utilizados na posterior proposta de um *benchmark* para DW armazenado em um banco de dados NoSQL. Ou seja, a teoria de *benchmark* fornece a fundamentação teórica para o trabalho desenvolvido. Dentro deste contexto, o capítulo está organizado da seguinte forma. Na Seção 3.2 são descritas as quatro principais etapas de um *benchmark* para DW, investigadas durante a execução deste projeto. Tendo em vista essas etapas, na Seção 3.3 são descritos três principais *benchmarks* para DW: o TPC-H (Seção 3.3.1), o TPC-DS (Seção 3.3.2) e o *Star Schema Benchmark* (SSB) (Seção 3.4), bem como a limitação deles para esta dissertação. Em especial, nas Seções 3.3.1.1, 3.3.1.2 e 3.4.1 a 3.4.4 são explicadas as adaptações desses *benchmarks* para serem aplicados em bancos de dados NoSQL distribuídos, considerados trabalhos fortemente relacionados ao estudo realizado. Na Seção 3.5 são detalhados os principais *benchmarks* para sistemas distribuídos e bancos de dados NoSQL, como o Yahoo! *Cloud Serving Benchmark* (YCSB) (Seção 3.5.1), o BigBench (Seção 3.5.2) e o TPCx-HS (Seção 3.5.3), novamente explicando a limitação de cada um deles em relação ao estudo realizado. Por fim, na Seção 3.6 são descritas as considerações finais do capítulo.

3.2 Etapas de um *Benchmark* de *Data Warehouse*

As quatro principais etapas de um *benchmark* de bancos de dados são (CIFERRI, 1995): (i) esquema e carga de trabalho; (ii) geração de dados; (iii) métricas e parâmetros; e (iv) validação. Na etapa (i) é definido o esquema de acordo com a aplicação que deseja-se avaliar. Considerando o ambiente de *data warehousing*, e sistemas relacionados, modela-se um DW seguindo o esquema estrela ou suas variações. Já em relação à carga de trabalho da etapa (i), são definidas operações de manipulação de dados frequentes, as quais são processadas na aplicação avaliada. Dentro do contexto de DW, essas operações são representadas por consultas OLAP, simbolizadas pelo processamento de agregações (considerando diferentes seletividades) e por operações como *drill-down* e *roll-up*.

Na etapa (ii) são gerados os dados que compõem o esquema estruturado durante a etapa (i). O processo de geração pode ser baseado em dados reais e/ou sintéticos. Os dados reais referem-se a uma aplicação contendo informações (ou parte delas) do ambiente que se deseja avaliar. Já os dados sintéticos estão relacionados à geração artificial de dados, o que possibilita a inserção de características especiais de modo a identificar possíveis deficiências

nos sistemas a serem analisados. Por exemplo, gerando-se dados artificialmente seguindo a distribuição uniforme, torna-se possível estimar e variar a seletividade das consultas OLAP, de forma a avaliar o impacto de consultas pouco seletivas sobre grandes conjuntos de dados ou de consultas muito seletivas sobre pequenos conjuntos de dados.

Na etapa (iii) são definidas métricas e parâmetros que devem ser analisados. As métricas são informações qualitativas e quantitativas extraídas durante a aplicação do *benchmark*, como a funcionalidade de algumas operações (testes qualitativos) e o intervalo de tempo para processar essas operações (testes quantitativos). Já os parâmetros são valores configuráveis do sistema ou da aplicação que podem ser variados. Um exemplo clássico de parâmetro é o volume de dados sintéticos analisado, o qual pode ser alterado para explorar o comportamento do DW frente ao seu crescimento. Outros exemplos incluem parâmetros de organização dos dados internamente aos sistemas, como presença de índices ou tamanho de blocos de dados contínuos alocados no disco.

Por fim, a etapa (iv) é denominada validação e consiste na aplicação do esquema e da carga de trabalho (i), juntamente à geração das tabelas e à inserção dos dados (ii), para explorar parâmetros a fim de coletar medidas qualitativas e quantitativas (iii), necessárias à comparação entre os diferentes sistemas analisados.

3.3 Benchmarks para Data Warehouse

Nessa seção são detalhados os *benchmarks* para avaliação de desempenho de DWs convencionais. Na literatura existem três principais *benchmarks* (BARATA; BERNARDINO; FURTADO, 2014): TPC-H, TPC-DS e *Star Schema Benchmark* (SSB). Os *benchmarks* desenvolvidos pela organização *Transaction Processing Performance Council*¹ (TPC) são detalhados nas Seções 3.3.1 e 3.3.2. Já o SSB, *benchmark* focado neste projeto, é descrito em detalhes na Seção 3.4.

3.3.1 TPC-H

O TPC-H (POESS; FLOYD, 2000) é um *benchmark* que define uma aplicação de DW relacionada a dados históricos referentes a pedidos e vendas de uma empresa de varejo, durante um certo período de tempo. O esquema é categorizado como constelação de fatos, conforme ilustrado na Figura 3.1, sendo composto por oito tabelas: duas tabelas de fatos (*LineItem* e *PartSupp*) e seis tabelas de dimensão (*Orders*, *Part*, *Supplier*, *Customer*, *Nation* e *Region*). Já a carga de trabalho é composta por diferentes consultas OLAP. Ao todo são 22 consultas, as quais abordam os seguintes aspectos (FORLANI, 2006): (i) determinação de preços e promoções; (ii) gerenciamento de oferta e procura; (iii) gerenciamento de lucros e ganhos; (iv) estudo da

¹ <http://www.tpc.org/>

satisfação do consumidor; (v) estudo da participação no mercado; e (vi) gerenciamento de remessas.

Este *benchmark* possui internamente um programa denominado *dbgen*, responsável pelo processo de geração dos dados sintéticos. Para cada uma das tabelas, este programa gera os dados em formato texto, em que cada linha representa uma instância na qual cada atributo é separado por uma barra vertical (“|”). A quantidade de dados gerados, ou seja, a cardinalidade de cada tabela varia de acordo com o fator escala (*scale factor* - SF) utilizado, em que esse valor é ilustrado na Figura 3.1, acima de cada uma das tabelas. Por exemplo, a tabela *LineItem* possui aproximadamente seis milhões de tuplas com SF = 1, ou doze milhões de tuplas com SF = 2.

Figura 3.1 – Esquema do *benchmark* TPC-H (adaptado de O’NEIL *et al.*, 2009). A cardinalidade de cada tabela é ilustrada acima dela. Esse valor pode ser estático, no caso das dimensões *Region* e *Nation*, ou baseado no SF (*scale factor*), como a tabela de fatos *LineItem*.

Levando em consideração o parâmetro SF, a principal métrica do TPC-H é o desempenho, calculada por meio do número de consultas executadas por hora. Essa métrica é expressa da forma *Query-per-Hour Performance Metric* (QphH@SF), tendo uma medida relativa a cada SF utilizado. O QphH@SF possibilita a comparação entre diferentes sistemas (usando o mesmo SF) ou a análise da variação de desempenho conforme a cardinalidade das tabelas aumentam (por meio de SFs distintos). Outra medida quantitativa pouco usual é denominada preço-desempenho, que calcula a razão entre o custo da infraestrutura do sistema pelo desempenho das consultas.

Uma vez que este *benchmark* foi desenvolvido para bancos de dados relacionais, para que seja possível utilizá-lo em bancos NoSQL, é necessário adaptá-lo. Nesse sentido, já existem

alguns trabalhos na literatura que têm esse objetivo. Esses trabalhos são considerados como trabalhos correlatos à presente dissertação, sendo que existem duas principais abordagens que investigam a avaliação de sistemas distribuídos usando o *benchmark* TPC-H (MOUSSA, 2012; MOUSSA, 2013). No primeiro trabalho, detalhado na Seção 3.3.1.1, MOUSSA (2012) descreve experimentos realizados por meio da execução do TPC-H no *framework* Hadoop MapReduce. Já no segundo trabalho, explicado na Seção 3.3.1.2, MOUSSA (2013) propõe novas métricas para um *benchmark* de ambientes de *data warehousing* distribuídos.

3.3.1.1 Processamento das consultas do TPC-H usando Hadoop MapReduce

No primeiro trabalho, MOUSSA (2012) analisa o armazenamento dos dados do DW no HDFS, ao mesmo tempo em que consultas são processadas. Neste cenário foram realizados testes de escalabilidade, analisando-se o desempenho das 22 consultas do TPC-H sobre um cluster de 3, 5 e 8 nós e utilizando-se como volume de dados os valores de SF = 1 (1.1 GB) e SF = 10 (11 GB). Os testes mostraram que, independente do número de nós, o tempo médio de cada consulta, considerando SF = 10, foi aproximadamente duas vezes maior que a média das consultas para SF = 1. Com base nesses resultados, MOUSSA (2012) argumenta que um ambiente distribuído é indicado para manipular grandes volumes de dados, uma vez que, aumentando-se o tamanho dez vezes, o tempo das consultas aumentou apenas duas vezes.

Outra investigação realizada foi a variação do esquema, realizando-se testes sobre o TPC-H desnormalizado, ou seja, juntando-se todas as tabelas do esquema na tabela de fatos *LineItem*. Sobre esse novo esquema, foram novamente realizados testes de escalabilidade sobre um *cluster* de 3, 5 e 8 nós. Já o volume, devido à redundância de dados causada pela desnormalização, foi de 4.5 GB (SF = 1) e de 45 GB (SF = 10). Nesta abordagem, para SF = 1, o *cluster* com 5 nós exibiu menores tempos para processamento das consultas, enquanto para SF = 10 a melhor configuração foi um cluster com 8 nós. MOUSSA (2012) concluiu que, conforme o volume de dados aumenta, o número de nós também deve aumentar. Adicionalmente, verificou-se que, se algum *cluster* com muitos nós armazenar poucos dados, o desempenho das consultas pode ser prejudicado.

Esse trabalho correlato é interessante por analisar dois aspectos: escalabilidade do volume de dados em relação ao tamanho do *cluster* (quantidade de nós) e análise de um DW desnormalizado, baseado no esquema do TPC-H. Porém, as análises efetuadas usaram como base apenas o sistema de arquivos distribuído HDFS. Não foram considerados bancos de dados NoSQL distribuídos, com os diferentes paradigmas de armazenamento, como o baseado em colunas. Neste estudo são investigados alguns parâmetros que influenciam no desempenho do processamento de dados, usando o *framework* Hadoop MapReduce, armazenados no HDFS (usando um banco de dados NoSQL ou não). Uma vez que é importante saber a relação desses parâmetros com o desempenho do processamento de consultas OLAP, eles são explorados durante esse estudo, cujos detalhes são descritos na Seção 4.4.

3.3.1.2 Proposta de métricas para avaliar ambientes distribuídos

No segundo trabalho, [MOUSSA \(2013\)](#) discute que o TPC-H não é suficiente para validar algumas características de ambientes distribuídos, como escalabilidade, elasticidade e tolerância a falhas. Nele são introduzidos os principais problemas do TPC-H ao avaliar sistemas distribuídos, dentre os quais é possível destacar três. O primeiro problema refere-se ao fato de que o TPC-H não reflete implementações modernas de sistemas para ambientes de *data warehousing*, considerando ferramentas para geração de relatórios e mineração de dados. Quanto ao segundo problema, destaca-se que o TPC-H não valida a escalabilidade do sistema, uma vez que a principal métrica utilizada é a quantidade de consultas por hora dado um volume estático de dados. O terceiro problema diz respeito ao fato de que TPC-H não é capaz de validar todas as combinações do teorema CAP (Seção 2.3), a saber: consistência e disponibilidade; consistência e tolerância a partições; e disponibilidade e tolerância a partições.

Assim, em [MOUSSA \(2013\)](#) são propostas novas métricas e requisitos de um *benchmark* distribuído, conforme descrito a seguir. Um *benchmark* para ambiente distribuído deve focar na análise sobre o desempenho do sistema sendo investigado, incluindo a velocidade de conexão e congestionamento de rede. Ele também deve realizar testes de escalabilidade com cargas de dados crescentes, medindo a taxa de transferência de dados. Adicionalmente, devido à elasticidade, o *benchmark* deve mensurar os impactos na adição ou remoção de recursos no sistema sem interromper o serviço. A disponibilidade do sistema é outra métrica a ser analisada. Ou seja, o *benchmark* deve permitir a análise do custo relacionado à quantidade de réplicas, à possibilidade de falhas e ao custo para recuperar-se de tais falhas. Por fim, o *benchmark* deve ponderar eficiência e desempenho, ou seja, deve permitir a validação do sistema não apenas em busca do menor tempo de execução (desempenho), mas procurando o maior poder computacional com o menor custo (eficiência).

Embora esse trabalho correlato seja direcionado ao *benchmark* TPC-H, ele pode ser aplicado ao SSB, porém ainda difere deste estudo por realizar uma análise apenas sobre a etapa (iii) de um *benchmark* para DW (Seção 3.2). Pensando-se nas outras etapas, não foram considerados alguns aspectos como os diferentes paradigmas de armazenamento e o impacto deles no esquema avaliado, conforme é investigado neste estudo e detalhado na Seção 4.2.1. Adicionalmente, apesar de as métricas descritas serem relevantes, faltou uma análise sobre os parâmetros que influenciam essas métricas, como o impacto do tamanho do bloco de dados distribuído sobre o desempenho do processamento das consultas, como é investigado neste estudo e detalhado na Seção 4.4.

3.3.2 TPC-DS

O TPC-DS ([POESS et al., 2002](#)) consiste em um *benchmark* para sistemas de suporte à decisão. O esquema consiste em uma constelação de fatos, possuindo um total de 24 tabelas, divididas entre 7 tabelas de fatos e 17 tabelas de dimensão compartilhadas entre as tabelas de

fatos. A Figura 3.2 representa uma simplificação deste esquema, exibindo apenas as tabelas de fatos e o relacionamento entre elas. Não são exibidos os atributos nem as tabelas de dimensão devido à elevada quantidade de elementos, que poderia prejudicar a legibilidade da figura. Em relação às consultas, o TPC-DS define ao todo 99 consultas distintas, categorizadas em quatro classes principais (NAMBIAR; POESS, 2006): (i) consultas *ad-hoc*; (ii) geração de relatórios; (iii) consultas iterativas (por exemplo, *drill-down* e *roll-up*); e (iv) consultas de extração de dados.

Figura 3.2 – Esquema resumido do *benchmark* TPC-DS (adaptado de BARATA; BERNARDINO; FURTADO, 2014). Devido à elevada quantidade de tabelas e atributos, esta figura ilustra apenas as tabelas de fatos, em que as dimensões como *Customer* ou *Item* não são exibidas.

Em relação à carga de trabalho, este *benchmark* engloba tanto consultas *ad-hoc* (adaptadas do TPC-H) quanto operações de manutenção dos dados (NAMBIAR; POESS, 2006). O termo *ad-hoc* refere-se às consultas que não podem ser pré-determinadas até o momento em que são executadas, diferentemente de consultas para geração de relatórios. O TPC-DS armazena dados sobre inteligência de negócios, contendo informações detalhadas de consumidores, pedidos e produtos, cada qual possuindo sua forma particular de venda: em lojas, por meio de catálogos ou pela internet.

Uma vez que o TPC-DS foi planejado para grandes volumes de dados, com SFs variando entre 100 e 100.000, a escalabilidade dos dados combina tanto cardinalidade da tabela quanto domínio de alguns atributos. Assim, conforme a cardinalidade das tabelas aumenta, o domínio de alguns dos atributos também pode crescer. A geração dos dados de cada tabela é similar à do TPC-H, realizada por meio da execução de um programa denominado *dbgen*. Por fim, o TPC-DS avalia três principais métricas (NAMBIAR; POESS, 2006): (i) desempenho: quantidade de consultas executadas por hora; (ii) preço-desempenho: razão entre o custo de manutenção de *hardware* e *software* e a quantidade de consultas por hora; e (iii) disponibilidade: período no qual o sistema esteve disponível para o cliente.

Contudo, similarmente ao TPC-H, este *benchmark* foi desenvolvido para bancos de dados relacionais. Adicionalmente, ele possui como único parâmetro o SF, em que seu principal diferencial consiste tanto na geração de um volume maior de dados quanto na adição de novas operações à carga de trabalho, como outros tipos de consultas além das definidas pelo TPC-H. Já existe um trabalho na literatura que adapta o *benchmark* TPC-DS para ambientes distribuídos. Esse trabalho, denominado BigBench, é considerado correlato ao trabalho desenvolvido nesta dissertação, e é detalhado na Seção 3.5.2.

3.4 Star Schema Benchmark

O *Star Schema Benchmark* (SSB) (O'NEIL *et al.*, 2009) é uma extensão do TPC-H, que modifica seu esquema, ilustrado na Figura 3.1, para um esquema estrela. Essa extensão foi realizada por meio das seguintes modificações (NASCIMENTO, 2013): (i) junção das tabelas de fatos *LineItem* e *Order*, resultando na tabela de fatos *LineOrder*, a qual possui todos os atributos das tabelas originais; (ii) exclusão da tabela de fatos *PartSupp* devido à diferente granularidade temporal, já que as atualizações feitas nas tabelas *LineOrder* e *PartSupp* não ocorrem ao mesmo tempo; (iii) exclusão de alguns atributos das tabelas *LineItem* e *Order* do TPC-H, já que se tratam de atributos do tipo texto não estruturados, que não podem ser armazenados para serem agrupados ou sumarizados, nem para contextualizar os fatos analisados; e (iv) adição da tabela de dimensão *Date*, devido à necessidade de armazenar o histórico de vendas da companhia.

Figura 3.3 – Esquema do benchmark SSB (adaptado de O'NEIL *et al.*, 2009). Acima de cada tabela está indicada a quantidade aproximada de tuplas, em que a dimensão *Date* possui um número fixo de instâncias (representando um período de 7 anos em dias), ao passo que a cardinalidade das outras tabelas está em função do SF (*scale factor*).

Diferentemente do TPC-H e do TPC-DS, o SSB possui as tabelas de dimensão desnormalizadas, proporcionando melhor desempenho no processamento de consultas, além de simplicidade e facilidade para os usuários entenderem e manipularem os dados do esquema (KIM-

BALL; ROSS, 2002). A Figura 3.3 ilustra o esquema deste *benchmark*, a qual representa a tabela de fatos *LineOrder* localizada visualmente no centro e as tabelas de dimensão *Customer*, *Supplier*, *Part* e *Date* nas extremidades do esquema.

Em relação à geração sintética de dados, o processo é similar ao TPC-H, onde a geração é realizada por um programa interno do SSB denominado *dbgen*. Este programa, dado um SF, produz dados para cada uma das tabelas no formato texto, em que cada linha representa uma instância e seus atributos são separados por uma barra vertical (“|”). Acima de cada tabela da Figura 3.3 há a informação sobre sua cardinalidade, de acordo com o SF utilizado, cujo valor pode variar desde 1 até 1000.

Outra característica importante da geração dos dados é que alguns atributos possuem distribuição uniforme, em que o tamanho de domínio desses atributos é mostrado na Tabela 3.1. Por meio dessa distribuição, é possível determinar a seletividade das consultas OLAP definidas pela carga de trabalho do SSB. Este *benchmark* define ao todo 13 consultas, das quais algumas foram adaptadas do TPC-H (considerando as adaptações no esquema) enquanto outras foram criadas para validar o novo esquema. Essas consultas são divididas em quatro classes, a fim de realizar cobertura funcional (diferentes tipos de consultas comuns ao esquema estrela) e testar variações tanto de seletividade quanto de nível de agregação.

Tabela 3.1 – Tamanho de domínio dos atributos do SSB, os quais possuem distribuição uniforme.

atributo	cardinalidade
<i>d_year</i>	7
<i>d_yearmonth</i>	84
<i>d_weeknuminyear</i>	53
<i>lo_discount</i>	11
<i>lo_quantity</i>	50
<i>s_region</i> e <i>c_region</i>	5
<i>s_nation</i> e <i>c_nation</i>	25
<i>s_city</i> e <i>c_city</i>	250
<i>p_mfgr</i>	5
<i>p_category</i>	25
<i>p_brand1</i>	1000

As consultas de classe 1 agregam dados da tabela de fatos *LineOrder* juntamente com a dimensão *Date* por meio da soma total (sem dividir os dados em diferentes grupos). Já as consultas de classes 2, 3 e 4 agregam os dados por meio da soma, os quais são separados em grupos e originários da união da tabela de fatos *LineOrder* com as dimensões *Date*, *Part* e *Supplier*, no caso da classe 2, *Date*, *Customer* e *Supplier*, no caso da classe 3, e todas as tabelas do esquema, no caso da classe 4. Por meio dessas consultas, a métrica do SSB consiste no desempenho, calculada pelo número de consultas executadas por hora. Assim como no TPC-H, essa métrica é dependente do SF utilizado durante o processo de geração dos dados.

O SSB é focado na análise de desempenho de bancos de dados relacionais, precisando ser estendido para bancos de dados NoSQL distribuídos, que são o foco desta pesquisa. Neste sentido, já existem alguns trabalhos na literatura que têm esse objetivo. Esta seção descreve as abordagens existentes para realizar-se tal extensão, consideradas como principais trabalhos correlatos a este projeto. A seção está organizada da seguinte forma: na Seção 3.4.1 é descrita a primeira adaptação do SSB, denominada CNSSB, para o paradigma armazenado em colunas usando o HBase; na Seção 3.4.2 é realizado um estudo sobre formas de adaptar a modelagem multidimensional de um DW em bancos de dados NoSQL orientados a documentos e a colunas; na Seção 3.4.3 são detalhadas variações do esquema CNSSB, considerando-se diferentes agrupamentos de atributos em famílias de colunas; e na Seção 3.4.4 é descrita uma adaptação do processo de geração de dados para NoSQL distribuídos denominada SSB+.

3.4.1 Columnar NoSQL Star Schema Benchmark

O trabalho de DEHDOUH; BOUSSAID; BENTAYEB (2014) propõe uma extensão do SSB, chamada *Columnar NoSQL Star Schema Benchmark* (CNSSB), a qual analisa o desempenho da geração de dados e a realização de consultas em bancos de dados NoSQL baseados em colunas. A principal adaptação dessa extensão consiste na desnormalização das tabelas do esquema estrela, unindo as tabelas de dimensão *Customer*, *Supplier*, *Part* e *Date* com a tabela de fatos *LineOrder*. Nesse esquema, foram removidas as cinco FKs da tabela *LineOrder* e o atributo *d_datekey* da tabela *Date* do esquema original do SSB, presente na Figura 3.3. Assim, todos os 53 atributos restantes foram organizados dentro da tabela *LineOrder*, cada qual dentro da família de colunas correspondente a sua tabela original (*fc_customer*, *fc_supplier*, *fc_part* e *fc_date*).

Além dessa adaptação no esquema, a carga de trabalho foi modificada. As 13 consultas do SSB foram ajustadas para esse sistema, adequando o nome das colunas (adicionando também as famílias de colunas) e removendo tanto as tabelas de dimensão quanto as FKs usadas para junção das tabelas, as quais não são mais necessárias. Para armazenar o esquema foi utilizado o banco de dados NoSQL HBase, ao passo que as consultas foram processadas por meio de duas interfaces SQL: Hive e Phoenix² (interface SQL similar ao Hive, porém específica para o HBase). A partir dessas adaptações, foram realizados dois experimentos.

O primeiro experimento enfocou o processo de geração de dados, comparando dois aspectos: volume de dados (de acordo com o SF) e tempo de geração. Em relação ao volume de dados, enquanto o SSB gerou 56.8 GB de dados com SF = 100 e 580 GB com SF = 1000, o CNSSB gerou 235 GB com SF = 100 e 2360 GB com SF = 1000. Ou seja, considerando os mesmos valores de SF, o CNSSB gerou um volume de dados bem maior (devido à desnormalização), o que pode ser interessante para simular grandes ambientes de *data warehousing* em NoSQL. Porém, em relação ao tempo, o CNSSB foi mais lento que o SSB para gerar os dados. Já o segundo experimento enfocou o tempo de execução das consultas, seguindo a métrica do SSB,

² <http://phoenix.apache.org/>

com a finalidade de comparar as interfaces Hive e Phoenix. Foi utilizado o sistema HBase em um *cluster* com 15 nós, sobre aproximadamente 1 TB de volume de dados. Os resultados indicaram que o Hive teve um pior desempenho no processamento das consultas. Isso ocorreu porque, ao integrar o Hive ao HBase, é necessário que os dados armazenados sejam migrados para o Hive antes de as consultas serem processadas, o que leva a um maior tempo de processamento.

Este trabalho correlato representou uma primeira adaptação do SSB para um banco de dados orientado a colunas, o qual geralmente é estruturado em famílias de colunas. Porém, a distribuição de atributos entre as famílias de colunas foi fixa, levantando o questionamento se existem impactos ao variar essa organização. A presente dissertação supre essa limitação, sendo que as investigações realizadas são detalhadas na Seção 4.2. Outra limitação do CNSSB é relacionada aos parâmetros, uma vez que essa primeira adaptação foi superficial e, portanto, não contém detalhes sobre o processamento distribuído e como ele impacta no processamento de consultas OLAP. Neste trabalho, por outro lado, investiga-se esses aspectos, conforme descrito na Seção 4.4.

3.4.2 Modelagem Multidimensional em bancos de dados NoSQL

O trabalho de CHEVALIER *et al.* (2015d) descreve uma abordagem para traduzir uma modelagem conceitual, mais especificamente um cubo de dados (como ilustrado na Figura 2.1), para bancos de dados NoSQL. Dentre os paradigmas de armazenamento, foram analisados os paradigmas orientado a coluna, usando o HBase, e orientado a documentos, usando o MongoDB. Em ambas as adaptações deste trabalho, foi considerado o esquema de dados completamente desnormalizado.

Em relação ao paradigma orientado a coluna, a adaptação consiste em armazenar cada tabela em uma família de colunas, fazendo com que essa abordagem seja similar ao CNSSB. Por ser parecida com o CNSSB, essa adaptação possui as mesmas vantagens, todavia herda as mesmas limitações (conforme descrito na Seção 3.4.1). Já em relação ao paradigma orientado a documentos, cada instância da tabela de fatos é considerada um documento (estruturado em atributos), em que os dados relacionados de cada dimensão são armazenados internamente como subdocumentos, associados a um dos atributos da instância. Ao comparar esses dois paradigmas, constatou-se que o HBase teve melhor desempenho tanto na carga de dados quanto no processamento de consultas.

Além dessa adaptação, CHEVALIER *et al.* (2015c) também analisaram variações de organização no MongoDB, possibilitando três configurações distintas: (i) armazenar todos os atributos de todas as tabelas em um mesmo documento (sem documentos internos); (ii) armazenar cada dimensão em um respectivo subdocumento, conforme explicado anteriormente; e (iii) armazenar os documentos separadamente, de forma normalizada (dados referentes à tabela de fatos separados das dimensões). Apesar de a abordagem (iii) possuir um volume menor de dados que as abordagens (i) e (ii), ela foi ineficiente durante o processamento de consultas OLAP que

possuíam junções, ou seja, agregavam dados de diferentes dimensões.

Uma vez que o foco dessa dissertação envolve bancos de dados orientados a coluna, as investigações sobre a adaptação de um DW ao banco de dados orientado a documentos MongoDB não foi aprofundada. Todavia, durante este estudo, foi analisada a divergência entre os desempenhos no processamento de consultas do HBase e do MongoDB. Esses experimentos adicionais estão presentes no Apêndice A. Eles têm como objetivo investigar a organização proposta na abordagem (ii) de [CHEVALIER et al. \(2015c\)](#), usando o MongoDB, e compará-la com as modelagens analisadas nesta dissertação, usando o HBase, descritas ao longo do Capítulo 4.

3.4.3 Distribuição de atributos em famílias de colunas

Antes de analisar o contexto de DW, [CAI et al. \(2013\)](#) avaliaram o desempenho do HBase considerando três características: (i) armazenamento dos dados em famílias de colunas; (ii) quantidade de blocos de dados; e (iii) fator de replicação. Com relação à característica (i), foram realizados testes variando o agrupamento de atributos em famílias de colunas, a fim de comparar o desempenho em operações de escrita e leitura. Quanto à característica (ii), ao variar o tamanho das partições de dados, foi alterada a quantidade de blocos distribuídos no HDFS, o que pode impactar o desempenho durante a manipulação desses dados. Em relação à característica (iii), analisou-se a influência da quantidade de réplicas dos dados no processamento de consultas. Como resultado, o desempenho das operações de leitura foi pior considerando-se principalmente as duas primeiras características: dados armazenados em diferentes famílias de colunas e/ou particionados em uma elevada quantidade de blocos de dados. A partir dessa análise, a característica (i) também é explorada em outros trabalhos correlatos descritos nessa seção. Neste estudo também investiga-se a característica (ii), sendo os resultados obtidos detalhados na Seção 4.4.

O trabalho de [CHEVALIER et al. \(2015b\)](#) avalia a distribuição de atributos em famílias de colunas no HBase considerando três abordagens: (i) agrupar todos os atributos de todas as tabelas em uma mesma família de colunas; (ii) agrupar os atributos de cada tabela em uma família de colunas distinta; e (iii) armazenar cada dimensão em uma tabela distinta, mantendo o esquema normalizado (separando a tabela de fatos das dimensões). Ao analisar essas três abordagens, constatou-se que a última, apesar de possuir um volume menor de dados, não teve um desempenho eficiente no processamento de consultas envolvendo junção. Adicionalmente, [CHEVALIER et al. \(2015b\)](#) afirmam que entre as abordagens (i) e (ii) não houve diferenças significativas de tempo. Mesmo realizando essa comparação entre diferentes agrupamentos de famílias de colunas, não foram analisados cenários que são beneficiados por cada abordagem, em termos de desempenho no processamento de consultas. Nesta dissertação, esse estudo de diferentes cenários empresariais, em que cada abordagem exibe os melhores desempenhos, é detalhado na Seção 4.2.1.

Por fim, o trabalho de [DEHDOUH et al. \(2015\)](#) faz uma análise similar ao trabalho de [CHEVALIER et al. \(2015b\)](#), considerando as mesmas três abordagens. Usando o esquema e os dados gerados pelo SSB e o banco de dados NoSQL o HBase, também verificou-se que a abordagem (iii) demora mais tempo para processar consultas envolvendo junção, em comparação com as outras abordagens. Todavia, diferentemente do trabalho de [CHEVALIER et al. \(2015b\)](#), notou-se uma diferença entre as abordagens (i) e (ii), que diz respeito ao processamento mais eficiente sobre uma família de colunas em contraste a várias famílias de colunas. Com base nesses resultados preliminares, levantou-se o questionamento sobre como estruturar o esquema de maneira efetiva, por meio da distribuição de atributos entre famílias de colunas. Esses questionamentos motivaram as investigações realizadas neste estudo, que são descritas na Seção 4.2.1.

3.4.4 SSB+

O trabalho de [CHEVALIER et al. \(2015a\)](#) estende o processo de geração de dados do SSB para bancos de dados NoSQL. Mais precisamente, ele adiciona a possibilidade no *dbgen*, programa interno do SSB, de gerar dados em outros tipos de formatos. Com isso, esses dados podem ser carregados diretamente em bancos orientados a documentos (no caso, o MongoDB) e em bancos orientados a coluna (como o HBase). No processo atual do SSB, são gerados cinco arquivos no formato texto, um para cada tabela. Porém, para carregar esses dados no HBase, por exemplo, é necessário desnormalizar os dados em uma única tabela, por meio da junção das tabelas de fatos e dimensões, o que consiste em um processo custoso. Já em relação ao paradigma orientado a documentos, esse formato de arquivo deve ser convertido para formatos como JSON ou XML antes de ser carregado no banco.

Tendo em vista essas limitações, [CHEVALIER et al. \(2015a\)](#) propõem o SSB+, que não só pode gerar os dados separados em cinco tabelas, como permite a geração desses mesmos dados desnormalizados e em vários formatos, como texto (estruturado em atributos), JSON e XML. Além disso, o SSB+ possibilita a geração distribuída desses dados, gerando-os diretamente no HDFS por meio do *framework* Hadoop MapReduce. Outra adaptação realizada consiste no volume de dados gerados dado um valor de SF, pois no SSB o SF = 1 não corresponde a 1 GB de dados, e sim aproximadamente 0.56 GB (fato que pode ser confuso para usuários que desejam usar este *benchmark*). [CHEVALIER et al. \(2015a\)](#) afirmam que essa divergência entre o valor de SF e volume de dados pode prejudicar uma análise detalhada do sistema avaliado.

Entretanto, em relação a este projeto, essa extensão está limitada a apenas uma das etapas de um *benchmark* de DW, uma vez que utiliza como esquema as modelagens propostas por [CHEVALIER et al. \(2015d\)](#), descritas na Seção 3.4.2. Em outras palavras, para adaptar um *benchmark* para DW em um banco de dados NoSQL, é preciso analisar várias outras questões além da geração de dados, como o impacto de diferentes esquemas e os fatores que influenciam o processamento distribuído dos dados. Essas e outras questões foram analisadas durante este

estudo e estão detalhadas ao longo do Capítulo 4. Além disso, outra limitação dessa extensão consiste na falta de análise sobre o balanceamento desses dados. Uma vez que os dados serão armazenados de forma distribuída, não foi analisada como esses dados ficaram divididos entre os nós de um *cluster*. Os detalhes sobre esta análise, realizada neste estudo, estão descritos na Seção 4.3.

3.5 Benchmarks para Sistemas Distribuídos e NoSQL

Nesta seção são detalhados *benchmarks* para validação de sistemas distribuídos e/ou bancos de dados NoSQL, sem necessariamente avaliarem o ambiente de *data warehousing*. Esses trabalhos estão relacionados com este projeto do ponto de vista de sistemas distribuídos e NoSQL, porém não definem um esquema multidimensional ou não foram aplicados a bancos de dados NoSQL (consistindo em uma avaliação apenas do processamento distribuído). Os três principais *benchmarks* são: Yahoo! *Cloud Serving Benchmark* (YCSB) (Seção 3.5.1), BigBench (Seção 3.5.2) e TPCx-HS (Seção 3.5.3).

3.5.1 YCSB

Considerado um dos *benchmarks* mais utilizados para comparação entre bancos de dados NoSQL (BARATA; BERNARDINO; FURTADO, 2014), o YCSB possibilita uma análise comparativa entre diferentes bancos de dados NoSQL. O esquema dele é composto por uma única tabela, sendo possível configurar a quantidade e o tamanho dos atributos. Já as cargas de trabalho do YCSB são divididas conforme cenários específicos de aplicação (BARATA; BERNARDINO; FURTADO, 2014; COOPER *et al.*, 2010), descritas resumidamente na Tabela 3.2.

Tabela 3.2 – Resumo das cinco cargas de trabalho definidas pelo YCSB, ilustrando a quantidade relativa de operações em cada carga.

carga de trabalho	quantidade de operações (em %)			
	leitura	leitura sequencial	inserção	atualização
atualização intensiva	50%	-	-	50%
leitura intensiva	95%	-	-	5%
apenas leitura	100%	-	-	-
leitura de dados recém-inseridos	95%	-	5%	-
pequenas leituras sequenciais	-	95%	5%	-

Em relação à etapa de parâmetros e métricas, os principais parâmetros configuráveis são a quantidade de instâncias da tabela (similar ao SF), o número de atributos de cada instância e o tamanho desses atributos (que consistem em campos de texto com conteúdo aleatório). Tendo em vista esses parâmetros, a principal métrica coletada durante a validação deste *benchmark* é o desempenho, definido pela quantidade de operações de inserção, leitura e atualização realizadas por segundo, bem como a latência de disco e rede.

Apesar de o YCSB possuir portabilidade para diversos bancos de dados NoSQL, como DynamoDB, HBase, MongoDB, Redis, Voldemort, entre outros, ele não possui esquema nem carga de trabalho característicos de um ambiente de *data warehousing*. As operações de leitura descritas acima consistem apenas em consultas que recuperam uma ou mais instâncias, sem operações de agregação e, conseqüentemente, sem o processamento de consultas OLAP. Uma vez que o foco desta dissertação consiste em analisar um *benchmark* para DW em bancos de dados NoSQL orientados a colunas, este *benchmark* é considerado muito genérico para o estudo realizado, podendo ser mais adequado para outras análises, como bancos de dados focados em operações transacionais sobre os dados.

3.5.2 BigBench

O BigBench consiste em um *benchmark* que engloba modelo e geração sintética de dados de forma a cobrir os aspectos de volume, variedade e velocidade definidos para *Big Data* (GHAZAL *et al.*, 2013). O volume refere-se à quantidade de dados gerados, baseado em um SF, ao passo que a variedade está relacionada ao fato de o banco ser capaz de armazenar três níveis de organização dos dados (estruturada, semiestruturada e não estruturada). Por fim, a velocidade é definida pela habilidade do banco em processar consultas analíticas e atualizações periódicas dos dados em um curto intervalo de tempo.

O esquema do BigBench é dividido em três partes (GHAZAL *et al.*, 2013; BARATA; BERNARDINO; FURTADO, 2014). A parte estruturada é derivada do modelo de dados presente no TPC-DS (Seção 3.3.2). Já a parte semiestruturada armazena, por meio de arquivos de *log*, o fluxo de requisições de um usuário na *web*, de forma estruturada para compras efetuadas (contendo, por exemplo, item comprado, usuário e data) e de forma não estruturada para outros tipos de navegação. Por fim, a parte não estruturada contém textos associados aos itens e à avaliação de produtos, gerados a partir do modelo de cadeias de Markov. A Figura 3.4 mostra as três divisões do esquema do *benchmark*, bem como as tabelas armazenadas em cada uma delas. Dentro da parte estruturada foi inserida uma tabela denominada *Marketprice*, a qual contém nomes e preços de concorrentes para cada produto armazenado. Esses dados são armazenados em forma de arquivos no HDFS, sendo processados usando a ferramenta Hive (Seção 2.4.1).

Em relação à carga de trabalho, o BigBench possui um total de 30 consultas, as quais cobrem diferentes categorias de análise em *Big Data* do ponto de vista de negócios. Sob a perspectiva técnica, as consultas visam validar os sistemas baseadas tanto na geração e carga dos dados quanto nos tipos de processamento de consultas e técnicas analíticas empregadas. Elas consistem em consultas SQL traduzidas pela ferramenta Hive ou até mesmo implementações customizadas de algoritmos de mineração de dados usando os *frameworks* Hadoop MapReduce e Spark. A partir dessas operações, este *benchmark* coleta a métrica de desempenho, reportando o tempo demorado em cada uma das operações.

Figura 3.4 – Modelo de dados do BigBench (adaptado de [GHAZAL et al., 2013](#)), ilustrando a divisão do esquema em dados estruturados, semiestruturados e não estruturados.

Contudo, apesar de o BigBench se considerado um *benchmark* de DW para sistemas distribuídos, ele está focado no processamento distribuído de dados. Por armazenar os dados em arquivos no HDFS, ele não analisa a organização do esquema em bancos de dados NoSQL, bem como a adaptação das tabelas nos diferentes paradigmas de armazenamento, como o paradigma orientado a coluna. Além dessas limitações, existem várias diferenças entre a proposta desse *benchmark* e o estudo realizado nesta dissertação, a começar pela análise de diferentes modelagens dos dados quando armazenados em bancos de dados NoSQL orientados a coluna (Seção 4.2.1) e orientados a documentos (Apêndice A). Outro estudo realizado nesta dissertação é a influência de alguns parâmetros no desempenho do processamento distribuído dos dados, usando o *framework* Hadoop MapReduce, conforme detalhado na Seção 4.4.

3.5.3 TPCx-HS

Este *benchmark*, também desenvolvido pela TPC, consiste na análise de desempenho do processamento distribuído usando o *framework* Hadoop MapReduce. O TPCx-HS ([NAMBIAR et al., 2015](#)) baseia-se em quatro etapas: (i) geração de dados, cujo volume é definido pelo SF; (ii) ordenação; (iii) validação; e (iv) verificação.

A etapa (i) consiste na geração sintética de palavras não ordenadas dentro de um arquivo de texto. A partir desse arquivo, armazenado no HDFS, a etapa (ii) refere-se à ordenação dessas palavras, usando como processamento o *framework* Hadoop MapReduce. Na etapa (iii) o arquivo ordenado é validado, analisando-se se foi mantida a cardinalidade e a integridade das palavras antes e depois da ordenação. Por fim, na etapa (iv) é realizada uma verificação do arquivo inteiro, de forma a identificar palavras que estejam fora da ordem lexicográfica. Durante cada etapa a principal medida coletada é o tempo decorrido. Com base no tempo total, é calculada a métrica denominada HSph@SF, similar à métrica do TPC-H, que é usada para comparação entre diferentes ambientes de processamento distribuído.

Entretanto, apesar do TPCx-HS ser um *benchmark* válido para o processamento distribuído usando o *framework* Hadoop MapReduce, ele não possui algumas das principais etapas de um *benchmark* para DW, como modelagem dos dados e operações de consulta OLAP. Esta dissertação realizou a maioria de suas investigações utilizando o *framework* Hadoop MapReduce para o processamento distribuído dos dados, sendo o TPCx-HS relevante por validar diretamente esse tipo de processamento, apesar de não ser aplicável nem a bancos de dados NoSQL nem ao ambiente de *data warehousing*.

3.6 Considerações Finais

Neste capítulo foram descritos importantes conceitos sobre a técnica de *benchmarking*, bem como os principais *benchmarks* para DW, sistemas distribuídos e bancos de dados NoSQL. Durante a explicação de cada um dos *benchmarks* para DW, foram detalhadas adaptações deles para bancos de dados NoSQL, usando paradigmas de armazenamento diferentes do relacional. Em especial, foi explicado sobre o SSB, o *benchmark* usado como base neste projeto, uma vez que possui um esquema estrela puro, principal em um ambiente de *data warehousing*, além de conter a maior quantidade de adaptações para bancos de dados NoSQL distribuídos. Adicionalmente, todas essas adaptações do SSB, considerando o paradigma de armazenamento orientado a colunas, usaram o HBase para validar suas propostas. Sendo assim, este foi o principal banco de dados NoSQL distribuído adotado durante a execução deste projeto.

No Capítulo 4 são detalhadas as investigações realizadas nesta dissertação sobre cada etapa de um *benchmark* de DW, levando-se em consideração as limitações identificadas em cada uma das adaptações do SSB descritas nas Seções 3.4.1 a 3.4.4.

INVESTIGAÇÃO DAS ETAPAS DE UM BENCHMARK PARA DATA WAREHOUSE

Neste capítulo é detalhado o trabalho desenvolvido durante o estudo realizado. São descritas as investigações sobre variações do esquema de um benchmark para DW considerando o paradigma de armazenamento orientado a colunas, além da proposta de novas consultas para a carga de trabalho definida pelo SSB e uma avaliação da distribuição dos registros gerados pelo SSB entre os nós do cluster. Também são analisados parâmetros do ambiente distribuído que influenciam no processamento de consultas OLAP. Além disso, são detalhadas diferentes técnicas para processamento e validação de ambientes distribuídos.

4.1 Considerações Iniciais

Conforme detalhado na Seção 3.2, um *benchmark* para DW é dividido em quatro etapas: (i) esquema e carga de trabalho; (ii) geração de dados; (iii) métricas e parâmetros; e (iv) validação. A principal pergunta desta dissertação é: ***"O que é necessário adicionar ao SSB, ou a outro benchmark de DW, para validar efetivamente ambientes de data warehousing baseados em bancos de dados NoSQL distribuídos e orientados a colunas?"***. Com base nessa pergunta, o objetivo deste estudo é identificar e investigar as limitações deste *benchmark* com relação a cada uma dessas quatro etapas. Portanto, vislumbra-se que as características necessárias a um *benchmark*, descobertas neste projeto, possibilitem um posterior desenvolvimento de um *benchmark* para DW para bancos de dados NoSQL distribuídos e orientado a colunas. Além disso, uma vez que as investigações das etapas (iii) e (iv) envolvem principalmente o processamento de dados distribuídos usando o *framework* MapReduce, os resultados exibidos podem ser aplicados a quaisquer outros *benchmarks* de DW.

Levando em consideração as quatro etapas de um *benchmark* para bancos de dados, as atividades desenvolvidas neste trabalho investigam a pergunta anteriormente especificada, dividindo-a nas seguintes perguntas:

- Quais os impactos que diferentes esquemas físicos têm sobre bancos de dados NoSQL distribuídos e orientados a colunas? Os resultados obtidos com a investigação dessa pergunta são descritos na Seção 4.2.1;
- A carga de trabalho, definida majoritariamente por consultas OLAP, é completa e suficiente para validar um DW? Os resultados obtidos com a investigação dessa pergunta são detalhados na Seção 4.2.2;
- Os dados gerados pelo SSB estão igualmente divididos entre as máquinas de um *cluster*? Os resultados obtidos com a investigação dessa pergunta são abordados na Seção 4.3;
- Quais parâmetros podem influenciar no processamento distribuído de consultas OLAP? Os resultados obtidos com a investigação dessa pergunta são descritos na Seção 4.4; e
- Como diferentes formas de processamento e validação de um *benchmark* podem influenciar no desempenho de operações OLAP? Os resultados obtidos com a investigação dessa pergunta são detalhados na Seção 4.5.

Nesse capítulo são apresentadas propostas e/ou análises referentes a cada uma das etapas, a fim de responder às perguntas descritas anteriormente. Adicionalmente, a Figura 4.1 ilustra graficamente cada uma dessas etapas, cada qual relacionada com a seção na qual são investigadas. Note que o objetivo deste trabalho não é propor um *benchmark* para DWs armazenados em bancos de dados NoSQL orientados a colunas, mas sim realizar um conjunto de propostas e análises majoritariamente baseadas em testes de desempenho experimentais que ofereçam fundamentos para a posterior proposta desse *benchmark*.

Durante os experimentos, o paradigma de armazenamento utilizado foi o orientado a colunas, devido à potencial eficiência deste paradigma para realizar majoritariamente operações de leitura. Com o objetivo de efetuar a validação das propostas, considerando a organização dos dados em famílias de colunas, o banco de dados NoSQL utilizado foi o HBase. Este banco foi empregado pois segue a mesma divisão lógica empregada por outros bancos, como Accumulo e Cassandra, além de já existirem trabalhos correlatos adaptando um ambiente de *data warehousing* a esse banco de dados NoSQL. Já para validação das hipóteses, o *benchmark* utilizado foi o SSB. No entanto, as análises realizadas nesse capítulo poderiam ter sido validadas em outros *benchmarks*, como o TPC-H ou TPC-DS, como os diferentes agrupamentos de atributos em famílias de colunas, o estudo sobre a influência de parâmetros de processamento distribuído e a verificação da distribuição balanceada dos dados entre os nós do *cluster*.

Conforme mostrado na Figura 4.1, esse capítulo está organizado da seguinte forma. Na Seção 4.2 são apresentadas as propostas referente ao esquema físico (Seção 4.2.1) e à carga de trabalho (Seção 4.2.2) definidos pelo SSB, relacionadas à etapa (i). Na Seção 4.3 é investigado como os dados gerados pelo SSB estão distribuídos entre os nós do *cluster* (atividade relacionada à etapa (ii)). Na Seção 4.4 são detalhadas as atividades relativas à etapa (iii), as quais detalham

três parâmetros de armazenamento e processamento distribuído que impactam no desempenho de consultas OLAP, como quantidade de processos Map executados simultaneamente em cada nó (Seção 4.4.1), número máximo de processos Reduce (Seção 4.4.2) e tamanho do bloco de dados distribuído (Seção 4.4.3). Por fim, na Seção 4.5 são descritas duas análises referentes à técnica de processamento distribuído, relacionadas ao ganho de desempenho considerando diferentes quantidades de nós no *cluster* (Seção 4.5.1), além de técnicas para otimização do processamento iterativo de consultas OLAP usando o *framework* Spark (Seção 4.5.2). Adicionalmente, foi feito um estudo preliminar sobre o banco de dados NoSQL orientado a documentos denominado MongoDB, descrito no Apêndice A, analisando como o SSB pode ser adaptado a este tipo de banco.

Figura 4.1 – Ilustração das quatro etapas de um *benchmark* de DW, relacionando cada uma delas com a seção em que elas são investigadas neste trabalho.

4.2 Esquema e da Carga de Trabalho

Essa seção descreve propostas e investigações realizadas sobre a etapa (i) de um *benchmark* para DW, consistindo na proposta e comparação de diferentes organizações físicas do esquema do SSB no HBase (Seção 4.2.1), além da elaboração de novas consultas para incorporar à carga de trabalho definida pelo SSB (Seção 4.2.2).

4.2.1 Organização de um DW em Famílias de Colunas

Proposta do Esquema FactDate

Uma vez que o agrupamento de atributos em famílias de colunas influencia no desempenho de operações de leitura, nesta dissertação é proposto um esquema parcial, com o objetivo de otimizar consultas OLAP específicas. Define-se um esquema parcial como a união de atributos de duas ou mais tabelas na mesma família de colunas, sem necessariamente juntar todos os atributos do esquema em uma mesma família. Conforme descrito na literatura, discutido na Seção 3.4.3, os trabalhos relacionados a este princípio agrupam as colunas por meio de duas abordagens antagônicas: todos os atributos na mesma família de colunas ou os atributos de cada tabela em uma família distinta.

Considerando o esquema proposto, em contraste às abordagens presentes na literatura, investigou-se também os benefícios e limitações que cada um proporciona em diferentes cenários, cada qual com requisitos específicos. Para definir esses cenários, é utilizado o esquema do SSB, ilustrado na Figura 3.3, que possui quatro dimensões: *Date*, *Customer*, *Supplier* e *Part*. A partir desse esquema, são definidos dois cenários:

- **Cenário A:** Uma empresa focada em analisar seu rendimento diário e/ou mensal, ou seja, interessada em calcular a quantidade total de produtos vendida por dia (dimensão *Date* apenas), ou interessada na quantidade vendida por categoria de produto por mês (dimensões *Part* e *Date*, respectivamente). Neste cenário, o tipo mais frequente de consulta OLAP acessa até duas dimensões do modelo multidimensional;
- **Cenário B:** Uma empresa interessada em análises mais detalhadas, como a quantidade de produtos vendidos por marca de produto por cliente por semestre (dimensões *Part*, *Customer* e *Date*), ou consultas que envolvam todas as dimensões do esquema estrela. Neste cenário, predominam consultas OLAP envolvendo três ou mais dimensões.

(a) Esquema proposto neste trabalho, denominado FactDate.

(b) Esquema correlato denominado CNSSB, descrito na Seção 3.4.1.

(c) Esquema correlato, denominado nessa dissertação de SameCF, descrito na Seção 3.4.2.

Figura 4.2 – Representação da organização física do esquema proposto, em contraste com os esquemas existentes na literatura. O esquema FactDate é focado em otimizar consultas que envolvam pelo menos a dimensão *Date*, agregação frequente devido à temporalidade dos dados de um DW.

O esquema parcial proposto é denominado FactDate, e é ilustrado na Figura 4.2(a). Nele, os atributos pertencentes à tabela de fatos *LineOrder* e à tabela de dimensão *Date*, presentes na Figura 3.3, são agrupados na mesma família de coluna. Essa junção foi realizada tendo em

vista a característica de temporalidade de um DW, sendo predominantes consultas que envolvam ao menos um intervalo de tempo. Já as outras dimensões (*Customer*, *Part* e *Supplier*) são armazenadas em famílias de colunas exclusivas. Na Figura 4.2 também são representados outros agrupamentos físicos presentes na literatura, para efeito de comparação. O CNSSB (Seção 3.4.1), que consiste em dividir os atributos de cada tabela em famílias de colunas distintas é mostrado na Figura 4.2(b). Já o agrupamento de todos os atributos na mesma família de colunas, denominado aqui SameCF, é ilustrado na Figura 4.2(c).

Proposta de mapeamento das consultas do SSB para o framework Hadoop MapReduce

A partir dessas três organizações físicas dos dados, outra contribuição desta pesquisa é a implementação, usando o *framework* Hadoop MapReduce, das consultas OLAP definidas pelo SSB. O processamento dessas consultas foi baseado no MRBench, descrito na Seção 2.4.1, adaptado para acessar os dados do HBase e para realizar as consultas OLAP definidas pelo SSB. Essas consultas são realizadas em duas etapas. Na primeira etapa, considerada recuperação e filtragem dos dados, o processamento da cláusula *where* da consulta SQL é realizada pelo próprio banco de dados HBase. Porém, por não possuir internamente algoritmos para realizar a agregação dos dados, a segunda etapa é executada por meio do *framework* Hadoop MapReduce, o qual possibilita a agregação simples dos dados (operações de soma total, por exemplo) ou a agregação dos dados dividida em categorias (ou seja, usando a cláusula *group by* de uma consulta SQL).

Algoritmo 1: mapQ1

Data: Tupla tupla

Result: Par (nulo, $\text{desconto} \times \text{preco_estendido}$)

1 **begin**

2 $\text{desconto} = \text{tupla.acessaColuna}(\text{CF}, \text{lo_discount});$

3 $\text{preco_estendido} = \text{tupla.acessaColuna}(\text{CF}, \text{lo_extendedprice});$

4 $\text{emite Par}(\text{nulo}, \text{desconto} \times \text{preco_estendido});$

O Algoritmo 1 representa o processamento de uma consulta responsável por efetuar a agregação simples de soma total, de acordo com as consultas de classe 1 definidas pelo SSB. O termo CF refere-se a uma variável de tipo texto que identifica a qual família de colunas o atributo pertence. Os possíveis valores de CF estão definidos na Figura 4.2. No esquema FactDate, por exemplo, usa-se $\text{CF} = \text{LINEDATE_CF}$ para acessar os atributos *lo_discount* e *lo_extendedprice*. Como essas consultas não possuem uma cláusula *group by*, o par chave-valor emitido neste processo Map tem como chave o elemento vazio e como valor a multiplicação dos atributos numéricos *lo_discount* e *lo_extendedprice*. Durante o processo *Shuffle*, todos esses pares de mesma chave são agrupados em um mesmo processo Reduce, o qual é mostrado no Algoritmo 2. Esse processo Reduce é responsável por efetuar a soma dos valores pertencentes a uma chave,

da mesma forma como foi ilustrado na Figura 2.9 (ou seja, por meio de uma soma dos elementos de um vetor).

O Algoritmo 3 processa consultas que realizam a soma dos dados dividida em categorias, as quais são definidas na cláusula *group by* do comando SQL. Essas consultas são baseadas na classe 2 da carga de trabalho definida pelo SSB, cujas categorias consistem na combinação dos atributos *d_year* e *p_brand1*. Para esse tipo de consulta, as chaves geradas durante a execução do processo Map contêm as categorias desejadas, formadas pela concatenação desses dois atributos. A partir disso, o processo *Shuffle* agrupa os pares chave-valor de mesma chave, enviando cada categoria para um processo Reduce distinto. Esse processo Reduce, por consistir na agregação de soma, é também definido pelo Algoritmo 2. As outras consultas definidas pelo SSB, pertencentes às classes 3 e 4, agregam diferentes atributos, sendo que cada uma delas é processada de forma análoga ao Algoritmo 3, apenas adaptando os atributos combinados.

Algoritmo 2: reduceSOMA

Data: Texto chave, Lista<número> valores

Result: Par (chave, soma)

```

1 begin
2   soma = 0;
3   while número ∈ valores do
4     soma += número;
5   emite Par(chave, soma);

```

Algoritmo 3: mapQ2

Data: Tupla tupla

Result: Pair (*atributosAgrupamento*, *rendimento*)

```

1 begin
2   atributosAgrupamento = tupla.acessaColuna(CF, d_year) ∪
                           tupla.acessaColuna(CF, p_brand1);
3   rendimento = tupla.acessaColuna(CF, lo_revenue);
4   emite Par(atributosAgrupamento, rendimento);

```

Testes de desempenho: Ambiente de Teste

O ambiente distribuído em que foram realizados os testes de desempenho voltado à análise das propostas descritas nessa seção é composto por quatro máquinas (também chamadas de nós), sendo um *NameNode* e três *DataNodes*. Cada um desses nós possui um processador i5-3330, 16 GB de memória RAM, 1 TB de disco SATA (7200 RPM) e conexão de 1Gb/s. O sistema operacional de cada máquina é o CentOS (versão 7.0), com o banco de dados NoSQL HBase (versão 0.98.13) e o *framework* Hadoop MapReduce (versão 2.4.1).

A base de dados utilizada durante os experimentos foi gerada pelo *dbgen* do SSB, sendo necessário um pós-processamento dos dados a fim de desnormalizar o esquema (unir a tabela de

fatos com as quatro dimensões). Durante o pós-processamento, também foi efetuada a projeção dos atributos, removendo colunas não utilizadas (por exemplo, os atributos FKs). A Tabela 4.1 detalha os volumes de dados gerados, em GB, dos valores 10, 20, 40 e 80 de SF. Para cada um deles, são exibidos o tamanho do arquivo texto desnormalizado e o volume ocupado em cada um dos três esquemas físicos avaliados. Note que ao inserir os dados no HBase, em virtude do formato *<PK, família, coluna, timestamp, valor>*, o volume dos dados armazenados é maior do que o arquivo texto original.

Tabela 4.1 – Volumes de dados utilizados durante os experimentos.

	fator escala (SF)			
	10	20	40	80
arquivo texto desnormalizado (GB)	28,0	55,0	109,0	218,0
FactDate (GB)	58,4	116,9	233,9	468,2
CNSSB (GB)	58,5	117,0	234,2	469,9
SameCF (GB)	57,9	115,8	231,8	464,0

Com base nesse ambiente distribuído e nos diferentes volumes de dados, foram realizados dois experimentos. O primeiro experimento teve como objetivo encontrar o esquema físico mais eficiente para o processamento de consultas OLAP, considerando os cenários A e B descritos anteriormente. Já o segundo experimento consistiu em uma análise de escalabilidade, verificando a divergência de desempenho entre os esquemas físicos conforme varia-se o volume de dados. Em ambos os experimentos, cada consulta foi executada 20 vezes para coletar o tempo médio em segundos.

Análise comparativa entre os esquemas físicos em termos do desempenho no processamento de consultas

Para analisar comparativamente os três esquemas físicos mostrados na Figura 4.2, foi coletada a média do tempo de processamento de consultas em cada um deles. As bases de dados analisadas nesse primeiro experimento correspondem aos valores 10 e 20 de SF, conforme descrito na Tabela 4.1. Em virtude dos cenários A e B inicialmente propostos, as consultas foram divididas em duas categorias:

- **Consultas de baixa dimensionalidade** (Cenário A): essa categoria engloba consultas que agregam poucas dimensões do modelo multidimensional, o qual foi traduzido para o esquema estrela. Mais especificamente, elas processam dados de apenas uma tabela de dimensão (chamadas consultas unidimensionais) ou de duas tabelas de dimensão (consultas bidimensionais) sobre o esquema estrela ilustrado na Figura 3.3 e adaptado ao HBase de acordo com a Figura 4.2; e

- **Consultas de alta dimensionalidade** (Cenário B): essa categoria abrange consultas que agregam simultaneamente várias dimensões desse modelo multidimensional. Nesta categoria, são processados dados de três ou mais dimensões (por exemplo, consultas tridimensionais).

A partir dessas duas categorias, a Figura 4.3 mostra o tempo médio decorrido das consultas de baixa dimensionalidade. As Figuras 4.3(a) e 4.3(b) ilustram o tempo de processamento de três consultas unidimensionais (denominadas Q1.1, Q1.2 e Q1.3), as quais são definidas pela carga de trabalho original do SSB. Já as Figuras 4.3(c) e 4.3(d) ilustram o tempo de processamento de duas consultas bidimensionais (denominadas Qnew1 e Qnew2), as quais são propostas e usadas neste trabalho porque o SSB não define consultas sobre duas dimensões. Essas consultas bidimensionais são detalhadas na Seção 4.2.2.

Figura 4.3 – Comparação entre os três esquemas avaliados dentro da categoria de consultas de baixa dimensionalidade, mostrando o tempo médio decorrido no processamento dessas consultas. Nesta análise, o esquema proposto FactDate possuiu um desempenho superior às outras abordagens.

Dentro da categoria de consultas de baixa dimensionalidade, o esquema proposto FactDate exibiu melhor desempenho que as outras duas abordagens. Ao comparar os esquemas

FactDate e CNSSB, o ganho de desempenho das consultas unidimensionais variou de 25% a 33%. Isto ocorreu pois enquanto o FactDate acessa apenas uma família de colunas, o CNSSB envolve duas famílias, sendo necessária a junção delas e, conseqüentemente, elevando o tempo de processamento da consulta. Já em relação ao esquema SameCF, este mostrou um desempenho bem inferior aos outros, devido ao fato de que, ao agrupar todos os atributos na mesma família, o arquivo *HFile* resultante contém um volume elevado de dados, fator que prejudica seu desempenho. No caso das consultas bidimensionais, a diferença de tempo foi menor, sendo que o esquema CNSSB exibiu pior desempenho que os demais. O FactDate possui os melhores tempos mesmo processando dados de duas famílias de colunas, em comparação ao SameCF, sendo que os ganhos de desempenho variaram de 6% a 14%.

Figura 4.4 – Comparação entre os três esquemas avaliados dentro da categoria de consultas de alta dimensionalidade, mostrando o tempo médio decorrido no processamento dessas consultas. Para esse tipo de cenário, o esquema SameCF obteve melhor desempenho, uma vez que nele não é necessária a junção entre duas ou mais famílias de colunas durante o processamento das consultas.

Em relação à segunda categoria, a Figura 4.4 mostra o tempo médio decorrido de consultas de alta dimensionalidade. As Figuras 4.4(a) e 4.4(b) ilustram o tempo de processamento de sete consultas tridimensionais (denominadas Q2.1, Q2.2 e Q2.3, Q3.1, Q3.2, Q3.3 e Q3.4), as

quais são definidas pela carga de trabalho original do SSB. Já as Figuras 4.4(c) e 4.4(d) ilustram o tempo de processamento de três consultas, as quais envolvem todas as quatro dimensões do esquema (denominadas Q4.1, Q4.2 e Q4.3), também definidas pelo SSB.

Na categoria de consultas de alta dimensionalidade o resultado se inverte, pois conforme dimensões são adicionadas às agregações realizadas, o desempenho do processamento de consultas nos esquemas FactDate e CNSSB piora. Isto ocorre em razão do tempo do processamento adicional para reconstruir a tupla (unindo famílias de colunas distintas), em que esse tempo supera o de processamento de grandes arquivos *HFiles* (presentes no SameCF). O ganho de desempenho do SameCF em comparação ao FactDate variou de 14% a 38% no caso tridimensional, enquanto que no caso em que todas as quatro dimensões são agregadas, esse ganho aumentou para a faixa de 47% a 54%.

As análises efetuadas mostraram como cada esquema físico pode favorecer determinadas categorias de consultas, considerando cenários distintos. O impacto no desempenho do processamento de diferentes esquemas físicos deve ser considerado em um *benchmark* para mais bem modelar o cenário de uso onde ele deve ser aplicado.

O próximo experimento visa analisar a escalabilidade dos esquemas, de forma a verificar se essa divergência permanece constante ou aumenta conforme o volume de dados cresce.

Análise comparativa entre os esquemas considerando a escalabilidade de dados

Nesse segundo experimento, foi avaliado o comportamento das consultas em diferentes volumes de dados processados, analisando uma maior variação desse volume. A Figura 4.5 mostra o tempo de processamento médio das consultas considerando os valores 10, 20, 40 e 80 de SF, descritos na Tabela 4.1. A Figura 4.5(a) ilustra o comportamento das consultas unidimensionais. Para cada SF desse gráfico foi calculada a média de tempo das consultas Q1.1, Q1.2 e Q1.3. Nota-se que conforme o volume de dados aumenta, o tempo de processamento dos dados organizados de acordo com o esquema SameCF cresce de forma mais acentuada do que os outros esquemas, evidenciando o impacto existente relacionado ao tamanho do arquivo *HFile*. Comparando o esquema proposto FactDate com seu concorrente mais próximo, o esquema CNSSB, organizar os dados de acordo com o esquema FactDate otimizou o desempenho no processamento das consultas unidimensionais em aproximadamente 20%. Isso está relacionado ao fato de que o esquema FactDate organizou os dados necessários às consultas em apenas uma família de colunas, sem conter muitos outros atributos desnecessários à consulta.

Já a Figura 4.5(b) exhibe o comportamento das consultas bidimensionais. Para cada SF desse gráfico foi calculada a média de tempo das consultas Qnew1 e Qnew2. Nesta análise, o esquema FactDate novamente provê melhor desempenho do que os outros esquemas, uma vez que ele realiza a união de apenas duas famílias de colunas. Nota-se que, em relação aos experimentos realizados, as consultas que acessaram mais de duas famílias de colunas (esquema CNSSB) possuem piores tempos quando comparado às consultas que processaram arquivos

HFile possuindo um elevado volume de dados (esquema SameCF). O ganho de desempenho, em consultas bidimensionais, do FactDate comparado ao seu concorrente mais próximo, o esquema SameCF, variou de 6% a 11%.

A Figura 4.5(c) mostra o comportamento das consultas tridimensionais. Para cada SF desse gráfico foi calculada a média de tempo das consultas Q2.1, Q2.2, Q2.3, Q3.1, Q3.2, Q3.3 e Q3.4. Nesse experimento, tanto o esquema FactDate quanto o esquema CNSSB organizaram os dados necessários às consultas em mais do que duas famílias de colunas, fato que prejudicou o desempenho das consultas realizadas nesses esquemas. Isso ocorre devido ao tempo de processamento adicional para a reconstrução de cada tupla, o qual supera o tempo para processar um arquivo *HFile* volumoso (presente no esquema SameCF). A partir dessa dimensionalidade, processar os dados armazenados no esquema SameCF garantiu os menores tempos e, consequentemente, melhores desempenhos, com ganhos na faixa de 21% a 31% quando comparado ao FactDate.

Por fim, a Figura 4.5(d) ilustra o comportamento das consultas que envolvem todas as quatro dimensões. Para cada SF desse gráfico foi calculada a média de tempo das consultas Q4.1, Q4.2 e Q4.3. Neste experimento, nota-se que os ganhos de desempenho do SameCF foram cada vez maiores, conforme o volume de dados aumentou, comportamento contrário ao exibido na Figura 4.5(a). Nas consultas envolvendo quatro dimensões, o SameCF superou o CNSSB e o FactDate em aproximadamente 54%.

Conclusão

Nesta seção foi analisado o impacto que diferentes organizações físicas têm sobre o desempenho de consultas OLAP, dependendo da quantidade de dimensões envolvidas. Para agregar os atributos em famílias de colunas, devem ser analisados os casos de uso mais frequentes de uma empresa, uma vez que essa agregação pode tanto beneficiar quanto prejudicar o desempenho das consultas cotidianas. Caso a maioria das análises envolva simultaneamente poucas dimensões, é vantajoso que os atributos dessas tabelas estejam agrupados na mesma família de colunas. Todavia, se muitas dimensões desnecessárias a essas consultas cotidianas forem adicionadas na mesma família, processar o elevado volume do arquivo *HFile* afeta o tempo dessas consultas, impactando negativamente seu desempenho. Caso contrário, quando as análises frequentes envolvem simultaneamente várias dimensões, minimizar a quantidade de famílias de colunas acessadas é uma abordagem que proporciona maior desempenho durante a agregação dos dados.

Dentro da etapa (i) de um *benchmark* para DW, foi investigado a maneira com que diferentes organizações físicas, relacionadas ao agrupamento de atributos em famílias de colunas, impactam no desempenho de consultas. Um *benchmark* para DW deve ser capaz de modelar consultas envolvendo diferentes combinações de dimensões com cardinalidades distintas quando for aplicado em uma empresa, a fim de simular com mais assertividade o cotidiano dessa empresa.

(a) Tempo médio das consultas unidimensionais.

(b) Tempo médio das consultas bidimensionais.

(c) Tempo médio das consultas tridimensionais.

(d) Tempo médio das consultas envolvendo todas as quatro dimensões do esquema.

Figura 4.5 – Comparação do tempo médio decorrido das consultas, divididas pela quantidade de dimensões agregadas pelas consultas.

A segunda parte da etapa (i) de um *benchmark* para DW consiste na carga de trabalho, na qual são propostas as consultas Qnew1 e Qnew2, usadas nesses experimentos, que são detalhadas na Seção 4.2.2.

4.2.2 Novas consultas para a Carga de Trabalho do SSB

A carga de trabalho do SSB é composta por 13 consultas OLAP, as quais envolvem a tabela de fatos *LineOrder* e agregam dados de diferentes dimensões. Essas consultas, categorizadas em quatro classes, promovem análises sobre as dimensões descritas a seguir: (i) a primeira classe (Q1.1, Q1.2 e Q1.3) agrega dados apenas da dimensão *Date*; (ii) a segunda classe (Q2.1, Q2.2 e Q2.3) envolve as dimensões *Date*, *Part* e *Supplier*; (iii) a terceira classe (Q3.1, Q3.2, Q3.3 e Q3.4) acessa atributos das dimensões *Customer*, *Supplier* e *Date*; e (iv) a quarta classe (Q4.1, Q4.2 e Q4.3) processa dados de todas as quatro dimensões do esquema. Todavia, não existem consultas que envolvam simultaneamente duas dimensões do esquema, e sim apenas consultas que acessam uma, três e quatro dimensões. A fim de avaliar o desempenho dessas consultas com duas dimensões, é proposta aqui uma nova classe de consultas ao SSB, composta

por duas consultas bidimensionais envolvendo as dimensões *Date* e *Part*.

A Tabela 3.1 mostra a cardinalidade de cada um dos atributos presentes no esquema do SSB e utilizados na filtragem das consultas. Cada um deles possui distribuição uniforme, cujo tamanho do domínio é constante e independente do valor de SF utilizado durante a geração de dados. Por meio desses atributos, é possível determinar e variar a seletividade das consultas executadas, podendo até mesmo definir a quantidade de registros retornados, dado um valor de SF. Neste trabalho, seletividade é definida como a quantidade de registros retornados, os quais serão agregados, dada a cardinalidade dos atributos utilizados nos predicados das consultas. A partir desse valor, se multiplicado pela quantidade total de tuplas da tabela *LineOrder*, é possível estimar quantas tuplas serão agregadas durante o processamento da consulta. A proposta das consultas Qnew1 e Qnew2, descritas nessa seção, utiliza como base esses valores de seletividade.

A primeira consulta proposta, denominada Qnew1 e descrita na Figura 4.6, calcula o rendimento máximo e mínimo dos produtos menos vendidos, agrupados por sua marca, dado um ano e uma categoria do produto. Os predicados dessa consulta são definidos sobre os atributos *d_year*, *p_category* e *lo_quantity*, tornando a seletividade da consulta equivalente a $\frac{1}{7} \times \frac{1}{25} \times \frac{25}{50} = 2,85 \times 10^{-3}$. Essa seletividade é próxima, pela mesma ordem de magnitude, das consultas Q2.2 ($1,60 \times 10^{-3}$) e Q3.2 ($1,37 \times 10^{-3}$), porém envolve apenas duas dimensões.

```
SELECT p_brand1, max(lo_revenue), min(lo_revenue)
FROM LineOrder, Date, Part
WHERE lo_orderdate = d_datekey
      AND lo_partkey = p_partkey
      AND d_year = 1993
      AND p_category = 'MFGR#11'
      AND lo_quantity < 25
GROUP BY p_brand1
ORDER BY p_brand1;
```

Figura 4.6 – Consulta Qnew1, responsável por agregar dados das dimensões *Date* e *Part*.

```
SELECT p_brand1, max(lo_revenue), min(lo_revenue)
FROM lineorder, dates, part
WHERE lo_orderdate = d_datekey
      AND lo_partkey = p_partkey
      AND d_yearmonthnum = 199301
      AND p_category = 'MFGR#11'
      AND lo_quantity < 25
GROUP BY p_brand1
ORDER BY p_brand1;
```

Figura 4.7 – Consulta Qnew2, a qual consiste na operação de *drill-down* sobre a consulta Qnew1, tornando-se dez vezes mais seletiva.

Já a segunda consulta proposta, denominada Qnew2 e descrita na Figura 4.7, é baseada na operação *drill-down* sobre a hierarquia de tempo, alternando a análise de ano para mês. A nova seletividade dessa consulta é maior em uma ordem de magnitude, uma vez que a seletividade do predicado envolvendo igualdade no atributo $d_yearmonthnum$ é $\frac{1}{84}$, fazendo com que a seletividade da consulta Qnew2 seja $\frac{1}{84} \times \frac{1}{25} \times \frac{25}{50} = 2,38 \times 10^{-4}$. Essa seletividade é próxima à seletividade das consultas Q1.2 ($6,49 \times 10^{-4}$) e Q2.3 ($2,00 \times 10^{-4}$). A Tabela 4.2 resume a seletividade das consultas definidas pelo SSB e propostas aqui.

Tabela 4.2 – Resumo da seletividade das consultas OLAP definidas pelo SSB, juntamente com as duas consultas propostas Qnew1 e Qnew2.

consulta	cálculo	seletividade	consulta	cálculo	seletividade
Q1.1	$\frac{1}{7} \times \frac{3}{11} \times \frac{25}{50}$	$1,95 \times 10^{-2}$	Q3.3	$\frac{2}{250} \times \frac{2}{250} \times \frac{6}{7}$	$5,49 \times 10^{-5}$
Q1.2	$\frac{1}{84} \times \frac{3}{11} \times \frac{10}{50}$	$6,49 \times 10^{-4}$	Q3.4	$\frac{2}{250} \times \frac{2}{250} \times \frac{1}{84}$	$7,62 \times 10^{-7}$
Q1.3	$\frac{1}{53} \times \frac{1}{7} \times \frac{3}{11} \times \frac{10}{50}$	$7,35 \times 10^{-5}$	Q4.1	$\frac{1}{5} \times \frac{1}{5} \times \frac{2}{5}$	$1,60 \times 10^{-2}$
Q2.1	$\frac{1}{25} \times \frac{1}{5}$	$8,00 \times 10^{-3}$	Q4.2	$\frac{2}{7} \times \frac{1}{5} \times \frac{1}{5} \times \frac{2}{5}$	$4,57 \times 10^{-3}$
Q2.2	$\frac{8}{1000} \times \frac{1}{5}$	$1,60 \times 10^{-3}$	Q4.3	$\frac{1}{5} \times \frac{1}{25} \times \frac{2}{7} \times \frac{1}{25}$	$9,14 \times 10^{-5}$
Q2.3	$\frac{1}{1000} \times \frac{1}{5}$	$2,00 \times 10^{-4}$	Qnew1	$\frac{1}{7} \times \frac{1}{25} \times \frac{25}{50}$	$2,85 \times 10^{-3}$
Q3.1	$\frac{1}{5} \times \frac{1}{5} \times \frac{6}{7}$	$3,43 \times 10^{-2}$	Qnew2	$\frac{1}{84} \times \frac{1}{25} \times \frac{25}{50}$	$2,38 \times 10^{-4}$
Q3.2	$\frac{1}{25} \times \frac{1}{25} \times \frac{6}{7}$	$1,37 \times 10^{-3}$			

Essas consultas foram propostas para avaliar análises bidimensionais, as quais não existem na carga de trabalho do SSB e são necessárias nos experimentos da Seção 4.2.1. O processo Map das consultas Qnew1 e Qnew2 é análogo ao Algoritmo 3, porém agregando diferentes atributos, os quais são determinados pela cláusula *group by*. Por se tratar de agregações de máximo e mínimo, o processo Reduce dessa agregação é detalhado no Algoritmo 4. Nele, dado um vetor contendo uma lista de elementos numéricos referentes ao atributo a serem agregados, são coletados os valores de máximo e mínimo, os quais são combinados e emitidos pelo processo Reduce.

Esta seção descreveu as análises e propostas sobre esquemas físicos e carga de trabalho, relacionadas à etapa (i) de um *benchmark* para DW. A Seção 4.3 investiga sobre a distribuição dos dados gerados pelo SSB, referente à próxima etapa do *benchmark*, ou seja, etapa (ii).

Algoritmo 4: reduceMaximoMinimo**Data:** Texto chave, Lista<Número> valores**Result:** Par (chave, $máximo \cup mínimo$)

```

1 begin
2   soma = 0;
3   máximo =  $-\infty$ ;
4   mínimo =  $+\infty$ ;
5   while número  $\in$  valores do
6     if mínimo  $\geq$  número then
7       mínimo = número;
8     if máximo  $\leq$  número then
9       máximo = número;
10  emite Par(chave,  $máximo \cup mínimo$ );

```

4.3 Distribuição dos dados gerados pelo SSB

Nessa seção é realizada uma análise sobre o processo de geração de dados do SSB. Mais precisamente, deseja-se verificar como é realizada a divisão dos dados gerados pelo *benchmark*. O objetivo desse estudo é determinar se ocorre desbalanceamento de dados entre os nós do *cluster*, ou seja, se os dados que atendem a uma determinada consulta OLAP estão localizados majoritariamente em apenas algumas das máquinas. Com isso, os dados de certas máquinas podem não contribuir com os dados agregados pela consulta. Os experimentos realizados na análise consideraram os registros armazenados dentro de cada *RegionServer* no HBase. Foram utilizados os valores 10 e 20 de SF. Para o valor 10 de SF, o esquema físico *FactDate* divide os aproximadamente 60 milhões de registros em 16 regiões, ao passo que, com o valor 20 de SF, as 120 milhões de tuplas são separadas em 32 regiões.

A Figura 4.8 mostra, para o valor 10 de SF, a quantidade percentual de tuplas necessárias à execução de cada consulta que são extraídas de cada *RegionServer*. Por exemplo, para a consulta Q1.1 da carga de trabalho do SSB são agregados dados de aproximadamente um milhão e duzentas mil tuplas, em que cada uma das 16 regiões contém, em média, 74.6 mil dessas instâncias (6,25%). Nota-se que, por conta da distribuição uniforme dos dados gerados pelo SSB, os registros necessários a cada uma das consultas ficaram igualmente divididos entre os *RegionServers*. Apenas a consulta Q3.4 mostra uma elevada variação na quantidade de tuplas recuperadas de cada região, contendo desde regiões que não possuem nenhum registro necessário à consulta (0%) até regiões que contêm mais de 11% desses registros.

Esse comportamento exibido pela consulta Q3.4 ocorreu devido à sua seletividade ser da ordem de magnitude de 1×10^{-7} , sendo a maior dentre todas as consultas avaliadas, conforme ilustrado na Tabela 4.2. Por conta dessa seletividade, dos 60 milhões de registros existentes no esquema, apenas foram selecionadas e agregadas 53 registros, o que representa uma quantidade ínfima de tuplas, cujo volume reduzido impactou o balanceamento desses dados entre as regiões

Figura 4.8 – Quantidade percentual de registros que são agregados de cada *RegionServer*, agrupados por consulta, considerando SF = 10. Nesse gráfico nota-se que a distribuição de tuplas entre as regiões ocorre de forma balanceada, com as consultas mais seletivas mostrando maiores variações.

em que estão armazenados. Desconsiderando a consulta Q3.4, as outras consultas bastante seletivas, na ordem de magnitude de 1×10^{-5} , são as consultas Q1.3, Q3.3 e Q4.3. Elas também mostraram uma maior variação que as demais, exibindo valores de 5,67% a 7%. Outras consultas com variação considerável são as consultas Q1.2, Q2.3 e Qnew2, cuja seletividade é da ordem de magnitude de 1×10^{-4} . A partir dessa seletividade, consultas com seletividade da ordem de magnitude de 1×10^{-3} ou menor exibiram uma variação ínfima.

Já a Figura 4.9 mostra esse mesmo comportamento após dobrar-se o volume de dados utilizado, ou seja, armazenando os 120 milhões de registros, correspondentes ao valor 20 de SF. Considerando que nesse novo volume os dados foram divididos entre 32 regiões, cada um dos *RegionServers* contribuiu em média com aproximadamente 3,13% dos registros agregados pelas consultas OLAP. Novamente a consulta Q3.4 mostrou a maior variação por conta de sua elevada seletividade, exibindo valores entre 0% e 7,95%. Já as consultas Q1.3, Q3.3 e Q4.3 também mostraram uma maior variação que as demais, com os valores na faixa de 2,74% a 3,70%, assim como consultas Q1.2, Q2.3 e Qnew2, com valores na faixa de 2,83% a 3,35%.

Por meio desse estudo foi possível avaliar que, ao inserir os dados gerados pelo SSB no HBase, esses dados foram igualmente divididos entre os *RegionServers*. Isso possibilita que os dados necessários para responder às consultas definidas pela carga de trabalho do SSB, juntamente às consultas Qnew1 e Qnew2 propostas na Seção 4.2.2, estejam igualmente divididas

entre as regiões. Essa distribuição apenas foi prejudicada em consultas muito seletivas, como no caso da Q3.4. Porém, como essa consulta seleciona uma quantidade muito pequena de registros para serem agregados, esse desbalanceamento não representa um impacto muito grande na distribuição dos dados.

Figura 4.9 – Quantidade percentual de registros que são agregados de cada *RegionServer*, agrupados por consulta, considerando $SF = 20$. Mesmo dobrando o volume de dados, o relacionamento da seletividade com a distribuição balanceada dos dados entre as regiões se manteve.

4.4 Parâmetros do Processamento Distribuído

Nessa seção é analisada a etapa (iii) de um *benchmark* para DW, que define os parâmetros utilizados e as métricas coletadas. A principal métrica quantitativa do SSB, assim como a medida fundamental de todos os *benchmarks* descritos no Capítulo 3, é o desempenho. Esse desempenho é calculado pela quantidade de consultas executadas por unidade de tempo, mais precisamente, o tempo decorrido em cada uma das consultas. Relacionada a essa duração, o parâmetro essencial é o volume de dados, o qual é definido pelo valor de SF .

Todavia, existem outros parâmetros, além do volume de dados, que impactam no desempenho reportado por um *benchmark*. Nesta seção são investigados três parâmetros importantes que influenciam no processamento distribuído de dados usando o *framework* Hadoop MapReduce: a quantidade de processos Map executando simultaneamente em cada nó do *cluster* (Seção 4.4.1); a quantidade total de processos Reduce (Seção 4.4.2); e o tamanho do bloco

de dados distribuído (Seção 4.4.3). Os experimentos dessa seção foram realizados no *cluster* definido na Seção 4.2.1.

4.4.1 Processos Map executados simultaneamente

Essa investigação consiste em analisar a influência, no desempenho das consultas, da quantidade de processos Map executados simultaneamente em cada nó do *cluster*. O experimento foi realizado no banco de dados NoSQL HBase, armazenando os dados com o esquema físico FactDate, proposto na Seção 4.2.1. O volume de dados utilizado nesse experimento corresponde ao valor 20 de SF, conforme mostrado na Tabela 4.1. A fim de configurar a quantidade de processos Map, deve-se determinar a memória alocada por cada processo. Uma vez que cada nó possui 16 GB de memória primária, sendo 4 GB reservados para o sistema operacional e para o HBase, existem disponíveis 12 GB de memória para serem alocados pelos processos Map.

A Tabela 4.3 exibe as três configurações utilizadas nessa investigação, considerando o total de memória disponível em cada nó. Variou-se a quantidade de processos Map simultâneos entre os valores 2, 4 e 6. A configuração máxima foi de seis processos Map por nó, uma vez que acima desse valor, ou seja, alocando menos de 2 GB de memória por processo, resulta na insuficiência de recursos computacionais e falha de execução nas consultas. Para cada configuração, foi calculada a média de tempo de 10 execuções de cada consulta definida pelo SSB, além das consultas propostas na Seção 4.2.2.

Tabela 4.3 – Três configurações investigadas, referentes à quantidade de processos Map que podem executar simultaneamente em um mesmo nó, juntamente com a memória alocada para cada processo Map.

configuração	memória para cada processo Map
2 Map / nó	6 GB
4 Map / nó	3 GB
6 Map / nó	2 GB

A Figura 4.10 exibe o tempo médio de processamento das consultas de classe 1 da carga de trabalho do SSB. O desempenho dessas consultas, responsáveis por realizar a agregação de soma total, melhorou conforme maior a quantidade de processos Map executando simultaneamente. Em relação ao ganho de desempenho dessas consultas, na configuração de seis processos Map simultaneamente, o ganho foi de aproximadamente 20% quando comparado à pior configuração (dois processos Map simultaneamente). Uma característica importante dessas consultas é que, por não definirem categorias, elas são processadas utilizando-se o Algoritmo 1, cujos resultados intermediários entre os processos Map e Reduce possuem um volume de dados reduzido (devido à chave possuir valor nulo).

A Figura 4.11 exibe o tempo médio de processamento das consultas de classe 2, 3 e 4 da carga de trabalho do SSB. Nessas consultas, o cenário inverteu-se, sendo que os piores desempenhos ocorreram usando a configuração de seis processos Map simultaneamente. Para as

Figura 4.10 – Tempo médio de processamento de consultas que agregam dados de soma total, mostrando que conforme mais processos Map são executados simultaneamente, melhor o desempenho.

consultas tridimensionais, o tempo de processamento usando dois processos Map simultâneos foi aproximadamente 20% menor que a configuração de seis em paralelo. Já no caso em que todas as quatro dimensões são acessadas, o ganho de desempenho aumentou para 31%. Essa inversão ocorreu pois, como essas consultas realizam a agregação dos dados divididos em categorias (Algoritmo 3), os resultados intermediários entre os processos Map e Reduce são maiores em termos de volume de dados. Esse aumento no volume intermediário é causado pelo fato do campo chave possuir a combinação dos atributos a serem agrupados. Uma vez que cada nó do *cluster* contém apenas um disco, isso resultou em uma maior concorrência de recursos para armazenamento e leitura dos resultados parciais originados pelos processos Map. Além disso, durante a etapa de *shuffle*, a comparação é realizada sobre chaves com maiores tamanhos, sendo mais custosa, e a transferência de dados entre as máquinas é mais intensa, devido ao tamanho desses dados.

Por fim, a Figura 4.12 exibe o tempo médio de processamento das consultas Qnew1 e Qnew2, propostas na Seção 4.2.2. Diferentemente dos casos anteriores, essas consultas resultaram em um caso intermediário, possuindo como configuração de melhor desempenho a utilização de quatro processos Map simultaneamente. Esse comportamento ocorreu devido ao fato de que as consultas Qnew1 e Qnew2 possuem grupos pequenos, compostos por apenas um atributo, se comparado às consultas do SSB, onde os grupos são definidos baseados em dois ou mais atributos combinados. Por utilizar grupos menores, o desempenho das consultas usando a configuração de quatro processos Map foi levemente melhor (aproximadamente 9%), porém, ao aumentar essa quantidade de processos simultâneos, o desempenho piorou.

Por meio do experimento descrito nesta seção, verificou-se que existem outros parâmetros do processamento distribuído de dados usando o *framework* Hadoop MapReduce que devem ser

Figura 4.11 – Tempo médio de processamento de consultas que agregam dados divididos em categorias, ou seja, possuem a cláusula *group by*. Isso aumenta o volume de dados intermediários entre os processos Map e Reduce, resultando em uma maior competição de disco e, consequentemente, degradação do desempenho das consultas.

analisados durante a aplicação de um *benchmark* para DW, além do volume de dados. Também mostrou-se que a quantidade de processos Map executados simultaneamente é influenciada pelo tipo de consulta realizada. Quanto mais processos Map são processados ao mesmo tempo, melhor o desempenho médio das consultas. Porém uma nova limitação pode surgir: a competição por disco durante a escrita dos resultados intermediários entre os processos Map e Reduce. Outro parâmetro investigado neste trabalho é a quantidade total de processos Reduce utilizado nas consultas, assunto detalhado na Seção 4.4.2.

Figura 4.12 – Tempo médio de processamento de consultas que agregam dados divididos em categorias, porém possuindo grupos de tamanhos pequenos (compostos pela concatenação de apenas um atributo) e realizando duas agregações simultaneamente.

4.4.2 Quantidade total de processos Reduce

Continuando a investigação dos parâmetros descritos na Seção 4.4.1, foi estudada a influência da quantidade total de processos Reduce executados durante o processamento das consultas. O esquema utilizado nesse experimento também foi o *FactDate*, com o volume de dados correspondente ao valor 20 de SF, além de usar a configuração de quatro processos Map simultaneamente. Esta configuração foi utilizada por ter apresentado resultados de desempenho intermediários para as consultas analisadas na seção anterior, possuindo assim, em média, os melhores desempenhos para todas as classes de consultas. No processo de agregação dos dados, orientado pelas consultas SQL, a quantidade de grupos resultantes é fixa. A Tabela 4.4 exibe o número de grupos gerados para as consultas definidas pela carga de trabalho do SSB, juntamente com as consultas Qnew1 e Qnew2 propostas na Seção 4.2.2.

Tabela 4.4 – Número de categorias geradas por consulta, considerando o volume de dados correspondente ao valor 20 de SF.

consulta	número de grupos	consulta	número de grupos
Q1.1	1	Q3.1	150
Q1.2	1	Q3.2	600
Q1.3	1	Q3.3	24
Qnew1	40	Q3.4	4
Qnew2	40	Q4.1	35
Q2.1	280	Q4.2	100
Q2.2	56	Q4.3	800
Q2.3	7		

O número de grupos é diretamente relacionado à quantidade de chaves distintas geradas pelo processo Map. Cada chave é enviada a um processo Reduce específico, em que a mesma chave não pode ser processada em diferentes processos Reduce, porém chaves distintas podem ser associadas a um mesmo processo Reduce. Partindo dessa regra, as consultas da classe 1 (Q1.1, Q1.2 e Q1.3), por exemplo, só podem ser executadas por um processo Reduce, enquanto que as consultas Qnew1 e Qnew2 podem utilizar no máximo 40 processos Reduce distintos, cada qual processando apenas uma chave e agregando os respectivos valores por meio da soma.

A Figura 4.13 exibe o tempo médio de 10 execuções das consultas pertencentes à classe 4 (Q4.1, Q4.2 e Q4.3) da carga do trabalho do SSB. Essas consultas foram avaliadas devido ao elevado volume de dados, pois acessam todas as quatro dimensões do esquema, ou seja, todas as famílias de colunas definidas. Ter apenas um processo Reduce implica que os dados intermediários, resultantes dos três *DataNodes*, devem ser enviados a um único nó a fim de agregá-los, resultando em um tempo adicional para transferência dos dados (na etapa *shuffle*) e, conseqüentemente, perda de desempenho. Porém, se a quantidade de processos Reduce for muito elevada, o tempo de processamento aumenta pois será necessário empilhar e iniciar vários processos Reduce, cada qual agregando um pequeno volume de dados. Adicionalmente, a partir

de dez processos Reduce, o desempenho das consultas piora conforme aumenta-se a quantidade total definida de processos Reduce. A consulta Q4.1, ao definir mais de 20 processos Reduce, permanece com tempo constante pois possui 35 chaves distintas, ou seja, mesmo definindo 40 processos Reduce, não são alocados mais do que 35, fazendo com que o tempo não aumente mais a partir desse valor.

Figura 4.13 – Tempo decorrido das consultas de classe 4 (Q4.1, Q4.2 e Q4.2) conforme varia-se a quantidade máxima de processos Reduce definidos. O gráfico mostra que determinar entre cinco e dez processos Reduce exibiu os melhores desempenhos entre as consultas analisadas.

Conclui-se que outro parâmetro importante de um *benchmark* para DW, considerando o processamento pelo *framework* Hadoop MapReduce, é a quantidade total de processos Reduce definidos. Esse experimento mostrou que, se essa quantidade for muito pequena, o desempenho dessas consultas é baixo pois apenas poucos nós do *cluster* processarão a agregação de um elevado volume de dados. Porém, ao elevar muito essa quantidade, são alocados muitos processos Reduce para processar um pequeno volume de dados, em que o tempo adicional para transferir dados e gerenciar esses processos impacta no desempenho da consulta. Por fim, a quantidade de processos Reduce é limitada pelo número de grupos retornados na consulta, onde esse parâmetro não tem influência caso seja maior que a quantidade de chaves distintas da consulta.

O último parâmetro avaliado neste trabalho é o tamanho do bloco de dados distribuído e a influência dele em consultas OLAP. A investigação sobre esse parâmetro é descrita na Seção 4.4.3.

4.4.3 Tamanho do bloco de dados distribuído

Quando um arquivo é submetido ao HDFS, os dados desse arquivo são divididos em blocos de tamanho fixo. Conforme descrito na Figura 2.9, para cada bloco é alocado um processo Map distinto. Ao variar o tamanho desse bloco, altera-se a quantidade de processos Map necessários para processar os dados. Considerando o arquivo texto desnormalizado, relacionado ao valor 20 de SF (Tabela 4.1), a Tabela 4.5 mostra a quantidade de blocos gerados, considerando três configurações distintas de tamanho de blocos de dados: 128, 256 e 512 MB.

Tabela 4.5 – Quantidade de blocos de dados distribuídos de acordo com o tamanho deles. Durante o processamento, para cada bloco é criado um processo Map.

tamanho bloco de dados	quantidade de blocos
128 MB	433
256 MB	217
512 MB	109

A Figura 4.14 exibe o tempo médio de dez execuções das 13 consultas presentes na carga de trabalho do SSB, juntamente com as consultas Qnew1 e Qnew2 propostas na Seção 4.2.2. Neste experimento, quanto maior o tamanho do bloco de dados, melhor o desempenho das operações de leitura e, conseqüentemente, menor o tempo de processamento das consultas. Ao comparar o tempo necessário para processar blocos de tamanho 512 MB, em relação aos de tamanho 128 MB, o tempo da primeira configuração foi três vezes menor que o da segunda. Quanto maior o tamanho desses blocos, mais memória é necessária aos processos Map, logo, para tamanhos acima de 512 MB, a memória alocada para cada processo Map foi insuficiente (considerando a configuração de quatro processos Map por nó, ou seja 3 GB).

Tendo em vista esse ganho de desempenho ao variar o tamanho de bloco, investigou-se o impacto desse tamanho em um banco de dados NoSQL. A Figura 4.15 exibe o tempo decorrido pelas mesmas consultas da Figura 4.14, porém com os dados armazenados no HBase, usando o esquema FactDate. Neste novo gráfico, não há uma influência significativa do tamanho do bloco no tempo de processamento das consultas. Isto ocorreu pois a organização dos dados do HBase sobrepôs a configuração do tamanho de blocos do HDFS, uma vez que os dados do HBase são distribuídos horizontalmente, organizados em *RegionServers*. Independentemente do tamanho de blocos de dados utilizados, por meio do esquema FactDate os dados são organizados em 32 regiões, cada qual com tamanho de 2 GB, o que fixa o processamento desses dados em 32 processos Map.

Tendo em vista a falta de influência do tamanho de bloco ao armazenar os dados em um banco NoSQL, surge o questionamento sobre os parâmetros internos ao banco. Mais especificamente, surge a necessidade de se analisar o impacto na variação do tamanho dos *RegionServers*, os quais alteram a quantidade de processos Map necessários para processar a consulta, similar ao comportamento do tamanho de bloco de dados no HDFS. A Figura 4.16 exibe o tempo médio de

Figura 4.14 – Tempo decorrido de consultas OLAP sobre um arquivo de texto desnormalizado, correspondente ao valor 20 de SF. Nele variou-se o tamanho de bloco de dados distribuídos, mostrando que quanto maior esse valor, melhor o desempenho dessas consultas.

Figura 4.15 – Tempo decorrido de consultas OLAP sobre o esquema FactDate do HBase. Nesse gráfico, mesmo variando o tamanho do bloco de dados, o tempo de processamento das consultas não exibiu diferença significativa.

processamento das consultas considerando três tamanhos de *RegionServers*, 1 GB, 2 GB e 4 GB, que correspondem respectivamente a 64, 32 e 16 regiões. Ao aumentar o tamanho da região de 1 GB para 2 GB, o tempo das consultas permaneceu constante para a maioria das consultas. A influência dessa configuração ficou evidente apenas nas consultas da classe 4, as quais processam um volume maior de dados, ou seja, todas as famílias de colunas do esquema. O ganho de desempenho máximo dessa primeira variação foi de 14%, na consulta Q4.2. Já a variação de tamanho de *RegionServer* de 2 GB para 4 GB degradou o desempenho de processamento de

todas as consultas. Isso ocorre pois o volume de dados processados em cada processo Map é maior que a memória alocada por eles, o que resulta em erros nas conexões com o banco e, consequentemente tempo de processamento adicional para restabelecer essas conexões.

Figura 4.16 – Tempo decorrido de consultas OLAP sobre o esquema FactDate do HBase, variando o tamanho dos *RegionServers*. A influência dessa configuração é evidente nas consultas de classe 4, em que a variação de 2 GB para 4 GB proveu ganho de desempenho de 14% na consulta Q4.2.

Este teste analisou a influência do tamanho de blocos de dados no tempo de processamento de consultas. Verificou-se que quanto maior o tamanho desses blocos, menor a quantidade de processos Map necessários e melhor o desempenho dessas consultas. Contudo, esse benefício vem com um custo, que consiste na memória adicional necessária para processar esses dados, o que pode tornar inviável a alocação de recursos. Adicionalmente, mesmo que um parâmetro do sistema de arquivos influencie no desempenho de processamento de consultas, ele pode não afetar o tempo em um banco de dados NoSQL. Essa influência demonstra mais um importante parâmetro a ser analisado por um *benchmark* para DW, o qual deve ser incorporado em sua validação, a fim de promover comparações justas de desempenho entre diferentes bancos de dados NoSQL distribuídos e orientados a colunas.

Após definir os três parâmetros anteriormente investigados para a agregação dos dados distribuídos, usando o *framework* Hadoop MapReduce, ainda existe a necessidade de se investigar a etapa (iv), que consiste no processamento e validação dos dados. Esta última etapa é detalhada na Seção 4.5.

4.5 Processamento e Validação

Essa seção descreve propostas e investigações realizadas sobre a etapa (iv) de um *benchmark* para DW. Mais especificamente, são detalhadas duas análises referentes ao processamento

de dados distribuídos. Na Seção 4.5.1 investiga-se o ganho de desempenho relacionado à quantidade de nós do *cluster* utilizado durante o processamento das consultas, bem como esse ganho considerando diferentes volumes de dados. Já na Seção 4.5.2 é proposta uma otimização de consultas OLAP iterativas, por meio do uso do *framework* Spark.

4.5.1 Escalabilidade contra variação do volume de dados

Nessa seção é analisado o tempo de processamento das consultas em relação à variação tanto de quantidade de nós do *cluster* (ou seja, escalabilidade) quanto de volume de dados. O objetivo dessa investigação é comparar a diferença de tempo ocasionada pela quantidade de nós responsáveis pelo armazenamento e processamento das consultas OLAP. Além disso, avalia-se qual o comportamento dessas divergências no tempo de acordo com o crescimento do volume de dados. A Figura 4.17 mostra o tempo médio, de 10 execuções, do processamento das consultas da carga de trabalho do SSB, juntamente com as consultas Qnew1 e Qnew2 propostas na Seção 4.2.2. O esquema utilizado nessa análise foi o FactDate, proposto na Seção 4.2.1, com o volume de dados correspondente ao valor 5 de SF (28 GB dentro do HBase). As consultas foram divididas em quatro categorias, referentes à quantidade de dimensões (ou melhor, famílias de colunas) acessadas durante a execução das consultas.

Na Figura 4.17(a), o tempo decorrido do processamento das consultas unidimensionais foi aproximadamente 50% menor, quando compara-se o *cluster* contendo 4 nós contra a infraestrutura com apenas 2 nós. A consulta Q1.1 demorou aproximadamente 53,23% menos tempo ao se dobrar o tamanho do *cluster*, à medida que a consulta Q1.2 ficou 48,38% mais rápida e a consulta Q1.3 obteve ganhos de desempenho de 42,76%. Nota-se também, que na consulta Q1.1 a diminuição do tempo decorrido entre as configurações de 2 e 3 nós foi 22,61% menor que as outras Q1.2 e Q1.3, com 41,49% e 39,71% respectivamente. Comparando esses resultados com a seletividade das consultas envolvidas, exibida na Tabela 4.2, tem-se que a consulta Q1.1 é menos seletiva que as demais em pelo menos duas ordens de magnitude.

Na Figura 4.17(b) é exibida a diminuição no tempo de processamento das consultas bidimensionais. Nela, ao aumentar de 2 para 3 nós, a diminuição foi de aproximadamente 43%, com Qnew1 e Qnew2 mostrando ganhos de desempenho de respectivamente 44,29% e 42,43%. Ao comparar a redução no tempo decorrido entre as infraestruturas de 2 e 4 nós, essa diminuição aumentou para aproximadamente 57%, com Qnew1 e Qnew2 obtendo ganhos de desempenho de 57,17% e 57,94% respectivamente. Já na Figura 4.17(c) a redução dos tempos médios decorridos foi na faixa de 68,56% a 72,51%. Nesta figura em particular, nota-se também o impacto da seletividade das consultas, uma vez que a consulta Q3.1 possui uma das menores seletividades, da ordem de magnitude de 1×10^{-2} , quando comparada com as outras consultas tridimensionais (Q3.2, Q3.3 e Q3.4), presentes na Tabela 4.2. Por fim, na Figura 4.17(d) o ganho de desempenho médio foi de 75%, considerando a diminuição de tempo de processamento das infraestruturas contendo 2 e 4 nós.

Figura 4.17 – Teste de escalabilidade, considerando a comparação do tempo médio de processamento das consultas sobre uma mesma base de dados, variando a quantidade de nós utilizados tanto no armazenamento quanto no processamento das consultas.

Com base nesses resultados, relacionados à diminuição percentual do tempo médio das consultas à medida que varia-se a quantidade de nós, foi analisado o comportamento desse ganho de desempenho conforme aumenta-se o volume de dados analisado. A Figura 4.18 mostra que a influência das famílias de colunas exibida na Figura 4.17 diminuiu conforme o volume de dados aumentou. Após usar uma base proporcional ao valor 10 de SF (57,9 GB), o ganho de desempenho de todas as consultas convergiu para a faixa de 50% a 60%. Já no volume correspondente ao valor 15 de SF (84,1 GB), esse ganho se manteve nessa faixa. Isso mostra que, para grandes volumes de dados, adotar uma infraestrutura de 4 nós tende a diminuir o tempo de processamento pela metade, quando comparado com a execução dessas mesmas consultas em um *cluster* de 2 nós.

A última investigação realizada sobre a etapa (iv) de um *benchmark* para DW consiste na proposta do uso de visões materializadas para otimizar o processamento de consultas OLAP iterativas, proporcionada pelo *framework* Spark. Os detalhes desta proposta estão na Seção 4.5.2.

Figura 4.18 – Percentual de ganho de desempenho das quatro categorias de consultas, ao comparar o tempo de processamento de uma consulta em 4 nós com o tempo decorrido dessa mesma consulta em 2 nós. Os gráficos mostram o comportamento desse ganho conforme aumenta-se o volume de dados avaliado.

4.5.2 Uso de Visões Materializadas em Consultas OLAP Iterativas

Nessa seção é proposto o armazenamento de atributos, juntamente com suas respectivas hierarquias, em Visões Materializadas (VM). Durante o processamento de uma consulta, são acessados atributos tanto de projeção quanto de filtragem dos dados. Esses atributos podem ser armazenados em uma VM, para que futuras execuções dessa mesma consulta, usando os mesmos atributos com diferentes predicados, não precisem processar o conjunto de dados original, melhorando o desempenho. Contudo, apenas são otimizadas as consultas praticamente idênticas às executadas anteriormente (em termos de atributos utilizados). Por meio das VMs propostas nessa seção, é possível otimizar futuras consultas sobre o conjunto de dados, porém em outros níveis de agregação. Nesta dissertação, define-se como consultas OLAP iterativas o processo de agregação dos dados por meio de uma consulta *ad-hoc*, seguido de operações OLAP (como *drill-down* e *roll-up*) iterando sobre as hierarquias de atributos desse conjunto de dados. Para avaliar o benefício do uso de VMs no processamento de consultas OLAP iterativas, são avaliados três cenários:

- **Sem VM:** este primeiro cenário é formado pela execução sequencial de consultas OLAP,

sem a criação ou uso de uma visão materializada. Ele corresponde ao cenário considerado em todos os demais testes de desempenho descritos neste capítulo;

- **VM hierárquica:** o cenário proposto consiste em gerar uma visão materializada contendo os atributos necessários para a consulta sendo atualmente processada, além de outros atributos pertencentes às hierarquias envolvidas; e
- **VM por consulta:** este último cenário consiste em gerar uma visão materializada para cada consulta processada, contendo apenas os atributos necessários à consulta.

Para possibilitar o uso de visões materializadas, as consultas foram implementadas usando o *framework* Spark. O Spark possibilita o uso de RDDs, que são coleções particionadas de objetos entre os nós de um *cluster*, permitindo o processamento iterativo e distribuído desses dados. A abstração proposta nesse trabalho é considerar um RDD como uma visão materializada, a fim de otimizar o processamento de consultas OLAP iterativas. Nesse sentido, optou-se pelo *framework* Spark ao invés do *framework* Hadoop MapReduce, sendo este último caracterizado por realizar um processamento em lote dos dados, precisando ser adaptado para gerar a visão materializada em um processo MapReduce, a fim de sua reutilização em futuros processos MapReduce. Adicionalmente, entre os processos MapReduce, toda a visão materializada seria armazenada e relida do disco integralmente, ao passo que no *framework* Spark parte dessas visões, que ainda estão em memória primária, são reaproveitadas durante o processamento.

A Tabela 4.6 mostra os atributos pertencentes a cada VM, agrupadas pelas classes de consultas do SSB, juntamente com as consultas Qnew1 e Qnew2 detalhadas na Seção 4.2.1. Por exemplo, na classe 1 as visões materializadas VM11, VM12 e VM13 correspondem às consultas Q1.1, Q1.2 e Q1.3 respectivamente, contendo apenas os atributos acessados em cada uma dessas consultas. Já a visão materializada VM1 possui todos os atributos necessários para atender a essas três consultas.

A Figura 4.7 exibe o volume ocupado pelas visões materializadas, tanto em memória primária quanto em memória secundária. Esses volumes correspondem aos arquivos texto desnormalizados, os quais estão relacionados aos valores 10, 50 e 100 de SF (28 GB, 122 GB e 244 GB, respectivamente). No menor volume de dados (SF = 10), todas as VMs são armazenadas apenas em memória primária. Acima desse volume, é necessário armazenar a parte excedente dessas visões em memória secundária (disco), devido às limitações de memória primária máxima disponível. Uma vez que as VMs hierárquicas propostas nesse trabalho (VM1, VM2, VM3, VM4 e VMnew) armazenam atributos de diferentes níveis de agregação, o volume total delas é igual ou superior às VMs específicas de cada consulta relacionada.

A Figura 4.19 mostra o tempo médio de 15 execuções das consultas de classes 1, 2, 3 e 4 da carga de trabalho do SSB, juntamente com as consultas Qnew1 e Qnew2, usando SF = 50. Ao comparar os cenários sem o uso de VMs com a criação e utilização de VMs específicas para

Tabela 4.6 – Atributos presentes nas Visões Materializadas (VMs) específicas para cada consulta, juntamente com os atributos armazenados nas VM propostas para otimização de operações OLAP iterativas.

Atributos das Visões Materializadas (VMs) específicas				VM proposta
VM11	VM12	VM13		VM1
d_year	d_yearmonthnum	d_year		d_year
		d_weeknuminyear		d_yearmonthnum
lo_extendedprice	lo_extendedprice	lo_extendedprice		d_weeknuminyear
lo_discount	lo_discount	lo_discount		lo_extendedprice
lo_quantity	lo_quantity	lo_quantity		lo_discount
				lo_quantity
VM21	VM22	VM23		VM2
s_region	s_region	s_region		s_region
p_category				p_category
p_brand1	p_brand1	p_brand1		p_brand1
d_year	d_year	d_year		d_year
lo_revenue	lo_revenue	lo_revenue		lo_revenue
VM31	VM32	VM33	VM34	VM3
c_region				c_region
c_nation	c_nation			c_nation
	c_city	c_city	c_city	c_city
s_region				s_region
s_nation	s_nation			s_nation
	s_city	s_city	s_city	s_city
d_year	d_year	d_year	d_year	d_year
			d_yearmonth	d_yearmonth
lo_revenue	lo_revenue	lo_revenue	lo_revenue	lo_revenue
VM41	VM42	VM43		VM4
s_region	s_region			s_region
	s_nation	s_nation		s_nation
		s_city		s_city
c_region	c_region	c_region		c_region
c_nation				c_nation
p_mfgr	p_mfgr			p_mfgr
	p_category	p_category		p_category
		p_brand1		p_brand1
d_year	d_year	d_year		d_year
lo_revenue	lo_revenue	lo_revenue		lo_revenue
lo_supplycost	lo_supplycost	lo_supplycost		lo_supplycost
VMnew1	VMnew2			VMnew
p_brand1		p_brand1		p_brand1
p_category		p_category		p_category
d_year				d_year
		d_yearmonthnum		d_yearmonthnum
lo_revenue		lo_revenue		lo_revenue
lo_quantity		lo_quantity		lo_quantity

Tabela 4.7 – Volume ocupado pelas visões materializadas (VMs), mostrando o espaço ocupado tanto em memória primária quanto em memória secundária.

visão materializada	SF = 10		SF = 50		SF = 100	
	memória primária (GB)	memória secundária (GB)	memória primária (GB)	memória secundária (GB)	memória primária (GB)	memória secundária (GB)
VM11	0,543	-	2,715	-	4,607	0,824
VM12	0,599	-	2,995	-	4,605	1,385
VM13	0,599	-	2,995	-	4,602	1,388
VM1	0,766	-	3,833	-	4,580	3,086
VM21	1,693	-	4,408	4,053	4,409	12,512
VM22	1,302	-	4,458	2,046	4,471	8,539
VM23	1,302	-	4,458	2,046	4,471	8,539
VM2	1,693	-	4,408	4,053	4,409	12,512
VM31	1,983	-	4,377	5,529	4,402	15,415
VM32	2,339	-	4,386	7,309	4,388	19,005
VM33	1,548	-	4,494	3,246	4,487	10,995
VM34	1,939	-	4,404	5,293	4,397	14,996
VM3	3,491	-	4,262	13,187	4,259	30,646
VM41	1,921	-	4,391	5,212	4,385	14,824
VM42	2,313	-	4,251	7,306	4,282	18,836
VM43	2,645	-	4,252	8,973	4,248	22,208
VM4	3,757	-	4,046	14,735	4,063	33,504
VMnew1	1,367	-	4,498	2,341	4,490	9,189
VMnew2	1,423	-	4,484	2,634	4,499	9,739
VMnew	1,535	-	4,481	3,196	4,497	10,858

cada consulta, nota-se que o custo adicional para criar uma VM, antes de executar cada consulta, aumentou o tempo acumulado para processar todas as consultas de forma iterativa. Já no cenário contendo as VMs hierárquicas, o tempo adicional de criação foi um pouco maior que o das VMs específicas, uma vez que as VMs hierárquicas possuem a mesma quantidade de atributos ou mais. Contudo, por armazenarem atributos de diferentes níveis de agregação, essas VMs podem ser reaproveitadas em futuras consultas, melhorando o desempenho acumulado de análises iterativas. Quanto maior o reaproveitamento dessas VMs, menor o tempo total das consultas que a utilizam.

Comparando os tempos totais obtidos, na Figura 4.19(a) o uso da VM hierárquica otimizou o tempo de processamento de consultas que não utilizam nenhuma VM em aproximadamente 57,92%, ao passo que o uso de VMs específicas por consulta piorou o tempo de processamento das consultas que não utilizaram nenhuma VM em 14,49%. Já na Figura 4.19(b), o ganho de desempenho da VM hierárquica foi de 48,98%, quando comparado ao processamento de consultas sem usar VMs, à medida que na Figura 4.19(c) esse ganho foi de aproximadamente 51,19%. Todavia, na Figura 4.19(d) esse ganho de desempenho foi pequeno, em torno de 37,19%.

Figura 4.19 – Tempo de processamento das consultas OLAP iterativas, usando SF = 50, organizada por classe de consultas. Note que, por causa de algumas visões materializadas, algumas consultas são processadas em um espaço de tempo muito curto.

Este último caso ocorreu pois as consultas da classe 4 manipulam uma grande quantidade de atributos, oriundos de todas as dimensões do esquema, conforme mostrado na Tabela 4.6. Em decorrência disso, as VMs hierárquicas ficam muito volumosas, prejudicando o desempenho de sua criação e utilização. Por fim, na Figura 4.19(e) o ganho de desempenho total das consultas usando VM hierárquicas em relação às consultas sem utilizar VMs foi de 33,17%, que ocorreu devido ao fato de que a VM criada apenas responde a duas consultas OLAP.

Figura 4.20 – Análise do tempo das consultas iterativas, variando o volume de dados nos valores 10, 50 e 100 de SF. Conforme o volume de dados aumentou, o tempo acumulado das consultas no cenário das VMs hierárquica foi menor que os outros dois cenários.

Analisando os diferentes volumes de dados descritos, os gráficos da Figura 4.20 exibem o tempo acumulado das consultas, categorizadas em classes. Nesse experimento foram testados os volumes de dados correspondentes aos valores 10, 50 e 100 de SF. O crescimento do tempo decorrido no cenário de VMs específicas por consulta foi maior que o todos os outros cenários, pois a criação e gerenciamento de cada VM requer um tempo adicional. Porém, conforme o volume de dados aumentou, o uso das VMs hierárquicas exibiu resultados cada vez melhores,

quando comparados ao processamento de consultas sem VM. Esse comportamento se manteve mesmo nos casos em que a visão foi parcialmente armazenada em disco, por limite de memória primária. Nota-se nesses casos que não foi definida qualquer estratégia de *buffer-pool* para melhorar a troca de páginas de disco entre a memória primária e a memória secundária, o que poderia melhorar ainda mais o desempenho das consultas. Nesses gráficos, o ganho de desempenho entre o uso da VM hierárquica em relação ao processamento de consultas sem usar VMs variou entre 33% e 58%.

Nessa seção foi descrita uma otimização para o processamento distribuído de consultas OLAP, usando o *framework* Spark. A construção de uma VM hierárquica, armazenando os atributos utilizados na consulta, juntamente com os outros presentes na respectiva hierarquia de atributos, possui um maior volume de dados e conseqüentemente, pior desempenho para processar apenas uma consulta individualmente. Entretanto, ao realizar operações OLAP sobre essas consultas, sendo elas aplicadas sobre um mesmo conjunto de dados, em um diferente nível de agregação, economiza-se no reproprocessamento completo da base de dados.

Do ponto de vista da técnica de *benchmarking*, esta última análise é importante pois investiga uma possível otimização sobre o processamento distribuído dos dados. Considerando uma aplicação que executa consultas OLAP iterativas, agregando dados em diferentes níveis de agregação, validar essas características pode auxiliar durante a avaliação de um ambiente de *data warehousing*. Isso pode tornar o *benchmark* mais adequando à aplicação que deseja-se validar, pois aborda um cenário não explorado nos *benchmarks* detalhados no Capítulo 3.

4.6 Considerações Finais

O objetivo desta dissertação consiste em realizar um conjunto de propostas e análises majoritariamente baseadas em testes de desempenho experimentais que ofereçam fundamentos para a posterior proposta de um *benchmark* para DWs armazenados em bancos de dados NoSQL orientados a colunas. Dentro deste objetivo, nesse capítulo foram descritos os resultados obtidos considerando cada uma das etapas de um *benchmark*, a saber: (i) esquema e carga de trabalho; (ii) geração de dados; (iii) parâmetros e métricas; e (iv) validação. Motivados pelos questionamentos deste projeto, descritos na Seção 4.1, as investigações e experimentos realizados durante este trabalho consistiram em:

- Proposta de um esquema físico para DW, denominado *FactDate*, considerando o armazenamento em um banco de dados NoSQL orientado a colunas. Também foi realizado um estudo comparativo dele contra os outros esquemas físicos existentes na literatura, propondo cenários em que cada esquema físico provê melhor desempenho no processamento das consultas OLAP. Durante esse estudo, foram propostas duas consultas adicionais à carga de trabalho do SSB, envolvendo duas dimensões do esquema lógico definido pelo *benchmark*;

- Estudo sobre o balanceamento dos dados gerados pelo SSB entre os nós do *cluster*, mostrando que os dados necessários às consultas do SSB, juntamente às duas consultas propostas, estavam igualmente distribuídos entre os *RegionServers* do HBase, considerando o esquema físico *FactDate*;
- Identificação de três parâmetros importantes do processamento de dados distribuído usando o *framework* MapReduce, e a análise da influência desses parâmetros na principal métrica quantitativa de um *benchmark* denominada desempenho. Essa métrica é calculada pelo tempo de processamento das consultas; e
- Testes de desempenho referentes à etapa de validação, envolvendo a variação física de recursos computacionais (máquinas alocadas no *cluster*). Além disso, foi proposta uma otimização para consultas OLAP iterativas por meio de visões materializadas hierárquicas.

Retomando as perguntas descritas no começo desse capítulo, segue a síntese das respostas adquiridas ao longo do estudo realizado:

- **Pergunta:** Quais os impactos que diferentes esquemas físicos têm sobre bancos de dados NoSQL distribuídos e orientados a colunas?
Resposta: Na Seção 4.2.1 detalhou-se a comparação entre três esquemas físicos distintos, mostrando diferenças de desempenho das consultas OLAP processadas sobre os dados contidos neles. Cada esquema físico distinto pode beneficiar o desempenho no processamento de certos tipos de consultas OLAP, em detrimento a outros tipos de consultas realizadas sobre esse mesmo esquema.
- **Pergunta:** A carga de trabalho, definida majoritariamente por consultas OLAP, é completa e suficiente para validar um DW?
Resposta: Não, pois durante os experimentos referentes à primeira pergunta, houve a necessidade de propor consultas bidimensionais não existentes na carga original do SSB, as quais foram descritas na Seção 4.2.2.
- **Pergunta:** Os dados gerados pelo SSB estão igualmente divididos entre as máquinas de um *cluster*?
Resposta: A análise descrita na Seção 4.3 mostrou que processar as consultas OLAP definidas pelo SSB apresentou um processamento uniformemente distribuído entre os nós de um *cluster*. Isso porque a quantidade de registros retornados de cada nó do *cluster* eram similares, ao se utilizar o esquema proposto *FactDate*, no HBase.
- **Pergunta:** Quais parâmetros podem influenciar no processamento distribuído de consultas OLAP?
Resposta: Os três parâmetros, relacionados ao processamento distribuído usando o

framework Hadoop MapReduce, investigados e explicados na Seção 4.4, são: a quantidade de processos Map simultâneos, a quantidade total de processos Reduce e o tamanho de bloco de dados distribuídos.

- **Pergunta:** Como diferentes formas de processamento e validação de um *benchmark* podem influenciar no desempenho de operações OLAP?

Resposta: Durante os experimentos referentes à etapa de validação, verificou-se o impacto de uma maneira alternativa de processar os dados a fim de otimizar o processamento de operações OLAP, a qual consiste em utilizar visões materializadas no *framework* de processamento Spark. Adicionalmente, pensando na etapa de validação, na Seção 4.5 foi analisado o desempenho de processamento das consultas do SSB variando tanto o volume de dados quanto o tamanho do *cluster*. Com isso, verificou-se comparativamente os resultados da etapa de validação envolvendo diferentes quantidades de dados e de máquinas, analisando a tendência do ganho de desempenho das consultas ao se dobrar o tamanho do cluster, conforme aumenta-se o volume de dados processado.

Levando em consideração esses resultados, no Capítulo 5 são descritas as contribuições, conclusões gerais e trabalhos futuros dessa dissertação de mestrado.

CONCLUSÕES

Neste capítulo são descritas as conclusões finais desta dissertação. Estão sumarizadas a conclusão geral do projeto, juntamente à listagem das principais contribuições e dos trabalhos futuros.

5.1 Conclusão Geral

Esta dissertação tem como objetivo realizar um conjunto de propostas e análises majoritariamente baseadas em testes de desempenho experimentais que ofereçam fundamentos para a posterior proposta de um *benchmark* para DWs armazenados em bancos de dados NoSQL. Para investigar esse objetivo foi usado como base o SSB, o qual foi desenvolvido para bancos de dados relacionais centralizados, e foi verificada a sua aplicabilidade a DWs armazenados no banco de dados NoSQL distribuído e orientado a colunas HBase. As investigações e os testes de desempenho experimentais realizados foram motivados pelo seguinte questionamento: ***"O que é necessário adicionar ao SSB, ou a outro benchmark de DW, para validar efetivamente ambientes de data warehousing baseados em bancos de dados NoSQL distribuídos e orientados a colunas?"***. Foram analisadas e exploradas limitações do SSB em cada uma das quatro etapas de um *benchmark* para DW: (i) esquema e carga de trabalho, (ii) geração de dados, (iii) parâmetros e métricas, e (iv) validação.

Apesar das investigações terem sido focadas no SSB, a maioria das análises realizadas pode ser aplicada a outros *benchmarks* para DW, como o TPC-H ou TPC-DS. Por exemplo, a proposta de esquemas físicos distintos para DW, originados pelos diferentes agrupamentos de atributos entre famílias de colunas, pode ser aplicada de forma análoga a outros *benchmarks*. Outro exemplo refere-se à análise dos parâmetros de processamento distribuído dos dados via *framework* Hadoop MapReduce, relacionando-os à métrica de desempenho reportada por praticamente todos os *benchmarks* estudados. Por fim, a proposta de VMs hierárquicas também pode ser aplicada a outros *benchmarks*, sendo necessário analisar seu esquema, juntamente com as respectivas hierarquias de atributos presentes, a fim de estruturar VMs que possibilitem o processamento das consultas OLAP iterativas.

Note também que, apesar dos experimentos terem sido realizados usando o HBase, este banco de dados NoSQL segue uma organização lógica dos dados em famílias de colunas similar a outros bancos de dados NoSQL orientados a colunas descritos nesta dissertação, como o Accumulo e o Cassandra. Este fato indica a possibilidade de portabilidade desses experimentos para os outros bancos de dados NoSQL supracitados.

5.2 Contribuições

As principais contribuições decorrentes do desenvolvimento desta pesquisa de mestrado são:

- Proposta de um esquema físico de DW, denominado *FactDate*, focado em melhorar o desempenho de consultas unidimensionais e bidimensionais que envolvam a dimensão *Date*. Essa dimensão é considerada necessária à maioria das consultas OLAP devido à característica temporal dos dados de um ambiente de *data warehousing*;
- Identificação de diferentes cenários, agrupados pela quantidade de dimensões acessadas durante o processamento das consultas, nos quais cada esquema físico de DW avaliado exibiu os melhores resultados de desempenho;
- Proposta de consultas bidimensionais à carga de trabalho do SSB, necessárias à análise realizada sobre os três esquemas físicos de DW avaliados;
- Estudo sobre a distribuição dos dados gerados pelo SSB entre os nós do *cluster*, mostrando que os dados necessários às consultas avaliadas são igualmente distribuídos entre os *RegionServers* do HBase;
- Identificação da influência de três principais parâmetros do armazenamento e processamento distribuído dos dados, usando o *framework* Hadoop MapReduce, sobre o desempenho de consultas OLAP. Cada um desses três parâmetros pode tanto melhorar quanto degenerar o desempenho das consultas realizadas;
- Identificação do comportamento de ganho de desempenho, no processamento de consultas, conforme são adicionados novos nós em um *cluster*. Esse ganho pode passar de 70% em algumas consultas considerando-se pequenos volumes de dados. Porém, conforme aumenta-se esse volume, o ganho de todas as consultas converge para a faixa de 50% a 60%; e
- Proposta do uso de visões materializadas, usando o *framework* Spark, para otimização do processamento de consultas OLAP iterativas.

5.3 Divulgação de Resultados

Os resultados relacionados à etapa (i) de um *benchmark* para DW, os quais especificam tanto o esquema *FactDate* quanto as consultas bidimensionais propostas, foram submetidos ao evento internacional *International Conference on Enterprise Information Systems* (ICEIS 2016), o qual possui classificação CAPES Qualis Ciência da Computação B1. Esse artigo, intitulado “*Physical DataWarehouse Design on NoSQL Databases - OLAP query processing over HBase*”, foi aceito como *short paper* de 8 páginas para ser apresentado na conferência. Os experimentos e análises efetuadas nesse artigo, juntamente à proposta do esquema *FactDate*, estão detalhados na Seção 4.2.

5.4 Trabalhos Futuros

Em relação à etapa (i) do SSB, além do esquema proposto, denominado *FactDate*, podem ser avaliados outros esquemas físicos intermediários. Por exemplo, supondo-se um cenário em que sejam comuns análises sobre a quantidade vendida de produtos por data e por cliente, pode ser interessante avaliar um esquema denominado *FactDateCustomer*, que agrupe os atributos tanto da tabela de fatos quanto das dimensões *Date* e *Customer*. Outro estudo que pode ser realizado futuramente, sobre a etapa de esquema, consiste em separar, por famílias de colunas, os atributos mais usados dos menos usados de uma tabela. Essa análise é focada em diminuir o tamanho dos *HFiles*, considerando-se os atributos mais frequentemente utilizados nas consultas OLAP.

Outras análises futuras consistem na combinação dos três parâmetros avaliados na etapa (iii), verificando-se, por exemplo, o desempenho das consultas conforme variam-se tanto a quantidade de processos Map executando em paralelo quanto o tamanho dos blocos de dados no HDFS. Ademais, outro trabalho futuro baseia-se em avaliar a influência desses parâmetros enquanto variam-se os esquemas físicos utilizados, como o *FactDate*, *CNSSB* e *SameCF*. Por fim, outro trabalho futuro, relacionado à etapa (iv), consiste na extensão da proposta do uso de visões materializadas, aplicando-se a mesma ideia a outros *benchmarks*, além de analisar maiores volumes de dados.

Existe também o trabalho futuro citado na Seção 5.1, relativo à portabilidade dos experimentos e propostas contidos nesta dissertação a outros bancos de dados NoSQL orientados a coluna, não só para validar a aplicabilidade deles, mas também possibilitar novas propostas e análises. Pensando em outros paradigmas de armazenamento, um trabalho futuro consiste em verificar quais desses experimentos e propostas poderiam ser aplicados aos paradigmas de armazenamento orientados a documentos e a grafos. Por exemplo, conforme discutido na Seção 3.4.2, existem modelagens distintas no MongoDB similares à abordagem explorada nessa dissertação, relacionada às famílias de colunas, porém abordando a distribuição de atributos em um mesmo documento, ou em documentos distintos (subdocumentos). Levando em considera-

ção o experimento adicional descrito no Apêndice A, que testou uma modelagem de DW no MongoDB, este trabalho futuro consiste em explorar outras modelagens, analisando o impacto de cada uma delas no desempenho de processamento de consultas OLAP.

Por fim, o último trabalho futuro é salientado ao longo de toda a dissertação, que é a proposta de um *benchmark* para DWs armazenados em bancos de dados NoSQL orientados a colunas, condensando os resultados e análises realizadas nesta dissertação, e dos outros trabalhos futuros discutidos nesta seção. Adicionalmente, como já descrito nesta seção, esse trabalho futuro também abrange estender este *benchmark* para os outros paradigmas de armazenamento, como o paradigma de armazenamento orientado a documentos.

REFERÊNCIAS

ABADI, D. Consistency tradeoffs in modern distributed database system design: Cap is only part of the story. **Computer**, IEEE Computer Society Press, v. 45, n. 2, p. 37–42, 2012. Citado na página 34.

ABADI, D. J.; MADDEN, S. R.; HACHEM, N. Column-stores vs. row-stores: How different are they really? In: **ACM SIGMOD**. NY, USA: [s.n.], 2008. p. 967–980. ISBN 978-1-60558-102-6. Citado 3 vezes nas páginas 22, 35 e 109.

ARIYACHANDRA, T.; WATSON, H. J. Key organizational factors in data warehouse architecture selection. **Decision Support Systems**, v. 49, n. 2, p. 200–212, 2010. Citado na página 21.

ARMBRUST, M.; XIN, R. S.; LIAN, C.; HUI, Y.; LIU, D.; BRADLEY, J. K.; MENG, X.; KRAFT, T.; FRANKLIN, M. J.; GHODSI, A.; ZAHARIA, M. Spark SQL: Relational data processing in spark. In: **Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data**. [S.l.: s.n.], 2015. p. 1383–1394. Citado na página 43.

BARATA, M.; BERNARDINO, J.; FURTADO, P. Survey on big data and decision support benchmarks. In: **Database and Expert Systems Applications**. [S.l.: s.n.], 2014. p. 174–182. Citado 4 vezes nas páginas 47, 51, 58 e 59.

BOG, A. **Benchmarking Transaction and Analytical Processing Systems: The Creation of a Mixed Workload Benchmark and Its Application**. 1. ed. [S.l.]: Springer Publishing Company, Incorporated, 2013. Citado na página 45.

BREWER, E. A. Towards robust distributed systems (abstract). In: **Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing - PODC**. [S.l.]: ACM, 2000. p. 101–130. Citado na página 34.

CAI, L.; HUANG, S.; CHEN, L.; ZHENG, Y. Performance analysis and testing of hbase based on its architecture. In: **IEEE/ACIS 12th International Conference on Computer and Information Science**. [S.l.: s.n.], 2013. p. 353–358. Citado na página 56.

CHAUDHURI, S.; DAYAL, U. An overview of data warehousing and OLAP technology. **SIGMOD Record**, v. 26, n. 1, p. 65–74, 1997. Citado 3 vezes nas páginas 21, 29 e 30.

CHAUDHURI, S.; DAYAL, U.; NARASAYYA, V. R. An overview of business intelligence technology. **Communications of the ACM**, v. 54, n. 8, p. 88–98, 2011. Citado na página 21.

CHEN, M.; MAO, S.; LIU, Y. Big data: A survey. **Mobile Networks and Applications**, v. 19, n. 2, p. 171–209, 2014. Citado na página 34.

CHEVALIER, M.; MALKI, M. E.; KOPLIKU, A.; TESTE, O.; TOURNIER, R. Benchmark for OLAP on NoSQL Technologies. In: **IEEE 9th International Conference on Research Challenges in Information Science**. [S.l.: s.n.], 2015. Citado na página 57.

_____. Implementation of multidimensional databases in column-oriented NoSQL systems (regular paper). In: **East-European Conference on Advances in Databases and Information Systems**. [S.l.]: Springer, 2015. p. 79–91. Citado 2 vezes nas páginas 56 e 57.

_____. Implementation of multidimensional databases with document-oriented nosql. In: MARDIA, S.; HARA, T. (Ed.). **Big Data Analytics and Knowledge Discovery**. [S.l.]: Springer International Publishing, 2015, (Lecture Notes in Computer Science, v. 9263). p. 379–390. Citado 2 vezes nas páginas 55 e 56.

_____. Implementing Multidimensional Data Warehouses into NoSQL. In: **17th International Conference on Enterprise Information Systems**. [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 55 e 57.

CIFERRI, C.; CIFERRI, R.; GÓMEZ, L.; SCHNEIDER, M.; VAISMAN, A.; ZIMÁNYI, E. Cube algebra: A generic user-centric model and query language for olap cubes. **International Journal of Data Warehousing and Mining**, v. 9, n. 2, p. 39–65, 2013. ISSN 1548-3924. Citado na página 33.

CIFERRI, C. D. A. **Distribuição dos dados em ambientes de data warehousing: o sistema WebD²W e algoritmos voltados à fragmentação horizontal dos dados**. Tese (Doutorado) — CIn, UFPE, 2002. Citado 3 vezes nas páginas 27, 28 e 32.

CIFERRI, R. R. **Um benchmark voltado a análise de desempenho de sistemas de informações geograficas**. Dissertação (Mestrado) — UNICAMP, Campinas, SP. In Portuguese, 1995. Citado 2 vezes nas páginas 22 e 46.

COOPER, B. F.; SILBERSTEIN, A.; TAM, E.; RAMAKRISHNAN, R.; SEARS, R. Benchmarking cloud serving systems with ycsb. In: **Proceedings of the 1st ACM symposium on Cloud computing**. [S.l.: s.n.], 2010. p. 143–154. Citado na página 58.

CORDOVA, A.; RINALDI, B.; WALL, M. **Accumulo: Application Development, Table Design, and Best Practices**. 1. ed. [S.l.]: O'Reilly Media, Inc., 2015. Citado na página 36.

DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. In: **Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation**. [S.l.: s.n.], 2004. v. 6, p. 137–150. Citado 2 vezes nas páginas 24 e 39.

DEDE, E.; GOVINDARAJU, M.; GUNTER, D.; CANON, R. S.; RAMAKRISHNAN, L. Performance evaluation of a mongodb and hadoop platform for scientific data analysis. In: **Proceedings of the 4th ACM Workshop on Scientific Cloud Computing**. [S.l.: s.n.], 2013. Citado na página 109.

DEHDOUH, K.; BENTAYEB, F.; BOUSSAID, O.; KABACHI, N. Using the column oriented NoSQL model for implementing big data warehouses. **The 21th International Conference on Parallel and Distributed Processing Techniques and Applications**, p. 469–475, 2015. Citado na página 56.

DEHDOUH, K.; BOUSSAID, O.; BENTAYEB, F. Columnar nosql star schema benchmark. In: **Model and Data Engineering**. [S.l.: s.n.], 2014, (Lecture Notes in Computer Science, v. 8748). p. 281–288. Citado 2 vezes nas páginas 36 e 54.

DESANI, D.; COSTA, V. G.; MARCONDES, C. A. C.; SENGHER, H. Black-box optimization of hadoop parameters using derivative-free optimization. In: **24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing**. [S.l.: s.n.], 2016. p. 43–50. Citado na página 43.

DOULKERIDIS, C.; NØRVÅG, K. A survey of large-scale analytical query processing in mapreduce. **The VLDB Journal**, v. 23, n. 3, p. 355–380, 2014. Citado 2 vezes nas páginas 38 e 39.

ENGFER, D. **An Introduction to the Hadoop Stack**. 2011 (acessado: 20/04/2016). Disponível em: <<http://pt.slideshare.net/davidengfer/intro-to-the-hadoop-stack-javamug>>. Citado na página 39.

FOLKERTS, E.; ALEXANDROV, A.; SACHS, K.; IOSUP, A.; MARKL, V.; TOSUN, C. Benchmarking in the cloud: What it should, can, and cannot be. In: **Proceedings of the 4th TPC Technology Conference on Performance Evaluation & Benchmarking**. [S.l.: s.n.], 2012. p. 173–188. Citado na página 45.

FORLANI, D. T. **Fragmentação Vertical de Dados em Data Warehouses no Sistema WebD²W**. Dissertação (Mestrado) — Universidade Estadual de Maringá, 2006. Citado na página 47.

GEORGE, L. **HBase: the definitive guide**. 1. ed. [S.l.]: O'Reilly Media, Inc., 2011. Citado 3 vezes nas páginas 35, 36 e 37.

GHAZAL, A.; RABL, T.; HU, M.; RAAB, F.; POESS, M.; CROLOTTE, A.; JACOBSEN, H.-A. Bigbench: Towards an industry standard benchmark for big data analytics. In: **Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data**. [S.l.: s.n.], 2013. p. 1197–1208. Citado na página 59.

GILBERT, S.; LYNCH, N. A. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. **SIGACT News**, v. 33, n. 2, p. 51–59, 2002. Citado 2 vezes nas páginas 33 e 34.

GOLFARELLI, M.; RIZZI, S. A survey on temporal data warehousing. **International Journal of Data Warehousing and Mining**, v. 5, n. 1, p. 1–17, 2009. Citado na página 21.

GOLFARELLI, M.; RIZZI, S.; CELLA, I. Beyond data warehousing: what's next in business intelligence? In: **Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP**. [S.l.: s.n.], 2004. p. 1–6. Citado na página 21.

GONZALES, M. L.; BAGCHI, K. K.; UDO, G. J.; KIRS, P. Diffusion of business intelligence and data warehousing: An exploratory investigation of research and practice. In: **Proceedings of the 44th Hawaii International International Conference on Systems Science**. [S.l.: s.n.], 2011. p. 1–9. Citado na página 21.

INMON, W. H. **Building the Data Warehouse**. 3. ed. New York, NY, USA: John Wiley & Sons, Inc., 2002. Citado 2 vezes nas páginas 28 e 29.

KAMBATLA, K.; PATHAK, A.; PUCHA, H. Towards optimizing hadoop provisioning in the cloud. In: **Proceedings of the 2009 Conference on Hot Topics in Cloud Computing**. [S.l.: s.n.], 2009. Citado na página 43.

KIM, K.; JEON, K.; HAN, H.; KIM, S.-g.; JUNG, H.; YEOM, H. Y. Mrbench: A benchmark for mapreduce framework. In: **Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems**. [S.l.: s.n.], 2008. p. 11–18. Citado na página 41.

KIMBALL, R.; ROSS, M. **The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling**. 2. ed. [S.l.]: Wiley Computer Publishing, 2002. ISBN 0471200247. Citado 5 vezes nas páginas 23, 29, 31, 32 e 52.

KONISHETTY, V. K.; KUMAR, K. A.; VORUGANTI, K.; RAO, G. V. P. Implementation and evaluation of scalable data structure over hbase. In: **Proceedings of the International Conference on Advances in Computing, Communications and Informatics**. [S.l.: s.n.], 2012. p. 1010–1018. Citado 2 vezes nas páginas 24 e 36.

LAKSHMAN, A.; MALIK, P. Cassandra: A decentralized structured storage system. **SIGOPS Operating Systems Review**, ACM, v. 44, n. 2, p. 35–40, 2010. Citado na página 36.

LEE, K.-H.; LEE, Y.-J.; CHOI, H.; CHUNG, Y. D.; MOON, B. Parallel data processing with mapreduce: a survey. **ACM SIGMOD Record**, v. 40, n. 4, p. 11–20, 2012. Citado 2 vezes nas páginas 39 e 40.

LIMA, J. d. C.; HIRATA, C. M. Multidimensional cyclic graph approach: Representing a data cube without common sub-graphs. **Information Sciences**, v. 181, n. 13, p. 2626–2655, 2011. Citado na página 30.

MOHAMMAD, S.; BRESS, S.; SCHALLEHN, E. Cloud data management: A short overview and comparison of current approaches. In: SCHMITT, I.; SARETZ, S.; ZIERENBERG, M. (Ed.). **Grundlagen von Datenbanken**. [S.l.: s.n.], 2012. (CEUR Workshop Proceedings, v. 850), p. 41–46. Citado na página 38.

MONIRUZZAMAN, A.; HOSSAIN, S. A. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. **The Computing Research Repository - CoRR**, v. 23, n. 3, 2013. Citado 3 vezes nas páginas 22, 34 e 35.

MOUSSA, R. Tpc-h benchmark analytics scenarios and performances on hadoop data clouds. In: **Proceedings of the 4th International Conference on Networked Digital Technologies**. [S.l.: s.n.], 2012. v. 293, p. 220–234. Citado na página 49.

_____. Benchmarking data warehouse systems in the cloud. In: **International Conference on Computer Systems and Applications**. [S.l.: s.n.], 2013. p. 1–8. Citado 2 vezes nas páginas 49 e 50.

NAMBIAR, R.; POESS, M.; DEY, A.; CAO, P.; MAGDON-ISMAIL, T.; REN, D. Q.; BOND, A. Introducing tpcx-hs: The first industry standard for benchmarking big data systems. In: NAMBIAR, R.; POESS, M. (Ed.). **Performance Characterization and Benchmarking. Traditional to Big Data**. [S.l.]: Springer International Publishing, 2015, (Lecture Notes in Computer Science, v. 8904). p. 1–12. ISBN 978-3-319-15349-0. Citado na página 60.

NAMBIAR, R. O.; POESS, M. The making of TPC-DS. In: **Proceedings of the 32th International Conference on Very Large Data Bases - VLDB**. [S.l.: s.n.], 2006. p. 1049–1058. Citado na página 51.

NASCIMENTO, S. M. **Spatial Star Schema Benchmark – Um Benchmark para Data Warehouse Geográfico**. Dissertação (Mestrado) — Universidade Federal de Pernambuco, Recife - UFPE, 2013. Citado na página 52.

O'NEIL, P.; O'NEIL, E.; CHEN, X.; REVILAK, S. The star schema benchmark and augmented fact table indexing. In: **Performance Evaluation and Benchmarking**. [S.l.: s.n.], 2009. p. 237–252. Citado 3 vezes nas páginas 23, 48 e 52.

POESS, M.; FLOYD, C. New TPC benchmarks for decision support and web commerce. **SIGMOD Record**, v. 29, n. 4, p. 64–71, 2000. Citado na página 47.

POESS, M.; SMITH, B.; KOLLAR, L.; LARSON, P. TPC-DS, taking decision support benchmarking to the next level. In: **Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data**. [S.l.: s.n.], 2002. p. 582–587. Citado na página 50.

PRITCHETT, D. BASE: An acid alternative. **ACM Queue**, v. 6, n. 3, p. 48–55, 2008. Citado na página 34.

RAJARAMAN, A.; ULLMAN, J. D. **Mining of Massive Datasets**. [S.l.: s.n.], 2011. Citado na página 40.

REDMOND, E.; WILSON, J. R. **Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement**. [S.l.]: Pragmatic Bookshelf, 2012. ISBN 1934356921, 9781934356920. Citado 2 vezes nas páginas 35 e 36.

SAKR, S.; LIU, A.; BATISTA, D. M.; ALOMARI, M. A survey of large scale data management approaches in cloud environments. **IEEE Communications Surveys & Tutorials**, v. 13, n. 3, p. 311–336, 2011. Citado na página 39.

SCHNEIDER, M.; VOSSSEN, G.; ZIMÁNYI, E. Data warehousing: from occasional olap to real-time business intelligence (dagstuhl seminar 11361). **Dagstuhl Reports**, v. 1, n. 9, p. 1–25, 2011. Citado na página 21.

SHVACHKO, K.; KUANG, H.; RADIA, S.; CHANSLER, R. The hadoop distributed file system. In: **Proceedings of the 26th Symposium on Mass Storage Systems and Technologies**. [S.l.: s.n.], 2010. p. 1–10. Citado na página 38.

SIVASUBRAMANIAN, S. Amazon dynamodb: a seamlessly scalable non-relational database service. In: CANDAN, K. S.; 0001, Y. C.; SNODGRASS, R. T.; GRAVANO, L.; FUXMAN, A. (Ed.). **SIGMOD Conference**. [S.l.: s.n.], 2012. p. 729–730. Citado na página 35.

SOUSA, F. R. C.; MOREIRA, L. O.; MACÊDO, J. A. F.; MACHADO, J. C. Gerenciamento de dados em nuvem: Conceitos, sistemas e desafios. In: **Proceedings of the 25th Simpósio Brasileiro de Banco de Dados**. [S.l.: s.n.], 2010. p. 101–130. Citado na página 34.

THUSOO, A.; SARMA, J. S.; JAIN, N.; SHAO, Z.; CHAKKA, P.; ANTHONY, S.; LIU, H.; WYCKOFF, P.; MURTHY, R. Hive: A warehousing solution over a map-reduce framework. **Proceedings of the VLDB Endowment**, v. 2, n. 2, p. 1626–1629, 2009. Citado na página 41.

VIEIRA, M. R.; FIGUEIREDO, J. M. d.; LIBERATTI, G.; VIEBRANTZ, A. F. M. Bancos de dados nosql: conceitos, ferramentas, linguagens e estudos de casos no contexto de big data. **Simpósio Brasileiro de Bancos de Dados**, 2012. Citado na página 34.

WHITE, T. **Hadoop: The definitive guide**. 3. ed. [S.l.]: O'Reilly Media, Inc., 2012. Citado na página 37.

XIN, R. S.; ROSEN, J.; ZAHARIA, M.; FRANKLIN, M. J.; SHENKER, S.; STOICA, I. Shark: Sql and rich analytics at scale. In: **Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data**. [S.l.: s.n.], 2013. p. 13–24. Citado na página [43](#).

ZAHARIA, M.; CHOWDHURY, M.; DAS, T.; DAVE, A.; MA, J.; MCCAULEY, M.; FRANKLIN, M. J.; SHENKER, S.; STOICA, I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: **Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation**. [S.l.: s.n.], 2012. p. 2–2. Citado na página [42](#).

ZAHARIA, M.; CHOWDHURY, M.; FRANKLIN, M. J.; SHENKER, S.; STOICA, I. Spark: cluster computing with working sets. In: **Proceedings of the 2nd USENIX conference on Hot topics in cloud computing**. [S.l.: s.n.], 2010. p. 10–10. Citado na página [42](#).

PROCESSAMENTO DE CONSULTAS EM BANCOS DE DADOS NOSQL ORIENTADO A DOCUMENTOS

Tendo em vista o trabalho de [ABADI; MADDEN; HACHEM \(2008\)](#), descrito na Seção [2.3.2](#), o paradigma de armazenamento de dados orientado a coluna pode promover elevados ganhos de desempenho durante o processamento analítico dos dados. Também motivado por esta informação, os experimentos realizados nesta pesquisa de mestrado focaram nesse paradigma de armazenamento. Contudo, questiona-se a aplicabilidade de um DW em outros paradigmas, como o paradigma de armazenamento orientado a documentos. Assim como foi discutido na Seção [3.4.2](#), relacionado à adaptação do SSB ao banco de dados NoSQL orientado a documentos MongoDB, surge a dúvida sobre como se comporta o desempenho de processamento de consultas OLAP neste outro paradigma de armazenamento, além de comparar esse desempenho com os experimentos realizados utilizando o HBase.

Neste apêndice são detalhados experimentos preliminares visando avaliar uma possível adaptação do SSB ao banco de dados NoSQL denominado MongoDB. O MongoDB é um banco de código aberto, desenvolvido sob a *10gen*, que armazena os dados utilizando o paradigma de armazenamento orientado a documentos ([DEDE et al., 2013](#)). Este banco não organiza as tabelas em linhas ou colunas, mas sim usando documentos, cada qual utilizando vetores associativos de valores, listas ou encapsulando outros vetores associativos. Os documentos do MongoDB são serializados como objetos *JavaScript Object Notation* (JSON) e armazenados como objetos binários denominados *Binary JSON* (BSON).

A abordagem adotada nesse experimento, introduzida na Seção [3.4.2](#), consiste em armazenar cada dimensão do esquema em um respectivo subdocumento. Para exemplificar essa abordagem, a Figura [A.1](#) mostra um registro do SSB convertido como documento JSON para ser armazenado no MongoDB. Esse documento é identificado pelo campo “*_id*”, que refere-se à

```

{
  "_id": "1-1",
  "CUSTOMER": {
    "_id": 36901,
    "C_NAME": "Customer#000036901",
    "C_ADDRESS": "8pPZto10LrT1",
    "C_CITY": "JORDAN 1",
    "C_NATION": "JORDAN",
    "C_REGION": "MIDDLE EAST",
    "C_PHONE": "23-644-998-4944",
    "C_MKTSEGMENT": "AUTOMOBILE"
  },
  "SUPPLIER": {
    "_id": 4137,
    "S_NAME": "Supplier#000004137",
    "S_ADDRESS": "yAsSoKR4P8J22Cz57",
    "S_CITY": "MOROCCO 5",
    "S_NATION": "MOROCCO",
    "S_REGION": "AFRICA",
    "S_PHONE": "25-533-761-8587"
  },
  "PART": {
    "_id": 310379,
    "P_NAME": "coral chiffon",
    "P_MFGR": "MFGR#3",
    "P_CATEGORY": "MFGR#31",
    "P_BRAND1": "MFGR#3122",
    "P_COLOR": "beige",
    "P_TYPE": "MEDIUM PLATED TIN",
    "P_SIZE": 14,
    "P_CONTAINER": "SM JAR"
  },
  "DATE": {
    "_id": 19960102,
    "D_DATE": "January 2, 1996",
    "D_DAYOFWEEK": "Wednesday",
    "D_MONTH": "January",
    "D_YEAR": 1996,
    "D_YEARMONTHNUM": 199601,
    "D_YEARMONTH": "Jan1996",
    "D_DAYNUMINWEEK": 4,
    "D_DAYNUMINMONTH": 2,
    "D_DAYNUMINYEAR": 2,
    "D_MONTHNUMINYEAR": 1,
    "D_WEEKNUMINYEAR": 1,
    "D_SELLINGSEASON": "Winter",
    "D_LASTDAYINWEEKFL": 0,
    "D_LASTDAYINMONTHFL": 1,
    "D_HOLIDAYFL": 0,
    "D_WEEKDAYFL": 1
  },
  "D_COMMITDATE": "February 12, 1996",
  "LO_ORDPRIORITY": "5-LOW",
  "LO_SHIPPRIORITY": 0,
  "LO_QUANTITY": 17,
  "LO_EXTPRICE": 2361912,
  "LO_ORDTOTALPRICE": 18150369,
  "LO_DISCOUNT": 4,
  "LO_REVENUE": 2267435,
  "LO_SUPPLYCOST": 83361,
  "LO_TAX": 2,
  "LO_SHIPMODE": "TRUCK"
}

```

Figura A.1 – Exemplo de uma instância do SSB, mostrando como essa tupla foi traduzida para o formato JSON, para que possa ser armazenada no MongoDB.

chave primária da tupla. Os outros atributos deste documento são compostos tanto por atributos simples, formados por campos numéricos ou textuais, quanto atributos compostos, estruturados por meio de subdocumentos. Esses subdocumentos são responsáveis por armazenar os atributos relacionados à tupla atual, porém pertencentes às dimensões do esquema. Por exemplo, no documento presente na Figura A.1, a data relacionada a ele refere-se ao dia 2 de janeiro de 1996, sendo esse campo acessado por meio do caminho “DATE.D_DATE”.

O ambiente distribuído usado foi o mesmo que o explicado na Seção 4.5.1, sendo composto por cinco máquinas, as quais possuem um processador i5-3330, 16 GB de memória RAM, 1 TB de disco SATA (7200 RPM) e conexão de 1Gb/s. O sistema operacional de cada máquina é o CentOS (versão 7.0), com o banco de dados NoSQL MongoDB (versão 3.0). Já o volume de dados corresponde ao valor 20 de SF, conforme mostrado na Tabela 4.1. Esses dados, armazenados em um arquivo texto desnormalizado de tamanho 55 GB, ao serem convertidos para o formato JSON, passam a ter o tamanho de 134 GB.

```
SELECT sum(lo_extendedprice*lo_discount) as revenue
FROM lineorder, dates
WHERE lo_orderdate = d_datekey
      AND lo_discount between 1 and 3
      AND lo_quantity < 25
      AND d_year = 1993;
```

(a) Consulta SQL original.

```
db.SF20.mapReduce(
  function(){
    emit(null,
      this.L_DISCOUNT * this.L_EXTPRICE);
  },
  function(key, values){
    return Array.sum(values)
  },
  {
    query:{
      "L_DISCOUNT":{$gte:1,$lte:3},
      "L_QUANTITY":{$lt:25},
      "DATE.D_YEAR":1993
    },
    out:"Q11"
  }
)
```

(b) Consulta traduzida para MapReduce.

```
db.SF20.aggregate(
  [ { $match: {"L_DISCOUNT":{$gte:1,$lte:3},
              "L_QUANTITY":{$lt:25},
              "DATE.D_YEAR":1993} },
    { $group: { _id: "Q11",
                total:
                  { $sum:
                    {
                      $multiply: ["$L_DISCOUNT",
                                "$L_EXTPRICE"]
                    }
                  }
    }
  ]
)
```

(c) Consulta traduzida com o comando Aggregate.

Figura A.2 – Consulta Q1.1 da carga de trabalho do SSB, representada de duas formas para ser processada dentro do MongoDB.

Em relação ao processamento de consultas, as Figuras A.2 e A.3 exemplificam como as consultas SQL originais da carga de trabalho do SSB foram traduzidas para serem executadas dentro do MongoDB. A Figura A.2(a) mostra a consulta Q1.1 e duas possíveis abordagens para processá-la dentro do MongoDB, enquanto que a Figura A.2(b) ilustra uma adaptação usando a interface MapReduce, em que são declarados os processos Map (primeira função) e Reduce (segunda função). Após isso, são atribuídos os filtros aos dados sendo processados, respeitando-se os predicados existentes na consulta SQL original. Já a Figura A.2(c) ilustra outra

possível adaptação, similar à primeira, porém usando-se o comando *Aggregate* do MongoDB. Durante os experimentos, ambas as abordagens utilizaram aproximadamente o mesmo tempo de processamento.

```
SELECT d_year, s_nation, p_category, sum(lo_revenue - lo_supplycost) as profit
FROM dates, customer, supplier, part, lineorder
WHERE lo_custkey = c_custkey
      AND lo_suppkey = s_suppkey
      AND lo_partkey = p_partkey
      AND lo_orderdate = d_datekey
      AND c_region = 'AMERICA'
      AND s_region = 'AMERICA'
      AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
      AND (d_year = 1997 or d_year = 1998)
GROUP BY d_year, s_nation, p_category
ORDER BY d_year, s_nation, p_category;
```

(a) Consulta SQL original, contendo cláusula *group by*.

```
db.SF20.mapReduce(
  function(){
    var all = [];
    all.push(this.DATE.D_YEAR);
    all.push(this.SUPPLIER.S_NATION);
    all.push(this.PART.P_CATEGORY);
    emit(all.join(","), this.L_REVENUE - this.L_SUPPLYCOST);
  },
  function(key, values) { return Array.sum(values) },
  { query: {
    $and: [ {"CUSTOMER.C_REGION": "AMERICA"},
            {"SUPPLIER.S_REGION": "AMERICA"},
            {$or: [{"PART.P_MFGR": "MFGR#2"}, {"PART.P_MFGR": "MFGR#1"}]},
            {$or: [{"DATE.D_YEAR": 1997}, {"DATE.D_YEAR": 1998}]}
          }
    },
  out: "Q42"
}
```

(b) Consulta traduzida para o MongoDB usando-se a interface MapReduce.

Figura A.3 – Consulta Q4.2 da carga de trabalho do SSB, traduzida para comandos MapReduce.

Partindo do princípio de que a Figura A.2 detalha uma consulta SQL de soma total, a Figura A.3 mostra a consulta Q4.2 da carga de trabalho do SSB. Essa consulta consiste na agregação de soma dividida em grupos, ou seja, contendo a cláusula *group by*. A Figura A.3 também exibe a adaptação dessa consulta para o MongoDB por meio da interface MapReduce. A principal diferença entre as Figuras A.2 e A.3 reside na primeira função, a qual especifica o processo Map. Nele, o campo chave deixa de possuir um valor nulo e passa a ser composto pela concatenação dos atributos que compõem os grupos determinados pela consulta.

Os experimentos realizados mostraram que o tempo de processamento médio de todas as consultas foi de aproximadamente 445 segundos. Esse comportamento foi similar ao esquema SameCF, descrito na Seção 4.2.1. Contudo, nesse mesmo ambiente, armazenar os dados usando-se o esquema SameCF possibilitou o processamento dessas mesmas consultas com um ganho de desempenho de 10%, conforme mostrado na Figura A.4. Já ao considerar o uso de famílias de colunas, adotando-se o esquema FactDate do HBase, nota-se o potencial ganho de desem-

penho de aproximadamente 55% em comparação ao MongoDB, considerando-se as consultas unidimensionais Q1.1, Q1.2 e Q1.3. Porém, conforme também discutido na Seção 4.2.1, caso a consulta envolva a união de muitas famílias de colunas, a junção dessas famílias tem um grande impacto no desempenho do processamento das consultas.

Por meio desses experimentos, obtém-se duas conclusões. Se o esquema for melhor adaptado às consultas realizadas, como no caso do esquema FactDate para consultas unidimensionais e bidimensionais envolvendo a dimensão *Date*, é possível otimizar o processamento dessas consultas em até 55%. Isso ocorre porque, segundo o paradigma de orientação baseado em colunas, os atributos podem ser agrupados em diferentes famílias de colunas de modo a evitar a leitura e o processamento de atributos não acessados pela consulta. A segunda conclusão consiste no comportamento similar com relação ao tempo de processamento das consultas sobre os dados armazenados no MongoDB e no HBase, usando-se o esquema SameCF, para o volume de dados e ambiente utilizados. A diferença entre eles foi que as consultas realizadas sobre o esquema SameCF exibe um tempo aproximadamente 10% menor quando comparado às consultas realizadas sobre o MongoDB.

Figura A.4 – Tempo de processamento das consultas no MongoDB, em contraste com o processamento dessas mesmas consultas no HBase, usando os esquemas SameCF e FactDate.