UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MATEUS NUNES DE BARROS MAGALHÃES

UMA AVALIAÇÃO DA REDUNDÂNCIA E DO PARTICIONAMENTO DE DADOS CONVENCIONAIS E GEOESPACIAIS EM DATA WAREHOUSES ORIENTADOS A COLUNAS

Recife
2022

MATEUS NUNES DE BARROS MAGALHÃES

UMA AVALIAÇÃO DA REDUNDÂNCIA E DO PARTICIONAMENTO DE DADOS CONVENCIONAIS E GEOESPACIAIS EM DATA WAREHOUSES ORIENTADOS A COLUNAS

Trabalho apresentado ao Programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para a obtenção do título de mestre em Ciência da Computação. Área de concentração: Bancos de Dados.

Orientador: Prof. Dr. Robson do Nascimento Fidalgo

Recife

2022

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M189a Magalhães, Mateus Nunes de Barros
Uma avaliação da redundância e do particionamento de dados convencionais e geoespaciais em data warehouses orientados a colunas / Mateus Nunes de Barros Magalhães. – 2022.
134 f.: il., fig., tab.

Orientador: Robson do Nascimento Fidalgo.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências e apêndices.

1. Banco de dados. 2. Data-warehouse. I. Fidalgo, Robson do Nascimento (orientador). II. Título.

025.04 CDD (23. ed.) UFPE - CCEN 2022 – 89

Mateus Nunes de Barros Magalhães

“Uma Avaliação da Redundância e do Particionamento de Dados Convencionais e Geoespaciais em Data Warehouses Orientados a Colunas”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 24/02/2022.

BANCA EXAMINADORA

Prof. Dr. Luciano de Andrade Barbosa
Centro de Informática / UFPE

Profa. Dra. Andreza Leite de Alencar
Departamento de Computação / UFRPE

Prof. Dr. Robson do Nascimento Fidalgo
Centro de Informática / UFPE
(Orientador)

RESUMO

A crescente demanda por processamento analítico alinhada à expansão dos casos de uso envolvendo Big Data colocou em xeque a capacidade dos bancos de dados relacionais de atenderem essas aplicações sem comprometerem o desempenho das mesmas ou incorrerem em elevados custos financeiros. Assim, os bancos NoSQL se mostraram uma alternativa promissora em razão de relaxarem muitos dos controles de consistência, integridade e transacionais em troca de disponibilidade, capacidade de processamento paralelo e escalabilidade horizontal. Estudos mostraram que os bancos NoSQL orientados a colunas são boas opções para aplicações analíticas em razão do particionamento vertical onde dados são armazenados de forma contígua e com separação física. O planejamento de esquemas para tais bancos foi objeto de diversos trabalhos acadêmicos, os quais, avaliaram os impactos de abordagens ditas normalizadas e desnormalizadas, bem como, a distribuição dos dados entre famílias de colunas, com o intuito de identificar as boas práticas no projeto esquemático. No entanto, não avaliaram cenários envolvendo o armazenamento e a consulta de dados geoespaciais. Vislumbrando tal lacuna, nós investigamos o projeto esquemático bem como a redundância de dados convencionais e geoespaciais em GDWs construídas sob bancos NoSQL orientados a colunas para finalidades analíticas. Propusemos 40 esquemas criados a partir de abordagens para a implementação dos relacionamentos entre a tabela de fatos e as dimensões convencionais, dimensões convencionais e as geoespaciais e entre as próprias geoespaciais. Adoramos o benchmark geográfico denominado Spadawan para geração de bases em conformidade com os esquemas propostos, e para fatores de escala equivalentes a 1 e 10 vezes o tamanho padrão do benchmark. Os esquemas foram avaliados em um cluster computacional rodando o Hadoop, HBase e Spark de acordo com os tempos demandados para a execução de uma carga de consultas, tempos de ingestão dos dados e os volumes ocupados por cada uma deles. Os resultados experimentais mostraram que a desnormalização das dimensões convencionais foi a abordagem que mais contribuiu para a redução dos tempos de execução, apesar de incorrer em bases maiores. A normalização das dimensões geográficas de menor seletividade trouxe impactos positivos, enquanto as mais seletivas não influenciaram significativamente de forma positiva ou negativa.

Palavras-chave: NoSQL; data-warehouse geográfico; redundância; seletividade.

ABSTRACT

The growing demand for analytical query processing and the multiplication of big data environments challenged the capabilities of relational databases to serve these purposes without worsening their performances or increasing financial costs significantly. NoSQL databases emerged as candidates to solve these issues given that they do not enforce consistency, integrity and transactional constraints in order to offer better availability, parallel computing power and horizontal scalability. Researchers found that column-oriented NoSQL databases are a better choice than row-oriented databases for the implementation of data warehouses due to the fact that they can read and aggregate data faster since columns are physically separated from each other on disk. The development of schemas for this kind of databases were the subject of multiples researchs that studied normalized approaches and denormalized approaches as well as the most optimized ways to distribute data among column families. However, they didn't investigate and evaluate scenarios involving geospatial data and the execution queries that had geospatial predicates. Observing this blank space in the literature, our work investigates the influence of data partitioning and redundancy of conventional and geospatial data in GDW built on top of column-oriented NoSQL databases. We conceived 40 different schemas that present distinct approaches for the implementation of the relationships between conventional dimensions and geospatial dimensões along with approaches for designing relationships between two geospatial dimensions. We used the Spadawan benchmark to create synthetic data that could be modeled according to our proposed schemas using scale factors equivalent to 1 and 10 times the standard size. Then, we used geospatial queries to evaluate them on a cluster running Hadoop, HBase, and also Spark. We've collected the schema size, loading time and running time of all our queries. The results indicated that denormalizing conventional dimensions was the best approach to reduce queries execution time, despite increasing the overall space needed. The normalization of low-selectivity geospatial data also contributed to lower execution times and, regarding high-selectivity geospatial data, the normalization or denormalization did not show significant positive or negative impacts.

Keywords: NoSQL; geospatial data warehouse; redundancy; selectivity.

LISTA DE FIGURAS

Figura 1 - Componentes esquemáticos do HBase 31

Figura 2 - Componentes arquiteturais do HBase 34

Figura 3 - Arquitetura do Spark e sumário do DAG 37

Figura 4 - Arquitetura de data warehouses 41

Figura 5 - Esquema redundante e esquema híbrido 45

Figura 6 - Fluxograma do processo de revisão literária 47

Figura 7 - Tempos de execução do NLA, DLA e DLA-CF 51

Figura 8 - Tempos de execução do DLA, DLA-CF e FactDate 54

Figura 9 - Tempos de execução algoritmos evolucionários 57

Figura 10 - Tempos de execução algoritmo k-means 59

Figura 11 - Abordagens para relacionar dimensões geoespaciais 68

Figura 12 - Combinações {CG1, CG2, CG3} X {GG1, GG2, GG3} 71

Figura 13 - Notação UML para a representação visual dos esquemas 75

Figura 14 Dimensionalidade, composições sucessivas e associações 76

Figura 15 - Combinações FDD-F1, FDD-FN, FDN-F1, FDH-F1 e FDH-FN ... 77

Figura 16 - Produto {CG1-GG1,CG1-GG3}X{FDD,FDN,FDH}X{F1,FN} 79

Figura 17 - Produto {CG2}X{GG1,GG2,GG3}X{FDD,FDN,FDH}X{F1,FN} 80

Figura 18 - Produto {CG3-GG1,CG3-GG2}X{FDD,FDN,FDH}X{F1,FN} 81

Figura 19 - Etapas envolvidas na execução do experimento 84

Figura 20 - Volume vs. tempo de carga para FDD, FDN e FDH 98

Figura 21 - Consultas vs. tempo médio de execução para CG-GG 106

Figura 22 - Consultas vs. tempo médio de execução para FDD, FDN e FDH 106

Figura 23 - Mapa de calor dos p-values comparando médias 107

Figura 24 - Os cinco menores e maiores volumes para o fator 1 129

Figura 25 - Os cinco menores e maiores volumes para o fator 10 130

Figura 26 - Os cinco menores e maiores tempos de carga para o fator 1 ... 131

Figura 27 - Os cinco menores e maiores tempos de carga para o fator 10 . 132

Figura 28 - Os cinco menores e maiores tempos de consulta para o fator 1 133

Figura 29 - Os cinco menores e maiores tempos de consulta para o fator 10 134

LISTA DE TABELAS

Tabela 1 - Lista dos trabalhos relacionados mais relevantes 48

Tabela 2 - Tempo de carga e o volume do NLA, DLA e DLA-CF 53

Tabela 3 - Linhas, colunas e o volume das tabelas do Spadawan 82

Tabela 4 - Tempo de carga, volume e as normalizações dos esquemas 95

Tabela 5 - Resultados do experimento com fator de escala 1 101

Tabela 6 - Resultados do experimento com fator de escala 10 102

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicity, Consistency, Isolation and Durability
CAP	Consistency, Availability and Partition tolerance
DW	Data warehouse
DLA	Denormalized logical approach
DLA-CF	Denormalized logical approach with column families
GDW	Geospatial data warehouse
I/O	Input/Output
NLA	Normalized logical approach
OLAP	Online Analytical Processing
SPADAWAN	Spatial Data Warehouse Benchmark
SOLAP	Spatial Online Analytical Processing
SSB	Star Schema Benchmark
SQL	Structured Query Language

SUMÁRIO

1	**INTRODUÇÃO**	**13**
1.1	MOTIVAÇÃO	17
1.2	OBJETIVOS	20
1.3	METODOLOGIA	21
1.4	ORGANIZAÇÃO	22
2	**FUNDAMENTAÇÃO TEÓRICA**	**23**
2.1	ABORDAGENS RELACIONAIS	23
2.2	ABORDAGENS NOSQL	27
2.2.1	**Apache HBase**	**30**
2.2.2	**Apache Spark**	**35**
2.3	DATA WAREHOUSE	39
2.4	SPADAWAN	42
3	**REVISÃO DA LITERATURA**	**46**
3.1	MÉTODO DE BUSCA E CRITÉRIOS DE ANÁLISE	46
3.2	TRABALHOS RELACIONADOS	47
3.2.1	**Grupo I: Abordagens Primárias**	**49**
3.2.2	**Grupo II: Abordagens Inteligentes**	**55**
3.2.3	**Grupo III: Abordagens Geoespaciais**	**60**
3.3	ESTADO DA ARTE	62
4	**PLANEJAMENTO EXPERIMENTAL**	**66**

4.1 PARÂMETROS GEOESPACIAIS 67

4.2 PARÂMETROS ESTRUTURAIS 72

4.3 MODELOS RESULTANTES 77

4.4 EXECUÇÃO DO EXPERIMENTO 82

4.4.1 Consultas 85

4.4.2 Coleta, Dados e Estatísticas 86

4.4.3 Ameaças e Validade do Experimento 89

5 RESULTADOS 93

5.1 VOLUME DE DADOS 93

5.2 TEMPO DE INGESTÃO 97

5.3 TEMPO DE EXECUÇÃO 99

5.3.1 Fator de Escala 1 99

5.3.2 Fator de Escala 10 104

5.3.3 Análise Consolidada 107

6 CONCLUSÃO 110

6.1 RETROSPECTO 110

6.2 TRABALHO DESENVOLVIDO 111

6.3 CONTRIBUIÇÕES 113

6.4 TRABALHOS FUTUROS 114

7 REFERÊNCIAS 115

APÊNDICE A - CONSULTAS EXECUTADAS 122

APÊNDICE B - MELHORES E PIORES ESQUEMAS 129

1 INTRODUÇÃO

Dados são produzidos, coletados e armazenados desde os primórdios da humanidade, sejam na forma de pinturas rupestres feitas em cavernas, equações trigonométricas cunhadas em tábuas de argila, hieróglifos grafados em papiros ou bits gerenciados por computadores. Eles abstraem nossa realidade e representam informações úteis sobre pessoas, lugares, objetos, acontecimentos e tudo mais que nos cerceia. Os computadores e suas modernas tecnologias de armazenamento fizeram com que não só conseguíssemos estocar mais dados mas também pudéssemos consultá-los com facilidade e empregar seus conhecimentos em prol dos interesses humanos.

As grandes corporações são as principais responsáveis pela coleta, estocagem e o processamento dos dados produzidos por cada um dos indivíduos que consomem produtos e serviços com alguma vertente digital [1]. Para elas, a análise de tais dados viabiliza o funcionamento de suas aplicações bem como a descoberta de novas oportunidades de negócios. Com dados abundantes e analistas qualificados, torna-se possível decidir com maior assertividade, validar decisões passadas, investir com mais confiança e, por conseguinte, assegurar vantagens competitivas sob concorrentes. Essa análise é feita com dados oriundos de diversas fontes, com grande volume, e envolve operações de junção, agregação, filtragem e o processamento de milhões de registros por consulta. Tal processo e os sistemas que tornam isto possível são chamados de OLAP (Online Analytical Processing) [2-4].

Um sistema OLAP, conceitualmente, organiza mensurações em matrizes multidimensionais que são popularmente chamadas de cubos. Cada dimensão do cubo, pode dispor de hierarquias com níveis variáveis de detalhamento, dos mais seletos (e.g. dia, bairro, produto) aos menos (e.g. mês, país, departamento). Assim, um analista consegue agregar medidas partindo de níveis hierárquicos, em operações conhecidas como *drill-down*, *roll-up*, *slice* e *dice*, que viabilizam a seleção e a navegação entre hierarquias. A implementação desse ferramental se preocupa em recuperar dados de forma eficiente para suportar a execução de consultas

complexas *ad-hoc* com tempos de processamento inferiores aos sistemas de bancos de dados convencionais.

Um *data warehouse* (DW) é um repositório de dados construído especialmente para o uso com os sistemas de suporte à decisão, como os do tipo OLAP. No DW, estão presentes dados provenientes de várias fontes, sendo organizados por assunto, integrados, não voláteis e que são acumulados ao passar do tempo [5]. O projeto de um DW, usualmente, se vale do modelo dimensional de Kimball e dos conceitos de fato e dimensão contidos nele. As tabelas de fatos. Os fatos são, tipicamente, atributos numéricos que são armazenados em uma única tabela, chamada tabela de fatos, e podem ser agregados para mensurar partes ou a totalidade de algum processo. Já as dimensões, são tabelas independentes contendo informações descritivas do negócio que ajudam na caracterização dos fatos. No modelo dimensional, as linhas da tabela de fatos (i.e. vendas de um varejo) possuem chaves estrangeiras que referenciam as tabelas dimensionais (i.e. produto, cliente, transportadora) para conectar as medidas aos dados descritivos que estão atrelados a elas.

Além de dados alfanuméricos, um DW também pode armazenar dados geoespaciais em uma ou mais de suas dimensões e, neste caso, se utiliza a terminologia *geospatial data warehouse* (GDW) para caracterizar tal banco. Entre os dados geoespaciais, ou geográficos, podem estar estruturas de dados que representem pontos, linhas e polígonos, abstraindo coisas do mundo real como um endereço, uma rua e o perímetro de um bairro, respectivamente. Um sistema analítico que utilize dados geográficos é chamado de SOLAP (Spatial OLAP). Ele deve ser capaz de processar consultas analíticas (e.g. drill-down, roll-up) com predicados espaciais (e.g. contém, está contido, intersecta).

A implementação subjacente de um DW, que é nada mais do que um tipo característico de banco de dados, foi baseada no modelo relacional de Codd durante muito tempo. Isto começou a mudar a partir do fenômeno do Big Data, quando o ritmo com que dados são criados caminhou para uma curva exponencial e a demanda por armazenamento explodiu. O então analista da Gartner, gigante estadunidense que presta consultoria, Doug Laney, contextualizou o Big Data com

três parâmetros básicos, em meados de 2001, são eles: volume, velocidade e variedade [6]. O volume cresceu de poucos megabytes para petabytes, a velocidade passou dos processamentos em lotes para cargas feitas quase em tempo real e a variedade saltou de simples dados alfanuméricos, guardados em simples tabelas, para dados complexos como áudios, imagens, vídeos, topografias faciais e outros.

As demandas dos ambientes de Big Data por mais volume, variedade e velocidade comprometeram a capacidade dos bancos de dados relacionais de atenderem tais aplicações sem que a qualidade dos serviços prestados fosse deteriorada. Algumas das soluções propostas para mitigar tais perdas envolveram técnicas de escalabilidade vertical mas, elas se mostraram muito caras e limitadas pelos avanços tecnológicos na produção de hardwares mais eficientes. Nas abordagens de escalabilidade horizontal, os controles de consistência, integridade, transacionais e a imposição das regras ACID se tornaram tarefas muito complexas de serem executadas em ambientes distribuídos. Assim, bancos de dados NoSQL ganharam destaque, pois, neles, não se faz necessário a normalização de tabelas e os demais controles mencionados podem ser relaxados [7] em prol de mais performance.

O relaxamento das restrições e controles supramencionados possibilitam uma maior escalabilidade aos bancos de dados NoSQL. Ela pode ser realizada através do emprego de *commodity hardware* na construção de *clusters* computacionais. No caso de aplicações transacionais, o uso de tais bancos é dificultado pela necessidade da implementação de funcionalidades adicionais para as quais eles não possuem suporte nativo mas, no caso de aplicações analíticas, o procedimento conhecido pela sigla em inglês ETL (Extract, Transform and Load) é realizado antes dos dados serem ingeridos para sanar algumas dessas demandas. Dessa forma, os ganhos de armazenamento e processamento, advindos da maior escalabilidade, são grandes atrativos para a adoção dos bancos NoSQL.

Bancos de dados NoSQL costumam ser divididos em quatro paradigmas: chave-valor, orientados a colunas, orientados a documentos e orientados a grafos. Cada um possui particularidades que lhes conferem vantagens e desvantagens, a depender da aplicação. No caso dos orientados a colunas, os bancos de dados

MonetDB [46] e MonetDB/X100 [47] que, apesar de terem um arcabouço do tipo relacional, foram os primeiros a demonstrarem vantagens de projetos orientados a colunas. Esses benefícios advieram de melhoras no uso das CPU e dos sistemas de cache, além da diminuição do número de operações de I/O, observadas durante testes experimentais com bancos comerciais e de código aberto em benchmarks como o TPC-H, por exemplo. Ainda no contexto de bancos orientados a colunas mas agora olhando para bancos que não sejam relacionais, no estudo de Abadi et al. [8] é avaliado e comparado o desempenho de um banco de dados orientado a linhas (row-store) tendo inúmeras configurações iniciais com um banco de dados NoSQL orientado a colunas, constatando a superioridade do segundo sob um benchmark para data warehouses. Isso acontece porque, neste paradigma, as operações de leitura só precisam manipular, efetivamente, os dados necessários para retornar uma resposta em função dos demais estarem separados fisicamente em setores distintos não contíguos da memória.

Neste estudo, abordaremos o uso do banco de dados NoSQL denominado HBase na construção de *data warehouses* geográficos. Ele é orientado a colunas e é tipificado como um banco de colunas largas, não relacional, distribuído e versionado, capaz de armazenar tabelas com bilhões de linhas e milhões de colunas em *clusters* de máquinas, com acesso randômico a leituras e escritas [9]. No caso dele e de seus pares NoSQL, os esquemas são frequentemente elaborados para atender as demandas específicas da aplicação. O projetista analisa os padrões de escritas e leituras, que serão empreendidos pelo cliente, e busca organizar os dados de forma a balancear a escalabilidade com o desempenho na execução dos caminhos de IO (Input/Output) mais frequentes. Como os bancos de dados NoSQL não costumam suportar a representação de relacionamentos e nem implementar junções nativamente [7], os esquemas acabam seguindo abordagens desnormalizadas para evitar que tais funcionalidades sejam criadas a nível de aplicação. Essa desnormalização de dados coloca ainda mais pressão sob o custo de armazenamento e precisa ser bem ponderada em GDWs.

O Apache Cassandra é um outro banco de dados orientado a colunas muito popular que também é um projeto de alto nível mantido pela Fundação Apache. Ele e o HBase tem muitas características em comum e diferem mais no que tange a

nomenclatura das estruturas de dados e a como foram implementadas suas arquiteturas. Optamos por trabalhar com o HBase em razão de termos feito a identificação prévia de mais estudos [10-19] relacionando o projeto esquemático com a redundância de dados, tangendo abordagens normalizadas e desnormalizadas, além de análises de desempenho de cunho similar que permeiam o escopo deste trabalho. Um outro aspecto importante que levou a escolha do HBase em detrimento ao Cassandra é o fato dele envolver o processamento de menos estruturas de dados no fluxo de recuperação de informações e contar com mecanismos de cache e de filtragem robustos, os quais, aceleram a velocidade dos acessos randômicos sem impactos mais negativos para as leituras em série. Tais ganhos são especialmente amplificados quando a carga de consultas envolve mais leituras ou uma equivalência entre leituras e escritas [48], o quê, é bastante desejável em sistemas construídos com finalidades analíticas.

1.1 MOTIVAÇÃO

A busca pela otimização contínua dos sistemas analíticos perpassa por melhoramentos em todas as camadas de software envolvidas em seus funcionamentos, desde a execução das operações de entrada e saída da memória até a apresentação de resultados em um monitor. Uma etapa passível de otimização, é o planejamento do esquema empregado no *data warehouse*. Nos relacionais, isto pode ocorrer através da criação de índices e desnormalização de tabelas [3]. Já em bancos NoSQL, conforme mencionado anteriormente, existe um direcionamento implícito para a adequação aos padrões de escritas e leituras das aplicações e a adoção de estratégias totalmente desnormalizadas que evitam os altos custos das junções. Alguns estudos científicos [13-17] avaliaram como promover tais otimizações no contexto dos bancos NoSQL orientados a colunas. Eles propuseram esquemas com abordagens variadas, algoritmos inteligentes capazes de otimizar a separação física dos dados nas famílias e técnicas de indexação para os casos envolvendo dados geoespaciais.

Seguindo a premissa de adequar o esquema de dados ao padrão de consultas trabalhadas, alguns estudos [10-12] utilizaram algoritmos inteligentes para separar os dados fisicamente com base nos atributos de cada padrão de consultas,

através das famílias de colunas. As famílias são grupos que particionam verticalmente as tabelas e separam fisicamente seus dados das demais famílias, logo, distribuir os dados entre famílias tem o apelo de evitar que os dados não efetivamente necessários sejam lidos durante a execução de uma consulta. Todos os estudos lograram êxito e conseguiram a redução do tempo demandado para a execução da carga de testes, no entanto, isto só foi possível porque tais cargas eram conhecidas previamente e alimentaram os algoritmos. Argumentamos que essas abordagens são incompatíveis com as premissas existenciais dos sistemas analíticos, uma vez que, um dos principais propósitos deles é auxiliar os especialistas do domínio na extração de dados e informações relevantes que ajudem no aprimoramento dos negócios de corporações. Afinal, se a carga de consultas fosse conhecida de antemão, a criação dos sistemas analíticos não se justificaria em termos financeiros e práticos, pois, bastaria que as consultas fossem executadas uma única vez sem qualquer interferência humana para a geração dos relatórios gerenciais.

Outros estudos [13-17], abstraíram a existência de uma carga de consultas preexistente e focaram na análise experimental comparativa de vários esquemas de dados, os quais, foram criados a partir de conhecimentos específicos sobre o paradigma orientado a colunas e embasados por pesquisas publicadas anteriormente. Foram propostos três esquemas nomeados de NLA, DLA e DLA-CF que seguem abordagem normalizada, desnormalizada com apenas uma família e desnormalizada com mais de uma família de colunas, respectivamente. O alto custo de tempo para execução de junções conferiu o pior desempenho ao NLA em todos os cenários avaliados. Contrapondo os outros dois, a conclusão dos pesquisadores foi de que poucas famílias no esquema beneficiam aquelas consultas que envolvam atributos de várias dimensões e, diametralmente, consultas envolvendo um número pequeno de dimensões se saem melhor quando existem muitas famílias.

Em [17], os autores propuseram uma heurística pontual que, a partir do esquema DLA-CF, consistia em deslocar as colunas das famílias sabidamente mais consultadas para a família dos fatos. Deram como exemplo a dimensão tempo, que, devido ao seu uso recorrente em consultas, deveria ser incorporada a família dos fatos para originar o modelo batizado de FactDate. O esquema proposto conseguiu

superar o DLA-CF em todos os cenários avaliados, mas só bateu os tempos de execução do DLA quando as consultas envolviam os atributos de no máximo duas dimensões.

Diante do exposto, a literatura parece convergir para o uso de esquemas que adotem abordagens desnormalizadas e façam uma distribuição pertinente das colunas em famílias, a fim de melhorar o desempenho dos *data warehouses* e, consequentemente, dos sistemas analíticos. Porém, quando dados geoespaciais passam a fazer parte do escopo das consultas, tal direcionamento pode levar a elaboração de modelos inconcebíveis dentro de parâmetros práticos. Por exemplo, tomando como base o modelo FactDate, poderíamos cogitar a criação de um modelo chamado FactGeo que adote abordagem desnormalizada e englobe dados geográficos na mesma família dos fatos. Geometrias de bairros, cidades e estados são exemplos de dados geoespaciais que podem ocupar dezenas de megabytes e são passíveis de existirem no nosso GDW hipotético que adota o FactGeo. No cenário postulado, a redundância de dados geoespaciais será elevada e isto poderá demandar a compra de mais máquinas e memórias para aliviar as pressões de escalabilidade e desempenho.

A implementação de sistemas SOLAP construídos com o suporte de um GDW sob bancos de dados do tipo NoSQL orientados a colunas ainda é um tópico pouco explorado na literatura. Alguns dos estudos identificados [18-19] direcionam seus esforços para a elaboração de índices e otimizações na codificação dos dados geográficos, mas não abordam uma das etapas precedentes e relevantes que é o projeto do esquema. Como exemplificado antes, através do FactGeo, muitas ponderações ainda precisam ser feitas para a criação de uma heurística ou metodologia que capture as nuances desses projetos e como eles devem ser conduzidos, a fim de, otimizar seus desempenhos.

Um estudo investigou o uso da seletividade de dados geoespaciais como parâmetro de calibração para o nível de redundância deste tipo de dado nas bases, especificamente, para bancos de dados orientados a documentos [20]. Eles constataram que desnormalizar apenas os dados geográficos mais seletivos e normalizar os menos seletivos culminava na concepção de modelos mais leves e

com melhores desempenhos na execução de consultas. Uma descoberta que vai na contramão da ideia de desnormalizar indiscriminadamente e tem o potencial de contribuir com o planejamento de esquemas de dados para os bancos orientados a colunas, pois, desnormalizar evita os custos da execução de junções mas tem o potencial de aumentar expressivamente o armazenamento exigido para suportar a alta redundância de dados. Um outro aspecto negativo da redundância é aumentar o número de registros que precisam ser avaliados para a determinação de um predicado, pois, as réplicas precisarão ser avaliadas de forma independente. Exemplificando, no cenário hipotético em que existam duas tabelas cujo relacionamento tenha cardinalidade N:1 (i.e. cliente e cidade) e nós desejemos filtrar os clientes com base na geometria da cidade, a avaliação será feita uma única vez nos registros das cidades e só depois ocorrerá a junção com os clientes. No entanto, se tivesse sido adotada uma abordagem desnormalizada onde os registros de cidades são replicados em todas as linhas que os referenciam, a avaliação da geometria teria de ser feita também para as réplicas.

1.2 OBJETIVOS

O objetivo primário deste trabalho é avaliar, experimentalmente, quais dados geoespaciais devem ser normalizados ou desnormalizados em função de suas seletividades, e, qual disposição deles em tabelas e famílias de colunas tem o potencial de reduzir os custos de armazenamento e execução de consultas analíticas em GDWs construídas sob o HBase. Esperamos que o nosso estudo consiga servir de orientação para as demais pessoas que trabalham na área e se preocupam em como um projeto do esquema de dados pode influenciar no desempenho de tais sistemas. Assim, traçamos objetivos intermediários que serviram de base para alcançarmos nosso objetivo global, são eles:

- Buscar, na literatura, estudos que tenham investigado as metodologias, considerações e as boas práticas que devem ser adotadas na construção de sistemas SOLAP implementados em bancos de dados NoSQL orientados a colunas. Sobretudo, aqueles com foco na análise de desempenho quando variados parâmetros experimentais.

- Analisar as decisões por trás da concepção de modelos de dados, ou esquemas, para tais bancos, a fim de, entender qual o estado da arte e como se dará sua expansão.

- Conceber esquemas que estendam os já existentes e conduzir testes experimentais em um *cluster* computacional que simule um ambiente passível de uso por uma aplicação real.

- Executar os experimentos e coletar dados que permitam a realização de uma análise comparativa entre os esquemas com o auxílio de testes estatísticos.

- Debater os resultados obtidos nos experimentos e as conclusões que puderam ser obtidas com base no estudo realizado. Indicando pontos fortes, fracos e aspectos relevantes que podem ser estudados em trabalhos futuros.

Neste caminho, almejamos identificar quais variáveis, dentre as selecionadas, têm o potencial de impactar negativamente ou positivamente o desempenho de um sistema analítico construído sob bancos de dados NoSQL orientados a colunas, no caso, representados pelo HBase. Esperamos que seja possível a criação de modelos capazes de balancear os custos de armazenamento advindos do processo de desnormalização com as penalidades impostas para a realização de junções.

1.3 METODOLOGIA

A fim de atingirmos os objetivos estipulados, nós concebemos esquemas para o HBase a partir das combinações entre parâmetros geoespaciais, chamados assim por promoverem abordagens para a implementação dos relacionamentos entre dimensões geográficas, e de parâmetros estruturais, os quais, tratam das abordagens possíveis para os relacionamentos entre a tabela de fatos e as demais dimensões convencionais. A combinação das abordagens escolhidas, para cada um dos parâmetros citados, imputou na construção de projetos esquemáticos únicos que puderam ser exaustivamente gerados e avaliados para a identificação das melhores e piores abordagens. Os esquemas propostos foram utilizados para a transformação do modelo conceitual híbrido do benchmark geoespacial que é

conhecido como pelo nome de Spadawan. Após serem geradas as bases em conformidade com os projetos esquemáticos, carregamos os dados em um *cluster* computacional contendo 3 máquinas configuradas como *datanodes* do Hadoop, *regionservers* do HBase e *workers* do Spark. Uma carga de consultas, tendo predicados geoespaciais, foi executada no *cluster* para cada um dos esquemas e os tempos de execução foram coletados. Em posse de tais dados, conduzimos testes estatísticos apropriados para validar a significância dos resultados e, a partir disso, determinamos quais foram as combinações de abordagens que resultaram nos melhores resultados, no tocante a execução da carga de consultas, tempo de carregamento dos dados e o volume de dados.

1.4 ORGANIZAÇÃO

A presente pesquisa está estruturada em seis capítulos. No segundo capítulo, trazemos conceitos teóricos relevantes para o acompanhamento deste trabalho. Em seguida, no capítulo três, temos estudos científicos relacionados que serviram de base e inspiração para nossos estudos. No quarto capítulo, serão discutidas as transformações realizadas sob o esquema conceitual do Spadawan e introduzidos os modelos resultantes das combinações dos parâmetros. No quinto, nós discorremos sobre a execução do experimento, testes estatísticos e os resultados obtidos. Finalmente, é posto o sexto capítulo contendo as considerações finais do trabalho e possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta conceitos teóricos importantes para o desenvolvimento deste trabalho e necessários para o entendimento do mesmo. Começamos abordando o modelo relacional, o qual, embora seja muito prestigiado e amplamente utilizado, pode não ser a melhor alternativa para a implementação de *data warehouses* voltados aos sistemas SOLAP (Seção 2.1). Em seguida, temos uma breve introdução sobre os bancos de dados NoSQL e como eles ganharam notoriedade com a perspectiva de alcançarem melhores desempenhos ante os relacionais nas aplicações envolvendo grandes massas de dados (Seção 2.2). Ainda na seção anterior, traremos duas subseções contendo informações sobre o banco de dados NoSQL chamado HBase e sobre o framework de computação distribuída chamado Spark. Fechamos o capítulo com duas seções abordando o conceito de data warehouse (Seção 2.3) e versando sobre o benchmark Spatial SSB que avalia de DWs com base em um esquema conceitual e cargas de consultas para avaliações experimentais (Seção 2.4).

2.1 ABORDAGENS RELACIONAIS

O modelo relacional de Codd [22] estabeleceu estrutura e linguagem consistentes com a lógica de primeira ordem para a representação de dados como função dos relacionamentos existentes entre tuplas. Ele viabilizou o emprego de métodos declarativos para definição e consulta de informações estruturadas em bancos de dados. O modelo ganhou tração em razão de sua simplicidade perante os usuários, que conseguiam abstrair detalhes da implementação de baixo nível e, com isso, eram capazes de desenvolver com mais agilidade e menos propensão a erros de programação [23]. Os primeiros bancos de dados relacionais criados com finalidade comercial surgiram no mercado nos primeiros anos da década iniciada em 1980. Muitos deles, continuam recebendo suporte nos dias atuais e ainda figuram entre os mais utilizados do planeta [24], fatos que atestam a importância do modelo relacional para os avanços tecnológicos obtidos na área.

Codd fundamentou-se no conceito de relação matemática da teoria dos conjuntos para expressar bancos de dados na forma de conjuntos de relações onde,

visualmente, cada relação é vista como uma tabela dotada de linhas e colunas. As linhas de uma tabela, formalmente chamadas de tuplas, representam as ocorrências de uma dada entidade e aglutinam dados correlatos a mesma. Já as colunas, ou atributos, servem para facilitar a interpretação das informações contidas em cada linha da tabela. Complementando tais aspectos estruturantes, o modelo relacional dispõe de operações derivadas da álgebra relacional (e.g. seleção, junção, união e subtração) para que as tabelas sejam manipuladas. Estas operações, quando executadas, afetam todas as linhas da tabela de uma única vez, ainda que, os parâmetros estabelecidos para a execução só contemplem poucas, uma ou até nenhuma linha existente.

O conceito de chave é outro importante aspecto do paradigma relacional. Ele designa algum dado presente em uma linha e identificado por uma coluna que seja utilizado para identificar, de forma única, a linha em questão e, neste caso, dizemos que se trata de uma chave primária. As tabelas de um banco relacional, sem exceção, precisam dispor de uma ou mais colunas que cumpram o papel de chave primária, a qual, deve ser única e não nula. Temos ainda, as chaves estrangeiras que são utilizadas para identificar linhas presentes em outras tabelas. Estas, podem ser nulas mas jamais deverão assumir valores que não sejam os de alguma chave primária existente em outra tabela do banco de dados. As restrições impostas às chaves primárias e estrangeiras são conhecidas como restrições de chave e de integridade referencial, respectivamente. São duas das várias restrições de integridade que existem no modelo relacional para garantir a exatidão e consistência.

A combinação entre fundamentação matemática e simplicidade presente nos conceitos de tabela, linha, coluna e chave também estão presentes na linguagem de consulta estruturada conhecida como SQL, a qual, foi criada alguns anos após o paradigma relacional para operacionalizar bancos de dados relacionais proprietários da IBM. O escopo de aplicação da linguagem compreende desde a definição do arcabouço de dados (i.e. criação e edição de esquemas), manipulação de dados por meio de comandos (i.e. inserção, atualização e remoção), consultas e até o controle de acesso aos dados. A SQL se tornou um padrão para os bancos de dados relacionais, tendo sido padronizada de forma conjunta pelo American National

Standards Institute (ANSI) e pela International Standards Organization (ISO) com a denominação de SQL-86 (ou SQL1) em 1986. Essa padronização agregou mais confiabilidade a linguagem e contribuiu indiretamente para o sucesso dos bancos de dados relacionais, uma vez que, o custo de migração entre diferentes sistemas comerciais foi reduzido e isso impulsionou a concorrência entre seus desenvolvedores [3].

Um sistema de gerenciamento de banco de dados (SGBD) é um software que agrega importantes funcionalidades aos bancos de dados, poupando o lado da aplicação de implementar tais recursos adicionais como a segurança da informação, controle de concorrência e a gestão do fluxo de dados entrando e saindo, por exemplo [25]. Além disso, o SGBD é o grande responsável por coordenar a execução de transações. Uma transação pode ser vista como uma coleção de escritas e leituras que desempenham alguma função lógica dentro de um programa computacional. Elas precisam obedecer propriedades conhecidas pelo acrônimo ACID (atomicidade, consistência, isolamento e durabilidade) para garantir a integridade e consistência da base, bem como, o bom funcionamento da sequência lógica presente na aplicação.

Uma análise superficial do modelo relacional revela inúmeras vantagens no tocante a simplicidade dos conceitos envolvidos, existência da poderosa linguagem SQL, transparência nos controles de consistência e no forte arcabouço matemático que elimina ambiguidades. Por outro lado, olhando para as potenciais desvantagens, constatamos que os mesmos controles de consistência, controles de integridade e a imposição das propriedades ACID geram um custo adicional de processamento que pode degradar o desempenho e inviabilizar a escalabilidade de tais sistemas [26]. O aumento do número de tabelas, linhas, colunas, relacionamento e transações atrelado aos gargalos citados coloca pressão sobre as metas de desempenho e faz com que os administradores busquem meios para reverter ou frear tal degradação (i.e. escalabilidade vertical e horizontal).

A escalabilidade vertical consiste na portabilidade do sistema para uma máquina com maior poder computacional ou na realização de melhoramentos pontuais na que está em uso, seja por meio da troca de memória, armazenamento

ou processador por versões mais robustas. Entretanto, apesar de ser uma abordagem direta e aparentemente fácil, existem limitações nas curvas de crescimento do poder computacional de tais componentes e isto pode criar uma disparidade entre a demanda por sistemas e *hardwares* cada vez mais avançados. Um segundo ponto negativo da escalabilidade vertical são os altos custos para a aquisição de tais componentes, uma vez que, muitas vezes, são produtos que não ganham escala e seus desenvolvimentos são lentos e custosos.

Além da abordagem vertical, a escalabilidade também pode ser construída de forma horizontal. No caso, o sistema conta com várias máquinas que conversam entre si para funcionar de forma coesa e transparente. Assim, quando o administrador julgar que precisa de mais recursos, basta acoplar mais máquinas ao *cluster*. Com alguns comandos de software, ou até mesmo automaticamente, o sistema é capaz de detectar a disponibilidade de mais *hardware* e mover tabelas ou pedaços delas para as máquinas ociosas, promovendo a distribuição da carga. A complexidade desta alternativa reside justamente no fato dos dados estarem distribuídos, logo, impor os controles característicos do modelo relacional (i.e. consistência e integridade) via rede impacta na disponibilidade geral do sistema, conforme iremos exemplificar a seguir.

O Teorema de Brewer, também conhecido como Teorema CAP, estipula que quaisquer sistemas de armazenamento que sejam distribuídos não conseguem impor, simultaneamente, garantias de sua consistência, disponibilidade e tolerância ao particionamento de dados na rede [27-28]. Segundo o teorema, apenas duas dentre as três podem coexistir numa aplicação distribuída. Dessa forma, nós podemos concluir que bancos que façam uso da escalabilidade horizontal precisam escolher entre relaxar as garantias de consistência ou as de disponibilidade.

A partir do momento em que a escalabilidade vertical se torna impossível ou impraticável e outras alternativas de otimização são exauridas, o mantenedor de um banco relacional precisará seguir o caminho da escalabilidade horizontal. Para isso, conforme postulado por Brewer, terá de abrir mão da disponibilidade em prol de um sistema distribuído e consistente. A área de pesquisa que avalia o desempenho de sistemas, postula que a disponibilidade de um sistema com componentes que são

interdependentes é dada pelo produto das disponibilidades de cada componente. Assim, um *cluster* com duas máquinas idênticas disponíveis 99.9% do tempo terá uma disponibilidade global de 99.8%, ou 86 minutos de inatividade por mês. Se incluirmos uma terceira máquina, teremos um sistema com disponibilidade de 99.7% e que fica inativo por 130 minutos no mês. Já para o cenário com 10 máquinas, o cluster ficaria inativo por pouco mais de 7 horas.

Os desafios de escalabilidade são um dos principais pontos a serem considerados antes da escolha da tecnologia de banco de dados que será utilizada por uma aplicação, especialmente, se ela está inserida em ambientes característicos de Big Data. O paradigma relacional possui muitas vantagens e é a escolha ideal em muitos cenários mas, quando a aplicação pode abrir mão dos seus recursos que também degradam o desempenho (i.e. controle de consistência, integridade e transacional), é muito provável que outros paradigmas se mostrem mais apropriados.

2.2 ABORDAGENS NOSQL

O termo NoSQL foi cunhado em 1998 por Carlo Strozzi para batizar um banco de dados autoral que seguia o paradigma relacional mas não utilizava SQL como ferramenta de operacionalização, logo, o nome do software foi dado como a concatenação do advérbio de negação "no" da língua inglesa e pelo nome da linguagem SQL [29]. Uma década depois, ele ressurgiu com nova conotação após os engenheiros de software Evans e Oskarsson o utilizarem para caracterizar bancos de dados que não se enquadrassem no padrão de relações tabulares do modelo relacional [30]. Hoje em dia, seu uso é mais associado à alcunha de *não-apenas-SQL*, para identificar bancos que não se baseiam no paradigma relacional mas, ainda assim, podem trabalhar com linguagens similares a SQL.

Bancos de dados categorizados como NoSQL, seguindo a definição atual, datam de antes mesmo do modelo relacional, apesar do termo não existir na época de suas concepções. Eles ganharam notoriedade à medida que ocorreu o crescimento da digitalização, popularização das redes sociais, o surgimento da internet das coisas e, consequentemente, o rápido aumento do número de dados

transacionados [31]. A expansão da quantidade de informações disponíveis fomentou a eclosão de programas computacionais orientados por dados (*data-driven apps*). Tais aplicações precisam ser alimentadas com um vasto universo de dados (i.e. transacionais, espaciais e sensoriais) extraídos de forma contínua da internet, as vezes, em tempo real. Estas demandas, cada vez maiores, foram responsáveis por evidenciar fragilidades nos bancos de dados relacionais, os quais, não tiveram o desempenho aprimorado em ritmo compatível com o esperado. A classe dos NoSQL se mostrou uma alternativa viável e tem se provado a mais adequada em muitos casos de uso.

Os sistemas de bancos de dados e SGBDs NoSQL são, comumente, produtos de código aberto que se distinguem dos relacionais por terem esquemas mais flexíveis, conseguirem prover um melhor controle de disponibilidade, oferecerem melhores implementações de escalabilidade horizontal e reduzirem o descasamento entre as estruturas de dados relacionais e as utilizadas na programação orientada a objetos [32]. Muitos deles favorecem a disponibilidade, tolerância ao particionamento e ganhos de velocidade em detrimento a manutenção de consistência. Aqueles que possuem seus controles nativos para a manutenção da consistência, adotam as propriedades ACID no projeto ou um modelo alternativo chamado BASE (Basically Available, Soft State, Eventually Consistent), que relaxa restrições de consistência oportunamente para aumentar a disponibilidade. Apesar de todos os aparentes benefícios, os sistemas NoSQL ainda possuem grandes barreiras de entrada, como o baixo nível de maturação das linguagens de consulta aceitas por eles, falta de operações nativas para a execução de junções *ad-hoc* entre tabelas, poucas interfaces de programação consistentes com as oferecidas pelos demais SGBDs já consolidados no mercado e os altos custos que já foram empreendidos na adoção dos sistemas relacionais por grande parte da indústria.

Bancos NoSQL são amplamente categorizados de acordo com as estruturas de dados empregadas em seus modelos. Podem, por exemplo, ser dos tipos chave-valor, orientados a grafos, orientados a documentos e orientados a colunas. O tipo chave-valor organiza os dados na forma de matrizes associativas, onde chaves referenciam estruturas de dados que não recebem nenhum tratamento especial do banco, como indexação. Ele é considerado um modelo fundamental que serve de

base para outros. Os orientados a documentos são, inerentemente, uma subclasse dos chave-valor mas suas chaves referenciam estruturas de dados denominadas documentos, as quais, podem aninhar outras chaves e também são analisadas para a extração de metadados que auxiliem na otimização de consultas e do armazenamento. Similarmente, os orientados a colunas podem ser vistos como matrizes multidimensionais cujas combinações de chaves servem para a disposição dos dados em arquivos físicos distintos. Por fim, os orientados a grafos são indicados para o uso com dados que possam ser representados por grafos, como os relacionamentos existentes entre usuários de uma rede social, mapas e topologias de redes.

As breves caracterizações dos tipos de bancos de dados NoSQL mostram que os dados podem ser armazenados em um vasto conjunto de estruturas, as quais, devem ser escolhidas seletivamente de acordo com as necessidades de cada aplicação para maximizar os ganhos de desempenho que cada umas das particularidades pode oferecer. Neste estudo, depositamos o nosso foco naqueles orientados a colunas, em especial, na variação conhecida como banco de dados NoSQL de colunas largas. Eles são mais uma opção dentre os tipos elencados e podem ser caracterizados em virtude de trazerem traços das estruturas tabulares do modelo relacional com a separação física dos dados baseada no grupo de colunas ao qual fazem parte. O termo colunas-largas remete justamente ao fato de que a separação física é feita para grupos de colunas ao invés da individualização de cada uma das colunas existentes na tabela, como ocorre nos orientados a colunas.

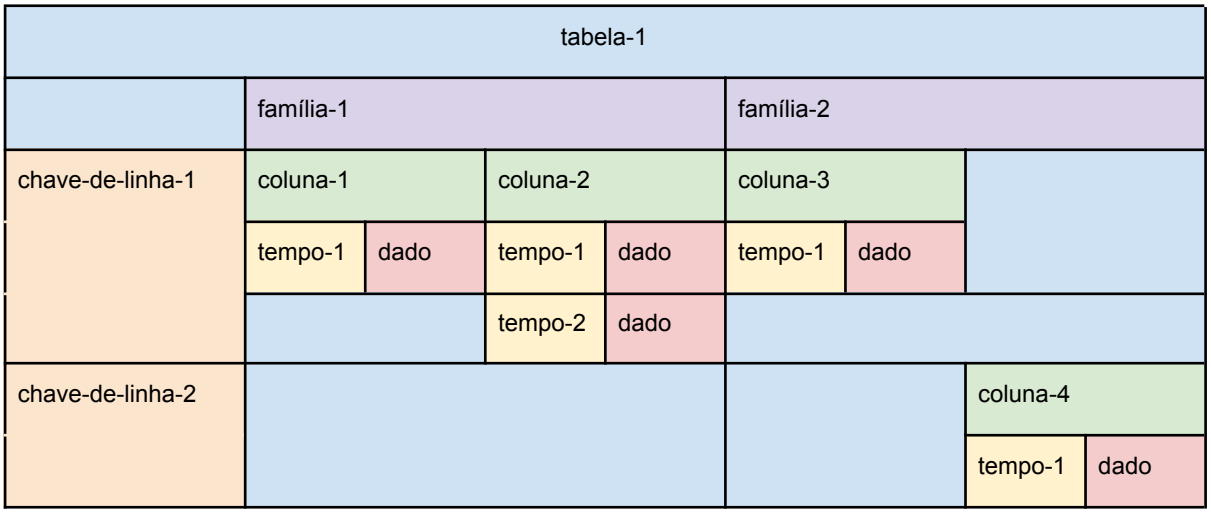
O modelo de banco de dados de colunas largas (*wide-column*) emprega as mesmas terminologias de tabela, linha e coluna que conhecemos do mundo relacional. Bem como nos relacionais, temos a noção de tabela como uma estrutura para armazenamento de dados, linhas para agrupar dados relacionados e colunas para identificar pedaços desses dados em cada linha. São introduzidos os conceitos de chave de linha, que é um campo utilizado para identificar as linhas de forma similar as chaves primárias, e família de colunas, que atua como um grupamento de colunas com padrão de acesso comum. Na subseção seguinte, iremos abordar esses conceitos sob a ótica funcional do banco de dados Apache HBase e discutiremos mais detalhes práticos dos *wide-column*.

2.2.1 Apache HBase

O Apache HBase, também chamado só de HBase, é um banco de dados *wide-column* caracterizado por ser de código aberto, escalável horizontalmente, consistente em alguns casos de uso, ter baixa latência e prover acesso randômico aos dados [9]. Ele é mantido pela Fundação Apache como um projeto de alto nível e recebe contribuições de várias instituições privadas. Sob a ótica do Teorema de Brewer, o HBase é considerado um sistema que garante a consistência dos dados no escopo das linhas e tolera o particionamento de dados na rede. Esse comprometimento parcial para garantir a consistência degrada a disponibilidade do banco mas, ainda assim, não impede que as latências de escrita e leitura sejam mantidas baixas. Outra característica importante do HBase é que ele utiliza a infraestrutura do Sistema de Arquivos Distribuídos Hadoop (HDFS) para armazenar seus dados na camada de mais baixo nível. O HDFS, que também é mantido pela Apache, foi desenvolvido para o armazenamento de arquivos com centenas de *megabytes* em ambientes distribuídos, logo, cabe ao HBase otimizar a entrada e saída de dados dos blocos de memória do HDFS e orquestrar o fluxo e volume de dados transacionados no banco de dados.

O modelo de dados do HBase é análogo ao de um banco de dados do tipo chave-valor que conte com múltiplas dimensões, ou seja, forme uma matriz associativa onde valores são acessados por meio do fornecimento sequencial de chaves que, juntas, designam o caminho até cada informação contida no banco. No HBase, existe uma hierarquia entre as chaves semelhante a existente entre os conceitos de tabelas, linha e colunas do modelo relacional. Os dados serão sempre lidos e escritos em tabelas identificadas por um nome e compostas por inúmeras linhas. Cada linha, engloba uma coleção de colunas e é identificada, unicamente, por uma coluna especial conhecida como chave de linha. As colunas de uma linha são dispostas em grupos denominados famílias de colunas e, por sua vez, podem conter múltiplos valores atrelados ao instante de tempo da escrita. Logo, o padrão de escritas e de leituras do HBase, envolve a especificação do nome de alguma tabela, da chave de linha, do nome da família de colunas, nome da coluna e da etiqueta de tempo.

Figura 1 - Componentes esquemáticos do HBase.



Fonte: autor (2022).

O relacionamento entre os diversos elementos esquemáticos apresentados no parágrafo anterior é ilustrado na Figura 1. Nela, visualizamos uma tabela (*tabela-1*) com duas linhas (identificadas por *chave-de-linha-1* e *chave-de-linha-2*) e duas famílias de colunas (*família-1* e *família-2*). Na primeira linha, podemos observar a existência de duas colunas (*coluna-1* e *coluna-2*) na *família-1* e apenas uma coluna (*coluna-3*) na *família-2*. Já na segunda linha, enxergamos uma única coluna (*coluna-4*) disposta na *família-2*. A combinação de tabela, chave de linha, família e coluna pode ser acrescida de etiquetas de tempo para finalizar o caminho de acesso aos dados. A célula localizada através do caminho *tabela-1*, *chave-de-linha-1*, *família-1* e *coluna-1* possui um único registro feito no instante *tempo-1*. Se olharmos a outra coluna, constatamos que existem dois registros escritos em *tempo-1* e *tempo-2*. O caminho completo trará o dado escrito no instante desejado e, a omissão da etiqueta de tempo, fará com que o banco retorne o valor mais recente.

A criação de tabelas no HBase demanda que sejam especificados os nomes da tabela e das famílias de colunas desejadas. Colunas, como visto na ilustração acima, existem de forma independente no escopo de cada linha e, por isso, podem ser criadas dinamicamente à medida em que dados sejam inseridos na tabela. Isto acontece porque, no modelo físico do HBase, os dados são particionados verticalmente de acordo com suas respectivas famílias e, cada família, origina um mapa associativo capaz de armazenar um número arbitrário de colunas. Tais mapas

são armazenados em arquivos especiais denominados de HFiles, os quais, além de atuarem como estruturas chave-valor, detém metadados, índices e filtros Bloom que possibilitam escritas e leituras aleatórias.

Além do particionamento vertical, baseado nas famílias, o HBase também realiza particionamentos horizontais com base nas chaves de linha. Eles são efetuados através da distribuição das linhas de uma tabela entre inúmeras partições que, por sua vez, são distribuídas equitativamente entre as máquinas do cluster. Quando alguma tabela, ou uma de suas partições, atinge o tamanho limite preestabelecido, o servidor responsável irá dividi-la ao meio, dando origem a duas novas partições. A execução sucessiva desses particionamentos acaba por distribuir pedaços de tabelas entre todas as máquinas do cluster, a fim de distribuir a carga de acesso, melhorar o desempenho do sistema e maximizar o uso do poder computacional disponível em cada nó. Essas partições recebem o nome de região e as máquinas que as gerenciam são chamadas de servidores de região.

A arquitetura do HBase possui três componentes principais, o nó mestre (HMaster), os servidores de região (RegionServers) e o ZooKeeper. Uma das máquinas do cluster, escolhida pelo usuário, irá assumir o papel de nó mestre, sendo responsável por gerenciar o cluster do HDFS, monitorar todos os servidores de região, ser a interface para alterações de metadados e cuidar da recuperação do sistema na ocorrência de falhas. É possível a criação de mestres adicionais, mas eles permanecerão dormentes até que o nó ativo falhe. Os servidores de região compreendem todos os outros nós do cluster, encarregados de operacionalizar escritas, leituras, atualizações e deleções originadas dos clientes. Neles, estarão presentes as HFiles juntamente com outras estruturas de dados relevantes que comentaremos nos próximos parágrafos. O último componente abordado é o ZooKeeper, um processo computacional que coordena, sincroniza e viabiliza a comunicação entre o nó mestre, os servidores de região e clientes. Ele mantém um registro atualizado de todas as regiões existentes, conseguindo direcionar as requisições para os servidores responsáveis sem ajuda do mestre. Uma visão completa da arquitetura com as interações entre componentes está ilustrada na Figura 2.

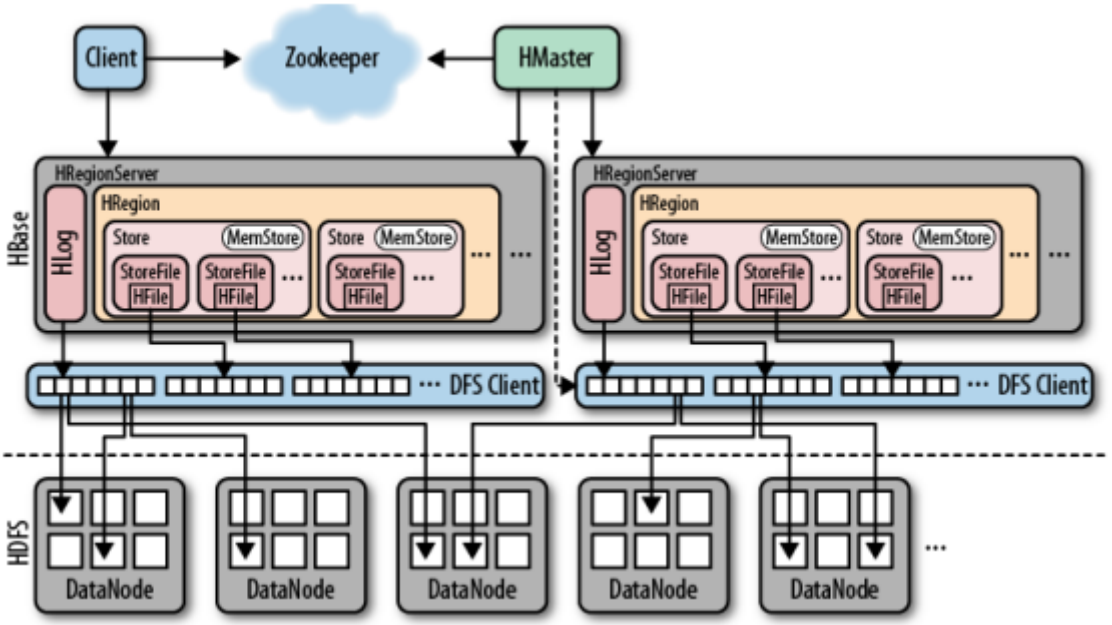
O caminho de escritas no HBase é projetado para suportar uma alta taxa de operações enquanto preserva a durabilidade, consistência e escalabilidade dos dados. Começa com o cliente solicitando ao ZooKeeper uma tabela de metadados que indica as localizações de todas as regiões distribuídas entre os servidores de região do cluster. Uma vez que o servidor de escrita tenha sido localizado na tabela, com base no ordenamento das chaves de linha, o cliente envia a requisição de escrita e fica aguardando que a persistência seja confirmada. No servidor, o dado é inicialmente escrito na WAL, uma estrutura de dados em disco semelhante a uma pilha e que garante a durabilidade dos dados em caso de falhas. Depois, se bem sucedido, o servidor grava o dado na MemStore, uma cache de escritas localizada na memória principal, e sinaliza o sucesso da operação. Após o cliente ter sido notificado, todas as leituras subsequentes obterão o valor que acaba de ser gravado ou um mais novo que tenha sido escrito posteriormente. Esta garantia alinhada à atomicidade da escrita em quaisquer colunas de uma mesma linha compreende o grau de consistência existente no HBase.

A partir do caminho de escritas traçado no parágrafo anterior constatamos que as estruturas WAL e MemStore irão crescer no mesmo ritmo até que o limite da memória principal do sistema seja atingido, logo, para garantir a escalabilidade, o HBase conta com as HFiles. Quando a MemStore se aproxima do limite preestabelecido, é realizado um procedimento chamado de descarga no qual o sistema persiste os dados existentes nela em HFiles e libera a memória principal. O problema da escalabilidade deixa de existir neste âmbito e retorna ao particionamento de tabelas e distribuição de suas regiões. Outro aspecto da descarga é que, a partir do momento que ela ocorre, não se faz mais necessário manter cópias dos dados na WAL e elas podem ser apagadas.

Apresentamos agora o caminho de leituras no HBase. Assim como nas escritas, o processo começa com o cliente solicitando as localizações das regiões e determinando em qual servidor de região o dado desejado está armazenado. Uma solicitação de leitura contendo os nomes da tabela, chave de linha, família de colunas, coluna e etiqueta de tempo é então enviada ao servidor e se aguarda a resposta com o dado. O servidor de região irá verificar se o dado existe na MemStore e, em caso negativo, precisará buscar nas HFiles. Como acessos ao

disco são ordens de magnitude mais lentos do que a memória principal, o HBase utiliza uma estrutura de dados chamada de Block Cache para guardar na RAM os blocos de memória lidos mais recentemente do disco, sempre acessando esta estrutura depois da MemStore e antes das HFiles.

Figura 2 - Componentes arquiteturais do HBase.



Fonte: HBase - Introduction to Distributed, Scalable, Big Data Store por Rakesh Jamdare (2014).

O projeto do modelo de dados para o HBase perpassa pela estruturação inteligente das chaves de linha, pois, é a partir delas que o sistema irá particionar tabelas em regiões e distribuir tais fatias entre os nós do *cluster*. Uma má distribuição poderá ocasionar problemas de *hotspotting*, que são observados quando grande parte da carga de acessos fica concentrada em um pequeno conjunto de máquinas enquanto outras estão ociosas. Um cenário, muito comum, em que isso pode ocorrer é quando são utilizadas chaves incrementais, de forma que os registros mais recentes ficam com as maiores chaves e os mais antigos com as menores. Assim, uma consulta que tenha janela temporal irá acabar selecionando apenas dados armazenados continuamente em poucas máquinas.

A definição das chaves de linha também é importante para a otimização do modelo em função dos padrões de acesso que serão empreendidos pela aplicação,

uma vez que, o projetista pode se valer dos baixos custos de armazenamento e leitura para eliminar ou reduzir a ocorrência das custosas operações de junção, varredura e busca. Exemplificando o primeiro dos cenários, relacionamentos com cardinalidade N:M poderiam ter os dados das entidades combinados em uma única instância e a chave de linha resultante seria a concatenação das duas chaves primárias, assim, a junção seria eliminada e restaria o custo da leitura. No caso das varreduras, o ordenamento lexicográfico feito nas chaves de linha pode ser utilizado para reduzir o escopo das varreduras através da inclusão de prefixos nas chaves. Finalmente, como o custo de armazenamento é baixo e o HBase é eficiente na leitura de dados, se torna possível a criação de tabelas adicionais que sirvam para indexar dados e evitar buscas linha a linha. Esses são apenas alguns exemplos de como as chaves de linha podem impactar no projeto do modelo de dados.

A flexibilidade do modelo de dados, falta de indexação avançada, ausência de tipagem de dados e a busca por escalabilidade contribuíram para que o HBase não oferecesse, nativamente, suporte a uma linguagem estruturada similar a SQL. Ele trabalha com operações primitivas que cumprem os papéis de leitura (GET), inserção (PUT), recuperação de linhas (SCAN) e remoção (DELETE). Não são disponibilizados métodos para a junção e agregação de dados. Apesar de tais limitações, o HBase é robusto o suficiente para que interfaces de consultas avançadas possam ser acopladas e consigam suprir essas demandas sem degradar expressivamente o desempenho do sistema. Softwares como o Apache Phoenix, HIVE e Spark são exemplos de aplicações que fazem tal papel. Também não há suporte nativo a execução de consultas geoespaciais mas, isso pode ser contornado com o uso de bibliotecas de terceiros que interagem diretamente com o HBase ou com algum dos softwares que possibilitam a execução de consultas avançadas como o GeoSpark, que é uma biblioteca do Spark.

2.2.2 Apache Spark

O Spark é um *software* de código aberto desenvolvido para o processamento de grandes volumes de dados em *clusters* computacionais, sendo capaz de prover uma interface para que tais sistemas sejam programados com poder de processamento paralelo e tolerância a falhas de forma implícita para os usuários

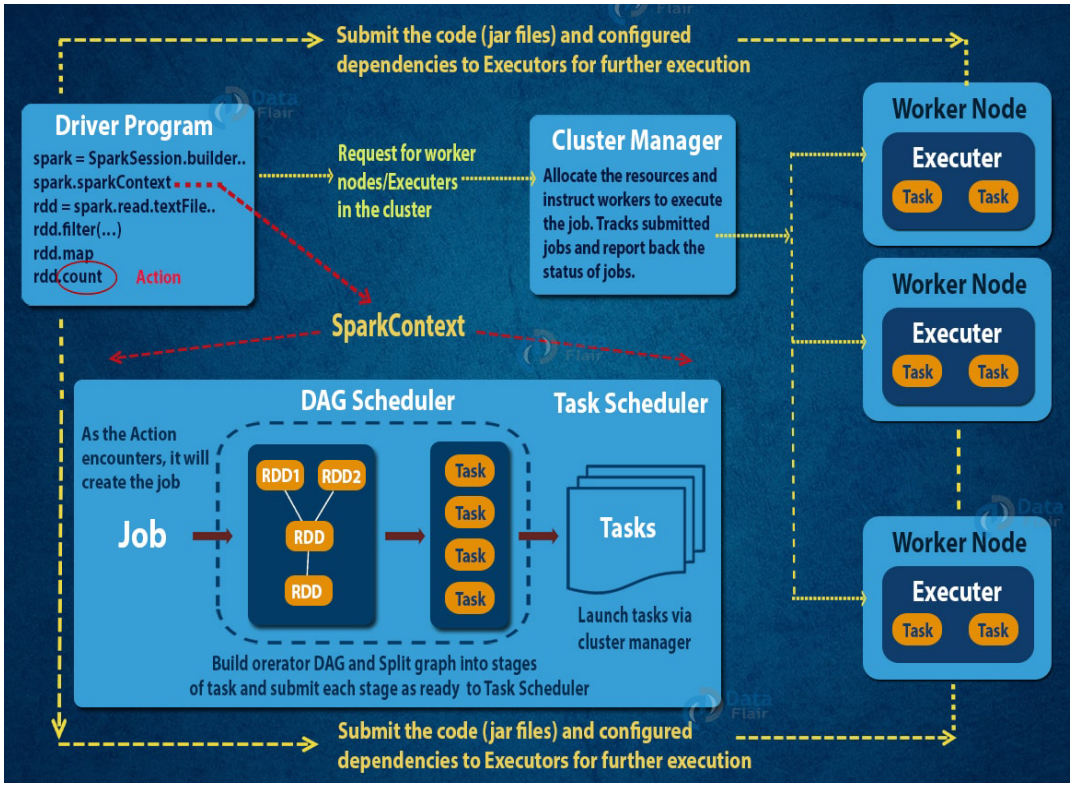
[33]. As principais funcionalidades do Spark são implementadas como módulos independentes construídos sob o Spark Core, o qual, é o módulo central que implementa funções elementares como as *map*, *reduce*, *filter* e *collect*. Outros componentes muito importantes e bem utilizados pela comunidade são o Spark SQL para a execução de consultas sobre dados que possam ser acessados pela ferramenta, o Spark Streaming para o processamento de dados que fluam em tempo real, o GraphX para processar estruturas de grafos e, por último, o MLlib com ferramentas para a execução de algoritmos de aprendizagem de máquina.

As aplicações executadas no Spark são geridas por três programas principais, são o Driver, Worker e o Cluster Manager. O Driver roda no nó principal do *cluster* e é responsável por fazer a integração da aplicação principal com as demais funcionalidades do Spark. O Worker é o nome dado aos nós disponíveis para a execução das tarefas que estão especificadas na aplicação, podem existir vários nós com esse papel e eles podem até ser incluídos ou removidos de forma dinâmica. Os recursos computacionais disponíveis em cada Worker são acompanhados e alocados pelo Cluster Manager de acordo com as demandas da aplicação. A adição de mais nós permite a divisão de tarefas entre um número maior de máquinas, o que contribui para melhorar o poder de processamento paralelo do sistema. Mais máquinas também significam mais memória disponível, logo, o Spark consegue reduzir as limpezas de *cache* e se beneficiar das leituras de leituras da memória mais rápidas.

Quando uma aplicação é submetida para o Spark, o código dela é analisado para a geração de um gráfico acíclico dirigido (DAG) que conterá todo o passo a passo, em alto nível, para a execução das transformações e funções contidas no programa. Os vértices dirigidos do DAG designam conjuntos de dados e as arestas representam as operações que deverão ser aplicadas aos dados. Esse mapa é passado para um programa chamado de DAG Scheduler que divide as operações em etapas ainda menores. Todo esse trabalho facilita a elaboração de um plano de execução da aplicação, em baixo nível, para promover otimizações que reduzam o trânsito de dados entre máquinas como forma de melhorar o desempenho do sistema. O Driver recebe tal plano e negocia com o Cluster Manager a obtenção dos recursos necessários para a execução do mesmo e, uma vez obtidos, ele é iniciado.

A Figura 3 ilustra os programas envolvidos na execução de uma aplicação no Spark, delineados em blocos com flechas indicando o fluxo de interações entre eles. No bloco Driver Program, é colocado um trecho de código que instancia um RDD (Resilient Distributed Dataset) e realiza transformações e ações com ele. O RDD nada mais é do que uma abstração para conjuntos de dados que o Spark distribui e gerencia nas máquinas do *cluster*. As transformações são operações que partem do RDD e produzem um novo RDD, elas sofrem o processo de *lazy loading* durante a execução da aplicação e só são computadas quando necessário. Já as ações, são executadas de imediato e produzem um resultado diferente da figura do RDD. Essa postergação das transformações viabiliza a criação do plano de execução que, como mencionado anteriormente, permite uma análise prévia do que será executado e a introdução de otimizações. Quando uma ação é encontrada no código, o Spark envia o DAG para o DAG Scheduler, o qual, divide as operações em estágios compostos por tarefas. Esses estágios são organizados pelo Task Scheduler, que negocia a execução das tarefas diretamente com o Cluster Manager uma vez que os recursos computacionais necessários estejam disponíveis.

Figura 3 - Arquitetura do Spark e sumário do DAG.



Fonte: Directed Acyclic Graph DAG in Apache Spark por Data Flair (2017).

O Spark SQL é um dos principais módulos do Spark. Ele integra o framework de Big Data e atua na execução de consultas SQL sob conjuntos de dados que possam ser acessados pelo Spark, estejam eles contidos em arquivos batch, formatados em JSON ou tabelas do HIVE, por exemplo. Por meio de um outro componente de software, denominado Conector Spark-HBase [34] e programado pela empresa Hortonworks, se torna possível usufruir das funcionalidades do Spark SQL com dados que estejam armazenados no HBase. O conector implementa múltiplas otimizações como localidade de referência, *partition pruning*, *predicate pushdown* e outras mais que tornam a integração entre os *softwares* coesa e menos custosa.

Quando instâncias do HBase e Spark estão acomodadas nos mesmos nós de um cluster, na forma de RegionServers e Workers, respectivamente, o Conector Spark-HBase consegue implementar a otimização de localidade de referência. Isto significa que ele é capaz de identificar em quais nós os dados demandados pelas consultas estão armazenados e consegue delegar a execução das etapas da consulta somente para tais nós, ou seja, cada nó irá processar majoritariamente os dados que estejam armazenados nele mesmo para evitar o tráfego desnecessário. Além disso, sempre que é possível, o conector se vale da técnica de *predicate pushdown* para extrair partes do predicado de consultas, que possam ser traduzidas em funções nativas do HBase, e executa essas operações via o banco de dados para serem ainda mais otimizadas.

O Spark SQL e o Conector Spark-HBase podem ser integrados sem afetar funcionalidades que estão disponíveis em outros módulos para a integração com aplicações do Spark. É o caso da biblioteca batizada de GeoSpark que é capaz de processar dados geoespaciais em larga escala [35]. Ela é feita para a execução de consultas SQL *ad-hoc* contendo predicados geoespaciais. Os dados geográficos podem ser importados em múltiplos formatos como os WKT, WKB, GeoJSON, Shapefile e HDF que estejam ao alcance do Spark. São aceitas geometrias de pontos, linhas, círculos e polígonos, além de coleções delas. São disponibilizadas funções já tradicionais de outras *engines* geográficas como para o cálculo de áreas, perímetros e distâncias bem como junções baseadas nos predicados dos tipos contém, intersecta e sobrepõe, por exemplo.

2.3 DATA WAREHOUSE

O processo de decisão nas grandes corporações comumente envolve a coleta e análise dos dados operacionais e de terceiros por longos períodos de tempo. Avanços tecnológicos alinhados à nova ubiquidade virtual no cotidiano das pessoas contribuíram com o rápido crescimento do volume de dados disponíveis em meios virtuais. Isto possibilitou o surgimento de aplicações computacionais que valem-se do volume, velocidade e variedade desses dados para a extração de conhecimentos únicos que possam ser empregados na tomada de decisão [36]. Essa matéria prima é composta de dados heterogêneos, advindos das mais diversas fontes e obtidos nos mais variados formatos que precisam ser armazenados de maneira apropriada para que possam ser consultados por analistas, cientistas e gestores, além de alimentarem outras aplicações como as de inteligência artificial e as de aprendizagem de máquina. Analisando as características e as finalidades pretendidas com esses sistemas, vemos indícios de que abordagens transacionais não sejam a alternativa mais apropriada para a implementação deles, assim, existe uma demanda para investigar quais abordagens são as mais adequadas e que tecnologias de bancos de dados irão prover os melhores desempenhos.

As consultas realizadas em ambientes tipicamente transacionais são, majoritariamente, operações simples, repetitivas, isoladas e atômicas que acessam poucos registros de cada vez e ocorrem com finalidades operacionais [37]. Os sistemas que processam regularmente este tipo de consultas são chamados de OLTP (On-Line Transaction Processing). Devido às finalidades operacionais, eles têm como prioridade a disponibilidade, velocidade, concorrência e a recuperação de falhas. Sistemas que lidam com vendas, controle de estoque e caixas eletrônicos são bons exemplos de ambientes transacionais com predominância desse tipo de consulta. Existe ainda uma outra classe com carga de consultas mais complexas tipificadas por envolverem muitas junções, agregações e a leitura de milhões de registros por vez. São as consultas analíticas que servem o propósito de analisar dados multidimensionais interativamente sob perspectivas variadas. São as consultas do tipo OLAP cujos sistemas privilegiam o armazenamento mais eficiente e inteligente com o propósito de otimizar a execução de consultas complexas e ad-hoc em tempos de processamento mais baixos.

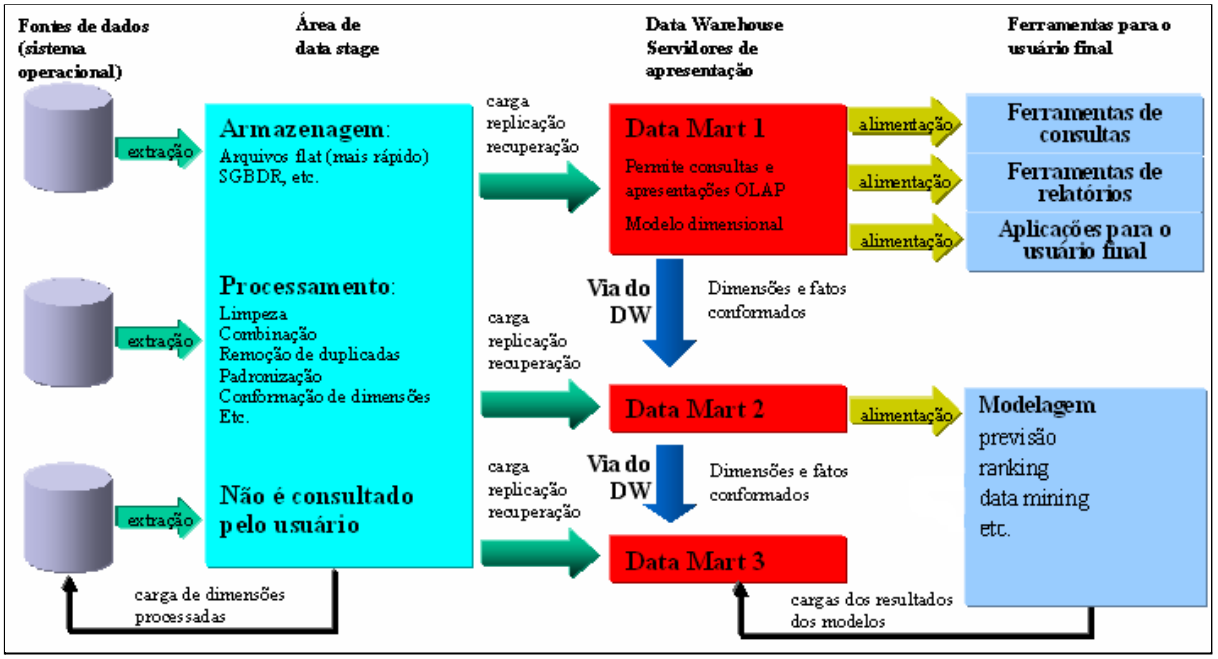
O *data warehouse* (DW) é um sistema computacional que atua como um depósito de informações orientado por assunto, integrado, não volátil e histórico, cujo objetivo principal é apoiar a tomada de decisão suportando o funcionamento de aplicações OLAP [5]. Os dados acumulados fornecem informações consolidadas sobre tópicos pertinentes à parte operacional de empresas (i.e. vendas, logística, marketing), mas não sobre a execução propriamente dita dessas atividades. Esses dados advém de integrações com os SGBDs responsáveis por operacionalizar o negócio no dia a dia ou de qualquer outra fonte cujos dados possam ser integrados de forma coerente, a fim de, aprimorar o processo de decisão. No ambiente de um *data warehouse*, os dados, após inseridos, sofrem pouca ou nenhuma atualização. As informações são acumuladas ao longo do tempo para a construção do histórico de atuação da companhia por horizonte temporal superior ao dos SGBDs convencionais.

Os componentes fundamentais que compõem a arquitetura de um data warehouse são ilustrados na Figura 4, são eles: I) fonte de dados, que pode ser uma composição de informações coletadas dos bancos transacionais da empresa com dados obtidos de terceiros; II) *softwares* que executam a extração, transformação e carga dos dados no DW, procedimento que é conhecido pelo acrônimo na língua inglesa de ETL; III) repositório de dados, ou *data mart* em inglês, que armazenam dados individualizados por tópico; IV) repositório de metadados, os quais, atuam como facilitadores para a realização de consultas através do fornecimento de características do banco e de seu respectivo esquema de dados; V) Ferramentas para a mineração, visualização e a consulta análitica.

O planejamento, desenvolvimento e as tecnologias envolvidas na criação das partes que compõem um *data warehouse* impactam diretamente no seu desempenho. No contexto deste trabalho, nós destacamos a escolha da tecnologia do banco de dados e o projeto do esquema dele como fatores muito importantes que afetam a quantidade de memória necessária para o armazenamento dos dados e o tempo requerido para a execução de consultas. Tecnologias mais consolidadas como as de SGBDs relacionais podem ser utilizadas na construção de *data warehouses* mas, de acordo com características do projeto (i.e. tamanho da base de dados), abordagens NoSQL e NewSQL viraram alternativas mais atrativas por

favorecerem ganhos de desempenho através da escalabilidade com paralelismo e menos sobrecargas no controle de consistência, integridade e transacional.

Figura 4 - Arquitetura de data warehouses.



Fonte: Retirado do artigo intitulado Data Warehouse publicado no Blog Dev Media (2009).

Independentemente do tipo de banco de dados, o projeto de esquemas para DWs costuma adotar os métodos, técnicas e conceitos da modelagem dimensional [38]. Ela gira em torno de dois tipos de tabelas, as de fatos e dimensões. Os fatos compreendem principalmente atributos numéricos que quantificam alguma métrica do negócio. Na tabela de fatos, além deles, ainda existem chaves estrangeiras que relacionam um dado fato a atributos descritivos que, por sua vez, são guardados nas tabelas de dimensões. Essas tabelas podem ser organizadas de três formas distintas, chamadas de esquemas estrela, floco de neve e constelação. No esquema dimensional do tipo estrela, temos a composição mais simples onde existe uma única tabela de fatos que contém referências para as dimensões. Já no floco de neve, além de fatos que referenciam dimensões, podem existir relações entre as próprias dimensões. Por fim, no esquema constelação, existem múltiplas tabelas de fatos que não se relacionam mas compartilham das mesmas dimensões.

Os esquemas estrela, floco de neve e constelação caracterizam o uso da modelagem dimensional na definição de esquemas para *data warehouse*s através dos relacionamentos estabelecidos entre as tabelas existentes. No entanto, além dos relacionamentos, as diferenças entre esquemas podem ser postuladas em função de seguirem abordagens normalizadas ou desnormalizadas. Seguir uma abordagem mais normalizada significa que o modelo preza pela diminuição de redundâncias com a contrapartida do aumento de referências, resultando em menos demanda por armazenamento e na possível ocorrência de mais junções, por exemplo. Já abordagens desnormalizadas prezam pela substituição de referências por cópias dos dados referenciados, ou seja, as chaves estrangeiras dão lugar aos dados originais trazidos de suas respectivas tabelas. Essas cópias sucessivas eliminam as custosas junções mas aumentam a quantidade de memória necessária para o armazenamento dos dados, tornam mais custosa a atualização de registros e dificultam o controle de consistência.

2.4 SPADAWAN

O Spadawan (Spatial Data Warehouse Benchmark) é um padrão proposto por pesquisadores para uniformizar a análise do desempenho de data warehouses geoespaciais através de procedimentos experimentais controlados que mensurem os custos para a junção de grandes tabelas de fatos com grandes tabelas de dimensões, execução de consultas contendo predicados geoespaciais e para a agregação de dados com base na granularidade de algum dado geográfico [21]. Ele deriva de outro benchmark muito popular chamado SSB (Star Schema Benchmark) [39]. Foram incluídos atributos geoespaciais (e.g. ponto, círculo, polígono) com base em dados textuais já existentes no SSB e que remetem a hierarquias geográficas (i.e. cidade, país, região). As principais contribuições do estudo foram a definição de consultas geoespaciais envolvendo as operações *roll-u*p e *drill-down* além da base de dados que pode ser gerada com base em um fator de escala, modificada e utilizada como parâmetro em análises de desempenho baseadas na execução da carga de consultas.

O esquema proposto no SSB segue o modelo dimensional estrela com uma única tabela de fatos denominada *lineorder*, a qual, congrega informações oriundas

de um sistema de vendas em que são registrados dados da compra, datas, produtos, clientes e fornecedores. Os atributos das quatro últimas entidades mencionadas são registrados em dimensões chamadas de *date*, *part*, *customer* e *supplier*, referenciadas por chaves estrangeiras contidas na tabela de fatos. Nas tabelas de clientes e fornecedores, existem atributos textuais que denotam endereços, cidades, nações e regiões dos registros. O Spadawan reutilizou as hierarquias existentes entre tais atributos para gerar polígonos denotando geometrias de regiões, polígonos das nações inseridas em cada região, polígonos das cidades de cada nação e pontos aleatórios representando endereços dentro de uma cidade. Esses são os atributos disponíveis na base para a execução de consultas com predicados geoespaciais.

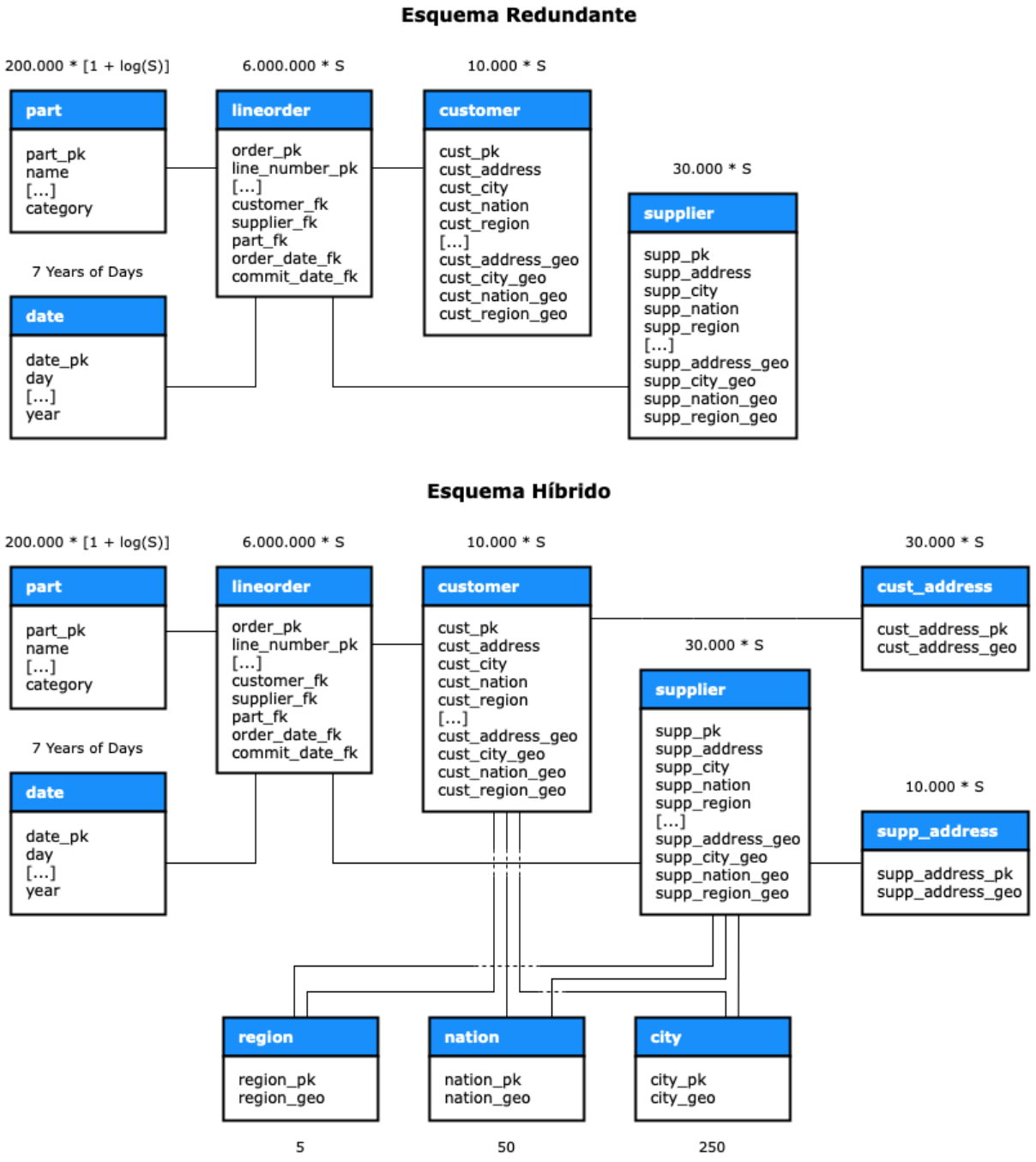
O primeiro esquema proposto pelos autores do Spadawan é denominado de Esquema Redundante e está ilustrado na Figura 5, nele, vemos que os atributos geográficos foram incluídos diretamente nas dimensões *customer* e *supplier* com os sufixos *geo*. Tal modelo é dito redundante em razão dos atributos geográficos acabarem sendo replicados em todos os registros de clientes e fornecedores da base de dados. Por exemplo, a geometria da região Europeia acabará sendo replicada em todas as ocorrências de clientes ou fornecedores situados no continente. Isto aumenta a quantidade de memória requerida para o armazenamento dos dados e também o número de registros analisados por consulta geoespacial, uma vez que, o predicado precisará ser verificado em um número maior de registros. Diante dessas desvantagens, foi proposto um novo esquema chamado de Híbrido com modelagem floco de neve em que os atributos geográficos são dispostos em dimensões separadas e passam a ser referenciados. Ele remedia os problemas anteriores mas incorre na necessidade de serem realizadas custosas junções entre as tabelas.

A geração da base de dados, conforme as proposições do Spadawan, é realizada por um programa de computador chamado *dbgen*. Ele gera as tabelas do SSB com dados sintéticos de acordo com o fator de escala, que serve para controlar a quantidade de registros populados em cada tabela, e só depois, um *software* proprietário do Spadawan é executado para integrar os dados convencionais com os geográficos. Observe que na Figura 5, acima de cada entidade, existe uma fórmula

que faz o cálculo do número de registros contidos em cada linha em função da variável *S* que representa o fator de escala. As tabelas de cidades, nações e regiões são criadas com números fixos de registros e não variam com o fator de escala. A base de dados sempre terá cinco regiões, cinco nações por região e dez cidades por nação, totalizando cinco, vinte e cinco e duzentos e cinquenta registros ao todo na base, respectivamente. Esses três atributos geográficos são tratados pelo Spadawan como geometrias de polígonos no formato WKT. Já os endereços, são representados através de pontos distribuídos aleatoriamente dentro das geometrias das cidades, sendo atribuídos unicamente aos registros de clientes e fornecedores.

Foram propostos quatro tipos de consultas, agrupadas em duas categorias. As consultas 1, 2 e 3 se baseiam na proposta Q2.3 do SSB e avaliam os seguintes aspectos: I) no mínimo, três junções entre tabelas; II) quatro predicados geoespaciais baseados em janelas *ad-hoc*; e III) agregação de dados de acordo com quatro níveis de granularidade geográfica. A consulta 4, única do segundo grupo, é uma adaptação da consulta Q3.3 do SSB e foi elaborada para conter dois predicados geoespaciais que avaliam duas janelas espaciais *ad-hoc*, adicionando mais complexidade. Os autores trocaram os parâmetros nominativos contidos nas duas consultas do SSB, Q2.3 e Q3.3, por predicados tendo operações geoespaciais dos tipos *interseção*, *contém* e *engloba*. Elas foram utilizadas em conjunto com os campos endereço, cidade, nação e região para execução das operações *drill-down* e *roll-up* com base na granularidade desses dados.

Figura 5 - Esquema redundante e esquema híbrido.



Fonte: Adaptado de Siqueira et al. Benchmarking Spatial Data Warehouses (2010).

3 REVISÃO DA LITERATURA

Neste capítulo, apresentamos estudos científicos que julgamos relevantes para a contextualização desta pesquisa por tratarem do projeto do modelo de dados para bancos NoSQL, em especial, os do tipo *wide-column*, e, da análise de desempenho em tais ambientes, independentemente, delas utilizarem dados convencionais ou geoespaciais. Abrimos a primeira seção do capítulo delineando a metodologia adotada para a identificação dos trabalhos relacionados e quais critérios utilizados para a inclusão ou exclusão dos mesmos em nossa lista de referências (Seção 3.1). Em seguida, nós destacamos as premissas, abordagens e conclusões dos estudos, os quais, categorizamos em três grupos com base na abordagem adotada (Seção 3.2). A subseção seguinte encerra o capítulo com um panorama dos estudos elencados e discute um último trabalho relacionado que inspirou esta pesquisa mas foi desenhado para bancos NoSQL orientados a documentos (Seção 3.3).

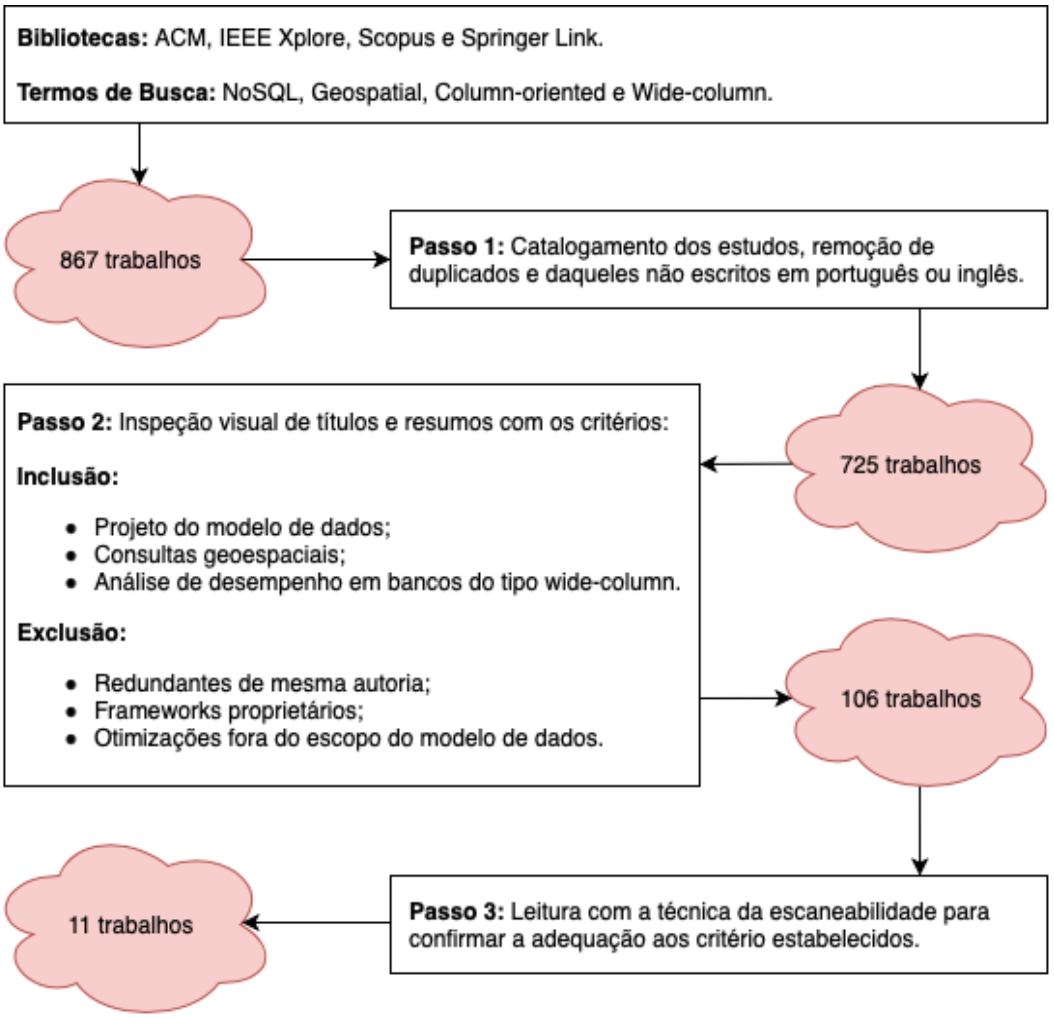
3.1 MÉTODO DE BUSCA E CRITÉRIOS DE ANÁLISE

Conjecturado um espaço na literatura e contextualizado o escopo do projeto de pesquisa, demos início a revisão bibliográfica através da busca por artigos científicos relativos ao tema e listados nas mais importantes bases acadêmicas. Elencamos os termos *NoSQL*, *Geospatial*, *Column-oriented* e também *Wide-column* como palavras chave a serem pesquisadas nas bases Scopus, Springer Link, IEEE Xplore e ACM Digital Library. Combinando tais termos, obtivemos um total de 867 resultados entre as fontes mencionadas. Desses, restaram 725 após a exclusão de resultados duplicados e dos que não estivessem disponíveis nos idiomas português ou inglês.

Os trabalhos acadêmicos que passaram no primeiro crivo, ainda muito numerosos, foram filtrados através da inspeção visual de seus títulos e resumos. Dos 725 iniciais, selecionamos 106 produções que indicavam tratar do projeto do modelo de dados para bancos wide-column, emprego deles na execução de consultas geoespaciais ou em análises de desempenho. Com essa redução expressiva do número de trabalhos, realizamos a leitura dos mesmos utilizando a

técnica da escaneabilidade para confirmar a conexão do conteúdo com os critérios adotados. O resultado desta última etapa foi a seleção de 11 artigos científicos altamente relevantes para nosso estudo. O processo de busca que fora descrito acima está ilustrado na Figura 6 e os trabalhos serão debatidos a seguir.

Figura 6 - Fluxograma do processo de revisão literária.



Fonte: autor (2022).

3.2 TRABALHOS RELACIONADOS

A lista final com os 11 artigos selecionados é apresentada na Tabela 1. Eles foram considerados de alta relevância em função de abordarem distintivamente aspectos do projeto do modelo de dados para os bancos do tipo *wide-column* e analisarem o desempenho de suas proposições através de testes experimentais. Com fins didáticos, categorizamos os artigos em três grupos de acordo com o

enfoque primário de cada um, são: I) trabalhos focados na proposta e análise de modelos criados a partir de noções arquiteturais dos bancos NoSQL e *wide-column*; II) aqueles com as mesmas metas dos primeiros mas que se valeram de algoritmos inteligentes para gerar modelos otimizados para a carga de consultas típica; III) finalmente, trabalhos focados na execução e otimização de consultas geoespaciais em bancos wide-column através de técnicas conhecidas como, por exemplo, a técnica de *geohashing* que possibilita a indexação de dados geoespaciais.

Tabela 1 - Lista dos trabalhos relacionados mais relevantes.

Nº	**Ref**	**Ano**	**Título**
1	13	2010	Transforming Relational Database into HBase: A Case Study
2	14	2014	Columnar NoSQL Star Schema Benchmark
3	15	2015	Using the Column-oriented NoSQL Model for Implementing Big Data Warehouses
4	16	2015	Implementation of Multidimensional Databases in Column-oriented NoSQL Systems
5	17	2016	Physical Data Warehouse Design on NoSQL Databases
6	10	2015	An Evolutionary Algorithm for Column Family Schema Optimization in HBase
7	11	2017	Logical Schema for Data Warehouse on Column-oriented NoSQL Databases
8	12	2018	A Data Partition Optimization Approach for Distributed Data Warehouses on Column Family NoSQL Systems
9	18	2020	State-of-the-Art Geospatial Information Processing in NoSQL Databases
10	19	2013	HGrid: A Data Model for Large Geospatial Data Sets in HBase
11	20	2019	Evaluating Redundancy and Partitioning of Geospatial Data in Document-oriented Data Warehouses

Fonte: autor (2022).

3.2.1 Grupo I: Abordagens Primárias

O primeiro grupo de trabalhos engloba cinco estudos científicos [13-17], os quais, foram elencados na tabela com os números de 1 até 5. Cada um deles propõe, pelo menos, uma abordagem para a concepção de modelos de dados para bancos do tipo wide-column. Eles desconsideram a presença de dados geoespaciais nas bases de dados mas, mesmo que tenham sido criados exclusivamente para dados convencionais, suas motivações, considerações e conclusões são muito relevantes para o nosso estudo. Afinal, os dados convencionais continuam constituindo parte expressiva das bases de dados que contenham dados geográficos e, além disso, podemos extrapolar as lições debatidas nesses estudos para os cenários que envolvam dados geoespaciais.

O artigo que encabeça a lista [13], discorre sobre um estudo de caso para o qual é proposta uma heurística para transformar modelos relacionais em modelos próprios para os bancos de dados do tipo *wide-column*. Foram especificadas três regras de transformação, são elas: I) Dados correlatos que compartilham dos mesmos padrões de escrita e leitura devem ser dispostos na mesma família de colunas, pois, a proximidade física dos registros dará margem para a otimização das entradas e saídas do disco e memória, além de um melhor aproveitamento das *caches*; II) Chaves estrangeiras devem ser incluídas em ambos os lados de um relacionamento com cardinalidade N:M para evitar a criação de tabelas intermediárias e junções; III) A terceira e última regra prega a combinação de tabelas ditas dependentes com aquelas da qual dependem. A dependência é caracterizada quando uma tabela somente é utilizada em conjunto com outra e só referencia a tal. É recomendado que a tabela dependente seja convertida em uma estrutura de dados (i.e. JSON, XML) que seja guardada em algum campo da tabela principal, evitando as custosas junções.

As regras propostas no primeiro trabalho expressam as preocupações dos projetistas em distribuir dados apropriadamente entre famílias de colunas e modificar relacionamentos a fim de otimizar o uso das memórias e reprimir junções, respectivamente. O trabalho seguinte [14], publicado 4 anos após o predecessor, endereça tais preocupações com a proposição de um benchmark denominado de

CNSSB (Columnar NoSQL SSB) para os bancos *wide-column*. Ele deriva da contraparte relacional chamada de Star Schema Benchmark (SSB) [39]. O CNSSB propôs uma abordagem desnormalizada que congregava todos os dados da base na mesma tabela de fatos, dispondo os atributos de cada dimensão em famílias de colunas distintas. Tal modelo eliminou as junções, que são custosas e em muitos casos não são implementadas nativamente por bancos NoSQL, mas provocou um aumento expressivo do armazenamento necessário, em razão da alta redundância de dados.

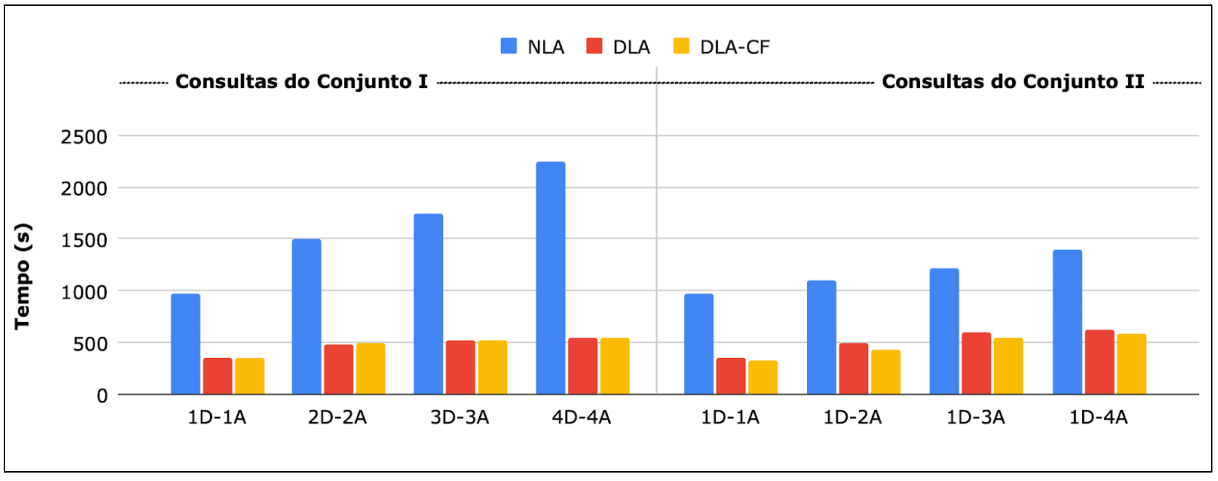
Após um ano, os autores proponentes do CNSSB publicaram a continuação do estudo [15] em que introduziram três abordagens para instanciar modelos lógicos de bancos wide-column partindo do modelo conceitual do SSB. São elas: I) *Normalized Logical Approach* (NLA), na qual, cada uma das tabelas existentes no modelo conceitual origina uma tabela com uma família e os relacionamentos entre tabelas são mantidos através da criação de colunas contendo as chaves de linha dos registros que precisam ser referenciados; II) *Denormalized Logical Approach* (DLA) onde os dados acessíveis por meio de referências são replicados na tabela da chave estrangeira até que só reste uma tabela com fatos e todos os demais dados dimensionais contidos nela; III) *Denormalized Logical Approach with Column Families* (DLA-CF) é uma abordagem similar a anterior mas, nela, os dados replicados de cada dimensão são dispostos em famílias de colunas distintas de acordo com a origem.

Os modelos DLA e DLA-CF seguem abordagens desnormalizadas e foram concebidos com o intuito de avaliar os impactos da desnormalização nos custos de armazenamento e execução de consultas para determinar se eles são compensados por eventuais ganhos de desempenho advindos da não realização de junções, uma vez que, todos os dados residem na tabela de fatos. O DLA-CF separa os dados oriundos de cada uma das dimensões em famílias de colunas distintas a fim de promover a separação física dos dados, objetivando eliminar sobrecargas no gerenciamento de memória que poderiam advir do processamento de dados não que foram efetivamente solicitados. O NLA reside no outro extremo dos dois primeiros, seguindo uma abordagem normalizada com os dados sendo majoritariamente referenciados e contendo pouquíssima redundância. Apesar disso,

ele implica na necessidade de mais junções e sofre com as penalidades para a execução delas.

O mesmo estudo que introduziu os modelos NLA, DLA e DLA-CF, realizou testes experimentais para avaliar como as três abordagens impactam o tempo demandado para a execução de uma carga de consultas analíticas em um data warehouse construído sob o HBase. Os testes foram realizados em um *cluster* computacional com 25 máquinas configuradas para a execução da stack composta pelos *softwares* Hadoop, HBase e Phoenix. Este último faz parte do ecossistema Apache e proporciona uma interface para a execução de consultas SQL. Após os dados terem sido gerados com um fator de escala equivalente a 1.000 vezes o tamanho padrão do SSB e serem inseridos no HBase, foram submetidas 8 consultas SQL através do Phoenix e coletados seus tempos de execução. As consultas podem ser subdivididas em dois conjuntos, onde, as do primeiro, abrangem um número crescente de dimensões e leem um único atributo por dimensão e, as do segundo, envolvem uma dimensão por vez mas com um número crescente de atributos lidos por consulta.

Figura 7 - Tempos de execução do NLA, DLA e DLA-CF.



Fonte: Adaptado de Dehdouh et al. Using the Column Oriented NoSQL Model for Implementing Big Data Warehouses (2015).

A Figura 7 ilustra um gráfico de barras verticais com os tempos de execução das consultas quando utilizados os modelos NLA, DLA e DLA-CF. As etiquetas ao longo do eixo das abscissas identificam as consultas através da formatação ***X****D*-***Y****A*, onde *X* denota o número de dimensões envolvidas e *Y* a quantidade de atributos requisitados. A análise dos resultados conduziu os autores a conclusão de que as abordagens desnormalizadas são muito superiores às normalizadas. Elas demandam espaço de armazenamento superior, são altamente redundantes e podem ocasionar a análise de número maior de registros para a determinação de predicados mas, ainda assim, os resultados corroboram com a visão de que o custo das junções em modelos normalizados são mais proibitivos. Já o páreo entre os modelos DLA e DLA-CF mostrou um ganho marginal de desempenho em favor do DLA-CF quando as consultas envolviam atributos da mesma dimensão, diferença que pode ser justificada pela diminuição do número de famílias processadas para a obtenção dos atributos.

O quarto trabalho da lista [16] foi publicado em data muito próxima ao anterior e introduziu ideias similares. Nele, foram apresentados três modelos de dados denominados MLC0, MLC1 e MLC2 que correspondem aos modelos DLA, DLA-CF e NLA, respectivamente. No entanto, o experimento feito no estudo comparou os desempenhos de cada modelo com base no tempo decorrido para geração dos eixos de um cubo OLAP. A base de dados foi gerada a partir do SSB para os fatores 1, 10 e 100 que serviram para a comparação dos tempos de carga e armazenamento exigidos, mas os testes só utilizaram o primeiro fator. Foram gerados cubos para todas as combinações possíveis de uma até três dimensões, incluindo dados descritivos e resultados das funções de agregação max, min, sum e count para todas as medidas. Bem como no estudo anterior, os resultados do teste experimental evidenciaram o melhor desempenho das abordagens desnormalizadas no tocante aos tempos de execução, assim, apresentamos apenas os demais dados, logo abaixo, na Tabela 2.

Tabela 2 - Tempo de carga e o volume do NLA, DLA e DLA-CF.

		NLA		DLA		DLA-CF	
SF[0]	**Nº Linhas**	**Tempo[1]**	**Tamanho[2]**	**Tempo[1]**	**Tamanho[2]**	**Tempo[1]**	**Tamanho[2]**
1	10^7	264	0,997	380	3,90	402	3,90
10	10^8	2.765	9,970	3.458	39,00	3.562	39,00
100	10^9	33.097	99,700	39.075	390,00	39.716	390,00

Fonte: Adaptado de Chevalier et al. Implementation of Multidimensional Databases in Column-Oriented NoSQL Systems (2015).

[0] fator de escala, [1] tempo em segundos (s), [2] tamanho em gigabytes (GB)

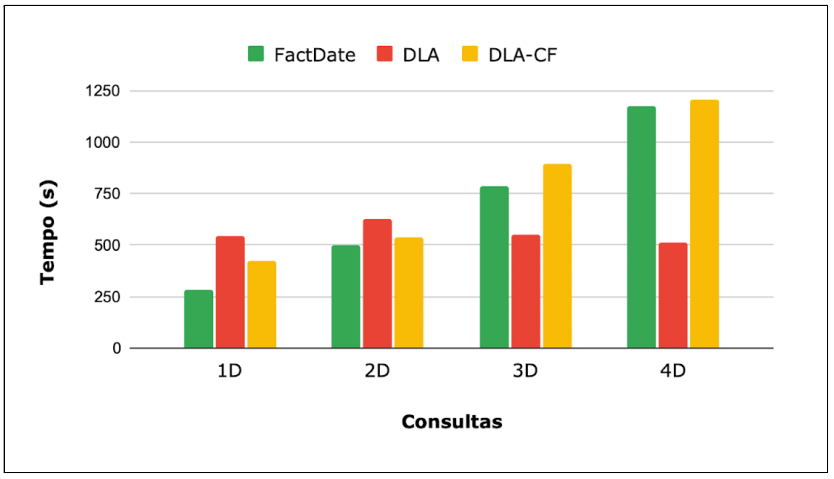
Como esperado, em razão da baixa redundância, o NLA demanda menos espaço do que os demais e também menos tempo de carga. Isto é justificado pela abordagem normalizada que pulveriza os dados entre tabelas e linhas, onde cada registro acaba ocupando menos espaço e sendo inserido mais rapidamente no banco. Comparando os outros dois modelos, o DLA e DLA-CF, vemos que as diferenças são inexpressivas. Apesar dos dados não refletirem, o DLA-CF ocupa um pouco mais de espaço por conta do maior número de famílias, que geram mais HFiles e estruturas de dados para a organização delas. Por conta disso, ele também demanda mais tempo de carga.

A última pesquisa disposta neste grupo [17] propôs uma abordagem para a distribuição de colunas entre famílias. Os autores argumentaram que as dimensões mais recorrentes entre os atributos da carga de consultas típica deveriam ter seus dados alocados na mesma família dos fatos, ajudando a manter o número de famílias baixo e evitando sobrecargas oriundas da abertura de vários arquivos do tipo HFile. Para ilustrar tal linha de raciocínio, eles apresentaram o modelo FactDate que segue uma abordagem desnormalizada e congrega, na mesma família de colunas, fatos e os atributos da dimensão tempo. O FactDate pode ser visto como um intermediário entre o DLA e o DLA-CF, uma vez que, ele começa com o mesmo número de famílias do DLA-CF mas elimina aquelas cujos dados estão mais

presentes nas cargas de consultas, incorporando as colunas das mesmas na família de fatos e, portanto, promovendo a redução da quantidade de famílias na tabela.

O modelo FactDate foi submetido a testes experimentais juntamente com o DLA e o DLA-CF para a avaliação e comparação dos seus respectivos desempenhos. Foi utilizada a carga de treze consultas originalmente propostas no SSB para avaliar os tempos de execução. Como elas abrangiam apenas uma, três e quatro dimensões, foram propostas consultas adicionais envolvendo duas dimensões para avaliar por completo os efeitos da variação do número de dimensões. Dentre as treze, foram selecionadas quatro para comporem o gráfico de barras apresentado na Figura 8, pois, as variações entre os tempos de consultas com o mesmo número de dimensões foram irrisórias. No gráfico, nós podemos visualizar que as consultas envolvendo uma ou duas dimensões se saíram melhores com os modelos FactDate e DLA-CF, que possuem um número maior de famílias. Para consultas tendo atributos de três ou quatro dimensões, o modelo DLA, que possui uma única família, se saiu muito melhor do que os demais. Estas observações vão de encontro as dos estudos anteriores, tem-se a percepção de que o tempo para a abertura e leitura dos arquivos de cada família traz um impacto notório no tempo de execução das consultas.

Figura 8 - Tempos de execução do DLA, DLA-CF e FactDate.



Fonte: Adaptado de Scabora et al. Physical Data Warehouse Design on NoSQL Databases (2016).

Apresentados os cinco primeiros trabalhos relacionados, atribuídos ao primeiro grupo em razão de suas similaridades, agora, podemos sumarizar suas proposições nos modelos NLA, DLA, DLA-CF, além do FactDate, que retratam uma dicotomia entre abordagens normalizadas e desnormalizadas acompanhadas de modificações pontuais para avaliar o papel das famílias no desempenho final do sistema analítico. Por um lado, a normalização gera modelos com pouca redundância, que exigem menos espaço de armazenamento, menos tempo de carga e podem ser atualizados de forma mais fácil mas incumbem na realização das custosas junções. Já a desnormalização, acaba eliminando as junções, mas gera muita demanda de armazenamento e tem o potencial de elevar a quantidade de registros avaliados na determinação de predicados por conta da redundância. O projeto do modelo de dados não precisa tender para tais extremos, abordagens totalmente normalizadas ou aquelas totalmente desnormalizadas, mas pode figurar entre um meio termo delas, onde sejam definidas regras para estabelecer quando e como é mais proveitoso normalizar ou desnormalizar.

3.2.2 Grupo II: Abordagens Inteligentes

Conforme tratado acima, os artigos inclusos no primeiro grupo propõem modelos que extrapolam conhecimentos sobre os bancos de dados NoSQL, fundamentos da arquitetura dos wide-column e aprimoramentos de estudos antecedentes. Nesta seção, nós apresentamos três artigos [10-12] que abordam a mesma problemática com nova ótica. Eles almejam promover ganhos de desempenho através da distribuição inteligente de colunas entre um número arbitrário de famílias, empregando algoritmos capazes de encontrar distribuições que minimizem os custos da execução de uma carga de consultas típica, a qual, é conhecida previamente e alimenta tais algoritmos.

Em [10] os autores partem do pressuposto de que, como os bancos *wide-column* dividem os dados entre famílias de colunas e essas são separadas fisicamente em arquivos, o tempo para a execução de uma consulta será diretamente proporcional ao número de famílias que estejam relacionadas entre os atributos envolvidos na consulta. Mais famílias implicam na abertura de mais arquivos, no carregamento das estruturas de dados contidas neles e também no

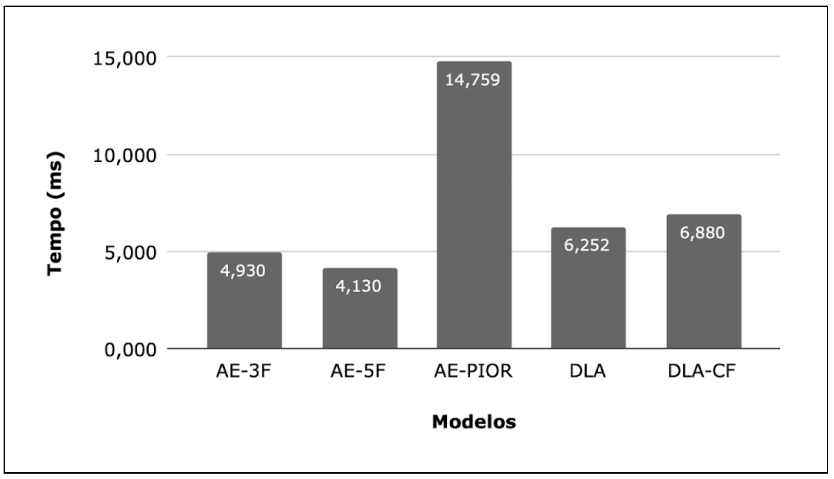
aumento da pressão realizada sob o disco, memória principal e caches. Essas consequências degradam o desempenho do sistema e, por isso, destacam a importância do número de famílias de colunas como parâmetro importante na construção dos modelos de dados. Apesar de tais conclusões, reduzir a quantidade de famílias indiscriminadamente renega as vantagens inerentes aos bancos orientados a colunas e pode afetar negativamente o desempenho. Assim, o estudo postulou o número de famílias como um problema de otimização, no qual, se deseja encontrar a quantidade ideal para potencializar ganhos e mitigar perdas de desempenho durante a execução das cargas de consultas.

Sabendo que tal problema de otimização é NP-difícil, a solução proposta pelos autores foi o uso de um algoritmo evolucionário capaz de encontrar configurações que se aproximam da ideal mas com tempo de processamento hábil. Os algoritmos evolucionários resolvem problemas de otimização global utilizando técnicas adaptadas da biologia evolutiva. Eles geram soluções para o problema de forma iterativa e as codificam na forma de indivíduos de uma população. A cada iteração, mede-se a qualidade das soluções e, as melhores, passam por processos de transformação para produzirem os novos indivíduos da iteração seguinte. A função de *fitness* mede a qualidade de cada indivíduo e é crucial para viabilizar o processo evolutivo. No contexto deste trabalho, ela foi desenvolvida para maximizar o número de famílias, igualar a quantidade de colunas atribuídas a cada família, reduzir o número de colunas duplicadas e minimizar o número de famílias demandadas por consulta.

Os testes experimentais para avaliar modelos otimizados pelo algoritmo evolucionário utilizaram a base de dados ZANOX AG, que possui mais de 2 milhões de registros de interações entre usuários e anúncios veiculados em páginas da internet. A base conta com 1,3 milhões de consultas feitas por pessoas reais que, em algum momento, buscaram extrair informações dela. Os dados e consultas alimentaram o algoritmo para que ele pudesse calcular a função de *fitness* e, a partir dele, foram selecionados dois modelos bem qualificados, com 3 e 5 famílias. Além deles dois, os testes levaram em consideração os modelos DLA, DLA-CF e mais um tendo a pior qualificação com base na função de *fitness*. Os cinco modelos foram inseridos em um cluster computacional e testados contra uma carga de consultas

para que os tempos de execução fossem coletados. A Figura 9 mostra o gráfico de barras comparando os tempos médios para a execução da carga de consultas quando utilizados dois modelos otimizados, o pior modelo possível segundo a função de *fitness* e os DLAs.

Figura 9 - Tempos de execução algoritmos evolucionários.



Fonte: Adaptado de Fangzhou Yang et al. An Evolutionary Algorithm for Column Family Schema Optimization in HBase (2015).

A análise dos dados mostra que os dois modelos otimizados pelo algoritmo (AE-3F e AE-5F) são os destaques positivos entre os pares, em especial o com 5 famílias que superou o segundo colocado com redução superior a 15%. É importante mencionar que, assim como o AE-5F, o modelo DLA-CF também possui cinco famílias de colunas, mas tem desempenho inferior em função da distribuição adotada e da possibilidade de colunas serem replicadas em famílias distintas. O AE-PIOR foi gerado a partir da pior qualificação passível de ser gerada pela função de *fitness*, não foram apresentados detalhes sobre ele. No entanto, apesar desta abordagem parecer promissora e definitiva, é preciso atentar ao fato de que ela só é viável porque a carga de consultas era conhecida previamente. Nos ambientes analíticos mais tradicionais, obter tal carga não é trivial e vai de encontro a expectativa de que as consultas realizadas nesses meios são *ad-hoc* e não podem ser pré-estabelecidas.

Os outros dois trabalhos selecionados que se valem de algoritmos inteligentes [11-12] incorporam as mesmas motivações e os objetivos do primeiro,

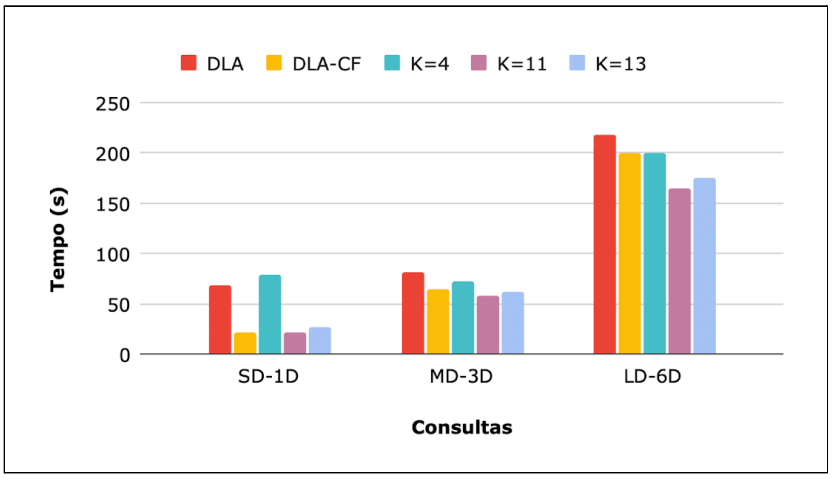
mas trilham estratégias diferentes. Em [11], foi utilizado o método de clusterização *k-means* para agrupar colunas em clusters, os quais, denotam as novas famílias de colunas. O algoritmo funciona de forma iterativa, avaliando as distâncias entre colunas e agrupando aquelas com maior proximidade. No caso, as proximidades foram calculadas com base em uma matriz de afinidades relacionando as consultas com as colunas especificadas em cada uma delas. Já em [12], o mesmo grupo de pesquisadores utilizou o algoritmo de clusterização *k-medoids* para primeiro agrupar consultas similares e depois distribuir as colunas em famílias com o algoritmo evolucionário PSO (Particle Swarm Optimization).

Os experimentos conduzidos nos dois últimos estudos citados utilizaram o benchmark TPC-DS que possui um total de 99 consultas na carga proposta. Ambos selecionaram um número de consultas menor do que a totalidade disponível e dividiram tal subconjunto em três categorias diferentes, de acordo com a quantidade de dimensões abrangidas por cada uma delas. A primeira tinha consultas com uma ou duas dimensões, a segunda com três ou quatro e a terceira com cinco ou seis, sendo identificadas com as abreviaturas SD, MD e LD, respectivamente. Na Figura 10, o gráfico ilustra os tempos de execução de três consultas, uma de cada categoria, para a análise de três modelos que foram otimizados com o algoritmo *k-means*. Os autores definiram os valores de *k* com sendo 4, 11 e 13, para que fossem criados um número *k* de *clusters* e, consequentemente, *k* famílias.

O gráfico de barras da Figura 10 mostra a superioridade do modelo com 11 famílias, que foi gerado a partir do *k-means*. Ele teve desempenho similar ao DLA-CF em consultas que envolviam poucas dimensões, mas se sobressaiu nos casos com mais de três dimensões. Analisando os tempos totais compilados das consultas, o modelo com 11 famílias teve o melhor desempenho e o com 13 veio logo em seguida na segunda posição. Já esperávamos que a superioridade de modelos otimizados pelo algoritmo fosse confirmada pois, conforme mencionado anteriormente, eles são se adaptam a demanda das consultas e minimizam o número de famílias processadas por consulta, no entanto, dependem que a carga seja conhecida previamente e desconsideram o aspecto ad-hoc inerente as consultas analíticas. Apesar disso, inesperadamente, os resultados do experimento

remontam a conclusões diferentes das encontradas anteriormente pelos estudos do primeiro grupo.

Figura 10 - Tempos de execução algoritmo k-means.



Fonte: Adaptado de Boussahoua et al. Logical Schema for Data Warehouse on Column Oriented NoSQL Databases (2017).

Nos estudos com os algoritmos *k-means* e *k-medoids*, a análise dos resultados consulta à consulta mostrou que a relação observada anteriormente entre os modelos DLA e DLA-CF não foi replicada com o benchmark TPC-DS. Os estudos do primeiro grupo concluíram que o DLA se saía melhor com consultas que envolvessem muitas dimensões e o DLA-CF as com poucas dimensões, mas o gráfico da figura acima e os demais resultados dos estudos mostram que o DLA-CF foi superior em todos os cenários postos nas consultas. Dessa forma, se faz necessário entender se os custos da presença de mais famílias foram subestimados ou se a alteração do benchmark justifica a anomalia.

Os trabalhos [14, 15, 16] utilizaram o modelo conceitual do SSB para conceber as bases de dados para os testes, enquanto [11, 12], fizeram uso do benchmark TPC-DS. A diferença entre os modelos dos dois benchmarks reside na quantidade de entidades dos dois modelos. Enquanto o SSB conta com apenas quatro dimensões, o TPC-DS possui oito dimensões ligadas à tabela Store Sales. Isso pode justificar a discrepância observada, uma vez que, a consulta com maior abrangência engloba atributos de todas as dimensões do SSB enquanto, no TPC-DS, sobram quatro dimensões sem uso quando a consulta envolve as mesmas

quatro dimensões do SSB. Assim, somos levados a concluir que existe um outro fator impactando negativamente o modelo DLA que, provavelmente, deve ser a quantidade de dados presentes em cada registro além do necessário.

3.2.3 Grupo III: Abordagens Geoespaciais

Nos estudos pautados acima, foram apresentadas algumas abordagens para a criação de modelos de dados para os bancos *wide-column*, as quais, desconsideravam a presença de dados geográficos entre as informações disponíveis. Essa categoria especial de dados compreende representações de pontos, linhas, polígonos, topografias e volumes, por exemplo. Cada um desses dados pode ocupar desde poucos bytes até dezenas de megabytes e, dependendo do grau de ocorrência, eles podem provocar impactos significativos na quantidade de memória necessária para o armazenamento dos mesmos e também na capacidade de processamento dos sistemas analíticos. Devido a importância e a necessidade de considerações especiais, tal categoria tipo de dados foi estudada [18, 19] para possibilitar a execução de consultas geoespaciais em tempo hábil.

O HBase, assim como outros bancos de dados wide-column, não possui as funcionalidades devidas para o armazenamento e a consulta de dados geoespaciais. Apesar disso, foram feitas pesquisas e aplicações que promovem a criação de índices baseados na técnica de geohash para atuarem em tabelas do HBase, o processamento de Big Data geoespacial com o combo HBase e Spark, criação de um modelo da dados geoespaciais e uma ferramenta para a execução de consultas geográficas sob índices nas chaves de linha [18]. Existe ainda um framework comercial chamado GeoMesa que foi projetado para o processamento de consultas analíticas geoespaciais em larga escala, tratando desde o armazenamento dos dados até a execução otimizada de predicados geoespaciais [40]. As muitas aplicações elencadas prestam contribuições relevantes para o entendimento da dinâmica atual entre o HBase e a execução de consultas geoespaciais, mas abordaremos brevemente apenas o modelo HGrid [19] que sumariza muitos dos esforços empreendidos em tal direção.

O HGrid é um modelo de dados proposto para o armazenamento e consulta de dados geoespaciais em bancos de dados *wide-column*. Ele adota uma abordagem híbrida de indexação que utiliza duas estruturas combinadas, uma *quad-tree* e um *grid* regular, nessa ordem, respectivamente. Antes de analisarmos a estrutura dos índices, precisamos entender a codificação dos dados. Um *grid* regular é uma das formas mais simples de indexação de dados geográficos, onde, um espaço retangular é repartido em células também retangulares de tamanho predeterminado. Cada registro geográfico é analisado e, com base em sua localização, é atribuído a uma das células do *grid*. Assim, é possível reduzir o espaço de busca para uma ou poucas células quando for realizada a execução de funções geoespaciais. A *quad-tree* funciona de forma similar, no entanto, sua estrutura de árvore permite o particionamento recursivo das regiões em novas regiões menores que permitem um refinamento mais avançado dos registros armazenados em cada um dos quadrantes.

A representação física do modelo HGrid pode ser contextualizada a partir da descoberta do menor retângulo que é capaz de englobar a área compreendida pelo conjunto de dados. Esse retângulo é decomposto em áreas retangulares de tamanho inferior e pré-definido, as quais, são indexadas em função da codificação decimal de uma *Z-order curve*. Os retângulos são novamente decompostos em um *grid* regular que atua como o segundo índice do modelo. Na prática, isto foi implementado através de uma tabela cuja chave de linha é a concatenação do *Z-value* com o número da linha em que o registro está localizado no *grid* regular. Já os nomes das colunas foram definidos como sendo a concatenação do número da coluna no *grid* com o identificador do objeto. Os dados geoespaciais e suas informações relacionadas foram codificadas na notação JSON como valor do caminho.

No estudo, os autores não debateram o uso das famílias de colunas na concepção do modelo. Elas foram relevadas na construção da tabela de dados geoespaciais, existindo apenas uma família sem qualquer contribuição para a otimização do desempenho. Outro aspecto que não foi abordado é o da normalização e desnormalização. O plano proposto para a execução de consultas geoespaciais envolve duas etapas. Na primeira, são selecionados os registros cuja

localização esteja dentro dos critérios estabelecidos até o nível de resolução alcançado pelos índices. Essa redução preliminar da quantidade de registros otimiza a avaliação dos demais critérios, inclusive, reduz o total de objetos que passam por alguma das dispendiosas funções geoespaciais. O uso de índices na primeira etapa garante que o segundo crivo analise menos registros mas, ainda assim, a segunda etapa precisa de análises criteriosas para a construção do modelo de dados.

3.3 ESTADO DA ARTE

Os primeiros cinco trabalhos relacionados [13-17] apresentaram técnicas para o projeto do modelo de dados de bancos *wide-column*. A dicotomia entre abordagens normalizadas e desnormalizadas levaram a criação dos modelos NLA e DLA, os quais, serviram para a validação dos benefícios e dos malefícios de cada abordagem, respectivamente. O NLA organiza os fatos e dimensões em tabelas distintas que possuem uma única coluna cada, isto culmina em pouca redundância e na realização de muitas junções. Já o DLA, une todos os dados em uma mesma tabela e, por isso, gera bastante redundância na base de dados mas elimina as custosas junções. O modelo DLA-CF muda de forma pontual o DLA, sendo desnormalizado mas com a disposição dos dados em famílias diferentes com base na dimensão de origem. A expectativa é que a existência de mais famílias, que são separadas fisicamente, melhore a otimização das operações de entrada e saída das memórias. E, por fim, foi apresentado o modelo FactDate, que defende a união das dimensões mais utilizadas em consultas com os fatos, em uma mesma família de colunas.

Os testes experimentais evidenciaram que o NLA foi o modelo com pior desempenho nos cenários avaliados, inferioridade que foi justificada pelos altos custos das junções. Apesar disso, precisamos ponderar que uma má distribuição das regiões de tabelas também pode impactar negativamente o desempenho de tais sistemas, especialmente, no caso do NLA, que se torna mais suscetível a essa problemática por conter uma quantidade maior de tabelas. Em relação às abordagens totalmente desnormalizadas, nos experimentos feitos a partir do SSB, os autores verificaram que o DLA se saia melhor quando o número de dimensões envolvidas nas consultas se aproximava da quantidade de dimensões contidas no

modelo. Diametralmente, o DLA-CF tinha os melhores resultados quando o número de dimensões abrangidas pelas consultas era pequeno, apenas um ou duas, e perdia para o DLA a medida em que tal número crescia. A proposição do modelo FactDate é condizente com os resultados dos predecessores, mas, na prática, os resultados dos experimentos mostraram que ele possui desempenho similar ao DLA-CF, se saindo melhor em consultas com uma dimensão.

Passando para as abordagens algorítmicas [10-12], vimos a adoção de modelos desnormalizados e a busca por meios de otimizar a disposição dos dados em famílias de colunas. Os estudos partiram do pressuposto de que seria possível utilizar uma carga de consultas conhecidas previamente para identificar afinidades entre atributos e utilizar tal dado para a construção das famílias. Os atributos com maior afinidade seriam aqueles que aparecessem com maior frequência juntos nas consultas e, portanto, deveriam ser dispostos na mesma família para se valer da separação física e reduzir a quantidade de informações processadas por consulta. Excluindo a abordagem com o *k-means*, as outras que utilizaram algoritmos genéticos propuseram funções capazes de quantificar a qualidade das soluções encontradas. Essas funções procuravam minimizar o número de famílias, distribuir de forma equitativa os dados de diferentes dimensões e minimizar o número de colunas duplicadas.

Os resultados dos testes experimentais atestaram a superioridade dos modelos construídos com o algoritmo evolucionário e com os algoritmos de clusterização. Eles reduziram o tempo exigido para a execução das cargas de consultas mas, o problema desse tipo de abordagem reside justamente no conhecimento prévio da carga de consultas, uma vez que, os sistemas analíticos são utilizados para a execução de consultas *ad-hoc* feitas por especialistas do domínio e não podem ser definidas tão facilmente com antecedência. Um outro ponto questionável é a falta de validação cruzada, os autores poderiam ter dividido a carga de consultas em conjuntos de treinamento e de testes para atestar a capacidade de generalização dos modelos concebidos. Isto, inclusive, contraporia o ponto anterior, pois, seria um indicativo que os modelos de dados não seriam afetados por consultas fora do escopo da carga que fora escolhida previamente para o treinamento.

Relacionamos dois estudos científicos que analisam o contexto de consultas geográficas em bancos wide-column, especialmente, como otimizar a execução delas [18, 19]. O primeiro faz uma revisão da literatura e apresenta os trabalhos mais promissores, os quais, focam na construção de índices geoespaciais. Apenas um deles [19], aborda a construção do modelo de dados para a realização de dois níveis de indexação. Esses estudos deixam de lado, ou discutem brevemente, como devem ser modelados os relacionamentos, a definição das chaves de linha, criação das famílias de colunas e os efeitos de abordagens mais normalizadas ou desnormalizadas. Dessa forma, suas contribuições podem ser vistas como estratégias complementares que podem ser utilizadas em conjunto com as proposições dos demais trabalhos relacionados.

Diante do exposto, fica evidente que existe uma lacuna na literatura para a avaliação dos impactos da normalização e desnormalização de dados geoespaciais em ambientes analíticos construídos sob bancos de dados NoSQL orientados a colunas do tipo *wide-column*. A opção entre abordagens mais normalizadas ou desnormalizadas também pode ser estendida para os dados convencionais e ainda ser combinada com abordagens para aspectos próprios dos bancos *wide-column* (i.e. as famílias de colunas). Nenhum dos estudos relacionados investigou abordagens para todas estas frentes mas, no escopo dos dados geoespaciais, o trabalho de Ferro et la. [20] investigou os cenários mais propícios para a redução da redundância de dados geográficos por meio da normalização e desnormalização baseada na seletividade. No caso dele, para os bancos NoSQL orientados a documentos.

A investigação da redundância de dados geoespaciais [20] foi inspirada em parâmetros abordados por três estudos científicos precedentes. Em [43], os autores avaliaram os impactos da redundância em um data-warehouse relacional através da análise comparativa entre dois esquemas que tinham abordagens normalizadas e desnormalizadas para os dados geográficos, respectivamente. Um outro estudo [44], de cunho similar, analisou se a complexidade de tais dados (i.e. ponto versus polígono) poderia servir de critério para a disposição deles em dimensões distintas das que forem utilizadas para dados convencionais. A partir de tais proposições, Ferro et al. propôs nove abordagens para a construção de esquemas para GDWs

orientadas a documentos, as quais, realizam a normalização e desnormalização em razão das dimensões mais e menos seletivas.

4 PLANEJAMENTO EXPERIMENTAL

A concepção de esquemas para os bancos de dados NoSQL wide-column preconiza o estudo prévio dos dados que serão armazenados, bem como, dos padrões mais frequentes de escritas e leituras que serão originados das aplicações [13, 16, 41]. Para Chevalier et al. [42], o projeto de esquemas perpassa pela criação de modelos conceituais multidimensionais com tabelas de fatos e dimensões para representar os dados de natureza quantitativa e qualitativa, respectivamente. Um modelo de cunho conceitual estabelece quais entidades, atributos e relacionamentos precisam estar presentes no sistema independentemente das tecnologias de armazenamento empregadas.

Os esquemas abordados anteriormente NLA, DLA, DLA-CF, FactDate e os demais são fruto de várias modificações realizadas a partir do modelo conceitual do SSB. Elas foram elaboradas com o intuito de alternar abordagens para a normalização, desnormalização, implementação de relacionamentos entre dimensões e distribuição de colunas em famílias. Essas diferentes escolhas constituem o que podemos chamar de parâmetros ou variáveis esquemáticas que, quando definidas, irão caracterizar os esquemas produzidos e podem sinalizar quais as melhores e piores decisões. Almejamos ajudar nesse processo de decisão através da proposição e avaliação de novos esquemas cujo projeto leve em consideração a presença de dados geoespaciais sendo armazenados e consultados a partir do sistema analítico. Esse tipo de informação geográfica gera implicações adicionais, como restrições de armazenamento, que inviabilizam a utilização dos modelos já disponíveis na literatura. Questões como o nível de redundância desses dados, a implementação dos relacionamentos entre eles e dos relacionamentos deles com dimensões convencionais precisam ser investigadas mais a fundo.

Nosso estudo sobre a inclusão de dados geoespaciais no projeto esquemático de bancos orientados a colunas, assim como ocorreu nos trabalhos relacionados, partiu do modelo conceitual híbrido do benchmark Spadawan. A partir de tal modelo, nós variamos exaustivamente o nível de redundância dos dados geográficos para que, cada abordagem de variação, originasse um novo modelo derivado do original, mas com características próprias. Em seguida, adotamos

diferentes abordagens para a implementação dos relacionamentos entre a tabela de fatos e as dimensões, além da disposição de colunas entre as famílias. O produto da variação de tais abordagens, quando aplicadas uma após a outra de forma independente, foram novos esquemas com particularidades próprias. Após serem analisados e terem suas viabilidades julgadas, puderam ser testados em um ambiente experimental para a determinação de quais escolhas influenciaram positivamente ou negativamente no volume de dados, tempo de ingestão na base e tempo de execução de consultas geoespaciais.

Neste capítulo, iremos tratar das abordagens escolhidas para a variação do nível de redundância de dados geoespaciais e iremos expor como as combinações delas deram origem aos novos modelos conceituais derivados do modelo híbrido do Spadawan (Seção 4.1). Em seguida, serão apresentadas as abordagens para transformar tais modelos em esquemas que contemplem maneiras distintas de relacionar tabelas de fatos e dimensões, bem como, duas abordagens para a disposição de colunas em famílias (Seção 4.2). Explicadas tais abordagens, vamos apresentar todos os esquemas gerados e discutiremos suas viabilidades (Seção 4.3). O capítulo é finalizado com detalhes sobre a carga de consultas escolhidas para os testes experimentais, informações sobre a execução deles e sobre os dados que serão coletados para a análise consolidada das abordagens (Seção 4.4).

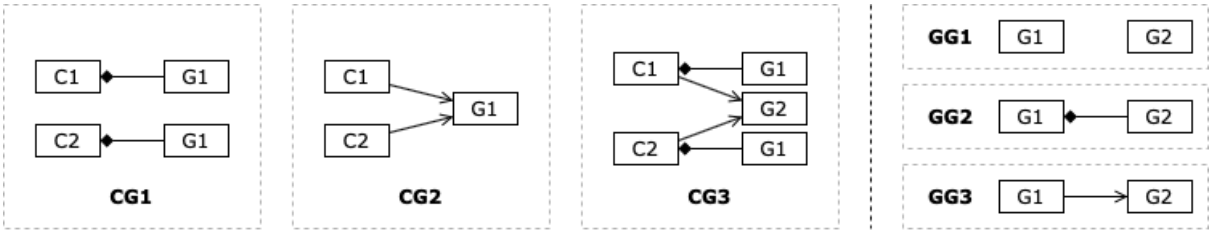
4.1 PARÂMETROS GEOESPACIAIS

Conforme exposto por Ferro et al. [20], o planejamento do nível de redundância geoespacial pode ser realizado em duas etapas. A primeira etapa consiste em determinar como serão implementados os relacionamentos entre as dimensões convencionais e as geoespaciais. Já a segunda consiste em escolher como serão implementados os relacionamentos entre duas ou mais dimensões de cunho geográfico. As abordagens propostas levam em consideração a seletividade dos dados, onde, um dado geográfico de alta seletividade (i.e. endereço) possui uma unicidade maior, ou seja, existem menos ocorrências desse dados na tabela, enquanto que, os pouco seletivos (i.e. cidade) possuem uma maior repetição com a presença em muitos dos registros da tabela.

O relacionamento entre dimensões convencionais e dimensões geoespaciais pode ser realizado de três formas distintas que, aqui, serão identificadas pelas siglas CG1, CG2, CG3, onde o prefixo CG é utilizado para denotar a natureza do relacionamento (convencional-geoespacial). A abordagem CG1 estabelece que um dado geográfico qualquer seja replicado em quaisquer dimensões convencionais que precisem dele, acarretando em muita redundância. A forma CG2 segue caminho oposto, onde as dimensões convencionais referenciam quaisquer dimensões geográficas pertinentes e isso leva a uma pequena redundância de referências. Na abordagem CG3, as dimensões mais seletivas são desnormalizadas e as menos seletivas normalizadas, ou seja, o nível de redundância será inferior ao existente na abordagem CG1 e superior ao da abordagem CG2, portanto, intermediário.

Similarmente a primeira etapa, adotaremos as siglas GG1, GG2 e GG3 para denotar as abordagens utilizadas para a implementação dos relacionamentos entre dimensões geográficas, onde, CG foi o prefixo utilizado para denotar a natureza do relacionamento (geoespacial-geoespacial). Na GG1, as dimensões geoespaciais não se relacionam entre si, pois, mantém relacionamentos apenas com as convencionais. A abordagem GG2 prega que as dimensões geográficas de menor seletividade sejam desnormalizadas nas de maior seletividade, ou seja, os menos seletivos são replicados em todos os registros de dados geoespaciais mais seletivos com os quais possuam alguma relação. Já na GG3, a dimensão geográfica de maior seletividade passa a referenciar as menos seletivas.

Figura 11 - Abordagens para relacionar dimensões geoespaciais.



Fonte: autor (2022). Abordagens para a implementação dos relacionamentos entre dimensões convencionais e geoespaciais (esquerda), e das relações entre as próprias dimensões geoespaciais (direita).

As ilustrações da Figura 11 retratam as abordagens CG e GG aplicadas em cenários fictícios onde, os blocos C1 e C2 representam dimensões convencionais, e, G1 e G2 representam dimensões do tipo geoespacial com a G1 tendo uma maior seletividade que G2. A notação utilizada na figura deriva do trabalho de Ferro et al. e será explana com mais detalhes em breve mas, no momento, para que sejam compreendidas as combinações entre abordagens CG e GG, basta salientar que os elementos da linguagem UML de associação e composição denotam cenários em que os dados da dimensão são compartilhados ou replicados quando demandados, respectivamente. A associação indica que a dimensão apontada é referenciada por qualquer registro que precise de seus dados, ou seja, não existe redundância. Já a composição, denota que os registros da dimensão base possuem cópias dos dados que precisem, existindo redundância deles. As associações e composições podem ser vistas como configurações que priorizam, nessa ordem, a normalização e desnormalização dos dados.

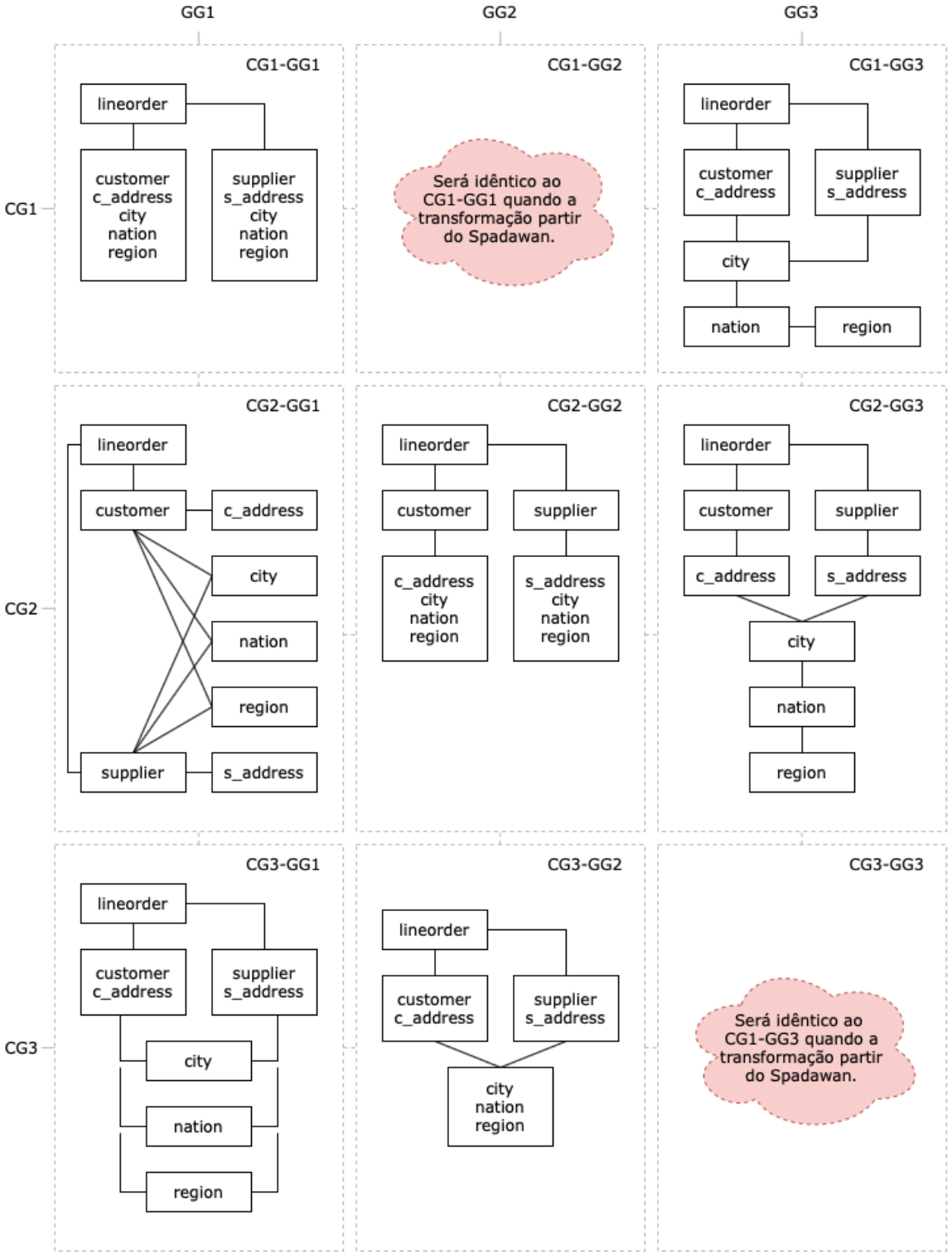
As transformações realizadas no modelo conceitual do Spadawan, de acordo com as abordagens apresentadas e as suas combinações, estão ilustradas na Figura 12 numa matriz. As linhas denotam as possíveis abordagens para a implementação do relacionamento entre dimensões convencionais e geoespaciais (i.e. CG1, CG2, CG3), e, as colunas, denotam as abordagens para os relacionamentos entre as dimensões geoespaciais (i.e. GG1, GG2, GG3). As células da tabela são o produto cartesiano de todas as abordagens {CG1, CG2, CG3} X {GG1, GG2, GG3} cujo resultado são os pares ordenados caracterizando novos modelos conceituais. Note que, por se tratar de um modelo conceitual, não foram utilizados os elementos de associação e composição, logo, as abordagens desnormalizadas devem ficar implícitas nas dimensões que compartilham das mesmas tabelas. Além disso, o desenho também não representa diretamente as dimensões *date* e *part*, que foram omitidas das ilustrações para simplificar a figura e conferir destaque as diferenças entre os modelos.

A primeira linha da matriz compreende os modelos conceituais gerados a partir da abordagem CG1 quando combinada com as abordagens GG1, GG2 e GG3, respectivamente indicadas em cada uma das colunas. Na combinação CG1-GG1, as dimensões geográficas *c_address*, *s_address*, *city*, *nation* e *region* foram

desnormalizadas e os seus dados replicados em cada um dos registros convencionais das tabelas *customer* e *supplier* que mantivessem alguma referência para eles. O modelo CG1-GG2 foi ilustrado de forma idêntica ao anterior porque, no modelo conceitual híbrido do Spadawan, as dimensões convencionais *customer* e *supplier* se relacionam com todas as dimensões geográficas e dessa forma, quando vamos aplicar a abordagem GG2, não restam dimensões geoespaciais a serem desnormalizadas em razão da CG1 já ter feito isto. Na abordagem CG1-GG3, as dimensões *c_address* e *s_address* são desnormalizadas por serem mais seletivas mas, as de menor seletividade, passam a ser referenciadas em um encadeamento ordenado pela seletividade das dimensões.

Os modelos gerados a partir da abordagem CG2 combinada com as abordagens GG1, GG2 e GG3 estão ilustrados na segunda linha da matriz. O modelo conceitual CG2-GG1 segue uma abordagem mais normalizada onde, as dimensões convencionais *customer* e *supplier* mantêm referências para as dimensões geoespaciais *c_address*, *s_address*, *city*, *nation* e *region*. A redundância de dados será mínima, pois, nenhum dado possui réplicas, com exceção das chaves. Na combinação CG2-GG2, as dimensões geoespaciais de maior seletividade *c_address* e *s_address* são normalizadas, mas as de menor seletividade são desnormalizadas e passam a serem replicadas juntamente aos registros das dimensões mais seletivas que estejam relacionadas a elas. Finalmente, a abordagem CG2-GG3 faz a normalização dos dados geográficos mais seletivos e cria referências encadeadas, pela ordem de seletividade, com início na dimensão *city* até a dimensão *region*.

Figura 12 - Combinações {CG1, CG2, CG3} X {GG1, GG2, GG3}.



Fonte: autor (2022). Modelos conceituais gerados a partir da combinação de abordagens para a modelagem dos relacionamentos entre dimensões convencionais e geoespaciais e entre as próprias dimensões geoespaciais.

Na última linha da tabela, temos os modelos oriundos das combinações entre a abordagem CG3 e as abordagens GG1, GG2 e GG3. O modelo conceitual CG3-GG1 se distingue pela desnormalização das dimensões geoespaciais mais seletivas *c_address* e *s_address*. Os dados delas são replicados nas dimensões convencionais *customer* e *supplier*. As dimensões geográficas menos seletivas continuam sendo referenciadas pelas dimensões convencionais e não se relacionam entre si. Na combinação das abordagens CG3-GG2, o modelo resultante segue a mesma premissa de desnormalizar os dados geográficos mais seletivos mas, dessa vez, a dimensão com a segunda melhor seletividade passa a ser normalizada e todas as demais são desnormalizadas junto a ela. Na última posição da matriz, o modelo CG3-GG3 foi ilustrado de forma idêntica ao CG1-GG3, pois, partindo do modelo conceitual do Spadawan, as dimensões geográficas mais seletivas são normalizadas e as de menor seletividade desnormalizadas em ambas as abordagens, produzindo modelos idênticos.

Na última linha da tabela, temos os modelos oriundos das combinações entre a abordagem CG3 e as abordagens GG1, GG2 e GG3. O modelo conceitual CG3-GG1 se distingue pela desnormalização das dimensões geoespaciais mais seletivas *c_address* e *s_address*. Os dados delas são replicados nas dimensões convencionais *customer* e *supplier*. As dimensões geográficas menos seletivas continuam sendo referenciadas pelas dimensões convencionais e não se relacionam entre si. Na combinação das abordagens CG3-GG2, o modelo resultante segue a mesma premissa de desnormalizar os dados geográficos mais seletivos mas, dessa vez, a dimensão com a segunda melhor seletividade passa a ser normalizada e todas as demais são desnormalizadas junto a ela. Na última posição da matriz, o modelo CG3-GG3 foi ilustrado de forma idêntica ao CG1-GG3, pois, partindo do modelo conceitual do Spadawan, as dimensões geográficas mais seletivas são normalizadas e as de menor seletividade desnormalizadas em ambas as abordagens, produzindo modelos idênticos.

4.2 PARÂMETROS ESTRUTURAIS

Introduzidas as abordagens com parâmetros geoespaciais e apresentadas as transformações feitas no modelo conceitual do Spadawan a partir delas, agora,

iremos tratar das novas abordagens que contemplam aspectos esquemáticos dos bancos de dados *wide-column*. Esses aspectos tangem as implementações possíveis dos relacionamentos entre a tabela de fatos e as dimensões, e, também a separação física dos dados das colunas por meio da alocação na mesma ou em diferentes famílias das tabelas. O objetivo dessas novas abordagens é, justamente, abranger mais possibilidades para a construção de esquemas e, com isso, garantir uma geração exaustiva de novos esquemas partindo das combinações entre os parâmetros geoespaciais e os novos parâmetros ditos estruturais.

Bem como fora introduzido na segunda seção do Capítulo 2, um projetista que esteja trabalhando na construção de esquemas para o HBase, precisará especificar em seu projeto como será realizada a distribuição dos dados em tabelas, dos dados de cada tabela em famílias de colunas e dos dados de cada família entre as colunas. Além disso, ele precisa considerar os padrões de escritas e leituras que ocorrem com maior frequência para, só então, definir quais dados devem ser designados como as chaves de linha da tabela. Os aspectos supramencionados fazem alusão aos principais elementos esquemáticos dos bancos *wide-column*, logo, as diferentes abordagens para a implementação dos mesmos configuram os parâmetros estruturais que buscamos. Assim, as abordagens que propomos nesta segunda etapa concernem a organização dos dados em tabelas, famílias, colunas e chaves.

A disposição de dados convencionais em tabelas será realizada apenas entre aqueles presentes nas tabelas de fatos e dimensões, uma vez que, a disposição de dados geográficos já foi tratada com as abordagens que foram propostas para os parâmetros geoespaciais. O arranjo dos dados será feito de três formas distintas que seguem abordagens normalizadas, desnormalizadas e um meio termo entre as duas primeiras. Utilizaremos as siglas FDN, FDD, FDH para indicar as três abordagens nessa ordem, respectivamente. O prefixo FD determina que tais abordagens tangem os relacionamentos entre fatos e suas dimensões associadas (fatos-dimensões) e, os sufixos, denotam se a abordagem é normalizada, desnormalizada ou híbrida (i.e. FDN , FDD, FDH). Na normalizada, existirá uma tabela para os fatos e tabelas próprias para cada uma das dimensões, as quais, serão referenciadas por os registros da tabela de fatos. Já na abordagem desnormalizada, os fatos e as

dimensões irão residir na mesma tabela, na qual, todos os dados dimensionais referenciados por um fato serão replicados juntamente aos dados de tal fato. E, finalmente, na abordagem híbrida teremos uma tabela para os fatos e outra para todas as demais dimensões. O relacionamento delas terá cardinalidade 1:1, com todos os dados dimensões referenciados por um dado fato estando agregados na única referência.

Os dados presentes nas tabelas, independentemente de serem fatos ou não, serão distribuídos em famílias de colunas de acordo com duas abordagens. Na primeira, todos os dados são congregados em uma única família de colunas. Diametralmente, na segunda abordagem proposta, serão criadas múltiplas famílias e cada uma conterá apenas fatos ou dados de uma mesma dimensão. Seguindo a mesma estratégia de nomear as abordagens com nomes alusivos, denotaremos as duas propostas para a distribuição dos dados entre famílias com as siglas F1 e FN, respectivamente. É importante frisar que a existência de mais ou menos famílias de colunas não afeta o nível de redundância dos dados, mas pode influenciar positivamente no tempo de execução de consultas.

Conforme mencionamos anteriormente, os dados de uma ou mais entidades podem ser arranjados em colunas a partir de diversas abordagens. Isto é possível em razão do HBase não demandar que sejam utilizados tipos primitivos de dados e tratar todos eles como sequências de bytes, assim, se torna possível armazenar estruturas de dados mais complexas (i.e. JSON, XML). Exemplificando, os relacionamentos com cardinalidade 1:N (i.e. a postagem de um blog e os comentários dela) podem ser convertidos em uma única tabela que tenha uma coluna designada para guardar todos os dados referenciados na forma de uma lista formatada em JSON. Embora isto seja possível e formula uma pergunta de pesquisa relevante, não iremos propor abordagens para tal caminho por acreditar que elas não contribuiriam de forma significativa para avaliar a redundância e o projeto esquemático de dados convencionais e geoespaciais no contexto proposto por esta pesquisa. O outro parâmetro que também não será discutido são as abordagens para a formulação das chaves de linha, pois, as chaves de linha seriam úteis na indexação de dados e nós não abordaremos isso neste momento.

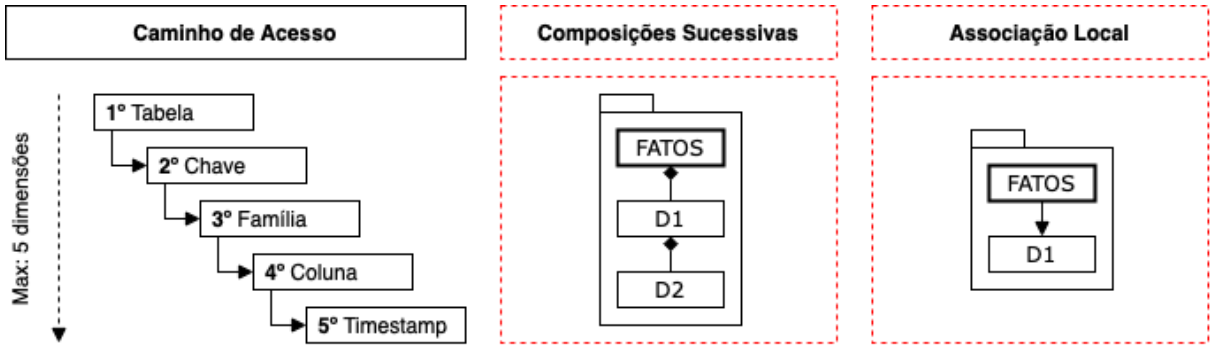
Agora que contextualizamos os parâmetros estruturais na forma de abordagens para modelarmos os relacionamentos entre as tabelas de fatos e as dimensões, além da disposição de colunas entre as famílias, iremos explicar como será feita a representação visual dos esquemas que foram criados a partir das combinações de parâmetros estruturais e geoespaciais. Isto se faz necessário por conta de não termos encontrado na literatura metodologias para a representação visual de tais esquemas em bancos de dados *wide-column*. Se faz necessário representar tabelas, relacionamentos, famílias e a disposição das colunas entre as famílias. Assim, utilizaremos a notação UML proposta no estudo de Ferro et al. [20] para os bancos orientados a documentos mas, ressignificada para o contexto dos bancos de dados orientados a colunas conforme as ilustrações na Figura 13. A notação UML possui os elementos construtores de pacote, classe, composição e de associação para representar, nessa ordem, as tabelas, famílias, famílias adjuntas e famílias referenciadas.

Figura 13 - Notação UML para a representação visual dos esquemas.

Fonte: autor (2022). Adaptado de Ferro et al. Evaluating Redundancy and Partitioning of Geospatial Data in Document-Oriented Data Warehouses (2020).

O construtor de pacote denota que uma dada tabela existirá no modelo físico, onde, ela pode conter uma ou mais famílias estando representadas pelo elemento classe. As classes representadas com linhas mais espessas denotam famílias de fatos e, as com linhas finas, denotam os dados oriundos de dimensões. É possível que duas ou mais dimensões precisem ocupar a mesma família de colunas em razão da adoção de abordagens desnormalizadas, logo, nestes casos, o elemento classe terá os nomes de todas as dimensões congregadas dispostas dentro do elemento gráfico. A ocorrência de tais cenários pode ser visualizada nas abordagens FDD-F1, FDN-F1 e FDH-F1 que foram ilustradas para exemplificar cenários, os quais, iremos discutir com mais detalhes em breve.

Figura 14 - Dimensionalidade, composições sucessivas e associações.



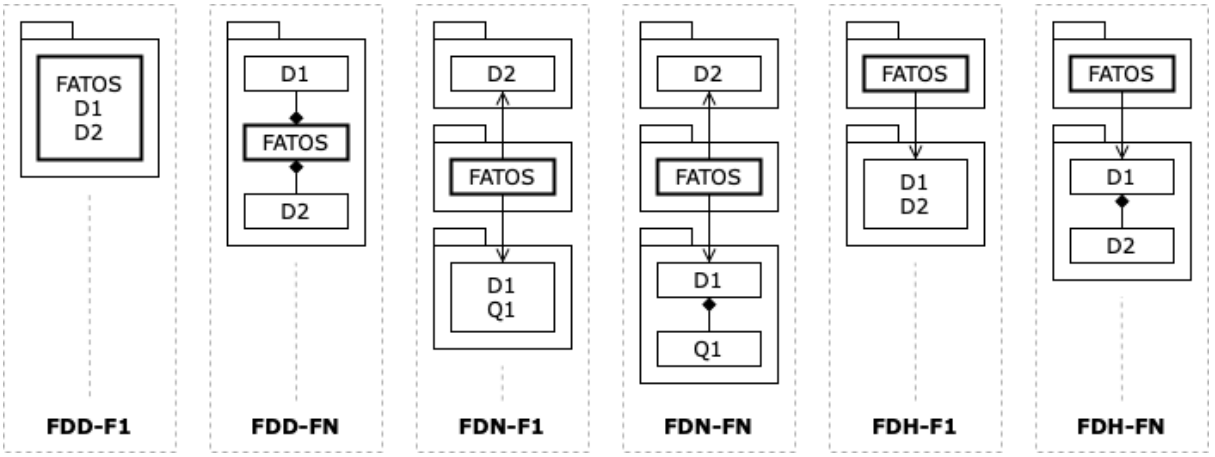
Fonte: autor (2022).

Outro aspecto importante dos bancos wide-column e que incorre no uso da notação é que, como tais bancos são mapas 5-dimensionais (tabela, chave de linha, família, coluna, timestamp), não se pode realizar o encadeamento de composições, conforme ilustrado na Figura 14. As composições sucessivas de famílias não podem ser realizadas por conta da dimensionalidade limitada no escopo dos componentes supracitados. O único caminho para expandir tal dimensionalidade é através do uso de estruturas de dados especiais (i.e. JSON) que poderiam ser armazenadas em uma coluna e abrigar mais dimensões mas, tal opção não é suportada por padrão no HBase. Complementando as excepcionalidades da notação, é importante mencionar que, embora seja possível a implementação de referências entre famílias de uma mesma tabela, tal construção não está documentada entre os estudos analisados por nós. Com base nos conhecimentos da arquitetura do HBase, o emprego das referências entre famílias de uma mesma tabela pode ser um facilitador para a implementação de técnicas de localidade de dados mas, até o momento, não detectamos vantagens em seu uso.

Agora que detemos do ferramental necessário para a representação visual dos esquemas que serão concebidos a partir de combinações dos parâmetros estruturais, iremos apresentar as combinações entre as abordagens FDD, FDN e FDH com as abordagens F1 e FN. Para isso, assumimos a existência de um modelo conceitual hipotético que possua uma tabela de fatos e duas tabelas convencionais associadas aos fatos. A Figura 15 ilustra tal modelo hipotético após serem seguidas as abordagens transformacionais oriundas do produto cartesiano {FDD, FDN, FDH}

X {F1, FN}. Especialmente para as abordagens FDN-F1 e FDN-FN assumimos a presença de uma dimensão qualquer que tivesse sido desnormalizada na mesma tabela de uma das dimensões que se relaciona com os fatos.

Figura 15 - Combinações FDD-F1, FDD-FN, FDN-F1, FDH-F1 e FDH-FN.



Fonte: autor (2022).

A abordagem FDD desnormaliza as dimensões referenciadas pela tabela de fatos, logo, culmina em uma maior demanda por armazenamento em função da alta redundância. Tais características estão presentes nas abordagens FDD-F1 e FDD-FN, as quais, diferem apenas pela quantidade de famílias nas tabelas. A normalização das dimensões referenciadas pelos fatos reduz drasticamente o espaço de armazenamento necessário em relação a abordagem desnormalizada, mas implica na execução das custosas operações de junção entre tabelas, este é o caso dos esquemas FDN-F1 e FDN-FN. Já nos cenários com a abordagem FDH, o nível de redundância será marginalmente acima da FDD mas ela se beneficiará nas consultas cujo predicado filtre os resultados com base na tabela de fatos.

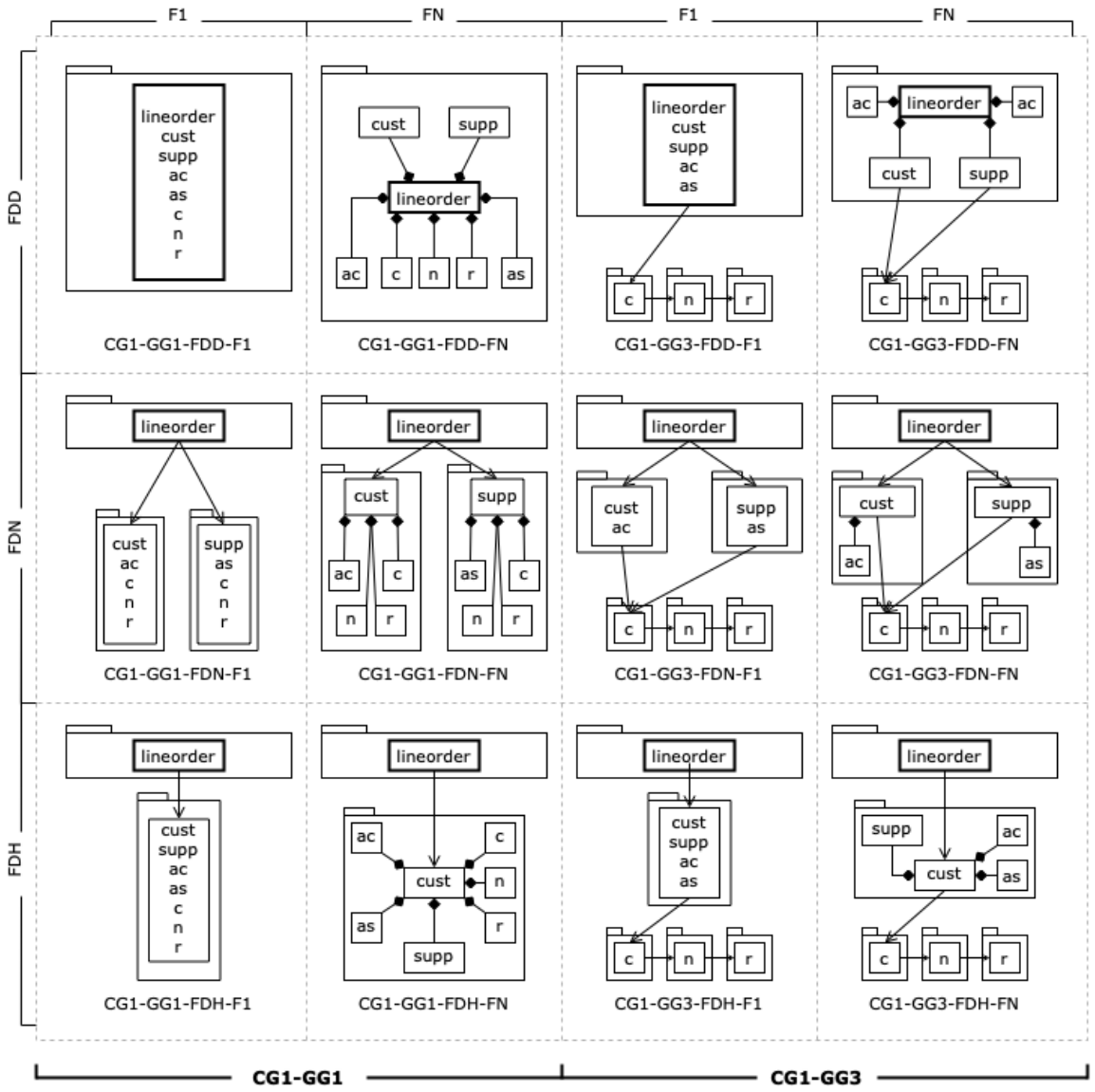
4.3 MODELOS RESULTANTES

Nas duas subseções precedentes, tratamos dos parâmetros geoespaciais na forma das abordagens denominadas CG1, CG2, CG3, GG1, GG2 e GG3, e, também, dos parâmetros estruturais através das abordagens FDD, FDN, FDH, F1 e FN. Após definir e especificar cada uma delas, nós utilizamos um programa computacional de autoria própria para transformar exaustivamente o modelo

conceitual do benchmark Spadwan em 54 novos modelos adequados para os bancos NoSQL colunares do tipo *wide-column* (i.e. {CG1, CG2, CG3} X {GG1, GG2, GG3} X {FDD, FDN, FDH} X {F1, FN}). Nomeamos os modelos como a concatenação das siglas (i.e. {CG2, CG3, FDH, FN} = CG2-GG3-FDH-FN). Apesar de o produto cartesiano indicar a existência 54 novos esquemas, precisamos excluir da conta os 12 que seriam gerados a partir das combinações CG1-GG2 e CG3-GG3 cujos modelos conceituais copiaram os modelos CG1-GG1 e CG1-GG3, respectivamente. Assim, totalizamos 42 esquemas únicos após a redução dos que resultariam em duplicatas (i.e. {CG1-GG2, CG3-GG3} X {FDD, FDN, FDH} X {F1, FN}).

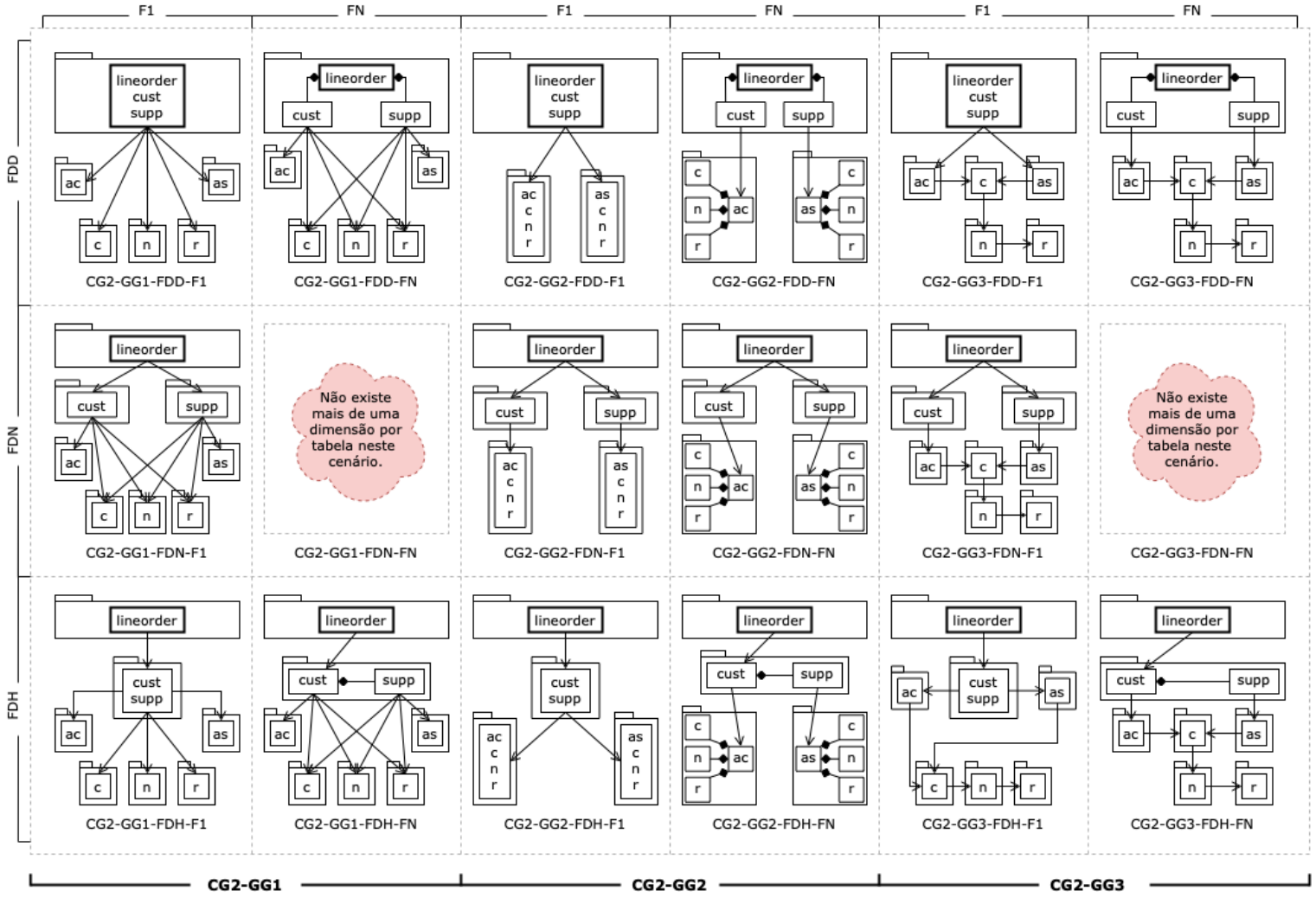
Os modelos lógicos gerados a partir dos produtos {CG1} X {GG1, GG3} X {FDD, FDN, FDH} X {F1, FN} foram ilustrados na Figura 16. A marcação lateral indica qual a abordagem adotada para a relação entre fatos e dimensões convencionais (i.e. FDD, FDN, FDH), a marcação superior indica como foi realizada a disposição dos dados em famílias (i.e. F1, FN) e, a inferior, evidencia a combinação base entre dimensões geoespaciais (i.e. CG1-GG1 e CG1-GG3). Seguindo a mesma sistemática da figura anterior, os modelos do produto {CG2} X {GG1, GG2, GG3} X {FDD, FDN, FDH} X {F1, FN} aparecem na Figura 17 e, finalmente, os oriundos do {CG3} X {GG1, GG2} X {FDD, FDN, FDH} X {F1, FN} estão representados na Figura 18. Para simplificar os desenhos e facilitar o entendimento, nós utilizamos as abreviações *cust*, *supp*, *c*, *n* e *r* para remeter aos nomes das dimensões *customer*, *supplier*, *city*, *nation* e *region*, respectivamente. Dado que o próprio modelo conceitual do Spadawan separa os endereços de consumidores e fornecedores em tabelas distintas, nós utilizamos as abreviações *ac* e *as* para denotar tais dimensões, nessa ordem. Já as dimensões *date* e *part* foram omitidas pelos mesmos motivos, assim, deve ficar implícito que elas se relacionam com os fatos da mesma forma que acontece com as dimensões *customer* e *supplier*, presentes nas ilustrações.

Figura 16 - Produto {CG1-GG1, CG1-GG3} X {FDD, FDN, FDH} X {F1, FN}.



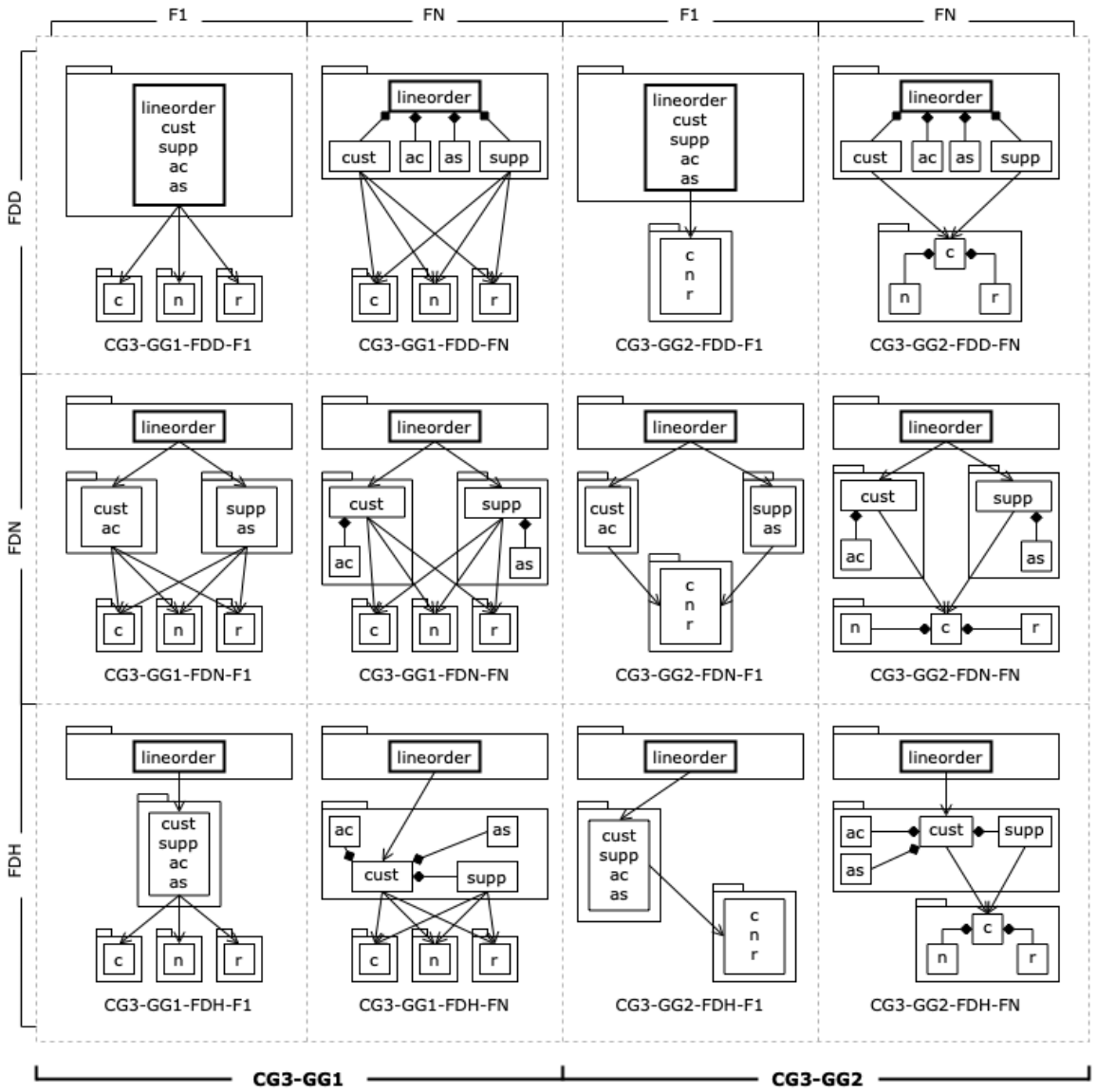
Fonte: autor (2022).

Figura 17 - Produto {CG2} X {GG1, GG2, GG3} X {FDD, FDN, FDH} X {F1, FN}.



Fonte: autor (2022).

Figura 18 - Produto {CG3-GG1, CG3-GG2} X {FDD, FDN, FDH} X {F1, FN}.



Fonte: autor (2022).

Conforme ilustrado na Figura 17, duas das dezesseis combinações oriundas do produto cartesiano entre as abordagens {CG2} X {GG1, GG2, GG3} X {FDD, FDN, FDH} X {F1, FN} foram omitidas e, em seus respectivos espaços, foram colocadas mensagens explanando que tais combinações envolvendo a abordagem FN não podem ser realizadas a partir do Spadawan e da metodologia construída para tal combinação de abordagens. Isto ocorre porque todas as dimensões acabam sendo dispostas em tabelas distintas e isso inviabiliza a distribuição de duas ou mais dimensões entre famílias diferentes de uma mesma tabela. Fazendo uma recapitulação das combinações e dos esquemas concebidos, a partir do produto

cartesiano {CG1, CG2, CG3} X {GG1, GG2, GG3} X {FDD, FDN, FDH} X {F1, FN} totalizamos um montante de 54 combinações possíveis, das quais, 14 foram excluídas pois iriam originar esquemas idênticos a algum outro dos demais quando tais transformações fossem realizadas para a construção de esquemas derivados do modelo conceitual do Spadawan.

4.4 EXECUÇÃO DO EXPERIMENTO

A preparação do experimento tem início com o uso do *software* SSB-DBGEN [50] para a geração da base de dados sintética do SSB, com fatores de escala equivalentes a 1 e 10 vezes o tamanho padrão. Tal programa recebe como parâmetro o fator e cria arquivos de texto correspondentes aos dados que devem ser utilizados para popular as tabelas *lineorder*, *customer*, *supplier*, *date* e *part*. Os dados de cunho geográfico são combinados posteriormente a partir de arquivos contendo as geometrias das cidades, nações e regiões, e de um programa computacional que gera endereços aleatórios para os registros de clientes e fornecedores. Esses recursos preparatórios estão disponíveis na página web do benchmark Spadawan [51] que trata do modelo conceitual híbrido. A Tabela 3 relaciona os arquivos originais com os respectivos números de linhas, colunas e volumes para os dois fatores de escala.

Tabela 3 - Linhas, colunas e o volume das tabelas do Spadawan.

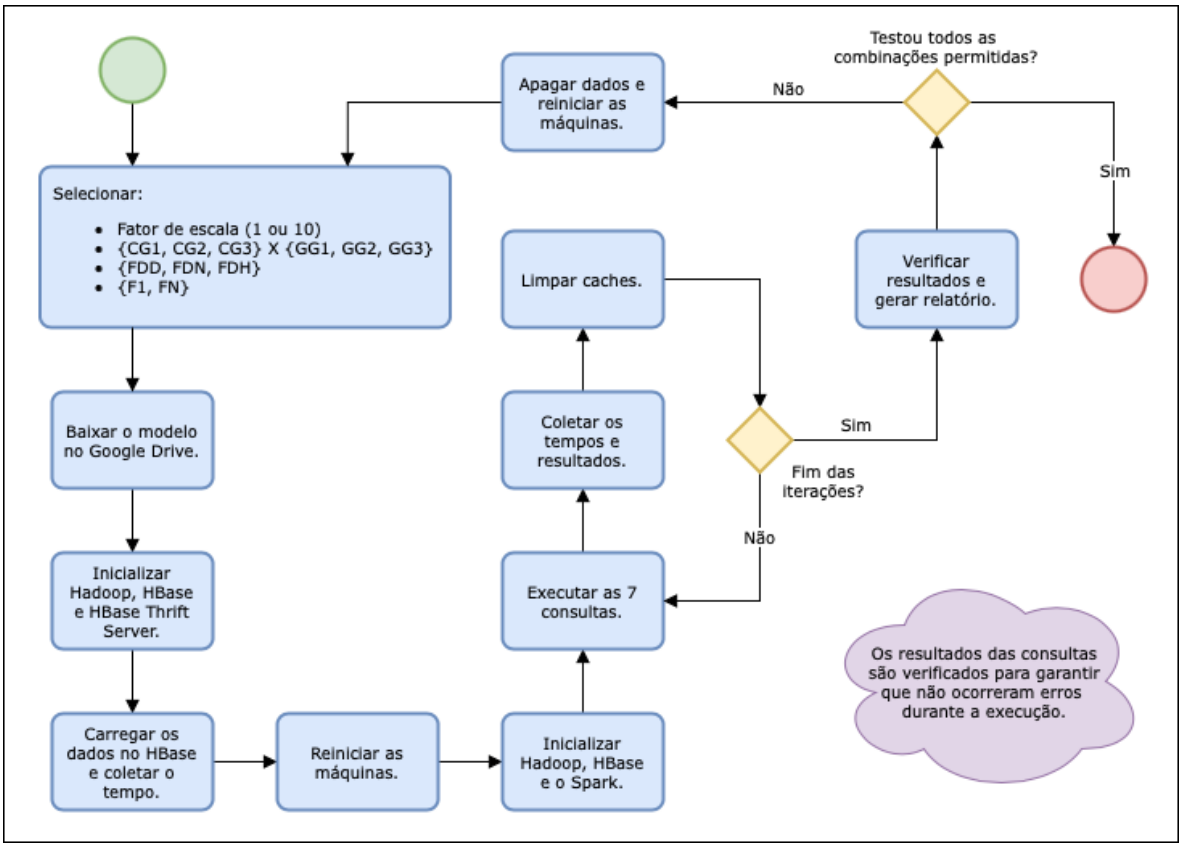
	Fator 1			**Fator 10**		
Tabela	**Registros**	**Colunas**	**Tamanho (MB)**	**Registros**	**Colunas**	**Tamanho (MB)**
lineorder	6.001.171	17	599,10	59.986.214	17	6.190,68
customer	30.000	12	2,80	300.000	12	28,70
date	2.556	17	0,23	2.556	17	0,23
part	200.000	9	17,10	800.000	9	68,90
supplier	2.000	11	0,17	20.000	11	1,70
c_address	30.000	2	2,50	300.000	2	25,60
s_address	2.000	2	0,16	20.000	2	1,70
city	250	2	10,40	250	2	10,40
nation	25	2	7,60	25	2	7,60
region	5	2	3,00	5	2	3,00

Fonte: autor (2022).

Em posse dos dados sintéticos gerados pelos benchmarks SSB e Spadawan, utilizamos um programa computacional de autoria própria para transformar tais dados do esquema estrela nos 40 esquemas propostos na subseção anterior. Como resultado das transformações, foram gerados arquivos tendo estruturas de dados especiais concebidas para codificar as tabelas, famílias, colunas e as fontes dos dados que deveriam ser utilizados para popular o banco de dados. Essas estruturas foram passadas como entrada para outro *software*, criado por nós, responsável por interpretar o esquema almejado e produzir arquivos que descrevam as inserções a serem realizadas no HBase. Esses arquivos foram compactados e armazenados no Google Drive, de onde, obtemos links de download para cada um deles e criamos um script para o *download* via linha de comando. Esse *script* recebe como entrada o nome do esquema, efetua o *download* dos arquivos e faz a descompactação. Após tal etapa feita no nós principal do cluster, iniciamos o Hadoop , HBase e o servidor Thrift para o carregamento dos dados. O servidor Thrift é uma ferramenta do HBase para a interação com o banco por chamadas de procedimentos remotos (RPC) que foi combinada com a HappyBase [52] para agilizar as inserções.

Após a inserção dos dados, encerramos todos os *softwares* utilizados e reiniciamos as máquinas, a fim de, eliminar qualquer potencial fonte de interferência antes de iniciarmos os testes. Uma vez reiniciadas, nós inicializamos o Hadoop, HBase e, dessa vez, o Spark. Submetemos um programa escrito em linguagem Scala que se utiliza do conector Spark-HBase e da biblioteca GeoSpark para executar as 7 consultas elaboradas. Elas são executadas, uma a uma, por n-vezes e seus tempos são registrados no arquivo de saída do programa. Uma vez finalizadas, nós executamos um programa que compara os resultados das consultas com uma tabela verdade, para garantir que não existiram erros durante a execução e, existindo erros, descartamos os resultados e repetimos o procedimento experimental. Esse mesmo programa lê o resultado das consultas e recupera os tempos que cada uma delas levou em cada iteração do experimento. A Figura 19 ilustra o fluxograma completo das atividades envolvidas na execução dos experimentos e que foram sumarizadas acima.

Figura 19 - Etapas envolvidas na execução do experimento.



Fonte: autor (2022).

O experimento foi conduzido em um *cluster* computacional composto por 4 máquinas conectadas por uma rede local de 1 Gbit/s. Três delas possuem a mesma configuração com o processador Intel Xeon E3-1230 3,50GHz e 48GB de memória principal. A quarta tem o processador Intel Xeon E5410 de 2,33GHz com 16GB de memória. O sistema operacional das máquinas é o CentOS versão 7.1 e as versões dos demais softwares foram Hadoop 2.7.6, HBase 2.1.8 e Spark 2.4.4. Nas configurações do Hadoop, HBase e Spark, as máquinas idênticas foram definidas como Datanode, RegionServer e Worker, respectivamente, enquanto a distinta recebeu o papel de coordenação. Não alteramos os parâmetros disponíveis para a configuração dos programas além do necessário para que atuassem em conjunto e usufruírem do hardware disponível no ambiente. Todos os programas de terceiros, programas criados por nós, dados utilizados, consultas, guias de instalação e os parâmetros para a configuração dos ambientes foram disponibilizados na página do projeto no Github [53].

4.4.1 Consultas

O modelo conceitual híbrido do Spadawan e os 40 esquemas, resultantes de combinações de várias abordagens utilizadas para a transformação do modelo, possuem duas dimensões geográficas com pontos (i.e. c_address, s_address) e três contendo polígonos (i.e. city, nation, region). Avaliaremos o uso de tais esquemas em data warehouses geográficas orientadas a colunas e de tais dimensões através da carga de consultas geoespaciais proposta por Ferro et al. [45], que também foi elaborada para os dados do Spadawan. São sete consultas contendo predicados geoespaciais que agregam os registros da tabela de fatos cujos clientes (listados na dimensão *customer*) estejam situados dentro dos polígonos das cidades, nações e regiões (dimensões *city*, *nation* e *region*, respectivamente) que tenham sido selecionadas por um clique aleatório no espaço. As descrições das consultas em texto são fornecidas abaixo e no Apêndice I é possível conferir a versão em linguagem SQL

1. Quantos produtos foram vendidos, por categoria, para clientes cujo endereço está situado dentro de uma região que engloba um ponto qualquer do espaço?

2. Quantos produtos foram vendidos, por categoria, para clientes cujo endereço está situado dentro de uma nação que engloba um ponto qualquer do espaço?

3. Quantos produtos foram vendidos, por categoria, para clientes cujo endereço está situado dentro de uma cidade que engloba um ponto qualquer do espaço?

4. Quantos produtos foram vendidos, por categoria, para clientes cujo endereço está situado dentro de um raio de 15 quilômetros centrado em um ponto qualquer do espaço?

5. Quantos produtos foram vendidos, por categoria e região, para clientes cujo endereço está situado dentro de uma nação que engloba um ponto qualquer do espaço?

6. Quantos produtos foram vendidos por categoria, região e nação para clientes que residem dentro de uma cidade que engloba um ponto qualquer do espaço?

7. Quantos produtos foram vendidos por categoria, região, nação e cidade para clientes que residem dentro de um raio de 15 quilômetros centrado em um ponto qualquer do espaço?

As consultas são ajustadas pontualmente para funcionarem com cada um dos 40 esquemas, dadas as especificidades de cada um, e depois executadas para podermos analisar quais as características que beneficiam ou prejudicam o tempo de execução. Podemos dividir as consultas em dois grupos em função das operações OLAP que realizam. O primeiro grupo compreende as consultas de 1, 2, 3 e 4 que realizam as operações *slice* e *dice*, selecionando dados das dimensões *part* e *customer* com níveis crescentes de seletividade geoespacial. A operação *slice* reduz a dimensionalidade por meio da seleção dos dados de uma dimensão e a *dice* extrai um subcubo selecionando os dados de mais de uma dimensão. Já no segundo grupo, temos as consultas 1, 5, 6 e 7 que também realizam *slice* e *dice* mas também executam as de *drill-down* e *roll-up*. Estas duas últimas consistem no aumento ou diminuição do nível de detalhamento com que os dados são agregados. Enquanto na consulta 1 os dados são agregados para a dimensão geoespacial menos seletiva, na consulta 5 eles levam em consideração os dois níveis menos seletivos, depois os três menos e assim sucessivamente.

4.4.2 Coleta, Dados e Estatísticas

Serão coletadas três informações principais: o volume de dados demandado para o armazenamento dos esquemas e das estruturas de dados utilizadas pelo HBase, o tempo de ingestão dos dados no banco e os tempos de execução das sete consultas. A determinação do espaço ocupado pelos dados e estruturas do HBase

foi feita através da aplicação denominada *hdfs*, a qual, está disponível na linha de comando e é capaz de calcular o espaço alocado em blocos do Hadoop para todos os dados que foram armazenados dentro da pasta designada para o HBase. Assumimos que o tempo de ingestão dos dados compreende a soma dos tempos necessários para a carga de cada tabela, nós auferimos eles com chamadas subsequentes a função *time.monotonic()* na aplicação Python responsável pela inserção dos dados. Finalmente, o tempo demandado para a execução das consultas foi calculado com a função *spark.time()*, invocada diretamente na aplicação Scala para cada consulta. Após serem finalizadas todas as consultas e todas as repetições, nós utilizamos um programa para condensar os dados num único arquivo para a análise posterior.

A análise dos dados se valerá de tabelas, gráficos, elementos da estatística descritiva e de testes de hipótese. Esperamos que as médias aritméticas dos tempos de execução de consultas, nas enésimas repetições, consigam explicitar diferenças entre as abordagens envolvidas na concepção de cada um dos esquemas avaliados. Desejamos, em especial, ser capazes de distinguir os impactos de escolhas que favoreçam mais a normalização ou desnormalização dos dados. As conclusões advindas de tais comparações serão suportadas estatisticamente através da realização do teste t de Student para a as diferenças entre as médias das amostras de quaisquer dois esquemas. A versão não pareada para dois grupos e duas amostras independentes com variâncias desconhecidas pode ser realizada com o intuito de determinar, com significância, se as médias dos dois grupos são diferentes. Supondo o cenário em que A e B sejam esquemas e tenhamos amostras dos seus tempos de execução, somos capazes de formular as hipóteses H_0: $\mu_A - \mu_B = 0$ e H_1: $\mu_A - \mu_B \neq 0$, onde μ_A e μ_B são as médias de A e de B, respectivamente. Assim, esperamos rejeitar a hipótese nula H_0 para obter evidência de que as médias diferem e, consequentemente, o projeto esquemático influenciou no resultado.

Uma das suposições para realização do teste t é a normalidade dos dados, podemos nos embasar no teorema do limite central para justificar o atendimento a essa condição. O teorema postula que a distribuição amostral das médias de uma população irá se aproximar da normal quando o número de amostras coletadas

aumentar. Isso pode ser intuído observando-se que a probabilidade de um resultado se distanciar da média diminui gradativamente ao passo em que se distancia dela. No que se refere ao tamanho amostral ideal, é possível estimar-se o menor tamanho necessário para serem alcançados níveis pré-determinados de poder estatístico (1-β) e de probabilidade da hipótese nula ser rejeitada indevidamente (α) mas, para isso, precisamos conhecer as médias dos dois grupos e os seus desvios-padrão, os quais, neste caso, são desconhecidos de antemão.

A fim de melhor fundamentar a escolha do número de iterações do experimento, nós executamos um teste piloto com alguns dos esquemas para obtermos valores reais de médias e desvios-padrão para o uso na estimação do tamanho amostral. Assumindo valores para α e β, e sabendo as médias e os desvios dos dois grupos amostrais, se torna possível calcular o tamanho da amostra n_A que irá conferir significância estatística ao teste. A equação apresentada abaixo faz tal cálculo, onde temos que *k* é a razão entre os tamanhos amostrais, σ_A é o desvio padrão do esquema A, σ_B o desvio do esquema B, α a taxa de erro do tipo I e β a taxa de erro do tipo II.

$$n_A = (\sigma_A^2 + \sigma_B^2 \cdot k^{-1}) \cdot [(Z_{1-\alpha} + Z_{1-\beta}) \cdot (\mu_A - \mu_B)^{-1}]^2$$

Adotamos α = 5% e β = 80% para efeitos de estimação, executamos o experimento piloto algumas vezes para esquemas diferentes e utilizamos a calculadora online HyLown [54] para estimar o tamanho amostral. O maior tamanho foi encontrado para os esquemas CG2-GG1-FDN-F1 e CG2-GG3-FDN-FN para o fator de escala equivalente a 1. Os dois esquemas compartilham de muitas similaridades em suas abordagens, logo, dadas as médias muito próximas e a magnitude dos desvios-padrão, são dois dos que precisam de um número elevado de repetições para obtermos significância estatística. Os valores obtidos foram μ_1 = 547.554,07, σ_1 = 64.028,62, μ_2 = 536.227,03, e σ_2 = 58.764,69 para as 30 execuções, logo, seriam necessárias aproximadamente 364 repetições. Um número muito grande se comparado com os obtidos nas demais análises. Assim, por conta da excepcionalidade do cenário e da limitação de tempo para a conclusão deste estudo, optamos por manter as 30 repetições para o fator de escala 1. Já no caso do

fator 10, quando analisados os mesmos esquemas anteriores, os valores obtidos são μ_1 = 2.042.726,20, σ_1 = 220.858,42, μ_2 = 2.326.231,00, e σ_2 = 167.595,74 para as mesmas 30 execuções. O cálculo do tamanho amostral para esse caso mostrou que seriam necessárias apenas 6 repetições para a rejeição da hipótese nula, logo, como nos limitados a testar um único cenário para o fator de escala 10 (o pior do fator 1), fixamos a quantidade de execuções em 10 repetições para que o experimento seja executado em tempo hábil e cubra cenários piores entre dois outros esquemas que não tenham sido avaliados no experimento piloto.

Essa metodologia utilizada na definição dos tamanhos amostrais incumbe ao experimento o risco de determinados resultados não poderem ser comparados com significância estatística e, portanto, as conclusões advindas de tais comparações perderem validade. Isso irá ocorrer para os casos em que a diferença das médias seja pequena quando comparada com os tamanhos dos desvios padrões das amostras, como é o caso dos dois esquemas mencionados no parágrafo anterior. No entanto, dado que não dispomos do tempo necessário para a coleta de todas as 364 amostras estimadas e isso se mostraria um esforço desnecessário para muitos dos cenários, escolhemos trabalhar com tamanhos amostrais de 30 e 10 amostras para os fatores de escala 1 e 10, respectivamente. Os casos que não tenham sido identificados no teste piloto e cujos tamanhos amostrais superem os escolhidos, serão identificados após a execução do experimento por meio da realização do Teste t de Student. Caso o número de tais ocorrências seja elevado, é possível que o experimento seja comprometido e se faça necessário uma nova execução com tamanhos amostrais maiores para termos significância.

4.4.3 Ameaças e Validade do Experimento

Nas subseções precedentes deste capítulo, contextualizamos o planejamento experimental criado para a avaliação e comparação dos diferentes caminhos existentes para a construção de esquemas voltados para aplicações analíticas envolvendo dados geoespaciais no banco colunar HBase. Sob a ótica experimental, quando nos referimos aos parâmetros estruturais e geoespaciais, podemos ver os mesmos como as variáveis independentes do experimento, onde, cada caminho proposto seria o tratamento possível que elencamos para ser avaliado. Esse

intercâmbio dos valores atribuídos as variáveis independentes, ou fatores, é o que caracteriza o processo experimental. Além delas, ainda nesse escopo, temos as variáveis independentes. Elas caracterizam as saídas do experimento e são o efeito da combinação dos valores assumidos pelas variáveis independentes. Podem ser tidas como os resultados obtidos e que servem de base para a comparação entre os diferentes tratamentos, no caso, conforme posto, seriam os tamanhos e os tempos de ingestão e de execução das consultas.

O âmbito do experimento é cercado de ameaças que podem invalidar as conclusões observadas por meio das variáveis dependentes, colocando em cheque os tratamentos (i.e. fatores). A discussão da validade de um experimento é destrinchada entre validade interna e externa. A validade interna é pertinente aos efeitos do tratamento, definindo se o relacionamento observado entre o tratamento e o resultado é causal e não decorre de fatores que não foram controlados e medidos [49]. Vendo de outro modo mais direto, a validade interna é violada quando existem outras explicações também plausíveis para os resultados. Já a validade externa, segundo Travassos et al. [49], define aspectos que limitam a habilidade de generalizar os resultados do experimento para a prática industrial, ou seja, se os resultados são válidos em ambientes reais.

Elencando e analisando as ameaças à validade interna, temos:

- A primeira ameaça que está presente em nosso estudo e em muitos dos que envolvem a execução de experimentos computacionais é a impossibilidade de replicar as condições do ambiente computacional entre diferentes rodadas de execução. Aspectos como operações do sistema operacional, tratamento de falhas de I/O, congestionamento da rede e outros não podem ser mantidos constantes e imutáveis entre execuções distintas, logo, eventuais anomalias podem surgir e impactar na inserção dos dados na banco, execução de consultas e até mesmo nas medições. Para minimizar esses potenciais efeitos danosos garantimos que, das aplicações que conseguimos controlar, apenas os softwares atrelados à execução do experimento estarão rodando durante a experimentação. As máquinas são reiniciadas a cada rodada e nenhum agente externo tem acesso ao cluster até que todos os resultados

tenham sido coletados para uma dada rodada. Esta problemática também é ajudada pelas repetições do experimento, uma vez que, podemos identificar a presença de *outliers* e agir para identificar se algo de errado aconteceu durante o experimento.

- Outra ameaça à validade interna do experimento consiste no uso da política padrão que é utilizada pelo HBase para a distribuição das regiões de tabelas. Em tal configuração, o HBase pode levar a concentração de regiões em poucas máquinas quando as tabelas em questão possuírem poucos registros. Desta forma, determinados esquemas podem ser mais afetados do que outros e os seus resultados inferiores podem advir da má distribuição de regiões, e não das escolhas realizadas no projeto esquemático dele. Conforme mencionado, a política de distribuição de regiões não entrou no escopo desse projeto de pesquisa e foi mantida a padrão do HBase. Um estudo futuro poderia adotar tal variável independente e investigar os efeitos de diferentes políticas quando combinadas com nossos projetos esquemáticos.

- Os swaps de memória também são uma ameaça à validade interna do experimento, pois, a movimentação dos blocos de memória entre a memória principal e o disco são operações que demandam quantidade expressiva de tempo e podem afetar o desempenho de certos esquemas que incorrem na realização demasiada de tais operações. Assim, bem como no caso da ameaça anterior, a razão para o desempenho inferior pode advir da ocorrência dos swaps de memória e não de escolhas ruins no projeto esquemático. É fato que as escolhas possam provocar os swaps mas, em ambientes com mais ou menos memória, poderíamos obter resultados diferentes em razão da menor ou maior ocorrência, respectivamente.

- A quarta e última ameaça elencada advém da definição das consultas. Nosso estudo pautou sete consultas envolvendo predicados geoespaciais, os quais, selecionam registros dentro de polígonos de cidades, países e regiões, além de registros englobados por um raio com o centro em um ponto qualquer do espaço. As sete consultas cobriram operações analíticas dos tipos *slice*, *dice*,

drill-down e *roll-up*, mas não foram diversas o suficiente para avaliar números variados de atributos e dimensões envolvidas, por exemplo. Além disso, os pontos escolhidos no espaço através da operação *point-to-polygon* [45] podem não representar de forma adequada a distribuição dos dados gerados pelo Spadawan, ou seja, as consultas podem ter sido executadas sobre áreas com quantidades muito pequenas ou muito grandes de registros, sem que tal aspecto tenha sido controlado no experimento.

Voltando a atenção para a validade externa do experimento, temos:

- A limitação do fator de escala utilizado, que fora limitado a 1 e 10 vezes o tamanho padrão do benchmark Spadawan. Existem aplicações reais envolvendo o HBase, como aplicações de crawling do Yahoo, que chegaram a armazenar mais de 1 PB. Os sistemas reais também costumam utilizar clusters mais poderosos, tendo dezenas ou centenas de máquinas, para os quais estão disponíveis mais poder de processamento paralelo e armazenamento.

- Uma outra ameaça à validade externa identificada advém do uso do Spadawan, uma vez que, ele não escala a quantidade e complexidade dos dados geoespaciais juntamente com o fator de escala. O número de regiões, nações e cidades permanece constante, apesar de serem criados mais registros de endereços quando o fator de escala aumenta. O número fixo de regiões retrata a realidade mas, numa aplicação real, a quantidade de países e das cidades contidas em cada um deles deveria escalar juntamente com o fator.

- Um ponto relevante sobre ameaças à validade externa presente em experimentos que usam múltiplos tratamentos é a perda do poder de generalização, uma vez que, os primeiros são passíveis de causarem um efeito cumulativo que mascara os efeitos dos subsequentes. Isso não ocorre em nosso experimento por conta dos projetos esquemáticos serem exaustivos dentro dos parâmetros que nos propusemos avaliar (i.e. estruturais e geoespaciais)

5 RESULTADOS

Neste capítulo, apresentaremos os resultados dos testes experimentais e discutiremos a influência de cada parâmetro estrutural e geoespacial no volume, tempo de ingestão e tempo para a execução das consultas analíticas. Inicialmente, trataremos dos impactos da normalização e desnormalização de dados convencionais e geoespaciais no volume de dados de cada esquema (Seção 5.1). Depois disso, iremos analisar o tempo de ingestão que cada esquema demandou e se tal variável tem uma correlação com o volume (Seção 5.2). Em seguida, trataremos dos tempos de execução da carga de consultas para cada um dos esquemas, focando nos impactos positivos e negativos das abordagens envolvidas na concepção deles (Seção 5.3). E, finalmente, as considerações finais (Seção 5.4).

5.1 VOLUME DE DADOS

A quantidade de armazenamento necessário para a implantação de um esquema é um dos aspectos mais importantes no projeto esquemático, o qual, precisa ser estimado a priori para não inviabilizar a escalabilidade do sistema nem incorrer em custos financeiros demasiados. Assim, a Tabela 4 traz uma lista com os 40 esquemas propostos ordenados do menor para o maior volume para o cenário com fator de escala equivalente a 1. Se forem ordenados com base no fator 10, teremos alterações pontuais na ordem de esquemas adjacentes. No entanto, a ordem dos melhores e piores colocados permanecerá inalterada. Essas diferenças não prejudicaram a análise porque são inferiores a 0,5% dos volumes envolvidos e, portanto, são inexpressivas. Ainda na mesma tabela, temos marcadores indicando quais dimensões geoespaciais foram normalizadas, bem como, se todas as convencionais também foram normalizadas em cada um dos esquemas.

É fato que a normalização contribui com a diminuição do volume de dados, uma vez que, diminui a redundância indiscriminada, pois, incorre apenas na redundância das referências. Desse modo, nós desejamos analisar de forma independente quais foram os impactos da normalização de dimensões convencionais e das geoespaciais no volume de dados demandado por cada esquema. Para melhor compreender o potencial impacto da normalização de tais

dimensões, remetemos aos números de registros existentes em cada dimensão que estão disponíveis na Tabela 3. Na tabela, vemos que as dimensões geoespaciais *c_address* e *s_address* possuem as mesmas quantidades de linhas das duas tabelas que as referenciam, respectivamente, *customer* e *supplier*. Dessa maneira, abordagens que preguem a normalização de tais dimensões não irão contribuir com a redução da redundância neste caso específico. Já a normalização das dimensões geográficas *city*, *nation* e *region* contribuirá para a redução da redundância, pois, possuem quantidades pequenas de registros quando comparadas com as quantidades das dimensões *customer* e *supplier*. Essa mesma linha de raciocínio se aplica a normalização das dimensões convencionais, que são mais de 300 vezes inferiores a tabela de fatos.

Dado que as dimensões geoespaciais mais seletivas *c_address* e *s_address* não se relacionam e têm a mesma hierarquia geográficas perante as demais, as abordagens propostas para a construção dos esquemas tratam as duas igualmente e, por isso, condensamos a informação sobre a normalização delas em uma mesma coluna da Tabela 4, intitulada de *address*. Conforme pode ser constatado na tabela, entre os oito primeiros colocados, a normalização ou a desnormalização da dimensão mais seletiva *address* (i.e. c_address e s_address) não alterou o volume de dados de forma expressiva. Os volumes desses oito esquemas flutuaram entre 3,79 GB e 4,00 GB, apesar de tal dimensão ter sido normalizada apenas no terceiro e quarto colocados. Essa observação pode ser justificada pelo fato das dimensões *customer* e *supplier* terem um relacionamento de cardinalidade 1:1 com as tabelas que armazenam os endereços de tais entidades, *c_address* e *s_address*, respectivamente. Assim, a normalização não irá diminuir a redundância e a desnormalização não irá aumentá-la.

Tabela 4 - Tempo de carga, volume e as normalizações dos esquemas.

							Fator 1		Fator 10	
Nº	**Modelo**	**address**	**city**	**nation**	**region**	**Conv.**	**Vol.**	**Carreg.**	**Vol.**	**Carreg.**
1	CG1-GG3-FDN-F1	-	✔	✔	✔	✔	3,79	48,18	39,09	475,17
2	CG1-GG3-FDN-FN	-	✔	✔	✔	✔	3,79	48,98	39,09	483,45
3	CG2-GG3-FDN-F1	✔	✔	✔	✔	✔	3,79	47,48	39,09	468,85
4	CG2-GG1-FDN-F1	✔	✔	✔	✔	✔	3,80	47,33	39,11	482,07
5	CG3-GG1-FDN-F1	-	✔	✔	✔	✔	3,80	48,20	39,11	484,57
6	CG3-GG1-FDN-FN	-	✔	✔	✔	✔	3,80	47,13	39,11	471,67
7	CG3-GG2-FDN-F1	-	✔	-	-	✔	4,00	48,60	39,30	486,87
8	CG3-GG2-FDN-FN	-	✔	-	-	✔	4,00	48,70	39,30	477,45
9	CG2-GG3-FDD-F1	✔	✔	✔	✔	-	16,92	192,37	174,55	1.933,75
10	CG2-GG3-FDD-FN	✔	✔	✔	✔	-	17,02	188,62	174,55	2.034,72
11	CG1-GG3-FDD-F1	-	✔	✔	✔	-	18,22	200,68	188,12	1.981,52
12	CG2-GG1-FDD-F1	✔	✔	✔	✔	-	18,22	203,67	188,22	2.050,82
13	CG1-GG3-FDD-FN	-	✔	✔	✔	-	18,32	203,45	188,22	1.983,92
14	CG2-GG1-FDD-FN	✔	✔	✔	✔	-	18,32	202,37	188,34	2.040,72
15	CG3-GG2-FDD-F1	-	✔	-	-	-	18,43	197,63	188,23	2.208,37
16	CG3-GG2-FDD-FN	-	✔	-	-	-	18,53	199,82	188,43	2.479,63
17	CG3-GG1-FDD-F1	-	✔	✔	✔	-	19,12	210,52	197,22	2.175,75
18	CG3-GG1-FDD-FN	-	✔	✔	✔	-	19,22	212,00	197,32	2.175,67
19	CG2-GG3-FDH-F1	✔	✔	✔	✔	-	25,62	201,67	266,15	2.036,38
20	CG2-GG3-FDH-FN	✔	✔	✔	✔	-	25,62	199,73	266,25	2.054,03
21	CG1-GG3-FDH-F1	-	✔	✔	✔	-	27,52	215,17	285,52	2.113,07
22	CG1-GG3-FDH-FN	-	✔	✔	✔	-	27,52	212,62	285,62	2.096,12
23	CG3-GG2-FDH-F1	-	✔	-	-	-	27,73	207,53	285,73	2.504,25
24	CG3-GG2-FDH-FN	-	✔	-	-	-	27,73	205,68	285,83	2.599,27
25	CG2-GG1-FDH-F1	✔	✔	✔	✔	-	27,82	215,90	288,64	2.147,77
26	CG2-GG1-FDH-FN	✔	✔	✔	✔	-	27,82	217,52	288,64	2.156,73
27	CG3-GG1-FDH-F1	-	✔	✔	✔	-	28,92	222,32	300,52	2.297,82
28	CG3-GG1-FDH-FN	-	✔	✔	✔	-	29,02	230,83	300,52	2.308,87
29	CG1-GG1-FDN-F1	-	-	-	-	✔	31,96	69,55	320,44	772,08
30	CG1-GG1-FDN-FN	-	-	-	-	✔	31,96	72,10	320,44	787,20
31	CG2-GG2-FDN-F1	✔	-	-	-	✔	31,97	74,88	320,44	816,00
32	CG2-GG2-FDN-FN	✔	-	-	-	✔	31,97	78,60	320,44	846,73
33	CG2-GG2-FDD-F1	✔	-	-	-	-	45,10	219,48	455,90	2.561,75
34	CG2-GG2-FDD-FN	✔	-	-	-	-	45,20	220,12	456,00	2.583,63
35	CG2-GG2-FDH-F1	✔	-	-	-	-	53,80	226,53	547,50	2.848,33
36	CG2-GG2-FDH-FN	✔	-	-	-	-	53,80	237,92	547,60	2.907,73
37	CG1-GG1-FDD-F1	-	-	-	-	-	>10TB	-	>10TB	-
38	CG1-GG1-FDD-FN	-	-	-	-	-	>10TB	-	>10TB	-
39	CG1-GG1-FDH-F1	-	-	-	-	-	>10TB	-	>10TB	-
40	CG1-GG1-FDH-FN	-	-	-	-	-	>10TB	-	>10TB	-

Fonte: autor (2022). Volume de dados (Vol.) em GB e tempo de carga (Carr.) em minutos.

Diferentemente das dimensões *c_address* e *s_address*, a normalização da dimensão *city* provocou a diminuição dos volumes de dados dos esquemas. Considerando a tabela em sua totalidade, temos que tal dimensão foi normalizada nos 28 primeiros colocados do total de 40 esquemas. Olhando as demais dimensões geográficas, percebemos que as dimensões *nation* e *region* foram normalizadas nos mesmos 28 primeiros colocados, com exceção de 6 deles. As exceções são esquemas oriundos da abordagem CG3-GG2, na qual, as dimensões menos seletivas *city*, *region* e *nation* acabam sendo organizadas em uma mesma tabela, onde, a dimensão mais seletiva (i.e. city) está normalizada e as demais desnormalizadas. São exceções porque a normalização da mais seletiva é suficiente para manter o volume de dados baixo em relação a potencial normalização das menos seletivas.

Bem como a normalização das dimensões geoespaciais, normalizar as dimensões convencionais foi um fator determinante para a redução do volume de dados. As dimensões *customer*, *supplier*, *date* e *part* aparecem normalizadas nos 8 melhores colocados. O maior volume dentre os esquemas em tal subgrupo foi de 4 GB, enquanto que, o nono colocado teve um aumento de 423% com o volume saltando para 16,92 GB em virtude dele não ter normalizado as convencionais. Na parte de baixo da tabela, entre os piores colocados, é possível constatar que nenhuma das dimensões convencionais ou geoespaciais foi normalizada. Eles compreendem os esquemas vindos das abordagens CG1-GG1 que desnormalizam todas as dimensões e, por isso, acarretam em muita redundância. Não tivemos capacidade de gerar e avaliar os quatro últimos em razão das nossas estimativas indicarem que eles precisariam de mais de 10 TB enquanto o nosso cluster só dispunha de 2,5 TB.

As constatações feitas podem ser sumarizadas em quatro pontos: (I) Normalizar ou desnormalizar dimensões geoespaciais de alta seletividade (i.e. c_address, s_address) não irá provocar impactos expressivos no volume de dados; (II) Normalizar dimensões geoespaciais de baixa seletividade leva a produção de esquemas que ocupam menos espaço de armazenamento; (III) A desnormalização de dimensões geográficas pouco seletivas não impacta negativamente o volume desde que elas não escalem registros de tabelas altamente seletivas; (IV) A

normalização de dimensões convencionais é importante para a redução dos volumes mas, não tanto, quanto a normalização das geográficas.

5.2 TEMPO DE INGESTÃO

O tempo para a ingestão dos dados no HBase é a segunda métrica que analisaremos. Esses dados estão dispostos na coluna "Carreg." da Tabela 4 para os dois fatores de escala. Como foi detalhado anteriormente, o processo de carga é todo realizado a partir do nó mestre do cluster e compreende etapas como o download e a descompactação de arquivos mas, para fins de comparação, apenas os tempos demandados para a execução dos comandos de criação de tabelas e inserção de dados são considerados no cálculo do tempo de ingestão. Ambas as operações são executadas através de uma aplicação que interage com o servidor Thrift do HBase. A execução ocorre de forma independente entre tabelas e, após o término, os tempos individuais são somados para comporem o total.

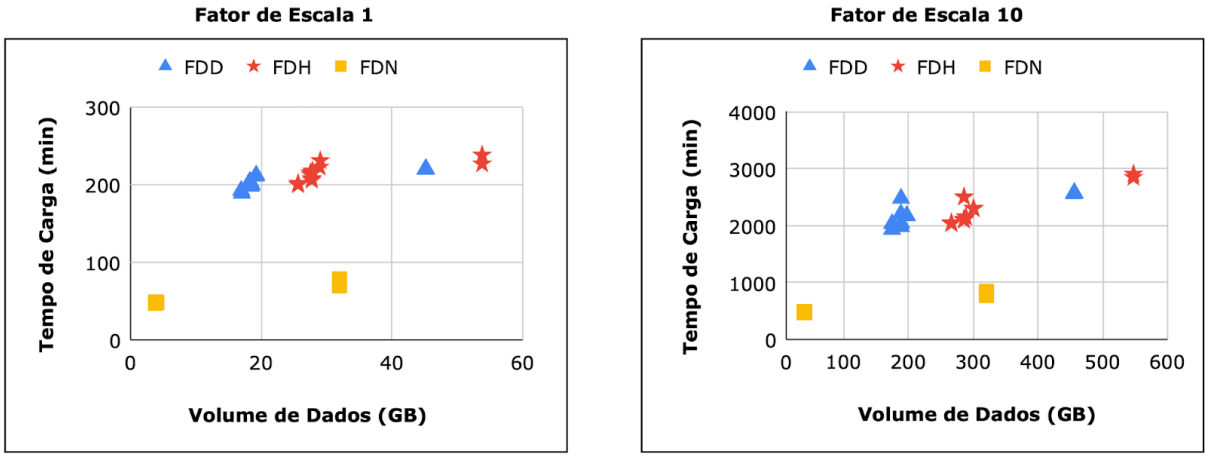
Apesar dos esquemas não estarem ordenados pelo tempo de ingestão, na Tabela 4, podemos notar a existência de uma aparente correlação entre os volumes de dados e os tempos de ingestão, uma vez que, os tempos crescem na medida que o volume cresce, salvo poucas exceções. Com o intuito de testar tal hipótese, calculamos o coeficiente de correlação de Pearson para as séries. Ele mede a relação estatística entre duas variáveis contínuas, onde, os resultados do teste são coeficientes que variam entre -1 e 1. Coeficientes negativos indicam que as variáveis são inversamente relacionadas e resultados positivos que elas são diretamente relacionadas. Quanto mais próximo dos extremos o coeficiente estiver, maior será a correlação. Realizando o teste de Pearson para as séries de volumes e tempos dois dois fatores de escala, encontramos os valores 0,60 e 0,69. Ambos os resultados são positivos, se aproxima do limite superior e sugerem a existência de uma correlação positiva.

O coeficiente de Pearson não foi maior em razão de duas excepcionalidades. A primeira decorre do descasamento entre a ordem de dois esquemas adjacentes quando ordenados pelas duas variáveis de forma independente. Como pode ser visualizado na Tabela 4 para o par composto dos esquemas CG2-GG3-FDD-F1 (9º)

e CG2-GG3-FDD-FN (10º), para o fator de escala 1, e o par de esquemas que é formado pelo CG2-GG3-FDD-FN (10º) e CG1-GG3-FDD-F1 (11º), para o fator 10. Nos dois pares, a ordem é invertida quando se ordena pelo volume ou tempo de ingestão, sugerindo que as variáveis possuem uma correlação negativa. No entanto, as diferenças entre os volumes e tempos é inferior a 5% dos valores envolvidos nesses casos, logo, eles pouco atestam contra a tese da correlação que constatamos. Já a segunda excepcionalidade, é muito mais expressiva. Ela é visualizada para o caso dos esquemas dispostos da 29º até a 32º colocação. Os volumes figuram entre os 12 piores, mas os seus tempos figuram entre os 12 melhores por terem normalizado as dimensões convencionais.

O impacto da normalização de dimensões convencionais no tempo de ingestão pode ser visualizado nos dois gráficos de dispersão apresentados na Figura 20. Neles, temos a relação entre os volumes e os tempos estratificados pela abordagem adotada para implementação dos relacionamentos dos fatos com as dimensões convencionais (i.e. FDD, FDH, FDN). Claramente, a normalização contribuiu para a redução dos tempos. Assim, temos que os tempos de ingestão aumentam significativamente na medida em que a redundância de dados convencionais aumenta, possivelmente, porque cresce o volume médio de dados que são inseridos para cada registro das tabelas.

Figura 20 - Volume vs. tempo de carga para FDD, FDN e FDH.



Fonte: autor (2022).

As constatações realizadas a partir da análise dos tempos de ingestão dos dados e o cruzamento de tais informações com os volumes de dados mostraram que: (I) Existe uma correlação positiva entre essas duas variáveis, logo, quanto maior o volume de dados, mais demorada será a ingestão; (II) As redundâncias impactam no volume e, consequentemente, nos tempos de ingestão. No entanto, os impactos são limiares desde que os dados redundantes ocupem pouco espaço e não incrementem de forma expressiva o volume médio de dados inseridos por linha da tabela; (III) A desnormalização das dimensões geoespaciais e o aumento do número de famílias por tabela não afetaram de forma negativa e significativa os tempos de ingestão tanto quanto ocorreu para os convencionais.

5.3 TEMPO DE EXECUÇÃO

Os tempos de execução da carga de consultas estão relacionados nas tabelas 5 e 6 para os cenários que utilizam os fatores de escala equivalentes a 1 e 10, respectivamente. Os esquemas aparecem ordenados, em ordem crescente, pela média de tempo para a execução das sete consultas. Iremos tratar primeiramente dos resultados obtidos para o fator de escala 1 e depois para o fator 10. Esses dois cenários serão analisados quanto aos impactos que cada parâmetro estrutural e geoespacial teve nos tempos de execução das consultas. Finalizaremos a seção com uma análise consolidada e comparativa entre os fatores de escala, parâmetros e resultados obtidos.

5.3.1 Fator de Escala 1

Os cinco melhores e cinco piores tempos de execução foram sinalizados com as marcas [+] e [-], ao lado esquerdo de seus valores na coluna $Q_{(1-7)}$ da Tabela 5. No grupo dos cinco melhores, temos os esquemas: CG2-GG1-FDH-FN (336,24 s), CG2-GG1-FDH-F1 (387,26 s), CG3-GG1-FDD-FN (403,69 s), CG2-GG1-FDD-FN (412,27 s) e CG3-GG2-FDD-FN (422,12 s). A diferença de tempo entre o melhor e o pior deles foi de 85,88 segundos enquanto que o desvio padrão médio foi de 26,29 segundos. Os cinco piores colocados da totalidade, excluindo os quatro que não puderam ser avaliados, foram os esquemas: CG1-GG1-FDN-FN (969,01 s), CG2-GG2-FDD-F1 (1018,74 s), CG3-GG2-FDD-F1 (1059,50 s), CG2-GG2-FDH-FN

(1.079,94 s) e CG2-GG2-FDH-F1 (3.559,14 s). O intervalo da diferença entre eles foi de 2.590,00 segundos com um desvio padrão médio de 178,00 segundos. Agora, olhando para a diferença entre o último dos primeiros e o primeiro dos últimos, obtemos 547,00 segundos com o desvio padrão da série oscilando entre 2,87% e 14,73% e a mediana deles de 8,81%.

Extrapolando dos cinco para os onze melhores resultados, nós podemos constatar que todos eles empregam uma abordagem desnormalizada para a implementação dos relacionamentos entre a tabela de fatos e demais dimensões convencionais (i.e. FDD, FDH). Um forte indicativo de que elas contribuem positivamente para a redução dos tempos de execução. Em relação ao uso das famílias de colunas para o particionamento vertical dos dados, dentre os cinco primeiros colocados, quatro designaram famílias distintas para dimensões distintas, portanto, um sinal de que os tempos são influenciados positivamente por tal abordagem (i.e. FN). Avaliando os impactos da organização dos dados geoespaciais, temos que a normalização dos menos seletivos (i.e. GG1, GG3) foi positiva e, a normalização ou desnormalização dos mais seletivos, não tiveram impactos sugestivos. A Figura 21 ilustra a superioridade da desnormalização das dimensões convencionais e, na Figura 22, temos que os melhores tempos foram obtidos para projetos esquemáticos que normalizaram, ao menos, uma das duas dimensões geográficas mais seletivas (i.e. c_address e s_address ou city).

Tabela 5 - Resultados do experimento com fator de escala 1.

Fator de Escala 1											
Nº	**Esquema**	**Vol.**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**$Q_{(1-7)}$**	**DP**
1	CG2-GG1-FDH-FN	27,82	46,26	45,17	42,24	46,67	45,37	50,68	59,85	[+] 336,24	29,02
2	CG2-GG1-FDH-F1	27,82	54,95	48,12	48,16	49,75	55,14	62,78	68,37	[+] 387,26	36,13
3	CG3-GG1-FDD-FN	19,22	49,09	45,46	48,01	74,19	51,37	56,00	79,58	[+] 403,69	18,56
4	CG2-GG1-FDD-FN	18,32	55,53	53,24	55,80	55,88	58,41	61,07	72,34	[+] 412,27	21,48
5	CG3-GG2-FDD-FN	18,52	50,09	51,34	51,42	72,59	56,06	57,26	83,37	[+] 422,12	26,24
6	CG3-GG2-FDH-F1	27,73	55,74	49,36	47,74	80,94	56,42	61,63	93,41	445,24	35,20
7	CG1-GG3-FDN-F1	(+) 3,79	71,21	68,13	63,97	69,14	69,54	59,83	63,13	464,94	51,03
8	CG2-GG3-FDH-F1	25,62	66,92	63,09	60,78	59,48	69,42	73,96	80,42	474,06	33,17
9	CG1-GG3-FDH-F1	27,52	61,50	54,74	53,86	83,65	63,23	69,36	97,35	483,68	42,37
10	CG1-GG3-FDD-FN	18,32	60,86	58,69	59,64	84,09	61,91	67,51	95,46	488,16	14,02
11	CG2-GG1-FDD-F1	18,22	71,27	63,44	60,82	69,02	72,70	72,18	81,78	491,20	39,64
12	CG1-GG3-FDN-FN	(+) 3,79	74,10	73,10	62,33	70,42	76,45	65,50	70,71	492,61	49,57
13	CG3-GG1-FDN-FN	3,80	75,09	67,73	68,60	73,52	77,16	68,55	67,68	498,34	63,34
14	CG2-GG3-FDH-FN	25,62	77,38	72,88	66,32	70,90	74,39	72,82	70,67	505,37	45,57
15	CG3-GG2-FDN-F1	4,00	69,12	80,98	75,00	66,53	73,11	69,25	74,05	508,05	74,85
16	CG3-GG2-FDN-FN	4,00	74,61	78,30	74,04	72,28	75,79	71,67	76,11	522,79	52,25
17	CG2-GG3-FDN-F1	(+) 3,79	81,05	75,62	73,27	75,20	81,98	76,68	72,41	536,22	58,76
18	CG3-GG1-FDN-F1	(+) 3,80	73,53	79,37	76,43	80,83	83,97	70,66	78,52	543,31	69,80
19	CG3-GG2-FDH-FN	27,73	82,38	82,41	75,93	73,91	80,63	77,20	74,91	547,36	43,10
20	CG2-GG1-FDN-F1	(+) 3,80	80,83	83,95	69,63	77,94	78,88	75,04	81,29	547,55	64,02
21	CG3-GG1-FDH-F1	28,92	76,96	74,45	77,37	90,15	76,79	77,99	103,40	577,11	43,13
22	CG1-GG3-FDH-FN	27,52	85,79	79,00	82,93	77,05	85,25	80,34	88,99	579,35	42,17
23	CG2-GG3-FDD-FN	17,02	80,18	76,06	74,21	77,82	81,83	91,79	101,73	583,62	28,32
24	CG3-GG1-FDD-F1	19,12	83,40	77,81	73,04	115,55	85,04	93,08	126,29	654,21	57,05
25	CG1-GG1-FDN-F1	31,96	192,78	100,27	68,17	66,82	104,99	66,24	72,01	671,28	59,45
26	CG3-GG1-FDH-FN	29,02	92,98	84,96	72,90	152,16	93,44	94,94	160,29	751,67	78,55
27	CG2-GG3-FDD-F1	16,92	105,33	102,52	98,65	107,44	118,65	118,85	125,82	777,26	71,97
28	CG2-GG2-FDN-F1	31,97	241,64	131,95	69,56	69,76	132,10	73,63	70,03	788,66	61,38
29	CG2-GG2-FDN-FN	(-) 31,97	309,72	159,79	48,75	44,50	166,29	47,43	49,43	825,91	59,36
30	CG2-GG2-FDD-FN	(-) 45,20	327,75	154,17	57,22	54,85	151,08	64,02	68,32	877,42	68,77
31	CG1-GG3-FDD-F1	18,22	102,76	97,76	102,74	174,18	115,93	136,43	202,94	932,73	96,67
32	CG1-GG1-FDN-FN	31,96	344,85	173,86	66,20	71,86	176,72	65,14	70,37	[-] 969,01	50,90
33	CG2-GG2-FDD-F1	(-) 45,10	243,88	126,80	107,98	116,07	134,36	145,92	143,74	[-] 1.018,74	82,47
34	CG3-GG2-FDD-F1	18,43	118,03	116,00	114,93	215,12	123,42	143,10	228,90	[-] 1.059,50	97,84
35	CG2-GG2-FDH-FN	(-) 53,80	343,89	177,77	82,32	79,42	171,95	109,49	115,11	[-] 1.079,94	92,33
36	CG2-GG2-FDH-F1	(-) 53,80	554,54	507,60	499,04	498,40	502,43	491,71	505,42	[-] 3.559,14	565,23
37	CG1-GG1-FDD-F1	>10TB	-	-	-	-	-	-	-	-	
38	CG1-GG1-FDD-FN	>10TB	-	-	-	-	-	-	-	-	
39	CG1-GG1-FDH-F1	>10TB	-	-	-	-	-	-	-	-	
40	CG1-GG1-FDH-FN	>10TB	-	-	-	-	-	-	-	-	

Fonte: autor (2022). O volume de dados (Vol.) em GB, tempo médio de execução por consulta (Q1…Q7) em segundos, a soma dos tempos (Q (1-7)) e o desvio padrão da soma (DP).

Tabela 6 - Resultados do experimento com fator de escala 10.

Fator de Escala 10											
Nº	**Esquema**	**Vol.**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**$Q_{(1-7)}$**	**DP**
1	CG1-GG3-FDN-FN	(+) 39,09	229	220	251	252	227	245	282	[+] 1.706	113,56
2	CG3-GG2-FDN-FN	(+) 39,30	320	255	262	277	238	276	285	[+] 1.913	215,05
3	CG3-GG1-FDN-FN	(+) 39,11	306	268	287	294	307	277	302	[+] 2.041	148,09
4	CG2-GG1-FDN-F1	(+) 39,11	340	282	300	278	287	300	256	[+] 2.043	329,14
5	CG3-GG1-FDN-F1	(+) 39,11	307	301	304	309	307	304	301	[+] 2.133	62,91
6	CG1-GG3-FDN-F1	(+) 39,09	310	352	336	339	269	327	318	[+] 2.250	115,03
7	CG3-GG2-FDN-F1	(+) 39,30	325	327	320	331	329	332	341	[+] 2.303	64,67
8	CG2-GG3-FDN-F1	(+) 39,09	340	348	381	294	298	349	316	[+] 2.326	167,59
9	CG3-GG1-FDD-FN	197,32	389	327	329	512	374	396	590	2.918	147,55
10	CG1-GG3-FDH-FN	285,62	642	598	587	654	598	566	641	4.286	503,49
11	CG1-GG3-FDD-FN	188,22	656	598	571	765	631	676	791	4.688	163,89
12	CG2-GG3-FDD-FN	174,65	733	788	776	749	750	714	704	5.213	627,47
13	CG2-GG2-FDN-FN	(-) 320,44	1.769	1.011	450	378	968	429	375	5.378	176,24
14	CG1-GG1-FDN-FN	(-) 320,34	2.005	1.180	426	321	1.148	462	380	5.921	364,94
15	CG3-GG1-FDH-FN	300,52	852	873	854	889	857	840	892	6.057	44,60
16	CG2-GG1-FDH-FN	288,64	915	902	901	926	903	905	920	6.372	32,86
17	CG3-GG2-FDD-FN	188,43	905	899	910	936	915	901	941	6.406	59,43
18	CG3-GG2-FDH-FN	285,83	956	1.050	986	1.092	986	983	942	6.996	220,89
19	CG2-GG1-FDD-FN	188,34	1.033	1.032	1.034	1.041	1.031	1.020	1.025	7.216	35,59
20	CG2-GG2-FDD-FN	(-) 456,00	2.540	1.379	640	486	1.318	640	567	7.570	430,98
21	CG2-GG3-FDH-FN	266,25	1.147	1.125	1.122	1.143	1.117	1.117	1.132	7.903	191,93
22	CG2-GG3-FDD-F1	174,55	1.170	1.185	1.165	1.174	1.178	1.196	1.183	8.250	48,30
23	CG2-GG1-FDD-F1	188,22	1.285	1.273	1.273	1.276	1.279	1.275	1.293	8.955	38,69
24	CG2-GG2-FDH-FN	(-) 547,60	3.192	1.482	856	777	1.263	876	784	9.230	282,53
25	CG1-GG3-FDD-F1	188,12	1.481	1.458	1.457	1.512	1.476	1.483	1.544	10.411	111,08
26	CG3-GG1-FDD-F1	197,22	1.711	1.698	1.677	1.723	1.692	1.686	1.728	11.914	137,52
27	CG3-GG2-FDD-F1	188,23	2.037	1.676	1.797	1.820	1.802	1.814	1.875	12.822	847,49
28	CG3-GG1-FDH-F1	300,52	1.909	1.896	1.888	1.871	1.881	1.843	1.865	13.153	57,53
29	CG3-GG2-FDH-F1	285,73	2.074	2.019	2.037	1.985	2.037	1.996	1.990	[-] 14.139	706,94
30	CG2-GG3-FDH-F1	266,15	2.592	2.551	2.588	2.599	2.581	2.578	2.597	[-] 18.086	176,08
31	CG1-GG3-FDH-F1	285,52	2.761	2.641	2.727	2.642	2.750	2.634	2.655	[-] 18.811	255,48
32	CG1-GG1-FDN-F1	(-) 320,44	2.824	2.755	2.846	2.852	2.825	2.839	2.844	[-] 19.785	207,48
33	CG2-GG1-FDH-F1	288,64	2.976	2.960	3.058	2.986	3.000	2.979	2.980	[-] 20.939	624,93
34	CG2-GG2-FDN-F1	(-) 320,44	3.400	3.248	3.409	3.519	3.364	3.590	3.491	[-] 24.022	753,09
35	CG2-GG2-FDD-F1	(-) 455,90	5.293	5.283	5.235	5.219	5.184	5.136	5.302	[-] 36.651	554,13
36	CG2-GG2-FDH-F1	(-) 547,50	6.442	5.636	5.813	6.751	7.187	6.472	6.112	[-] 44.412	1.266,30
37	CG1-GG1-FDD-F1	>10TB	-	-	-	-	-	-	-	-	-
38	CG1-GG1-FDD-FN	>10TB	-	-	-	-	-	-	-	-	-
39	CG1-GG1-FDH-F1	>10TB	-	-	-	-	-	-	-	-	-
40	CG1-GG1-FDH-FN	>10TB	-	-	-	-	-	-	-	-	-

Fonte: autor (2022). O volume de dados (Vol.) em GB, tempo médio de execução por consulta (Q1…Q7) em segundos, a soma dos tempos (Q (1-7)) e o desvio padrão da soma (DP).

A análise dos cinco piores resultados evidencia que a desnormalização das dimensões geoespaciais menos seletivas (i.e. GG2) foi a abordagem que mais influenciou negativamente os resultados, ela está presente em quatro dos cinco piores esquemas. Ainda sobre as dimensões geoespaciais, fazer a normalização das mais seletivas (i.e. CG2) contribuiu para aumentar os tempos das consultas, em especial, quando as menos seletivas foram desnormalizadas (i.e. CG2-GG2), o que, ocorreu em três dos cinco piores. A adoção de abordagens mais normalizadas ou desnormalizadas, para dimensões convencionais, não aparentam ter tido efeitos negativos. Já as abordagens que organizam os dados de dimensões distintas em famílias compartilhadas (i.e. F1), foram muito prejudiciais nos cenários que envolvessem desnormalizações. Em 5º lugar, temos o esquema CG3-GG2-FDD-FN (422,12 s) e em 34º o esquema CG3-GG2-FDD-F1 (1.059,50 s), onde, eles diferem apenas pela forma como são organizados os dados em famílias, mas o primeiro teve um desempenho 250% superior ao seu par.

As conclusões supramencionadas foram feitas a partir da comparação entre as médias dos tempos de execução da carga de consultas, no entanto, para conferir significância estatística aos resultados apresentados, nós utilizamos o teste t de Student para a comparação de médias. Dados quaisquer dois esquemas A e B, cujas séries de resultados tenham médias μ_A e μ_B, nós formulamos a hipótese nula H_0: $\mu_A - \mu_B = 0$ e a hipótese alternativa H_1: $\mu_A - \mu_B \neq 0$. Após isso, executamos o teste estatístico com nível de significância de 5%, no qual, desejamos rejeitar a hipótese nula para que seja indicada a existência de significância estatística na diferença entre as médias das duas amostras. Fizemos os testes para todos os esquemas, dois a dois, e ilustramos os resultados dos p-values na Figura 23, na forma de uma matriz quadrada. As linhas e colunas da matriz são identificadas pela numeração que segue a mesma ordem das tabelas 5 e 6 para os fatores de escala 1 e 10, respectivamente. Assim, a célula na linha 15 e coluna 20 contém o p-value para o teste t de Student quando realizado com as séries dos esquemas colocados na 15º e 20º posição da tabela, respectivamente.

No gráfico do lado esquerdo da Figura 23, temos a matriz de p-values para as execuções do teste t para as séries do experimento com fator de escala 1. Analisando a figura, vemos que as linhas dos cinco primeiros e cinco últimos

esquemas estão, predominantemente, pintadas de preto. Isto indica que as diferenças das médias deles possuem significância estatística (p-value < 0,05) e, portanto, as comparações deles com os demais são válidas. Um ponto de atenção deve ser a zona com coloração mais clara que vai do 9º até o 20º colocado. Analisando os tempos de tais esquemas, constatamos que eles diferiram pouco e as diferenças foram inferiores a um desvio padrão para vários deles. Isso mostra que não é possível extrair conclusões estatisticamente válidas apenas comparando as suas médias e, portanto, seria necessário aumentar o tamanho amostral para tais casos.

5.3.2 Fator de Escala 10

Bem como para a outra tabela, utilizamos as marcações [+] e [-] para sinalizar os melhores e piores tempos de execução e, as marcações (+) e (-), para os volumes. No grupo dos cinco melhores, temos os esquemas: CG1-GG3-FDN-FN (1.706 s), CG3-GG2-FDN-FN (1.913 s), CG3-GG1-FDN-FN (2.041 s), CG2-GG1-FDN-F1 (2.043 s) e CG3-GG1-FDN-F1 (2.133 s). O intervalo de tempo entre o 1º e o 5º foi equivalente a 427 segundos e o desvio padrão médio de 174 segundos. Já na parte de baixo, temos os modelos: CG1-GG1-FDN-F1 (19.785 s), CG2-GG1-FDH-F1 (20.939 s), CG2-GG2-FDN-F1 (24.022 s), CG2-GG2-FDD-F1 (36.651 s) e CG2-GG2-FDH-F1 (44.412 s). A diferença de tempo entre os extremos desse grupo foi de 24.627 segundos com desvio padrão médio de 628 segundos. Olhando o último colocado dentre os melhores e o melhor colocado dentre os piores, tivemos um intervalo de tempo equivalente a 17.653 segundos com a mediana dos desvios-padrão em 1,36%.

Analisando os esquemas mais bem colocados para o fator de escala 10, dispostos na parte superior da tabela 6, vemos uma aparente coincidência entre os menores volumes e os menores tempos de execução. Os volumes dos oito melhores colocados são quase 5 vezes inferiores ao volume do nono colocado e, a média de tempo deles, também é 40% inferior. Assim, podemos concluir que o volume baixo foi um fator determinante para os oito primeiros colocados terem conseguido finalizar a carga de consultas em menos tempo do que os demais. Os volumes baixos de tais esquemas decorrem da normalização das dimensões convencionais e grande parte

das geográficas, logo, as abordagens que levam a tais projetos esquemáticos (i.e. FDN) também contribuem para a redução dos tempos.

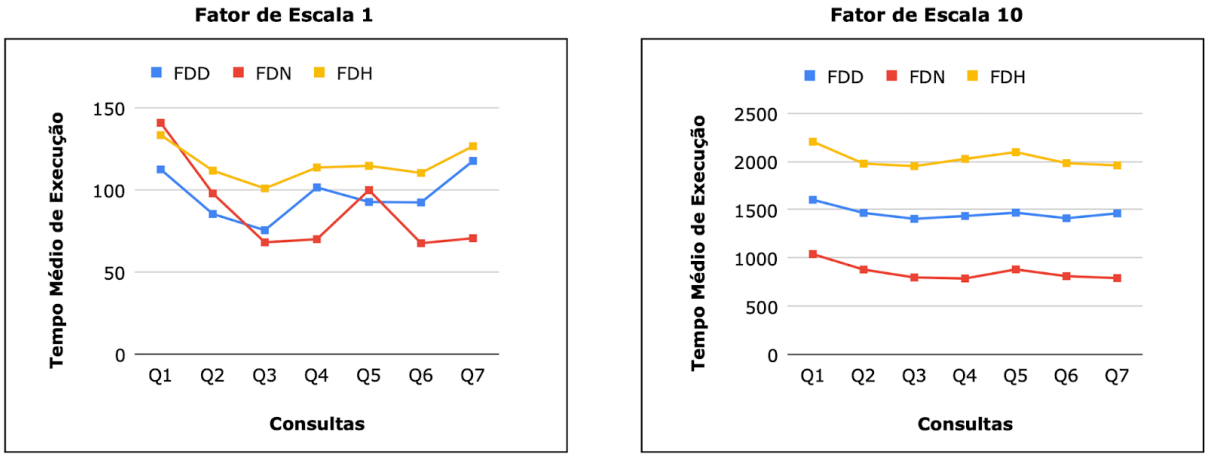
Dentre os oito melhores, as dimensões convencionais foram normalizadas em todos eles. Os dados geográficos mais seletivos foram normalizados em seis casos e, os menos seletivos, em cinco. Essa aparente vantagem obtida através da normalização é contraditória com as conclusões dos trabalhos relacionados e também com as constatações do experimento anterior, onde, os custos das junções incorridas pela adoção de abordagens normalizadas influenciaram negativamente. Essa discrepância decorre dos oito melhores colocados serem os únicos esquemas cujos dados cabem por inteiro na memória principal do nosso *cluster*, enquanto que, o processamento dos demais ultrapassa o limite dos 144 GB disponíveis e se faz necessário o uso da área de *swap*. Em resumo, o custo da execução de junções, quando todos os dados cabem na memória principal, foi inferior ao custo incorrido pelas operações de *swap* (trocas constantes de dados entre memória principal e secundária).

Desconsiderando os esquemas cujos volumes de dados cabem na memória principal (1º até o 8º), é possível constatar que as considerações feitas para os melhores resultados com fator de escala 10 continuam válidas. A partir do nono, temos: CG3-GG1-FDD-FN (2.918 s), CG1-GG3-FDH-FN (4.286 s), CG1-GG3-FDD-FN (4.688 s), CG2-GG3-FDD-FN (5.213 s) e CG2-GG2-FDN-FN (5.378 s). Os primeiros quatro esquemas desnormalizam as dimensões convencionais (i.e. FDD, FDH), os três melhores têm a dimensão geográfica mais seletiva desnormalizada e todos normalizaram, ao menos, a dimensão geoespacial com a segunda maior seletividade. O destaque vai para o primeiro esquema citado, que conseguiu superar o segundo em mais de 1.000 segundos, ou, uma redução de 32% no tempo.

Voltando a nossa atenção para os piores resultados, constatamos que os oito últimos fizeram uso de apenas uma única família de colunas por tabela (i.e. F1), os quatro piores normalizaram os dados geoespaciais mais seletivos (i.e. CG2) e os três piores desnormalizaram os dados geográficos menos seletivos (i.e. GG2). Conforme pode ser visto na Figura 22, as abordagens CG1-GG2 e CG2-GG2 são as

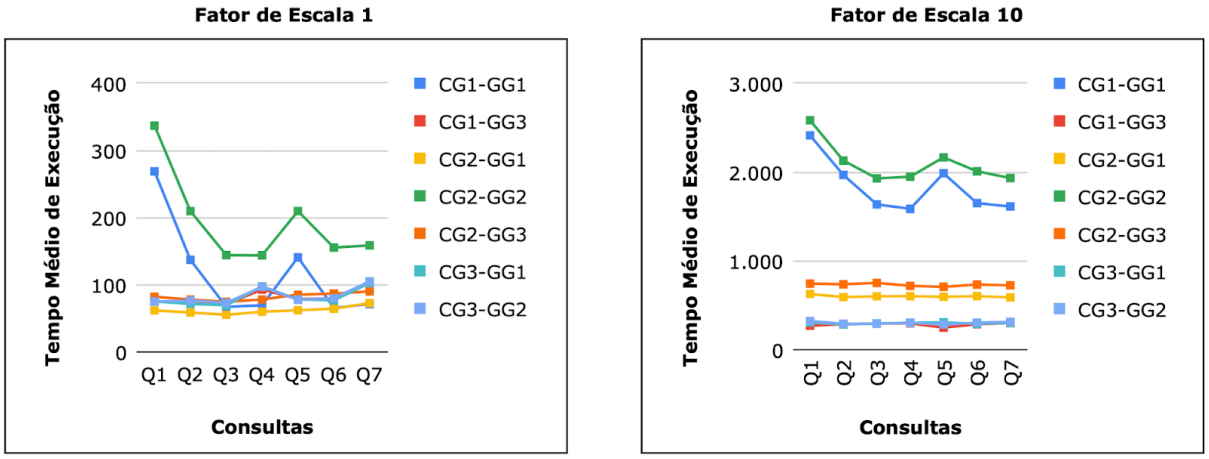
duas com o maior nível de redundância dos dados geoespaciais menos seletivos e foram as que tiveram os piores resultados em média. Novamente, da normalização e desnormalização, a adoção de apenas uma única família de colunas por tabela se mostrou uma escolha ruim. Como foi o caso dos esquemas CG2-GG1-FDH-FN (6.372 s) e CG2- GG1-FDH-F1 (20.939 s) que diferem unicamente por tal fator mas os tempos tiveram uma diferença de 14.567 segundos, ou seja, a organização dos dados entre famílias de colunas cortou o tempo de execução em um terço.

Figura 21 - Consultas vs. tempo médio de execução para CG-GG.



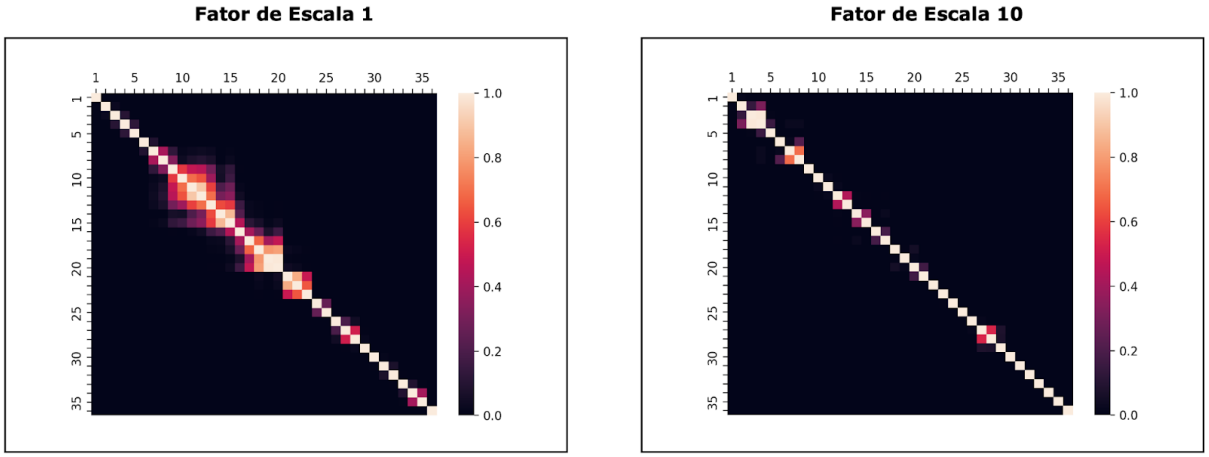
Fonte: autor (2022).

Figura 22 - Consultas vs. tempo médio de execução para FDD, FDN e FDH.



Fonte: autor (2022).

Figura 23 - Mapa de calor dos p-values comparando médias.



Fonte: autor (2022).

5.3.3 Análise Consolidada

Após a análise preliminar dos resultados, segmentados pelo fator de escala dos experimentos, nós iremos examinar e justificar as convergências e divergências encontradas. Em relação aos melhores resultados para o fator 1, determinamos que a desnormalização das dimensões convencionais foi a abordagem que mais influenciou positivamente o tempo de execução das consultas. A redundância de tais dados convencionais ajudou a eliminar ou, no mínimo, cortar drasticamente a quantidade de junções realizadas e isso ocasionou os reflexos positivos. Além disso, constatamos que normalizar as dimensões geoespaciais menos seletivas também contribuiu positivamente, enquanto que, fazer a normalização ou desnormalização das de maior seletividade não mostraram uma tendência clara de qual seria a melhor abordagem dentre elas. O último aspecto esquemático avaliado é a quantidade de famílias de colunas criadas por tabela. Nós identificamos que a disposição dos dados em famílias distintas, com base nas dimensões de origem, foi a melhor abordagem e contribuiu para a redução dos tempos de execução das consultas expressivamente em diversos casos.

As constatações supramencionadas corroboram com as proposições e conclusões de [14-17], dado que, a desnormalização das dimensões convencionais contribuiu expressivamente para a redução dos tempos de execução da carga de

consultas, uma vez que, tal abordagem reduziu as penalidades de tempo incorridas pela execução das junções. No escopo dos dados geoespaciais, a normalização das dimensões menos seletivas contribuíram para a redução dos volumes de dados ocupados pelos esquemas e dos tempos de execução consultas, conforme posto por [20], mas, diferentemente de tal estudo, nossos resultados não mostraram ganhos expressivos com a desnormalização das mais seletivas. Acreditamos que isso tenha ocorrido em razão dos ganhos obtidos com a desnormalização das dimensões convencionais serem muito mais significativos do que os obtidos desnormalizando a dimensão geoespacial mais seletiva para os casos de uso com o HBase. Ferro et al. comenta que no caso do MongoDB, os efeitos do swap de memória foram severos e os modelos que dependiam de tais execuções ficaram em piores colocações. Podemos fundamentar tal suposição ao observarmos que dos cinco melhores esquemas que não desnormalizaram as dimensões convencionais, todos realizaram a normalização da dimensão geográfica mais seletiva.

Os melhores resultados obtidos para o experimento realizado com fator de escala 10 não seguiram as mesmas abordagens esquemáticas dos obtidos para o fator 1. Analisando os dados, podemos ver que os esquemas com os menores tempos são, coincidentemente, aqueles que ocupam os menores volumes de armazenamento e, portanto, seguem abordagens normalizadas tanto para dimensões convencionais como para muitas das geoespaciais. Isso mostra uma discrepância com os resultados encontrados para o fator de escala 1 e com as conclusões dos trabalhos relacionados mas, a qual, é justificada pelo fato dos melhores esquemas serem justamente aqueles cujos dados cabem em sua totalidade na memória principal do *cluster*. As três máquinas que são efetivamente utilizadas como *datanodes* do Hadoop, *regionservers* do HBase e *workers* do Spark possuem um total de 144 GB de memória principal, mas precisam dividir tal recurso entre si e com os sistemas operacionais. Dessa forma, os esquemas cujo volume de dados supera tal marca, incorrem em mais operações de swap de memória e acabam sendo prejudicados perante os que não precisam delas.

Um aprofundamento da análise dos swaps de memória se faz necessário para confirmar as suspeitas levantadas, dado que, sem isolar tal variável, não temos como validar estatisticamente que exista uma correlação negativa entre um bom

desempenho na execução das consultas e a quantidade de swaps realizadas durante a execução das mesmas. Apesar de sabermos o espaço ocupado por cada um dos esquemas e a totalidade de memória principal disponível, a divisão de tais dados entre as máquinas do cluster e o uso de cada máquina não são perfeitamente balanceados pelo HBase e isso impacta nossa intuição inicial explanando a superioridade dos esquemas mais compactos.

Quando desconsideramos os esquemas cujos dados couberam na memória principal, para o caso do experimento com fator de escala 10, constatamos que as mesmas conclusões tiradas da primeira execução permanecem válidas. Nosso estudo não avaliou cenários com fatores de escala superiores a 10 mas, nesses casos, seria esperado que nenhum dos esquemas pudesse ser totalmente posto na memória e as abordagens desnormalizadas voltassem a figurar entre as melhores. Em aplicações reais que envolvem dados geoespaciais, os volumes de dados tenderão a serem muito superiores aos simulados neste estudo e com maior variedade de dados geográficos, logo, conduzir a avaliação experimental de tais condições irá complementar o nosso estudo.

Apesar dos melhores resultados terem divergido, por conta dos *swaps* de memória, os piores foram unânimes em apontar as abordagens que resultaram nos maiores tempos de execução. Na parte de baixo das tabelas 5 e 6, podemos constatar que os piores projetos esquemáticos tem em comum o fato de normalizarem as dimensões geoespaciais mais seletivas e desnormalizarem as demais, com a criação de uma única tabela, onde, os dados das dimensões *city*, *nation* e *region* foram replicados em todos os registros das dimensões *c_address* e *s_address* (i.e. CG2-GG2). Além de que, pudemos constatar que a condensação dos dados provenientes de diferentes dimensões numa única família de colunas (i.e. F1) influenciou negativamente nos tempos de execução.

6 CONCLUSÃO

Este capítulo sumariza as motivações, objetivos, metodologias empreendidas e as conclusões que puderam ser obtidas a partir do presente estudo. Inicialmente, fazemos um retrospecto bem breve do contexto no qual nosso trabalho está inserido e da lacuna existente na literatura para a qual nós buscamos elevar o conhecimento da comunidade (Seção 6.1) Em seguida, trazemos as conclusões do estudo (Seção 6.2), as contribuições (Seção 6.3) e, finalmente, os trabalhos futuros que poderão contribuir para um entendimento ainda mais amplo da temática abordada (Seção 6.4).

6.1 RETROSPECTO

A crescente demanda por processamento de dados com fins analíticos alinhada a multiplicação dos ambientes de Big Data e aos avanços tecnológicos da computação distribuída levaram a saturação dos sistemas de bancos de dados construídos com as premissas do modelo relacional. O aumento cada vez mais rápido do volume e variedade dos dados processados comprometeram a capacidade dos SGBDs relacionais de imporem os controles de consistência, integridade e os transacionais sem consequências severas em seus desempenhos, disponibilidade e escalabilidade. Assim, os holofotes se voltaram para os bancos de dados NoSQL, uma vez que, eles relaxam tais restrições objetivando mais escalabilidade e poder de processamento paralelo. Implementar aplicações analíticas em tais ambientes ainda é um desafio, pois, existem muitas variáveis que precisam ser controladas durante o projeto e a implementação deles (i.e. escolha do banco, projeto esquemático).

Bancos de dados NoSQL orientados a colunas trabalham com formatos tabulares semelhantes aos vistos no paradigma relacional, mas com a diferença de que os dados dispostos em cada coluna são armazenados em endereços de memória contíguos e separados fisicamente dos dados das demais colunas, logo, as escritas e leituras de dados só acarretam no processamento das estruturas e dos dados que são efetivamente necessários. Um estudo científico [8] avaliou o potencial de tal classe na implementação de sistemas analíticos, chegando a conclusão de

que seus representantes detém vantagens competitivas sob os bancos mais tradicionais que organizam seus dados orientados por linhas (row-oriented stores). Em [10-19], foram explorados diversos tópicos relacionados ao projeto de esquemas para os bancos orientados a colunas, especialmente, para os casos de uso envolvendo consultas analíticas. Esses esquemas foram concebidos com o intuito de possibilitar a avaliação de abordagens mais normalizadas ou desnormalizadas, bem como, a redundância de dados, mas eles não trataram de cenários envolvendo a presença de dados geográficos no escopo das consultas.

A presença de dados geográficos na base de dados de um sistema analítico incorre em ponderações adicionais durante o planejamento do mesmo, como, os impactos do esquema no armazenamento e tempo de execução das consultas, por exemplo. Um dado geográfico qualquer pode representar pontos, linhas, polígonos, além de outras geometrias que podem ocupar desde poucos bytes indo até dezenas de megabytes. Dessa forma, conforme exemplificado, o projeto esquemático de bancos de dados utilizados com finalidades analíticas e cujo escopo envolve consultas geoespaciais, precisa balancear a normalização e desnormalização de dimensões para manter o nível de redundância sob controle, evitar o aumento indiscriminado do volume de dados e, ao mesmo tempo, não prejudicar o desempenho do banco no tocante ao armazenamento, durações da ingestão e consulta.

6.2 TRABALHO DESENVOLVIDO

Nosso estudo objetivou a avaliação experimental dos impactos da normalização e desnormalização de dados convencionais e geoespaciais no projeto de esquemas para o banco de dados HBase, que é um representante da classe de bancos NoSQL orientados a colunas. Aplicamos transformações ao modelo conceitual do benchmark geoespacial Spadawan com o intuito de criar esquemas derivados dele, mas que adotassem novas abordagens para: os relacionamentos entre dimensões geográficas e convencionais (i.e. CG1, CG2, CG3), relacionamentos entre duas dimensões geográficas (i.e. GG1, GG2, GG3), relacionamentos entre fatos e dimensões (i.e. FDD, FDN, FDH) e, finalmente, disposição das colunas entre famílias distintas (i.e. F1, FN). Realizamos o produto

cartesiano de tais abordagens {CG1, CG2, CG3} X {GG1, GG2, GG3} X {FDD, FDN, FDH} X {F1, FN} e obtemos 54 combinações entre as abordagens, das quais, 40 puderam ser efetivamente concebidas e avaliadas.

Após definirmos quais esquemas seriam avaliados, usamos o gerador de dados do Spadawan para criar os arquivos com os dados sintéticos de cada uma das tabelas do esquema, para os fatores de escala equivalentes a 1 e 10 vezes o tamanho padrão. Esses arquivos, originalmente organizados em esquema estrela, foram modificados para conformarem com os aspectos esquemáticos propostos por cada uma das abordagens introduzidas. Em seguida, eles foram carregados e avaliados em um cluster computacional composto por um nó mestre e três máquinas configuradas como *datanodes* do Hadoop, *regionservers* do HBase e *workers* do Spark. Coletamos o volume de dados, tempo para a ingestão dos dados e o tempo para a execução da carga de consultas para cada um dos esquemas propostos, com 30 repetições para o fator de escala 1 e 10 repetições para o fator 10. A execução de um experimento piloto permitiu que utilizássemos o teste t de Student para estimar qual o número de repetições que confere significância estatística para compararmos as médias de tempo deles.

Considerando as evidências presentes e fundamentadas nas avaliações experimentais, constatamos que a desnormalização de dimensões convencionais (i.e. FDD, FDH) implicou no aumento do volume de dados armazenados mas, em contrapartida, influenciou positivamente na execução das consultas analíticas. O emprego de múltiplas famílias de colunas por tabela (i.e. FN) não influenciou o volume expressivamente, mas contribuiu substancialmente para a redução do tempo de execução da carga de consultas. A normalização das dimensões geoespaciais menos seletivas (i.e. CG1, GG3) também contribuiu positivamente para a redução dos volumes e tempos. Já no caso das mais seletivas, não observamos impactos relevantes advindos da normalização ou desnormalização delas.

A análise dos resultados mostrou que as nossas constatações foram condizentes com as conclusões de [13-17], em prol dos benefícios da desnormalização de dimensões convencionais, e de [20], em prol da normalização das dimensões geográficas menos seletivas. Quando analisamos quais foram as

piores abordagens, também existe consenso. A normalização da dimensão geoespacial de maior seletividade combinada com a desnormalização das menos seletivas (i.e. CG2-GG2) e o emprego de uma única família por tabela (i.e. F1) foram as abordagens que mais impactaram negativamente nos tempos de execução e contribuíram para o aumento do volume de dados armazenados. Esquemas cujas dimensões convencionais foram desnormalizadas (i.e. FDD, FDH) também aparecem entre os piores colocados, mas, apenas quando combinados com as abordagens negativas supracitadas.

6.3 CONTRIBUIÇÕES

As principais contribuições realizadas por esta pesquisa científica foram:

- A proposição de esquemas para bancos de dados NoSQL orientados a colunas, onde, foram combinadas abordagens normalizadas e desnormalizadas para modelar a implementação dos relacionamentos existentes entre dimensões geoespaciais. Essas abordagens também levaram em conta a seletividade de tais dados, bem como, as redundâncias incorridas.

- O uso da normalização e desnormalização de dimensões para modelar a implementação dos relacionamentos existentes entre a tabela de fatos e dimensões convencionais.

- Combinação de todas as abordagens propostas acima para a criação de 40 esquemas que foram concebidos e implementados a partir do benchmark Spadawan.

- Execução de testes experimentais com uma carga de consultas analíticas para avaliar todos os esquemas propostos acerca de: volume de dados demandado para o armazenamento, o tempo de ingestão dos dados e o tempo de execução de uma carga de consultas analíticas cujos predicados envolviam operações com os dados geográficos.

- Coleta dos dados em um cluster computacional composto de três máquinas rodando como *datanodes* do Hadoop, *regionservers* do HBase e *workers* do Spark.

- Análise dos resultados com tabelas, gráficos e testes estatísticos para identificarmos quais foram as melhores e piores abordagens segundo as métricas coletadas.

6.4 TRABALHOS FUTUROS

A análise de outros projetos de pesquisa e dos resultados obtidos sugerem que os seguintes pontos precisam ser investigados com maior profundidade e podem melhorar o entendimento acerca das boas práticas no projeto de esquemas para sistemas analíticos em bancos NoSQL colunares:

- Realizar experimentos com fatores de escala mais elevados (i.e. 100, 1000) para averiguar com mais detalhamento a discrepância entre abordagens normalizadas e desnormalizadas para as dimensões convencionais quando as tabelas não cabem inteiramente na memória principal do cluster.

- Realizar experimentos com outros benchmarks geoespaciais que tenham mais dimensões e mais variedade de dados geográficos complexos (i.e. cidades), dado que, no Spadawan o aumento do fator de escala não incorre no aumento dos elementos geográficos.

- Avaliar os impactos da distribuição das regiões entre os servidores, ou seja, entender como o particionamento das tabelas e a distribuição desses pedaços entre as máquinas do cluster impacta na execução das consultas.

REFERÊNCIAS

1. BRYANT, R.; KATZ R.H.; LAZOWSKA, E.D.; Big-data computing: Creating revolutionary breakthroughs in commerce, science and society. In: Computing Research Initiatives for the 21st Century, Computing Research Association, Ver. 8, 2008. Disponível em: <https://cra.org/ccc/wp-content/uploads/sites/2/2015/05/Big_Data.pdf>. Acesso em 15 de Janeiro de 2022.

2. KIMBALL, R.; ROSS, M.; The Data Warehouse Toolkit. 2nd. New York: Wiley Computer Publishing, 2002.

3. ELMASRI, R. E.; NAVATHE, S.; Sistemas de Banco de Dados. Addison-Wesley. p.744, 2005.

4. RIZZI, S.; Conceptual Modeling Solutions for the Data Warehouse. In: W. R. e C. Koncilia (Ed.). Data Warehouses and OLAP: Concepts, Architectures and Solutions. Hershey, Pennsylvania, USA: IRM Press, 2007. Chapter 1, p.1-26.

5. DATE, C. J.; An Introduction to Database Systems. 8. ed. Boston, MA, USA: Addison Wesley Longman Publishing Co., Inc., 2003. ISBN 0321197844.

6. Doug Laney, 3d data management: Controlling data volume, velocity and variety, META Group Research Note 6 (2001). Disponível em: <https://www.bibsonomy.org/bibtex/742811cb00b303261f79a98e9b80bf49>. Acesso em 15 de Janeiro de 2022.

7. SADALAGE, P. J.; FOWLER, M.; NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. 1st. ed. [S.l.] Addison-Wesley Professional, 2012. ISBN 0321826620, 9780321826626.

8. ABADI, D. J.; MADDEN, S. R.; HACHEM, N.; Column-stores vs. row- stores: How different are they really? In: Proc. of SIGMOD, pages 967–980, Vancouver, Canada, June 2008.

9. Apache HBase – Apache HBase™ Home. Disponível em <https://hbase.apache.org>. Acesso em 15 de Janeiro de 2022.

10. YANG, F.; MILOSEVIC, D.; CAO, J.; An evolutionary algorithm for column family schema optimization in HBase. In: Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on. IEEE, 2015, pp. 439–445.

11. BOUSSAHOUA, M.; BOUSSAID, O.; BENTAYEB, F.; Logical schema for data warehouse on column-oriented NoSQL databases. In: International Conference on Database and Expert Systems Applications, pp. 247–256. Springer, Cham (2017).

12. BOUSSAHOUA, M.; BENTAYEB, F.; BOUSSAID, O.; KABACHI, N.; A data partitioning optimization approach for distributed data warehouses on column family NoSQL systems. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA); 2018:54–60.

13. LI, C.; Transforming relational database into HBase: A Case Study. In: Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on. IEEE, 2010: 683-687.

14. DEHDOUH, K.; BOUSSAID, O.; BENTAYEB, F.; Columnar NoSQL Star Schema Benchmark. In: Model and Data Engineering, LNCS 8748, Springer, pp 281-288 (2014).

15. DEHDOUH, K.; BOUSSAID, O.; BENTAYEB, F.; Using the column oriented NoSQL model for implementing big data warehouses. In: The 21st International Conference on Parallel and Distributed Processing Techniques and Applications. 2015. p.469-475.

16. CHEVALIER, M.; MALKI, M. E.; KOPLIKU, A.; TESTE, O.; TOURNIER, R.; Implementation of Multidimensional Databases in Column-Oriented NoSQL Systems. In: DaWak, (2015), p 379-390.

17. SCABORA, L. C.; BRITO, J. J.; CIFERRI, R. R.; CIFERRI, C. D. A.; Physical Data Warehouse Design on NoSQL Databases OLAP Query Processing over HBase. In: ICEIS , pp. 111-118, (2016).

18. GUO, D.; ONSTEIN, E.; State-of-the-Art Geospatial Information Processing in NoSQL Databases. ISPRS Int. J. Geo-Inf. 2020, 9, 331.

19. HAN, D.; STROULIA, E. HGrid: A Data Model for Large Geospatial Data Sets in HBase. In: Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, 28 June–3 July 2013; pp. 910–917.

20. FERRO, M.; LIMA, R. FIDALGO, R.; Evaluating Redundancy and Partitioning of Geospatial Data in Document-oriented Data Warehouses. In: Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Genève, Switzerland, 2019; pp. 221–235.

21. SIQUEIRA, T. L. L.; CIFERRI, R. R.; TIMES, V. C.; CIFERRI, C. D. A.; Benchmarking spatial data warehouses. In: The 12th International Conference on Data Warehousing and Knowledge Discovery, pages 40–51, 2010.

22. CODD, E. F; A Relational Model of Data for Large Shared Data Banks. In: Communications of the ACM 13 (6): 377–387. Disponível em: <https://doi.org/10.1145/362384.362685>. Acesso em 15 de Janeiro de 2022.

23. TSICHRITZIS, D.; Comments on Advantages of the Relational View. In: Proc. ACM SIGMOD Workshop on Data Description, Access and Control, Data Models: Data-Structure-Set versus Relational, 1974.

24. DB-Engines Ranking. Disponível em: <https://db-engines.com/en/ranking>. Acesso em 15 de Janeiro de 2022.

25. CONNOLLY, T.; BEGG, C; STRACHAN, A; Database Systems: A Practical Approach to Design, Implementation & Management, Addison-Wesley Publishing Company, Edinburgh, 2005.

26. RAUT, A. B.; NOSQL Database and Its Comparison with RDBMS. In: International Journal of Computational Intelligence Research, Volume 13, Number 7 (2017), pp. 1645-1651.

27. BREWER, E. A.; Towards Robust Distributed Systems. In: ACM Symposium on Principles of Distributed Computing (PODC), p7, 2000.

28. GILBERT, S.; LYNCH, N.; Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. In: ACM SIGACT News 33(2):51-59, 2002.

29. LITH, A.; MATTSON, J.; Investigating storage solutions for large data: A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data. p. 70. Göteborg: Department of Computer Science and Engineering, Chalmers University of Technology (2010).

30. EVANS, R.; NoSQL. Disponível em <http://blog.sym-link.com/posts/2009/12/nosql_2009/>. Acesso em 15 de Janeiro de 2022.

31. BAJAJ, A.; BICK, W.; The Rise of NoSQL Systems: Research and Pedagogy. In: Journal of Database Management (JDM), 31(3), 67-82, 2020.

32. STRAUCH, C. H.; NoSQL Databases. Lecture Selected Topics on Software-Technology Ultra-Large Scale Sites, Manuscript. In: Stuttgart Media University, 2011, p. 149. Disponível em <http://www.christof-strauch.de/nosqldbs.pdf>. Acesso em 15 de Janeiro de 2022.

33. Apache Spark: Unified engine for large-scale data analytics. Disponível em <https://spark.apache.org>. Acesso em 15 de Janeiro de 2022.

34. Spark - HBase Connector. Disponível em <https://github.com/hortonworks-spark/shc>. Acesso em 15 de Janeiro de 2022.

35. Apache Sedona: A cluster computing system for processing large-scale spatial data. Disponível em <https://sedona.apache.org>. Acesso em 15 de Janeiro de 2022.

36. OUSSOUS, A.; BENJELLOUN, F. Z.; LAHCEN, A. A.; BELFKIH, S.; Big data technologies: A survey. In: J. King Saud Univ. Comput. Inf. Sci., vol. 30, no. 4, pp. 431–448, 2018.

37. SIQUEIRA T. L.; CIFERRI, C. D.; TIMES, V. C.; OLIVEIRA, A. G.; CIFERRI, R. R.; The Impact of Spatial Data Redundancy on SOLAP Query Performance. In: Journal of the Brazilian Computer Society 15:19–34, 2009.

38. LEHNER, W.; Modelling Large Scale OLAP Scenarios. In: Advances in Database Technology (EDBT), 1998, pp. 153-167.

39. O'NEIL, P. E.; O'NEIL, E. J.; CHEN, X.; The Star Schema Benchmark (SSB). Pat 200(0), 50 (2007).

40. HUGHES, J. N.; ANNEX, A.; EICHELBERGER, C. N.; FOX, A.; HULBERT, A.; RONQUEST, M.; GeoMesa: A distributed Architecture for Spatio-temporal Fusion. In: SPIE Defense+Security, 2015.

41. KHURANA, A.; Introduction to HBase Schema Design. In: Networked Syst, 37:29–36, 2012.

42. CHEVALIER, M.; MALKI, M. E.; KOPLIKU, A.; TESTE, O.; TOURNIER, R.; Implementing Multidimensional Data Warehouses into NoSQL. In: The 17th Int. Conf. on Enterprise Information Systems, vol. DISI, to appear, 2015.

43. SIQUEIRA, T. L.; CIFERRI, R. R.; CIFERRI, C. D.; TIMES, V. C.; Investigating the effects of spatial data redundancy in query performance over geographical data warehouses. In: Proceedings of the 10th Brazilian Symposium on Geoinformatics. SBC, Rio de Janeiro, pp 1–12, 2008.

44. MATEUS, R. C.; TIMES, V. C.; SIQUEIRA, T. L.; CIFERRI, R. R.; CIFERRI, C. D.; How does the spatial data redundancy affect query performance in geographic data warehouses? In: Journal of Information and Data Management 1:519–534, 2010.

45. FERRO, M.; FRAGOSO, R.; FIDALGO, R.; Document-oriented Geospatial Data Warehouse: An Experimental evaluation of SOLAP Queries. In: Proceedings of the 21st IEEE Conference on Business Informatics, CBI 2019, Moscow, Russia, 15–17 July 2019.

46. BONCZ, P. A.; KERSTEN, M. L.; MIL primitives for querying a fragmented world. In: VLDB Journal 8(2):101–119, 1999.

47. BONCZ, P.; ZUKOWSKI, M.; NES., N.; MonetDB/X100: Hyper-pipelining query execution; In: CIDR, 2005.

48. CUI, X.; CHEN, W.; Performance Comparison Test of HBase and Cassandra Based on YCSB; In: 2021 IEEE/ACIS 19th International Conference on Computer and Information Science (ICIS), 2021, pp. 70-77, doi: 10.1109/ICIS51600.2021.9516864.

49. TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. D. Introdução à Engenharia de Software Experimental. COPPE/UFRJ, Rio de Janeiro, 2002. (Relatório Técnico RT-ES-590/02 do Programa de Engenharia de Sistemas e Computação).

50. SSB DBGEN. Disponível em <https://github.com/vadimtk/ssb-dbgen>. Acesso realizado em 15 de Janeiro de 2022.

51. Spadawan Benchmark. Disponível em <http://gbd.dc.ufscar.br/spadawan/hybrid.htm>. Acesso realizado em 15 de Janeiro de 2022.

52. Happy Base. Disponível em <https://happybase.readthedocs.io/en/latest/>. Acesso realizado em 15 de Janeiro de 2022.

53. Github de @mateusnbm - Colunar. Disponível em <https://github.com/mateusnbm/colunar>. Acesso realizado em 15 de Janeiro de 2022.

54. Power and Sample Size Calculator. Disponível em <http://powerandsamplesize.com/Calculators/Compare-2-Means/2-Sample-1-Sided>. Acesso realizado em 15 de Janeiro de 2022.

APÊNDICE A - CONSULTAS EXECUTADOS NO EXPERIMENTO

CONSULTA Nº1

```
SELECT
    p_category,
    c_region,
    SUM(lo_quantity)
FROM
    customer,
    region,
    lineorder,
    part
WHERE
    ST_Contains(region_geo, ST_GeomFromText('POINT(-87.42 41.24)') ) AND
    region_pk = c_region_fk AND
    c_custkey = lo_custkey AND
    lo_partkey = p_partkey
GROUP BY
    p_category,
    c_region
ORDER BY
    p_category,
    c_region
```

CONSULTA Nº2

```
SELECT
    p_category,
    c_nation,
    sum(lo_quantity)
FROM
    customer,
    nation,
    lineorder,
    part
WHERE
    ST_Contains(nation_geo, ST_GeomFromText('POINT(-87.42 41.24)') ) AND
    nation_pk = c_nation_fk AND
    c_custkey = lo_custkey AND
    lo_partkey = p_partkey
GROUP BY
    p_category,
    c_nation
ORDER BY
    p_category,
    c_nation
```

CONSULTA Nº3

```
SELECT
    p_category,
    c_city,
    sum(lo_quantity)
FROM
    customer,
    city,
    lineorder,
    part
WHERE
    ST_Contains(city_geo, ST_GeomFromText('POINT(-87.42 41.24)') ) AND
    city_pk = c_city_fk AND
    c_custkey = lo_custkey AND
    lo_partkey = p_partkey
GROUP BY
    p_category,
    c_city
ORDER BY
    p_category,
    c_city
```

CONSULTA Nº4

```
SELECT
    p_category,
    c_address,
    sum(lo_quantity)
FROM
    customer,
    c_address,
    lineorder,
    part
WHERE
    ST_Distance(c_address_geo, ST_MakePoint(-87.42, 41.24)) <= 0.18 AND
    c_address_pk = c_address_fk AND
    c_custkey = lo_custkey AND
    lo_partkey = p_partkey
GROUP BY
    p_category,
    c_address
ORDER BY
    p_category,
    c_address
```

CONSULTA Nº5

```
SELECT
    p_category,
    c_region,
    c_nation,
    sum(lo_quantity)
FROM
    customer,
    nation,
    lineorder,
    part
WHERE
    ST_Contains(nation_geo, ST_GeomFromText('POINT(-87.42 41.24)') ) AND
    nation_pk = c_nation_fk AND
    c_custkey = lo_custkey AND
    lo_partkey = p_partkey
GROUP BY
    p_category,
    c_region,
    c_nation
ORDER BY
    p_category,
    c_region,
    c_nation
```

CONSULTA Nº6

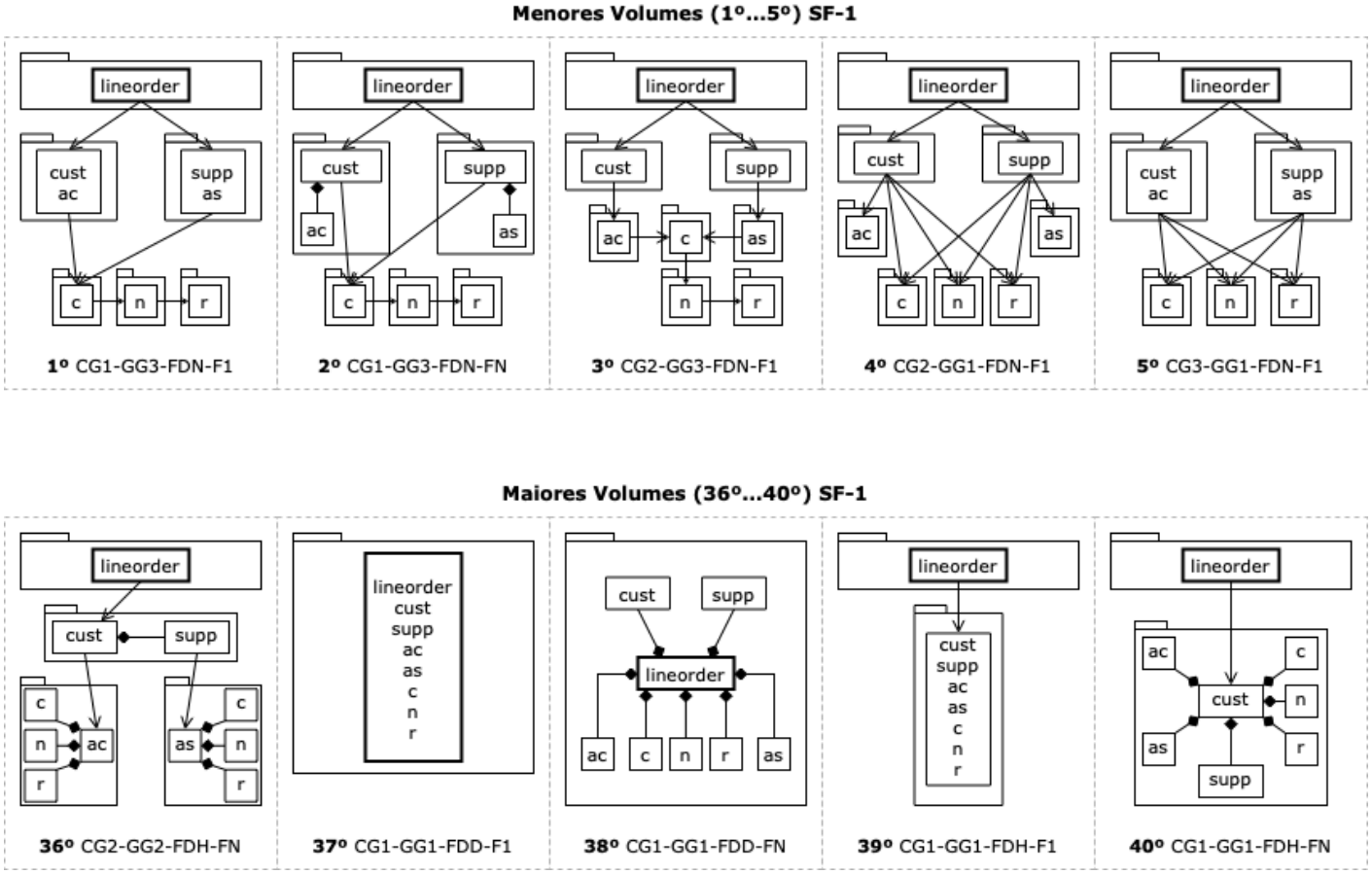
```
SELECT
    p_category,
    c_region,
    c_nation,
    c_city,
    sum(lo_quantity)
FROM
    customer,
    city,
    lineorder,
    part
WHERE
    ST_Contains(city_geo, ST_GeomFromText('POINT(-87.42 41.24)') ) AND
    city_pk = c_city_fk AND
    c_custkey = lo_custkey AND
    lo_partkey = p_partkey
GROUP BY
    p_category,
    c_region,
    c_nation,
    c_city
ORDER BY
    p_category,
    c_region,
    c_nation,
    c_city
```

CONSULTA Nº7

```
SELECT
    p_category,
    c_region,
    c_nation,
    c_city,
    c_address,
    sum(lo_quantity)
FROM
    customer,
    c_address,
    lineorder,
    part
WHERE
    ST_Distance(c_address_geo, ST_MakePoint(-87.42, 41.24)) <= 0.18 AND
    c_address_pk = c_address_fk AND
    c_custkey = lo_custkey AND
    lo_partkey = p_partkey
GROUP BY
    p_category,
    c_region,
    c_nation,
    c_city,
    c_address
ORDER BY
    p_category,
    c_region,
    c_nation,
    c_city,
    c_address
```

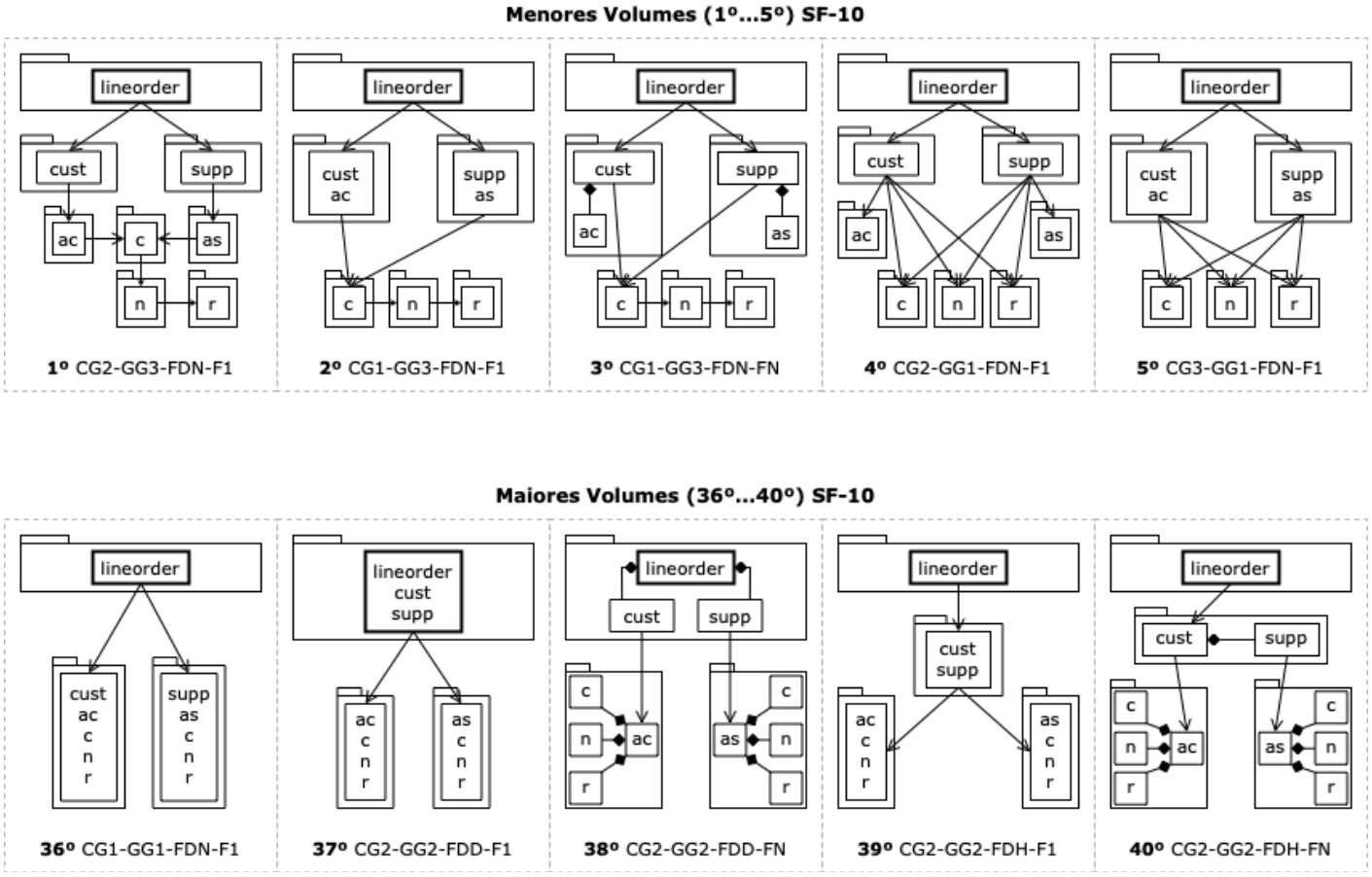
APÊNDICE B - MELHORES E PIORES ESQUEMAS DIAGRAMATICAMENTE

Figura 24 - Os cinco menores e maiores volumes para o fator 1.



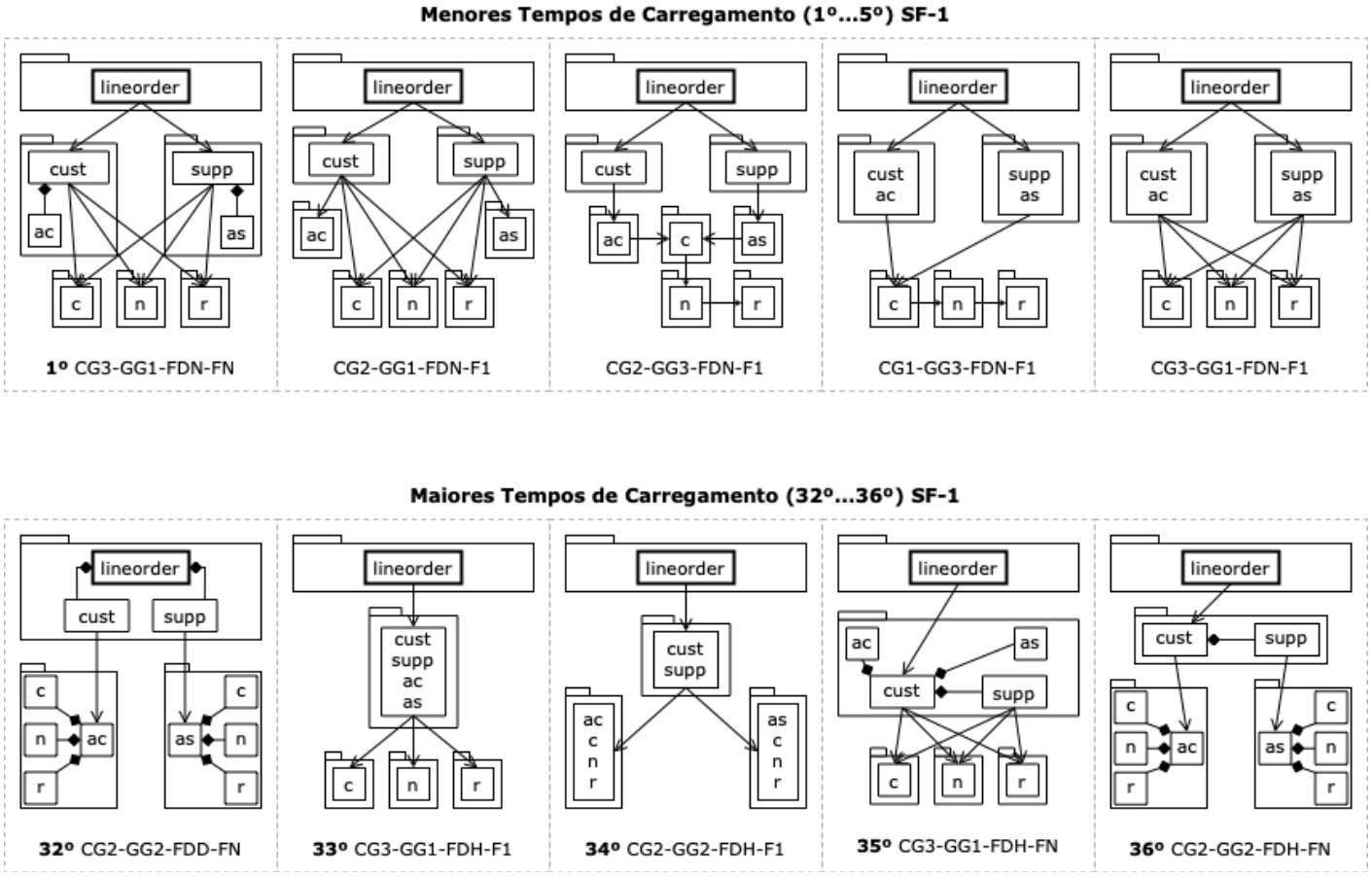
Fonte: autor (2022).

Figura 25 - Os cinco menores e maiores volumes para o fator 10.



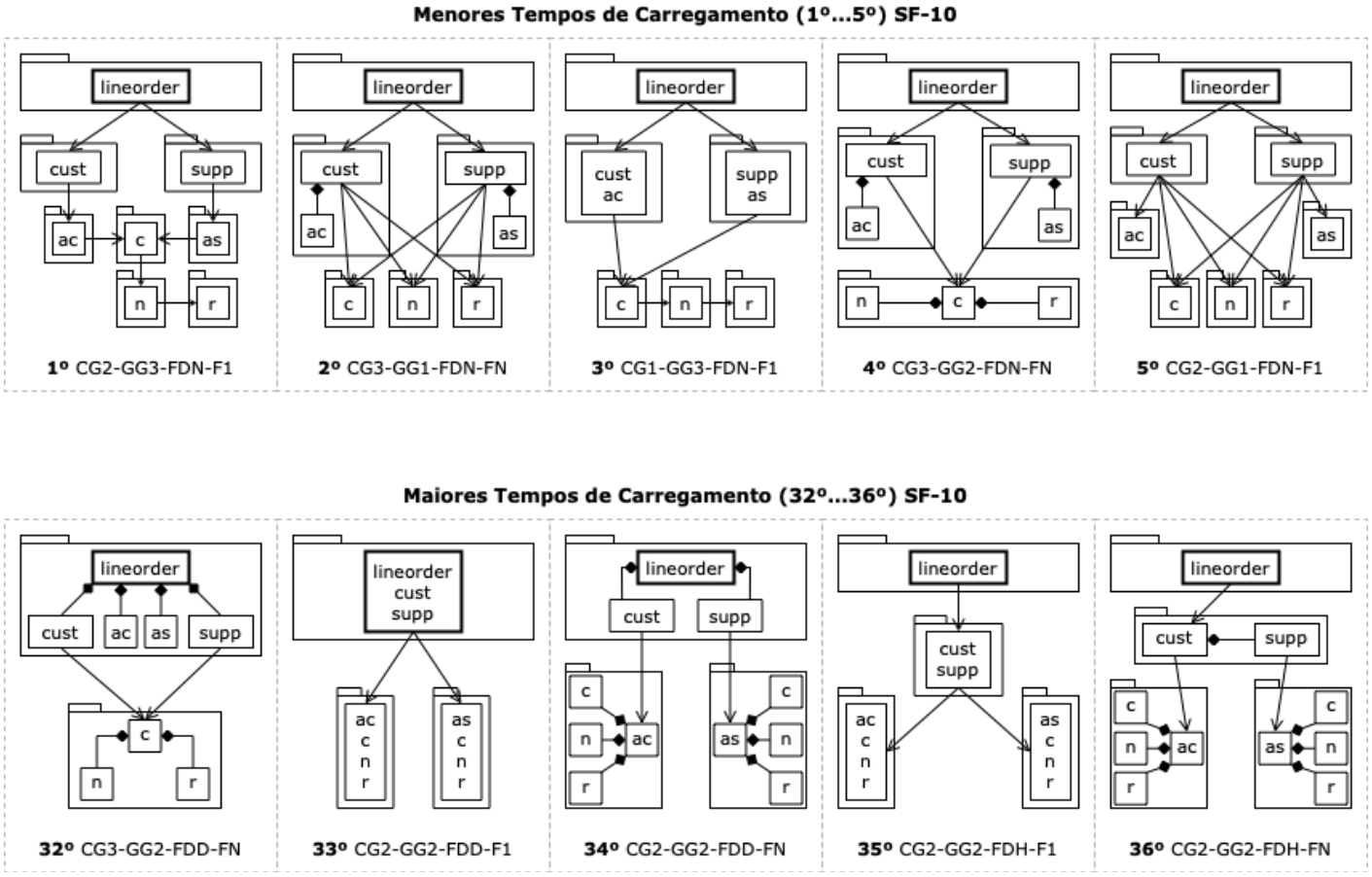
Fonte: autor (2022).

Figura 26 - Os cinco menores e maiores tempos de carga para o fator 1.



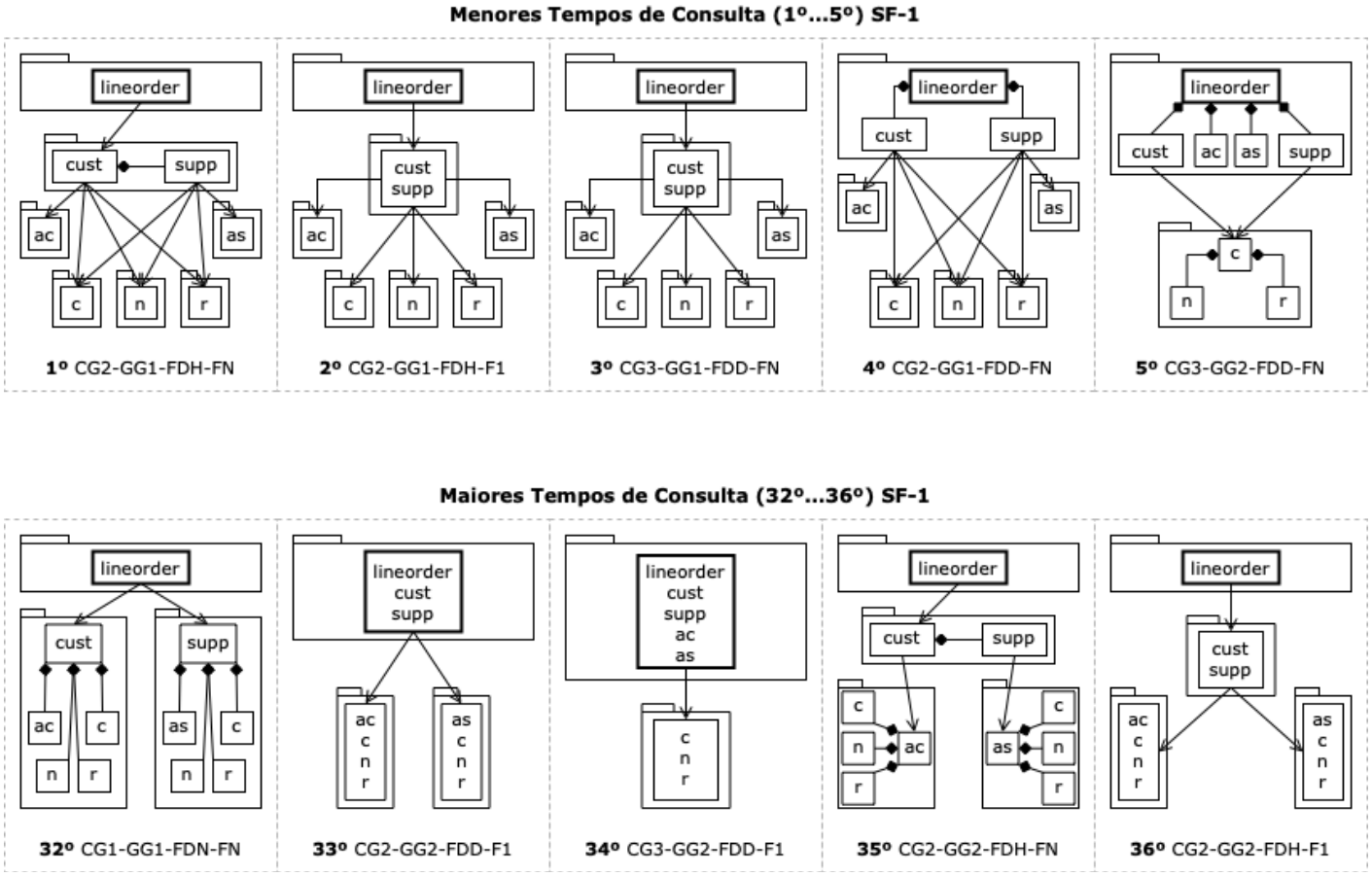
Fonte: autor (2022).

Figura 27 - Os cinco menores e maiores tempos de carga para o fator 10.



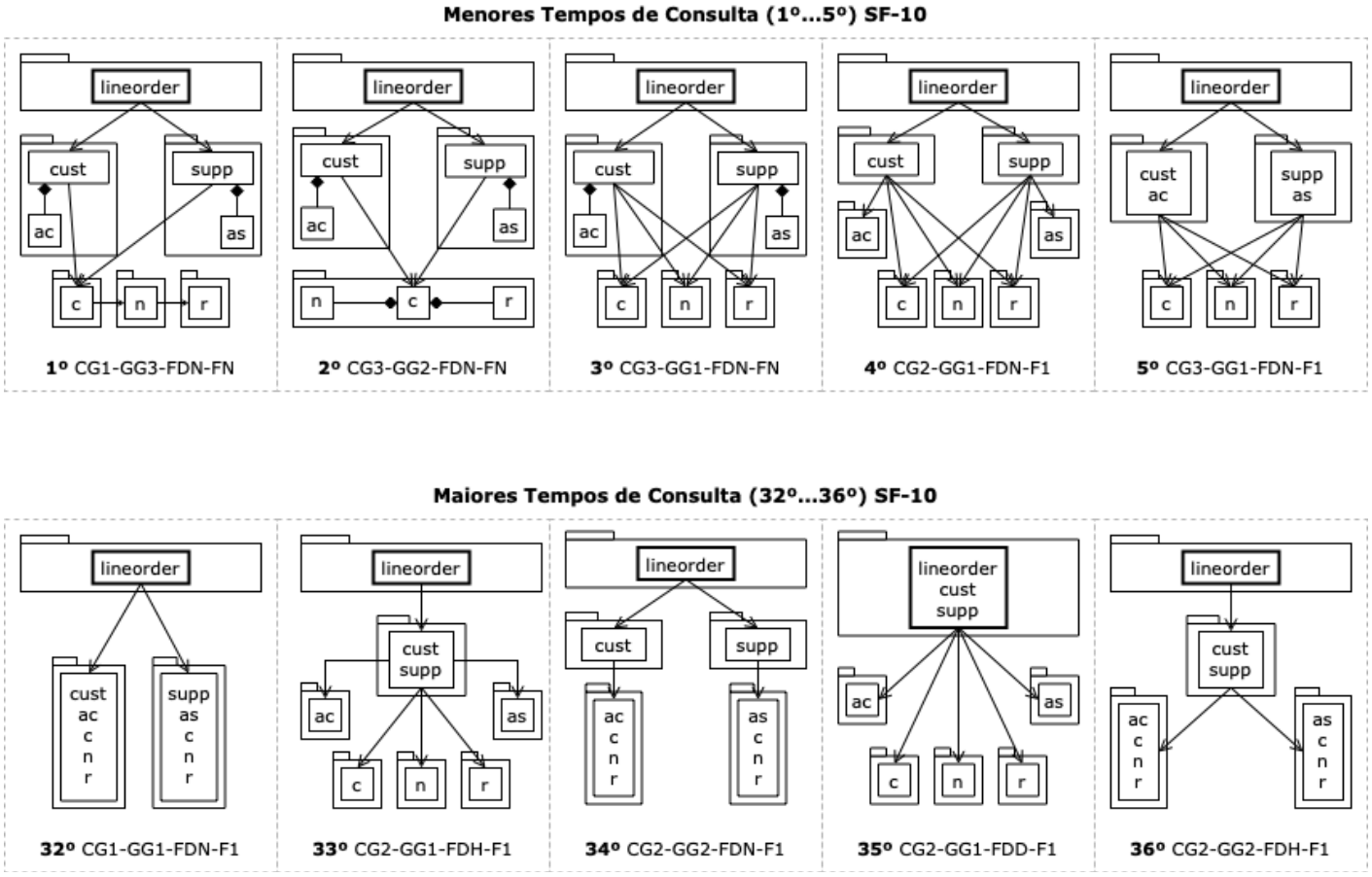
Fonte: autor (2022).

Figura 28 - Os cinco menores e maiores tempos de consulta para o fator 1.



Fonte: autor (2022).

Figura 29 - Os cinco menores e maiores tempos de consulta para o fator 10.



Fonte: autor (2022).