

UNIVERSIDADE FEDERAL DE MATO GROSSO  
INSTITUTO DE FÍSICA  
PROGRAMA DE PÓS GRADUAÇÃO EM FÍSICA AMBIENTAL

ANÁLISE DE SISTEMAS DE  
GERENCIAMENTO DE BANCO DE DADOS  
PARA ARMAZENAMENTO DE DADOS  
CLIMÁTICOS

Igor Antonio Kuhnen

Orientador: Prof. Dr. Josiel Maimone de Figueiredo

Cuiabá - MT  
Fevereiro/2016

UNIVERSIDADE FEDERAL DE MATO GROSSO  
INSTITUTO DE FÍSICA  
PROGRAMA DE PÓS GRADUAÇÃO EM FÍSICA AMBIENTAL

ANÁLISE DE SISTEMAS DE  
GERENCIAMENTO DE BANCO DE DADOS  
PARA ARMAZENAMENTO DE DADOS  
CLIMÁTICOS

Igor Antonio Kuhnen

Dissertação apresentada ao Programa de Pós Graduação em Física Ambiental da Universidade Federal de Mato Grosso, como parte dos requisitos para obtenção do título de Mestre em Física Ambiental.

Prof. Dr. Josiel Maimone de Figueiredo

Cuiabá, MT

Fevereiro/2016

### **Dados Internacionais de Catalogação na Fonte.**

K96a Kuhnen, Igor Antonio Kuhnen.  
Análise de sistemas de gerenciamento de Banco de Dados para armazenamento  
de dados climáticos / Igor Antonio Kuhnen Kuhnen. -- 2016  
70 f. ; 30 cm.

Orientador: Josiel Maimone de Figueiredo.  
Dissertação (mestrado) - Universidade Federal de Mato Grosso, Instituto de  
Física, Programa de Pós-Graduação em Física Ambiental, Cuiabá, 2016.  
Inclui bibliografia.

1. Dados micrometeorológicos. 2. NoSQL. 3. Benchmark. I. Título.

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

**Permitida a reprodução parcial ou total, desde que citada a fonte.**

## DEDICATÓRIA

*À minha família e principalmente à minha esposa.*

# Agradecimentos

Gostaria de agradecer primeiramente à UFMT e ao PGFA pelo apoio recebido.

Agradeço ao meu orientador Dr. Josiel Maimone Figueiredo por sempre me cobrar, e ter paciência.

Gostaria de agradecer a todas as pessoas do PGFA que fizeram parte deste trabalho, em especial Raphael, Allan, Thiago e Osvaldo.

Ao Diego Lima, pela ajuda na alteração dos códigos em Java para finalização deste trabalho.

Agradeço também ao professor Dr. José de Souza Nogueira (Paraná) pelo apoio e acolhimento no desenvolvimento deste trabalho.

Agradeço à minha família, mas principalmente a minha esposa, por estar sempre presente, com apoio, dicas e principalmente atenção a todos os detalhes para que esse trabalho fosse concluído.

# SUMÁRIO

|                                                          |          |
|----------------------------------------------------------|----------|
| LISTA DE FIGURAS                                         | I        |
| LISTA DE TABELAS                                         | II       |
| LISTA DE TABELAS                                         | II       |
| RESUMO                                                   | III      |
| ABSTRACT                                                 | IV       |
| <b>1 INTRODUÇÃO</b>                                      | <b>1</b> |
| 1.1 PROBLEMÁTICA . . . . .                               | 1        |
| 1.2 JUSTIFICATIVA . . . . .                              | 2        |
| 1.3 OBJETIVO . . . . .                                   | 4        |
| 1.3.1 Objetivo Geral . . . . .                           | 4        |
| 1.3.2 Objetivo Específico . . . . .                      | 4        |
| <b>2 FUNDAMENTAÇÃO TEÓRICA</b>                           | <b>5</b> |
| 2.1 DADOS METEOROLÓGICOS . . . . .                       | 5        |
| 2.2 <i>BIG DATA</i> . . . . .                            | 9        |
| 2.3 BANCO DE DADOS . . . . .                             | 10       |
| 2.3.1 Histórico . . . . .                                | 13       |
| 2.3.2 Banco de dados relacional . . . . .                | 14       |
| 2.3.2.1 Limitações dos Bancos de dados Relacionais . . . | 15       |
| 2.3.3 Banco de dados NoSQL . . . . .                     | 16       |
| 2.3.3.1 Classificação de banco de dados NoSQL . . . . .  | 17       |

|          |                                                                                 |           |
|----------|---------------------------------------------------------------------------------|-----------|
| 2.3.3.2  | Banco de dados Orientado a coluna ( <i>Column Family</i> ) . . . . .            | 18        |
| 2.4      | TESTES DE DESEMPENHO . . . . .                                                  | 19        |
| 2.4.1    | <i>Benchmark</i> . . . . .                                                      | 20        |
| 2.4.2    | <i>YCSB</i> . . . . .                                                           | 21        |
| 2.4.2.1  | Propriedades das Cargas de trabalho do Benchmark ( <i>WORKLOADS</i> ) . . . . . | 23        |
| 2.4.2.2  | Propriedades de tempo de execução ( <i>Runtime</i> ) . . . . .                  | 26        |
| 2.5      | TESTES ESTATÍSTICOS DE KRUSKAL WALLIS . . . . .                                 | 27        |
| <b>3</b> | <b>MATERIAL E MÉTODOS</b>                                                       | <b>29</b> |
| 3.1      | MATERIAIS . . . . .                                                             | 29        |
| 3.1.1    | <i>Hardware</i> . . . . .                                                       | 29        |
| 3.1.2    | Sistemas de Gerenciamento de Banco de Dados . . . . .                           | 30        |
| 3.1.2.1  | PostgreSQL . . . . .                                                            | 30        |
| 3.1.2.2  | Cassandra . . . . .                                                             | 30        |
| 3.1.3    | <i>YCSB</i> . . . . .                                                           | 31        |
| 3.2      | MÉTODOS . . . . .                                                               | 31        |
| 3.2.1    | Preparação do ambiente de teste . . . . .                                       | 31        |
| 3.2.2    | Teste de performance YCSB . . . . .                                             | 32        |
| 3.2.2.1  | Adaptações no código Java do YCSB . . . . .                                     | 33        |
| 3.2.2.2  | Configurações da linha de comando . . . . .                                     | 36        |
| 3.2.3    | Análise estatística . . . . .                                                   | 37        |
| <b>4</b> | <b>RESULTADOS</b>                                                               | <b>38</b> |
| 4.1      | <i>WORKLOAD A</i> – SOMENTE <i>INSERT</i> . . . . .                             | 38        |
| 4.2      | <i>WORKLOAD B</i> – SOMENTE <i>SCAN</i> . . . . .                               | 41        |
| 4.3      | <i>WORKLOAD C</i> – SOMENTE <i>READ</i> . . . . .                               | 45        |
| 4.4      | LATÊNCIA <i>WORKLOAD A, B e C</i> . . . . .                                     | 49        |
| 4.5      | FLUXO DE DADOS <i>WORKLOAD A, B e C</i> . . . . .                               | 50        |

|          |                             |           |
|----------|-----------------------------|-----------|
| <b>5</b> | <b>CONCLUSÃO</b>            | <b>51</b> |
| 5.1      | CONTRIBUIÇÕES . . . . .     | 51        |
| 5.2      | TRABALHOS FUTUROS . . . . . | 52        |
|          | <b>REFERÊNCIAS</b>          | <b>53</b> |

# LISTA DE FIGURAS

|    |                                                                                                                                                                                |    |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1  | Fluxo de atividades no monitoramento ambiental. . . . .                                                                                                                        | 7  |
| 2  | Atributos e Tuplas de uma Relação de uma Torre Meteorológica .                                                                                                                 | 14 |
| 3  | Esquema do YCSB . . . . .                                                                                                                                                      | 22 |
| 4  | Arquitetura do YCSB <i>Client</i> . . . . .                                                                                                                                    | 22 |
| 5  | As distribuições de probabilidade. Eixos horizontal representa os<br>itens em ordem de inserção, e os eixos verticais representam pro-<br>babilidade de ser escolhido. . . . . | 25 |
| 6  | Latência do <i>Workload A</i> - Somente <i>Insert</i> . . . . .                                                                                                                | 38 |
| 7  | Fluxo de Dados <i>Workload A</i> - Somente <i>Insert</i> . . . . .                                                                                                             | 40 |
| 8  | Latência do <i>Workload B</i> - Somente <i>Scan</i> . . . . .                                                                                                                  | 42 |
| 9  | Fluxo de dados do <i>Workload B</i> - Somente <i>Scan</i> . . . . .                                                                                                            | 44 |
| 10 | Latência do <i>Workload C</i> - Somente <i>Read</i> . . . . .                                                                                                                  | 46 |
| 11 | Fluxo de dados do <i>Workload C</i> - Somente <i>Read</i> . . . . .                                                                                                            | 48 |
| 12 | Latência <i>Workload A, B e C</i> . . . . .                                                                                                                                    | 50 |
| 13 | Fluxo de dados <i>Workload A, B e C</i> – Somente <i>Scan</i> . . . . .                                                                                                        | 50 |

# LISTA DE TABELAS

|   |                                                                                 |    |
|---|---------------------------------------------------------------------------------|----|
| 1 | Cargas de trabalho no YCSB Core Package - configuração padrão                   | 25 |
| 2 | Cargas de trabalho no YCSB desenvolvidas para o estudo . . . . .                | 32 |
| 3 | Teste de Kruskal-Wallis - Workload A (20.000 a 140.000 operações)               | 39 |
| 4 | Teste de Kruskal-Wallis - Workload A (400.000 a 15.000.000 operações) . . . . . | 40 |
| 5 | Teste de Kruskal-Wallis - Workload B (20.000 a 140.000 operações)               | 43 |
| 6 | Teste de Kruskal-Wallis - Workload B (400.000 a 15.000.000 operações) . . . . . | 43 |
| 7 | Teste de Kruskal-Wallis - Workload C (20.000 a 140.000 de operações)            | 47 |
| 8 | Teste de Kruskal-Wallis - Workload C (400.000 a 15.000.000 operações) . . . . . | 47 |

# RESUMO

KUHNEN, I. A. ANÁLISE DE SISTEMAS DE GERENCIAMENTO DE BANCO DE DADOS PARA ARMAZENAMENTO DE DADOS CLIMÁTICOS. Cuiabá, 2016, 59f. Dissertação (Mestrado em Física Ambiental) - Instituto de Física, Universidade Federal de Mato Grosso.

O monitoramento e estudo de dados meteorológicos podem auxiliar instituições públicas e privadas a traçar planos para maximização de sua produção ou mesmo preservar a vida e o patrimônio público em diversas áreas como a segurança civil, agrícola, energética. O Brasil, e em especial o estado de Mato Grosso, dentro deste contexto, possui maior responsabilidade por possuir em seu território três diferentes biomas: Amazônia, Cerrado e Pantanal. Um grande problema para o desenvolvimento de pesquisas na área de climatologia está na disponibilização e acesso aos dados, e ainda na confiabilidade dos dados, uma vez que alterações indevidas dos dados permanecem para pesquisas posteriores. Sistemas gerenciadores de Banco de dados permitem o acesso a dados históricos, provenientes de diferentes fontes, por diferentes usuários, integrando os dados e fornecendo segurança. O objetivo deste trabalho é identificar formas eficientes de armazenamento e recuperação de dados de Séries Temporais obtidos a partir de torres meteorológicas. A metodologia utilizada se fundamentou na preparação do ambiente computacional de teste e execução dos testes em dois diferentes Banco de dados: PostgreSQL e Cassandra, utilizando-se do Benchmark YCSB, com configuração de diferentes cargas de trabalhos comumente requeridas para gerenciamento de dados meteorológicos. Os resultados mostraram que o BD Cassandra possui maior eficiência no processamento das operações de Inserção e Seleção dos dados com estrutura equivalente às Séries Temporais obtidos a partir de torres meteorológicas, quando comparado ao Banco de Dados relacional PostgreSQL, com menor latência e maior fluxo de dados para todos os volumes de operações testados. No entanto, quando tratamos das operações de leitura o BD PostgreSQL revelou-se mais eficiente, tanto quanto a latência quanto ao fluxo de dados.

**Palavras-chave:** Banco de dados, Dados micrometeorológicos, NoSQL, *Benchmark*.

# ABSTRACT

KUHNEN, I.A.: Analysis of data base management systems for climate data storage. Cuiabá, 2016, 59f. Dissertação (Mestrado em Física Ambiental) - Instituto de Física, Universidade Federal de Mato Grosso.

The monitoring and study of weather data can help public and private institutions to draw up plans for maximizing your production or even preserve life and public property in areas such as civil security, agricultural, energy. The Brazil, especially the state of Mato Grosso, in this context, has greater responsibility by having in its territory three different biomes: Amazonia, Cerrado and Pantanal. A major problem for the development of research in climatology area is the availability and access to data, and also the reliability of the data, since tampering of the data remains for further research. Data Base Management Systems allow access to historical data from different sources for different users, integrating the data and providing security. The objective of this study is to identify efficient ways of storing and Time Series data recovery obtained from meteorological towers. The methodology used was based on the preparation of the computational test environment and execution of tests on two different database: PostgreSQL and Cassandra using the Benchmark YCSB with different workloads configurations commonly required for management of weather data. The results showed that the BD Cassandra has greater efficiency in the processing operations of insertion and selection of data with equivalent structure to the time series obtained from meteorological towers, when compared to the PostgreSQL relational database, with lower latency and increased data flow for all volume of operations tested . However, when dealing with operations of reading the BD PostgreSQL proved to be more efficient both for the latency and the data stream.

**Keywords:** Database, Meteorological Data, NoSQL, Benchmark .

# Capítulo 1

## INTRODUÇÃO

### 1.1 PROBLEMÁTICA

Relaciona-se ao aquecimento fenômenos ambientais extremos como o derretimento das geleiras e calotas polares ou ainda alterações nos processos biológicos naturais como os períodos de floração, produzindo enormes perdas econômicas e de vidas em diferentes partes do planeta (HOUGHTON et al., 2001). No Brasil, no ano de 2011, catástrofes climáticas relacionadas ao excesso de chuvas foram responsáveis por aproximadamente 1.000 mortes na região serrana do estado do Rio de Janeiro, e no início de 2012 deixaram mais de 40.000 desabrigados e mais de 100 cidades em estado de emergência somente na região sudeste do país (CEPED-UFSC, 2014). Destaque se dá ainda a escassez de chuvas ocorrida no ano de 2014, que baixaram os níveis de reservatórios e rios, provocaram seca de veredas, redução nas produções agrícolas e outros, tornando necessária a criação de medidas de redução de consumo e utilização da água.

O Professor Pedro Leite da Silva Dias, do Instituto Astronômico e Geofísico da USP, destaca que “os relatórios do IPCC apontam para grandes dificuldades na detecção de mudanças climáticas de origem antrópica em função da alta variabilidade natural do clima” e que existem “grandes incertezas quanto aos efeitos regionais, particularmente na América do Sul onde existe um enorme estoque de carbono armazenado no solo e na parte aérea da Floresta Amazônica” (DIAS, 2005).

Tais fatos confirmam que o monitoramento e estudo de dados meteorológicos são de extrema importância, pois podem auxiliar instituições públicas e privadas a traçar planos para maximização de sua produção ou mesmo preservar a vida e o patrimônio público em diversas áreas como a segurança civil, agrícola, energética, transportes, ecologia, saúde e na previsão do tempo (COUTINHO et

al., 2015). O Brasil, e em especial o estado de Mato Grosso, dentro deste contexto, possui maior responsabilidade por possuir em seu território três diferentes biomas: Amazônia, Cerrado e Pantanal.

O Programa de Pós-graduação em Física Ambiental (PPGFA) promove ações no sentido de aprimorar o conhecimento sobre os aspectos físicos (físico-químicos, biofísicos e geofísicos) do meio-ambiente, bem como sobre os impactos que a ocupação não planejada tem exercido sobre os ecossistemas e os correspondentes reflexos sobre a questão das mudanças climáticas globais. Nos últimos três anos o programa promoveu estudos relacionados a estimativas de fluxos de calor entre ecossistema e atmosfera (SANTANNA, 2013; RODRIGUES, 2014), fluxos de CO<sub>2</sub> (SILVA, 2013; ARRUDA, 2014; PEREIRA, 2013), balanço de energia (PEREIRA, 2013; ARRUDA, 2014; DANELICHEN, 2015), dinâmicas de inundação no pantanal (SILVEIRA, 2015) além de ilhas de calor urbanas (MACIEL, 2014; ROSSETI, 2013). A multidisciplinaridade do programa gera uma série de conflitos no momento em que busca-se instituir uma síntese entre as diversas áreas do conhecimento.

Segundo Oliveira (2015) o maior problema relacionado ao desenvolvimento de pesquisas científicas nesta área está na disponibilização e acesso aos dados. Vemos cientistas, a todo momento, buscando dados climáticos *in loco* através de torres meteorológicas, mas a utilização dos mesmos acaba se tornando restrita a manipulação em planilhas eletrônicas, caracterizadas por dados não-estruturados (xls, pdf, etc). Como consequência, a confiabilidade dos dados pode ser perdida, uma vez que alterações indevidas dos dados permanecem para pesquisas posteriores.

## 1.2 JUSTIFICATIVA

Para implementar sistemas de armazenamento de dados meteorológicos que sejam confiáveis, flexíveis e eficientes, instituiu-se nos grandes centros meteorológicos do mundo o uso da tecnologia de Banco de Dados (POTTIER, 1995; SANDERS, 1997; RAOULT, 1997). Esta tecnologia se tornou ainda mais necessária com a utilização de equipamentos de coleta digital e automática de dados, que elevaram o volume e a complexidade dos dados a serem armazenados, demandando ferramentas específicas que facilitem o processamento dessa informação, para depois interpretá-la de forma adequada (CUNHA, 2001).

A quantidade de dados gerada diariamente nas redes de sensores meteorológicos, dados de sensoriamento, entre outros, estão na ordem de algumas dezenas,

ou centenas, de Terabytes (VIEIRA et al., 2012). Segundo D’Andrea (2010) estes conjuntos de dados que crescem exponencialmente são chamados *Big Data* e não são possíveis de serem analisados por técnicas tradicionais de banco de dados devido seu elevado volume, brutalidade ou desestruturação, trazendo novos desafios na forma de manipulação, armazenamento e processamento de consultas em várias áreas de computação, e em especial na área de bases de dados, mineração de dados e recuperação de informação.

Os BDs relacionais, manipulados em linguagem SQL (*Structured Query language*) (SILBERSCHATZ et al., 2006), onde os dados são estruturados, armazenados, manipulados e recuperados em forma de tabelas normalizadas, e todas as transações seguem propriedades de forte consistência chamadas ACID (Atomicidade, Consistência, Isolamento, Durabilidade), apesar da riqueza de recursos, tendem a aumentar a complexidade de utilização com o aumento do fluxo de dados (ELMASRI et al., 2005).

Diante deste contexto os bancos de dados não relacionais, chamados NoSQL (*NotOnly SQL*), se apresentam como uma otimizada solução onde os bancos relacionais têm grande deficiência, eliminando algumas regras do modelo relacional (VIEIRA et al., 2012; STONEBRAKER et al., 2007). Apesar do rompimento com as regras, o modelo NoSQL garante um ganho de performance, que passou a ser fundamental para suprir os requisitos de alta escalabilidade necessária para o gerenciamento de grandes quantidades de dados, assim como para garantir uma alta disponibilidade dos mesmos (SOUZA; SANTOS, 2015).

Contudo, uma questão a ser considerada quando se trata da manipulação de dados climáticos é a interdependência entre observações vizinhas no tempo, comportamento característico de Séries Temporais (MADDALA; LAHIRI, 1992). A realização deste trabalho é justificada pela necessidade de analisar o desempenho dos bancos de dados NoSQL e Relacional no gerenciamento das operações de busca e inserção de dados de Séries Temporais. Este objetivo busca atender as futuras demandas oriundas da elevação do volume e complexidade dos dados climáticos provenientes das varias estações de coletas, não somente da Região Centro Oeste, mas dos demais programas ambientais do Brasil e do mundo, demandando padronização e confiabilidade dos dados armazenados.

## 1.3 OBJETIVO

### 1.3.1 Objetivo Geral

O objetivo deste trabalho é identificar formas eficientes de armazenamento e recuperação de dados de Séries Temporais obtidos a partir de torres meteorológicas.

### 1.3.2 Objetivo Específico

Para alcançar tal objetivo foi necessária a realização dos seguintes objetivos específicos:

- Estudo sobre as propriedades dos Dados Climáticos;
- Escolha dos Sistemas de Banco de Dados para realização dos testes;
- Escolha e configuração do Benchmark;
- Análise estatística dos resultados encontrados.

## Capítulo 2

# FUNDAMENTAÇÃO TEÓRICA

Neste item é feita uma abordagem geral sobre os conceitos básicos necessários para a compreensão dos principais temas abordados nesta pesquisa. De uma forma geral serão trabalhadas as tipologias de dados abordadas, os conceitos fundamentais em torno do tema Banco de Dados e Sistema de Gerenciamento de Banco de Dados - aprofundando-se no que tange as tipologias de Banco de dados Relacionais e NoSQL, englobadas nos objetivos desta pesquisa – os conceitos em torno dos testes necessários para avaliação de desempenho dos dois BDs em estudo e testes estatísticos utilizados na análise dos resultados.

## 2.1 DADOS METEOROLÓGICOS

Uma questão a ser considerada quando se trata de dados climáticos é a interdependência entre observações vizinhas no tempo, comportamento característico de Séries Temporais. Pode-se definir uma série temporal como a coleção de observações feitas sequencialmente ao longo do tempo (MADDALA; LAHIRI, 1992). O propósito da análise de séries temporais é estudar a dinâmica e a estrutura temporal dos dados. A análise de uma sequência única de dados é chamada de análise de série temporal univariada, e a análise de várias coleções de dados para a mesma sequência de períodos de tempo é chamada de análise multivariada (GUJARATI D. N.; PORTER, 2011).

Ehlers (2007) ilustra algumas características particulares a estes tipos de dados:

- Observações correlacionadas são mais difíceis de analisar e demandam técnicas específicas;
- Precisa ser considerada a ordem temporal das observações;

- Fatores complicadores como presença de tendências e variação sazonal ou cíclica podem ser difíceis de estimar ou remover;
- A seleção de modelos pode ser complicada, e as ferramentas podem ser de difícil interpretação;
- Dificuldade em lidar com observações perdidas e dados discrepantes devido à natureza sequencial.

O conjunto de observações ordenadas no tempo pode ser discreto como o volume total de chuvas em uma determinada estação ou o número mensal de casos notificados de uma doença específica; ou contínuo, como o registro de um eletrocardiograma de uma pessoa ou o registro dos valores de temperatura e umidade ao longo do dia (CARDOSO, 2001). Pode-se obter uma série temporal discreta a partir de uma amostra de pontos de uma série contínua ou por meio de um parâmetro como, por exemplo, a média de períodos fixos de tempo.

Segundo Cardoso (2001), na análise de uma série temporal, primeiramente deseja-se modelar o fenômeno estudado para, a partir daí, descrever o comportamento da série, fazer estimativas e, por último, avaliar quais os fatores que influenciaram o comportamento da série, buscando definir relações de causa e efeito entre duas ou mais séries.

Para o correto desenvolvimento e garantia de validade das análises de séries temporais oriundas de dados meteorológicos, a manipulação dos dados é realizada com um conjunto de operações interativas, nas quais é de vital importância o conhecimento do especialista de domínio para guiar essas operações (OLIVEIRA, 2015). Moresi et al. (2010) apresenta o ciclo desses dados no contexto do gerenciamento ambiental, conforme Figura 1.



Figura 1: Fluxo de atividades no monitoramento ambiental.  
 FONTE: Adaptado de Moresi et al. (2010).

O conjunto de atividades apresentado por Moresi et al. (2010) é formado por 5 etapas essenciais:

- **Planejamento e Direção:** Nesta etapa é realizado o planejamento das atividades a serem desenvolvidas com base nos objetivos a serem alcançados. Questões como escopo da pesquisa, dados a serem coletados e período de coleta são definidos neste momento. Esta etapa é determinante para que os dados coletados sejam capazes de responder aos questionamentos levantados na pesquisa.
- **Armazenamento e Processamento das Informações:** Neste momento os dados da pesquisa são armazenados e processados. São realizadas tarefas que viabilizam a interpretação dos dados, que é feita na próxima etapa. Deve-se garantir nesta etapa a fidelidade dos dados armazenados em relação aos coletados e uma organização que permita a correta análise dos mesmos.
- **Coleta e Relatórios Adequados:** É o momento em que os dados são organizados de forma a possibilitar a extração de informações úteis para uma

tomada de decisão futura.

- **Análise e Produção:** Nesta fase as informações extraídas na etapa anterior são analisadas. São testadas hipóteses levantadas na fase de planejamento, por exemplo. Os dados podem ainda ser submetidos a simulações para auxiliar as validações e questionamentos.
- **Disseminação:** Na etapa de disseminação os modelos já validados são aplicados, são tomadas decisões baseadas no conhecimento extraído do processo.

Os principais problemas no monitoramento de informações meteorológicas estão relacionados a recepção, armazenamento e processamento dessas informações. Os dados das estações meteorológicas são muitas vezes enviados para um servidor ou coletados manualmente na estação por um profissional, que fica responsável por realizar operações de armazenamento, tratamento, organização e disponibilização (COUTINHO et al., 2015). A total dependência na capacidade do profissional responsável acaba prejudicando a utilização desses dados por pesquisadores de diversas áreas.

Para implementar sistemas de armazenamento de dados meteorológicos que sejam confiáveis, flexíveis e eficientes, instituiu-se nos grandes centros meteorológicos do mundo o uso da tecnologia de Banco de Dados (POTTIER, 1995, 1995; SANDERS, 1997; RAOULT, 1997). De acordo com Date (2004), as vantagens de um sistema de bancos de dados sobre os métodos tradicionais baseados em papel são:

- **Densidade:** não há necessidade de arquivos de papel, possivelmente volumosos;
- **Velocidade:** a máquina pode obter e atualizar dados com rapidez muito maior que o ser humano;
- **Menor trabalho monótono:** grande parte do tédio de manter arquivos à mão é eliminada. As tarefas mecânicas são sempre feitas com melhor qualidade por máquinas;
- **Atualidade:** informações precisas e atualizadas estão disponíveis a qualquer momento para consulta.

Elmasri et al. (2005) destacam que os avanços tecnológicos dos últimos anos geraram aplicações inovadoras dos sistemas de banco de dados, possibilitando o armazenamento de figuras, vídeos e mensagens sonoras por banco

de dados multimídia, a análise de mapas, dados do tempo e imagens de satélite por sistemas de informações geográficas e extração e análise de informações úteis dos bancos de dados para tomada de decisões por sistemas de processamento analítico on-line.

Com a utilização de equipamentos de coleta digital e automática de dados, há a necessidade de ferramentas específicas que facilitem o processamento dessa informação, para depois interpretá-la de forma adequada, devido à grande quantidade de dados obtidos neste tipo de equipamentos (CUNHA, 2001).

## 2.2 *BIG DATA*

Segundo D’Andrea (2010), *Big Data* é um termo que se refere a conjuntos de dados que crescem exponencialmente e que não são possíveis de serem analisados por técnicas tradicionais de banco de dados relacionais devido seu elevado volume, brutalidade ou desestruturação. *Big Data* pode ser resumidamente definido como o processamento (eficiente e escalável) analítico de grande volume de dados complexos produzidos por (várias) aplicações (VIEIRA et al., 2012) como científicas e de engenharias, redes sociais, redes de sensores, dados de *Web Click*, dados médicos e biológicos, transações de comércio eletrônico e financeiros, entre outras (CUZZOCREA et al., 2011).

Um dos motivos que influenciaram o grande aumento de volume de dados é a difusão dos dispositivos de captação de dados, com armazenamento na ordem de Terabytes, e aumento de velocidade de transmissão nas redes. Tal difusão se dá principalmente pelo seu barateamento (redes de sensores, GPS, smartphones), enquanto que as redes aumentaram sua velocidade e abrangência geográfica. Outro fator importante é a facilidade de geração e aquisição de dados gerados digitalmente, como máquinas fotográficas digitais, smartphones, GPS, etc (CHAUDHURI, 2012).

Basicamente, podemos resumir as características do contexto *Big Data* em quatro propriedades (SINGH S.; SINGH, 2012):

- Dados na ordem de dezenas ou centenas de Terabytes (podendo chegar a ordem de Petabytes): envolve, entre outros aspectos, o requisito de alto poder computacional de armazenamento e processamento dos dados;
- Poder de crescimento elástico: está relacionada ao fato de que a quantidade de dados pode variar de alguns Megabytes a vários Terabytes (e vice-versa) em um espaço de tempo relativamente curto, fazendo com que a estrutura

de software/hardware se adapta sob demanda, e seja alocada/desalocada dinamicamente;

- Distribuição do processamento dos dados: os dados devem ser distribuídos de forma transparente em vários nós espalhados de forma a manter a integridade dos dados, o que demanda armazenamento, processamento e controle de falhas distribuído;
- Tipos de dados variados, complexos e/ou semiestruturados: relacionada a adoção de modelos mais apropriados, flexíveis e eficientes para o armazenamento de tipos de dados variados e semiestruturados. Vale ressaltar que o modelo relacional não é o mais adequado pois não possui flexibilidade para o armazenamento de dados e evolução no modelo para tais tipos de dados.

Mesmo com a grande quantidade de dados, um *Big Data* em si não garante a qualidade da informação, pois a análise continua, em grande parte, sendo subjetiva. Isso se deve ao fato que os dados não são autoexplicativos, e o processo de limpeza, amostragem, e relacionamento dos dados continua sendo crítico e passível a erros, aproximações e incertezas (FISHER et al., 2012). Por exemplo, a análise de dados da ordem de Petabytes (ou Exabytes) de cunho científicos (dados genômicos, física ambiental e simulações numéricas) tem se tornado muito comum hoje em dia, onde é aceitável que o resultado da análise contenha imprecisão (certos limites de erros), porém seja computado de forma (relativamente) rápida e/ou em tempo real.

## 2.3 BANCO DE DADOS

Segundo Korth et al. (1999), um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”, ou seja, sempre que for possível agrupar informações que se relacionam e tratam de um mesmo assunto, posso dizer que tenho um banco de dados (ALVES, 2014, (Apostila)).

Podemos tomar como exemplo situações comuns como uma lista telefônica, um catálogo de CDs ou um sistema de controle de RH de uma empresa. O banco de dados pode ser comparado a um armário de arquivamento, ou seja, ele é um recipiente para uma coleção de arquivos de dados computadorizados (DATE, 2004).

Um banco de dados permite colocar à disposição de usuários dados para uma consulta, uma introdução ou uma atualização, assegurando-se dos direitos

atribuídos a esses últimos. Tal ferramenta é ainda mais útil quando os dados são cada vez mais numerosos. A vantagem essencial da utilização dos bancos de dados é a possibilidade de possibilitar o acesso aos dados por vários usuários, simultaneamente, além de facilitar a busca de informações, eliminando os arquivos de papéis, integrando os dados de aplicações e fornecendo segurança (ALVES, 2014, (Apostila)).

Um modelo de dados é um conjunto de conceitos usados para descrever a estrutura e as operações em um banco de dados (ELMASRI; NAVATHE, 2014). O modelo busca sistematizar o entendimento que é desenvolvido a respeito de objetos e fenômenos que serão representados por um sistema informatizado. Os objetos e fenômenos reais, no entanto, são muito complexos e impossibilitam uma representação completa, dessa forma, é necessário construir uma abstração dos objetos e fenômenos do mundo real, de modo a obter uma forma de representação conveniente que seja adequada às finalidades das aplicações do banco de dados (BORGES; DAVIS, 2002). O sucesso da implementação em computador de um sistema de informação depende da qualidade desta transposição de entidades do mundo real e suas interações para um banco de dados informatizado (BORGES; DAVIS, 2002).

Ao longo dos anos, foram criados vários modelos de dados que, apesar de pretenderem se configurar como ferramentas genéricas, refletem as condicionantes tecnológicas da época de sua criação. Os modelos podem ser classificados, segundo Elmasri e Navathe (2014) em:

- Modelos de dados conceituais: são os mais adequados para capturar a semântica dos dados e, conseqüentemente, para modelar e especificar as suas propriedades. Eles se destinam a descrever a estrutura de um banco de dados em um nível de abstração independente dos aspectos de implementação (CHEN, 1976).
- Registra que dados podem aparecer no banco, mas não registra como estes dados estão armazenados. Modelos de dados lógicos: é uma representação lógica das informações em sua área, não é um banco de dados, é independente do modelo físico. Deve ser independente da tecnologia a ser desenvolvida.
- Modelos de dados físicos: são utilizados para descrever as estruturas físicas de armazenamento. É uma descrição de um banco de dados no nível de abstração visto pelo usuário do sistema. Assim, esse modelo depende do sistema que está sendo usado.

Com o objetivo de simplificar a criação e manutenção de um Banco de Dados surgiu os Sistemas Gerenciadores de Banco de Dados (SGBDs). Segundo Date (2004), os SGBDs têm a finalidade de gerenciar as informações de um banco de dados. O gerenciamento implica na definição dos mecanismos para manipulação dessas informações. Um SGBD deve garantir a segurança das informações armazenadas contra eventuais problemas, além de impedir tentativas de acesso não autorizadas (SILBERSCHATZ A.; SUDARSHAM, 2006). Um SGBD fornece aos usuários operações para inserir, alterar, excluir, obter e atualizar dados em um sistema. Existem muitas funcionalidades que um SGBD deve possuir, algumas delas são (ELMASRI et al., 2005):

- Controle de redundância;
- Restrição de acesso não autorizado;
- Garantia de armazenamento persistente para objetos programas;
- Garantia de armazenamento de estruturas para o processamento eficiente de consultas;
- Garantia de backup e restauração;
- Fornecimento de múltiplas interfaces para os usuários;
- Representar relacionamentos complexos entre os dados;
- Forçar as restrições de integridade;
- Permitir interferências e ações usando regras.

Uma das mais importantes funções de um SGBD é o gerenciamento de transações, definida como uma coleção de operações que desempenha uma função lógica dentro de uma aplicação do sistema de banco de dados, em outras palavras, um conjunto de operações de leitura ou escrita que são realizadas no banco de dados (LÓSCIO et al., 2011). As operações de banco de dados que formam uma transação podem estar embutidas em um programa de aplicação ou especificadas interativamente, via linguagem de consulta (ELMASRI et al., 2005).

Lóscio et al. (2011) destaca que a execução de transações em um SGBD deve obedecer a algumas propriedades a fim de garantir o correto funcionamento do sistema e a respectiva consistência dos dados. Estas propriedades são chamadas de propriedade ACID, definidas como:

### 2.3.1 Histórico

Segundo Takai et al. (2005), o primeiro SGBD comercial surgiu no final de 1960 baseado nos primitivos sistemas de arquivos disponíveis na época, os quais não controlavam o acesso por vários usuários ou processos.

Desde o surgimento dos SGBDs, diferentes modelos de dados foram propostos, os quais diferenciam-se pelos conceitos adotados na representação dos dados do mundo real. Inicialmente, foram propostos os modelos hierárquico e de rede (LÓSCIO et al., 2011). No modelo hierárquico os dados são representados como registros, organizados em árvores e relacionados por meio de associações do tipo pai-filho. O modelo em rede foi proposto como uma extensão ao modelo hierárquico, no qual não existe o conceito de hierarquia e um mesmo registro pode estar envolvido em várias associações (LÓSCIO et al., 2011).

No entanto, devido ao grande número de restrições ao se relacionar estruturas no mundo real, este modelo foi perdendo força para dar lugar aos bancos de dados relacionais, em 1970, os quais, segundo Takai et al. (2005), se firmaram como solução comercial para armazenamento e gerenciamento de dados convencionais, ou seja, dados caracterizados por uma estrutura fixa, bem definida e com tipos de dados simples (ex: sistemas de controle de estoque e folha de pagamento).

Brito (2010) destaca que, o modelo relacional de dados tem sido utilizado em larga escala pela maioria dos SGBDs. Outro ponto importante a salientar sobre o modelo relacional é a utilização das restrições de integridade que garantem a consistências dos dados em um banco de dados. Estas restrições, em sua grande maioria, são conhecidas como chaves primárias (*primary key*) e chaves estrangeiras (*foreign key*) (BRITO, 2010).

Posteriormente, com o surgimento da Web, novas aplicações de banco de dados começaram a ser desenvolvidas, originando novos requisitos de bancos de dados (LÓSCIO et al., 2011). Dentre esses requisitos destaca-se a necessidade de manipulação de grandes volumes de dados não estruturados ou semi-estruturados, bem como novas necessidades de disponibilidade e escalabilidade. Assim, de forma a corresponder aos requisitos destas aplicações, novas soluções para gerenciamento de dados começaram a ser propostas, como, por exemplo, os bancos de dados NoSQL (LÓSCIO et al., 2011).

Segundo Lóscio et al. (2011) as primeiras propostas de bancos de dados não relacionais foram desenvolvidas por pequenas empresas e por comunidades de software livre. Tais soluções receberam a terminologia “NoSQL” (*Not Only SQL*), que significa “não apenas SQL”. Este termo faz referência a SGBDs que não adotam o modelo relacional e são mais flexíveis quanto às propriedades ACID.

Esta flexibilidade torna-se necessária devido os requisitos de alta escalabilidade necessários para gerenciar grandes quantidades de dados, bem como para garantir a alta disponibilidade dos mesmos (LÓSCIO et al., 2011).

### 2.3.2 Banco de dados relacional

Modelo relacional (MR) é um modelo de dados representativo (ou de implementação) que foi proposto por Ted Codd, em 1970. O modelo baseia-se em conceitos matemáticos – teoria dos conjuntos e lógica de predicado (TAKAI et al., 2005). Os primeiros sistemas comerciais baseados no modelo relacional foram disponibilizados em 1980 e desde então são implementados em muitos sistemas, como Access, Oracle, MySql, entre outros (ELMASRI et al., 2005).

O modelo relacional surgiu devido à necessidade de se ampliar a independência de dados nos sistemas gerenciadores de banco de dados, prover um conjunto de funções apoiadas em álgebra relacional para armazenamento e recuperação de dados e possibilitar processamento dedicado (exclusivo), características não contempladas pelos bancos de dados até então disponíveis (CODD, 1970).

O Modelo relacional revelou-se ser mais flexível e adequado ao solucionar os vários problemas que se colocam no nível da concepção e implementação da base de dados além de ser mais simples, com estrutura de dados uniforme e formal (TAKAI et al., 2005).

A estrutura fundamental do modelo relacional é a relação (tabela). Uma relação é constituída por um ou mais atributos (campos) que traduzem o tipo de dados a armazenar. Cada instância do esquema (linha) é chamada de tupla (registro) (ELMASRI et al., 2005) (Figura 2).

| TORRE FÍSICA AMBIENTAL |                        |       |         |
|------------------------|------------------------|-------|---------|
| Data                   | Variável               | Valor | Unidade |
| 2012-01-10 10:10:00    | Temperatura do ar      | 27    | °C      |
| 2012-01-10 10:10:00    | Unidade relativa do ar | 54    | %       |
| 2012-01-10 10:10:00    | Velocidade do Vento    | 3     | m/s     |
| 2012-01-10 10:10:00    | Direção do Vento       | 138   | °       |
| 2012-01-10 10:10:00    | Radiação solar         | 600   | W/m²    |

Figura 2: Atributos e Tuplas de uma Relação de uma Torre Meteorológica  
 FONTE: Adaptado de Elmasri et al. (2005).

Outra característica do modelo relacional é o processo de Normalização com o objetivo de aplicar uma série de passos com determinadas regras sobre a tabela do banco de dados na forma de garantir o projeto adequado dessas tabelas. Um conceito básico da normalização consiste na separação de dados referentes a elementos distintos em tabelas distintas, conectadas através da utilização das chaves. Essas regras permitem um armazenamento consistente e, além disso, um acesso eficiente aos dados, reduzindo redundâncias e diminuindo as chances dos dados se tornarem inconsistentes (CODD, 1970).

A SQL (*Structured Query Language*) é a linguagem de consulta implementada pela maioria dos sistemas gerenciadores de banco de dados relacionais. Foi criada pela IBM Research, no início da década de 1970 (DATE, 2004). Baseada nas linguagens de Álgebra e Cálculo Relacional, e primeiramente denominada SEQUEL (*Structured English QUery Language*), a SQL é hoje a linguagem padrão para Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDR) (ELMASRI et al., 2005).

Todos esses diferentes recursos auxiliaram a manter os SGBDRs sempre em uma posição de predominância entre os diversos tipos de ambientes computacionais, mas ao mesmo tempo, não impediu a ocorrência de determinados problemas, isso devido ao crescimento do volume de dados presente nos bancos de dados de algumas organizações (CUZZOCREA et al., 2011).

### 2.3.2.1 Limitações dos Bancos de dados Relacionais

Os principais problemas encontrados com a utilização do modelo relacional estão principalmente na dificuldade de conciliar o tipo de modelo com a demanda de escalabilidade cada vez mais frequente (BRITO, 2010).

De acordo com Ferreira et al. (2000) a escalabilidade de uma base de dados relacional possui um limite pois, à medida que a base de dados é distribuída, a complexidade das ligações com tabelas, que não estejam na mesma máquina, aumenta. Do mesmo modo, à medida que o número de instâncias da base de dados aumenta, complica-se a gestão da garantia das propriedades ACID em transações distribuídas.

Por isso, as bases de dados relacionais começaram a evidenciar os seguintes problemas quando lidam com volumes enormes de informação:

- Escalabilidade: as bases de dados escalam, mas não escalam facilmente. É possível escalar vertical e/ou horizontalmente mas a complexidade exigida na gestão dos múltiplos nós da base dados, ou o custo do hardware para suportar a distribuição, são entraves.

- Um aumento na complexidade geral da base de dados e na sua gestão, quando esta é distribuída por várias instâncias.
- As propriedades ACID de uma base de dados relacional obrigam que tenham uma distribuição e não isolamento dos dados.

### 2.3.3 Banco de dados NoSQL

O NoSQL surgiu no contexto de grande quantidade de dados gerados em curto intervalo de tempo. Com isso, para a manipulação desses dados, os SGBDs necessitam de um grande poder de performance de forma eficiente e escalável. Pensando nisso os projetistas do NoSQL promoveram uma alternativa de alto armazenamento com velocidade de grande disponibilidade, eliminando algumas regras e estruturas que norteiam o modelo relacional. Apesar do rompimento com as regras, havia ganho de performance, flexibilizando os sistemas de banco de dados para as diversas características, que são peculiares de cada empresa, que passou a ser fundamental para suprir os requisitos de alta escalabilidade necessária para o gerenciamento de grandes quantidades de dados, assim como para garantir uma alta disponibilidade dos mesmos (SOUZA; SANTOS, 2015).

Segundo Tiwari (2011), NoSQL é um conceito vasto que engloba todas as bases de dados que não seguem os princípios das bases de dados relacionais e que estão ligados a grandes volumes de dados. Os bancos de dados NoSQL apresentam determinadas características consideradas importantes, que o tornam diferentes dos bancos de dados relacionais (FERREIRA et al., 2000; ANDERSON et al., 2010):

- Escalabilidade horizontal: ausência de bloqueios, o que permite a escalabilidade horizontal com uma maior facilidade e eficiência (ele não é afetado pelo aumento da concorrência). Uma alternativa muito usada para promover a escalabilidade horizontal é o *Sharding*, que particiona os dados em múltiplas tabelas a serem armazenadas ao longo de diversos nós da rede. O que esta técnica faz, na realidade, é interromper a cadeia de relacionamentos;
- Ausência de esquema (*Schema-free*) ou esquema flexível: é justamente esta ausência de esquema que facilita a alta escalabilidade e alta disponibilidade.
- Suporte nativo e a replicação: outra forma de prover a escalabilidade, pois, no momento em que permitimos a replicação de forma nativa o tempo gasto para recuperar informações é reduzido (LAKSHMAN A.; MALIK, 2010);

- Consistência eventual: nem sempre a consistência dos dados é mantida. Esta característica tem embasamento no teorema CAP (*Consistency, Availability e Partition tolerance*) que afirma que em um dado momento só é possível garantir duas dessas três propriedades, que seriam consistência, disponibilidade e tolerância a partição (BREWER, 2000). No mundo real, normalmente estas duas últimas são privilegiadas. Como consequência disto, as propriedades do ACID não são respeitadas simultaneamente, ao contrário disto, temos outro conjunto de projetos denominado Base (*Basically Available, Soft state, Eventual consistency*, ou seja Basicamente disponível, Estado leve e Consistência eventual) (BRITO, 2010), indicando que é necessário haver um planejamento para que o sistema possa tolerar inconsistências temporárias com o objetivo de priorizar a disponibilidade (PRITCHETT, 2008; SILVA, 2011).

O termo NoSQL surgiu em 1998, a partir de uma solução de banco de dados que não oferecia uma interface SQL, mas esse sistema ainda era baseado na arquitetura relacional. Uma das primeiras implementações maduras de um sistema realmente não-relacional surgiu em 2004 quando o Google lançou o BigTable (BRITO, 2010). Em 2008, iniciou-se outro importante projeto, o Cassandra. Este último foi desenvolvido pelo site Facebook para lidar com grandes volumes de dados (LAKSHMAN A.; MALIK, 2009). No início de 2010 o Cassandra desbancou o MySQL como banco de dados do Twitter, demonstrando sua importância cada vez mais crescente (LAI, 2010).

### 2.3.3.1 Classificação de banco de dados NoSQL

Apesar de possuírem certas características em comum, como serem livres de esquema, promoverem alta disponibilidade e maior escalabilidade, os sistemas de bancos de dados NoSQL existentes possuem diversas singularidades (BRITO, 2010).

Segundo o mesmo autor, quanto à distribuição de dados, certos sistemas promovem o particionamento e a replicação dos dados, enquanto outros deixam essa tarefa para o usuário. A maioria das soluções é distribuída, como é caso do Amazon Dynamo, do CouchDB, do MongoDB, do BigTable e do Cassandra.

Quanto ao modelo de dados, os bancos de dados NoSQL podem ser geralmente classificados em quatro grandes grupos (CATTELL, 2011):

- Modelo chave/valor (*key-value*): o mais simples deles, onde todos os itens do banco são guardados como um atributo (ou chave) junto com o seu

valor, como é o caso do Dynamo, Azure Table Storage, Couchbase Server, Riak, Redis, LevelDB, Chordless, GenieDB, Scalaris, Tokyo Cabinet/Tyrant, GT.M, Scalien, Berkeley DB, Voldemort, Dynamite, KAI, MemcacheDB, Faircom C-Tree, HamsterDB, STSdb, Tarantool/Box, Maxtable, Pin caster, RaptorDB, TIBCO Active Spaces, allegro-C, nessDB, HyperDex, Mnesia, LightCloud, Hibari, BangDB.

- Modelo orientado a documentos: os documentos são as unidades básicas de armazenamento e não utilizam necessariamente qualquer tipo de estruturação pré-definida. Neste modelo cada item é um documento, podendo um documento conter outros documentos, em formato similar a do XML, entre os quais temos o MongoDB, CouchDB, BigCouch, RavenDB, Clusterpoint Server, ThruDB, TerraStore, RaptorDB, JasDB, SisoDB, SDB, SchemaFreeDB, djondb;
- Modelo orientados a grafos: para quando os dados são armazenados em nós de um grafo cujas arestas representam o tipo de associação entre esses nós, como são os casos do Neo4J, Infinite Graph, Sones, InfoGrid, HyperGraphDB, DEX, Trinity, AllegroGraph, BrightStarDB, BigData, Meronymy, OpenLink Virtuoso, VertexDB, FlockDB;
- Modelo orientados a colunas (*column Family*): no qual muda-se o paradigma de orientação a registros (ou tuplas) para orientação a atributos (ou colunas). Possui colunas (tuplas que contém nome, valor e *timestamp*), famílias de colunas (um repositório para colunas, análogo a uma tabela do Modelo Relacional) e super-colunas (compostas por arrays de colunas) (LAKSHMAN A.; MALIK, 2009), que tem como exemplos o Hbase, Cassandra, Hypertable, Accumulo, Amazon SimpleDB, Cloudata, Cloudera, SciDB, HPCC, Stratosphere.

Como trataremos de dados climáticos o modelo orientado a coluna foi escolhido para estudo, devido sua estrutura de tabelas comportarem a forma de armazenamento dos dados coletados pelos sensores, contendo nome, valor e *timestamp*.

### 2.3.3.2 Banco de dados Orientado a coluna (*Column Family*)

Demonstra maior complexidade que o de Chave-valor. Este tipo de banco de dados foi criado para armazenar e processar uma grande quantidade de dados distribuídos em diversas máquinas (WEBER, 2010).

Aqui existem as chaves, mas neste caso, elas apontam para atributos ou colunas múltiplas. Neste caso, os dados são indexados por uma tripla (coluna, linha e *timestamp*), sendo que a coluna e a linha são identificadas por chaves e o *timestamp* permite diferenciar múltiplas versões de um mesmo dado (WEBER, 2010).

Como o próprio nome sugere, as colunas são organizadas por famílias da coluna. Vale destacar que as operações de escrita e leitura são atômicas, ou seja, os valores associados a uma linha são considerados em sua execução independentemente das colunas que estão sendo lidas/escritas, reunindo assim colunas que armazenam o mesmo tipo de informação (WEBER, 2010).

## 2.4 TESTES DE DESEMPENHO

Testes de desempenho consistem em testar um sistema quanto aos seus requisitos de desempenho. Alguns exemplos desses requisitos, segundo Denaro et al. (2004) são:

- Latência: que é o tempo entre uma requisição e a completude de resposta da operação requisitada, medida em (*us*) milissegundos;
- Vazão (*throughput*): o número de operações que o sistema é capaz de completar em um dado período de tempo;
- Escalabilidade: a característica que indica sua capacidade de crescente de trabalho de forma uniforme;

Para a correta execução desse tipo de teste é necessário que haja um conjunto bem definido de objetivos para esses requisitos, do contrário, esses testes serão medições cegas (WEYUKER; VOKOLOS, 2000). Apesar de um teste de desempenho completo e ideal depender da existência de um sistema totalmente integrado e funcional, no contexto do seu funcionamento, testes de desempenho são frequentemente aplicados em todos os passos do processo de construção do sistema (PRESSMAN, 2011). Isso se fundamenta no fato de que quanto mais cedo uma falha é detectada, mais eficiente e barata é a sua solução, principalmente levando em consideração o fato de que a maioria dos problemas críticos de desempenho advém de decisões feitas em estágios iniciais do ciclo de desenvolvimento do software (DENARO et al., 2004).

### 2.4.1 *Benchmark*

Um *benchmark* consiste em um padrão de testes para medir ou avaliar algo, sendo utilizado também para analisar o desempenho de um sistema. Pode ser implementado como uma instrução, um programa ou uma análise específica de pedidos de interações com um componente de *software* (GUIMARAES, 2007).

Segundo Svobodova (1976), a finalidade da avaliação de desempenho de um Sistema é verificar a efetividade com que os recursos de *hardware* são utilizados para atender aos objetivos do *software*. Esta avaliação poderá ser comparativa, quando o desempenho do sistema é avaliado mediante o desempenho de outro sistema, ou analítica, quando o sistema é avaliado com vários parâmetros de carga e de configuração do próprio sistema.

Os programas de *benchmark* efetuam um número de operações pré-definidas (carga de trabalho) e apresenta um resultado em algum formato (métrica), que relata o comportamento do sistema. As métricas podem calcular a velocidade para completar a carga de trabalho ou vazão, que seriam quantas cargas de trabalho por unidade de tempo foram medidas (PIRES et al., 2006). Segundo Gray (1993), um *benchmark* deve atender os seguintes critérios:

- Relevância: As medidas verificadas deverão descrever a funcionalidade e expectativa de desempenho sobre o produto submetido ao *benchmark*. É necessário especificar com clareza o domínio da utilização da ferramenta e as operações a serem submetidas;
- Portabilidade: Na elaboração de um *benchmark*, as definições e os termos utilizados devem estar em um nível de abstração que permita transportá-lo para as implementações de diferentes ferramentas, tornando o *benchmark* aplicável a uma gama maior de ferramentas a serem avaliadas;
- Escalabilidade: O padrão das medições deve atender a pequenos ou grandes sistemas, independente do nível de complexidade dos mesmos. Fatores como monoprocessamento ou multiprocessamento, *clock* e tamanhos de memórias não devem tornar as medidas inconsistentes, nem influenciar na aplicabilidade do *benchmark*;
- Simplicidade: Os critérios definidos no *benchmark* devem ser simples e de fácil compreensão. Porém, a relevância das métricas não deve ser negligenciada. Medidas muito simples ou complexas podem não refletir boas características de julgamento de uma ferramenta.

Existem várias implementações de *benchmarks* para realizar medidas de desempenho, nos mais diversos sistemas, como servidores WEB, SGBDs, aplicações gráficas, entre outros. Como exemplo, pode-se citar: *benchmarks* para banco de dados (AS3AP, DBT-2, OSDB, TPCC-UVA, TPC-C, Wiscosin, 007, JMetrics, YCSB) e *benchmarks* para arquiteturas de CPUs (SPEC 95, SPEC 2000).

### 2.4.2 YCSB

O Benchmark escolhido para realização dos testes foi o *Yahoo! Cloud Serving Benchmark (YCSB)*, utilizado por Souza e Santos (2015) e Cooper et al. (2010). Trata-se de um *framework* capaz de realizar *benchmarks* em diferentes sistemas gerenciadores de banco de dados, consistindo em um cliente gerador de cargas de trabalho com diferentes características, tais como cargas de trabalho com muitas leituras, muitas escritas e muitos escaneamentos. Uma das grandes vantagens do YCSB é a capacidade de criar ou adaptar cargas de trabalho (COOPER et al., 2010).

Esse *benchmark* foi desenvolvido para comparar o PNUTS (COOPER et al., 2008), um SGBD próprio do Yahoo!, com outros SGBD em nuvem, principalmente os NoSQL (COOPER et al., 2010), considerando o fato de os bancos NoSQL que estavam surgindo terem modelos de dados diferentes entre si, o que dificultava a comparação através dos *benchmarks* existentes (LEMOS P.H.S.; FIGUEIREDO, 2014).

O YCSB possui uma única tabela, chamada *data* representada na Figura 3, na qual as cargas de trabalho são executadas. O *benchmark* possui 6 cargas de trabalho pré-definidas, com alterações sobre as operações e frequências na base de dados. Além dos existentes, o YCSB permite que novas cargas de trabalho sejam criadas, de forma a atender a necessidades específicas (LEMOS P.H.S.; FIGUEIREDO, 2014).

| usertable |                      |
|-----------|----------------------|
| PK        | <u>YCSB KEY int</u>  |
|           | FIELD1 VARCHAR(100)  |
|           | FIELD2 VARCHAR(100)  |
|           | FIELD3 VARCHAR(100)  |
|           | FIELD4 VARCHAR(100)  |
|           | FIELD5 VARCHAR(100)  |
|           | FIELD6 VARCHAR(100)  |
|           | FIELD7 VARCHAR(100)  |
|           | FIELD8 VARCHAR(100)  |
|           | FIELD9 VARCHAR(100)  |
|           | FIELD10 VARCHAR(100) |

Figura 3: Esquema do YCSB

O YCSB possui um programa Java denominado “Client” que gera os dados a serem carregados para o banco de dados, e gera as operações que compõem as cargas de trabalho *Workloads*. A arquitetura do YCSB “Client” é mostrada na Figura 4.

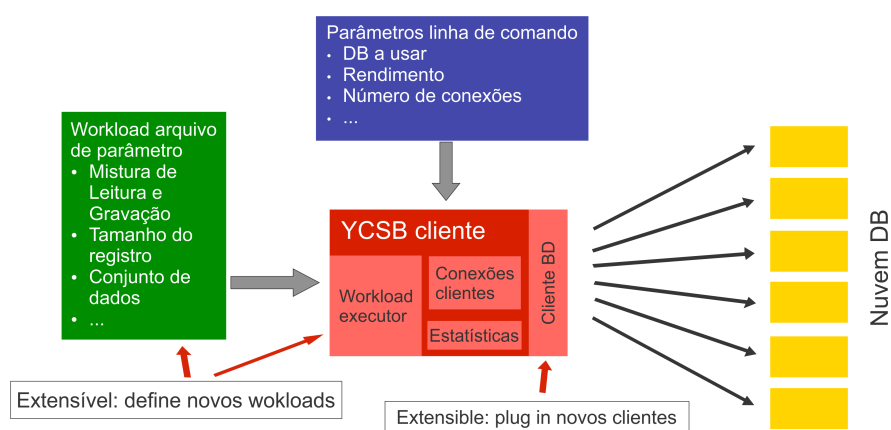


Figura 4: Arquitetura do YCSB *Client*  
 FONTE:(COOPER et al., 2010)

O YCSB “Client” deve fazer várias escolhas aleatórias quando está gerando cargas, tais como: qual operação realizar (inserção, atualização, leitura ou sele-

ção), qual registro ler e quantos registros verificar. Essas decisões ficam a cargo de distribuições, que serão tratadas nas seções que seguem.

O *client* tem uma série de propriedades que definem o seu funcionamento. Por convenção, dividimos essas propriedades em dois grupos: Propriedades da Carga de Trabalho (*Workload*) e Propriedades de tempo de execução (*Runtime*).

As *Workloads* são propriedades que definem a carga de trabalho, independente de um determinado banco de dados ou experimento em processamento. Por exemplo, o mix de leitura / gravação do banco de dados, a distribuição a ser usada, tamanho e número de campos em um registro. Já as *Runtimes* são propriedades específicas para um determinado experimento. Por exemplo, a camada de interface de base de dados a ser usado (Cassandra, PostgreSQL, etc.), propriedades inicialização desta camada (como os nomes dos caminhos dos serviços do banco de dados), o número de *threads* (conexões) de cliente, etc.

#### 2.4.2.1 Propriedades das Cargas de trabalho do Benchmark (*WORKLOADS*)

As cargas de trabalho no *YCSB Core Package* são variações de um mesmo tipo de aplicações básicas. Nessas aplicações, existe uma tabela de registros, cada um com  $F$  campos. Cada registro é identificado por uma chave primária, que é uma *string* como "user234123". Cada campo é chamado Field0, Field1 e assim por diante. Os valores de cada campo são uma sequência aleatória de caracteres ASCII de comprimento  $L$ . Por exemplo, nos resultados apresentados neste estudo, construíram-se 15.000.000 de registros usando  $F = 20$  campos, cada um com  $L = 100$  bytes.

Cada operação sobre o armazenamento dos dados é escolhida entre as seguintes (COOPER et al., 2010):

- *Insert* (Inserção): Inserir um novo registro.
- *Update* (Atualização): Atualizar um registro, substituindo o valor de um campo.
- *Read* (Leitura): Ler um registro, em um ou outro campo escolhido aleatoriamente ou em todos os campos.
- *Scan* (Seleção): Verificar os registros em ordem, começando com uma chave de registro escolhido aleatoriamente. O número de registros para fazer a Seleção é escolhido aleatoriamente.

Quanto a distribuição das escolhas por determinados registros na realização das operações, o YCSB possui várias distribuições pré-definidas (COOPER et al., 2010):

- *Uniform*: Escolha de um item uniformemente ao acaso. Por exemplo, ao escolher um registro, todos os registros no banco de dados possuem a mesma probabilidade de serem escolhidos.
- *Zipfian*: Escolha de um item de acordo com a distribuição Zipfian. Por exemplo, ao escolher um registro, alguns registros se mostrarão extremamente populares (cabeça da distribuição), enquanto a maioria dos registros serão impopulares (cauda).
- *Latest*: Como a distribuição Zipfian, os registros mais recentemente inseridos estão na cabeça da distribuição.
- *Multinomial*: As probabilidades para escolha de cada registro pode ser especificada. Por exemplo, poderíamos atribuir uma probabilidade de 0,95 para a operação de *read*, uma probabilidade de 0,05 para a operação *Update*, e uma probabilidade de 0 para *Scan* e *Insert*. O resultado seria uma carga de trabalho de leitura pesada.

A Figura 5 ilustra a diferença entre as distribuições *Uniform*, *Zipfian* e *Latest*. Os eixos horizontais na figura representam os registros que podem ser escolhidos, para uma operação de *Insert*, enquanto que as barras verticais representam a probabilidade de cada registro ser escolhido (COOPER et al., 2010).

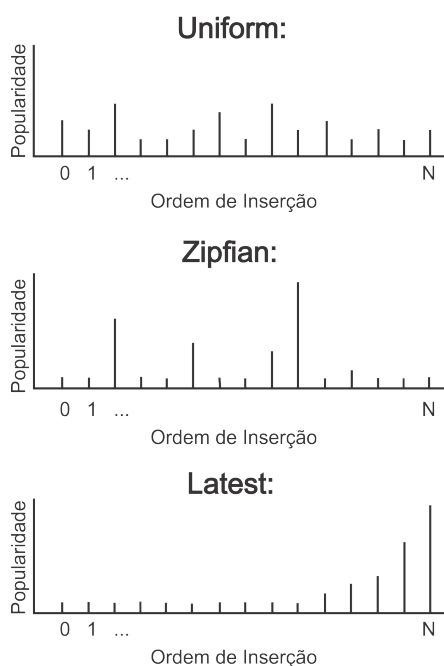


Figura 5: As distribuições de probabilidade. Eixos horizontal representa os itens em ordem de inserção, e os eixos verticais representam probabilidade de ser escolhido.

Fonte: (COOPER et al., 2010)

As definições das cargas de trabalho no YCSB Core Package com atribuição das operações são apresentadas na Tabela 1.

Tabela 1: Cargas de trabalho no YCSB Core Package - configuração padrão

| Carga de Trabalho (Worload) | Operações                        | Seleção de registros |
|-----------------------------|----------------------------------|----------------------|
| A - Leitura e Modificação   | Leitura: 50% Modificação: 50%    | Zipfian              |
| B - Leitura e Modificação   | Leitura: 95% Modificação: 5%     | Zipfian              |
| C - Somente Leitura         | Leitura: 100%                    | Zipfian              |
| D - Leitura e Inserção      | Leitura: 95% e Inserção 5%       | latest               |
| E - Seleção e Inserção      | Seleção: 95% e Inserção 5%       | Zipfian/Uniform      |
| F - Leitura e Leitura Prop. | Leitura 50% e Leitura Prop.: 50% | Zipfian              |

Os arquivos de propriedade usados com os geradores de carga de trabalho do *YCSB Core Package* podem especificar valores para algumas propriedades do *Workflow* (COOPER, 2015):

- *FieldCount*: o número de campos em um registro (utilizado 20, conforme proposto por (COOPER, 2015) para processamento de testes de longa duração);
- *Fieldlength*: o tamanho de cada campo (30 bytes - considerando as demandas provenientes da quantidade de caracteres geradas em uma linha de

dados climáticos);

- *Readallfields*: deve-se ler todos os campos (true) ou apenas um (false) (mantido padrão: true, uma vez que o teste busca simular o stress gerado pela utilização do BD em sua totalidade);
- *Operationcount*: número de operações a realizar (foram utilizadas quantidades diferentes de operações, variando de 2.000 à 14.000, de forma a possibilitar o entendimento do comportamento dos bancos sob diferentes demandas de operações conforme proposto por Cooper et al. (2010) e Souza e Santos (2015));
- *Table*: o nome da tabela (utilizado "data");
- *Recordcount*: número de registros a serem carregados inicialmente no banco de dados (utilizado 15.000.000, conforme proposto por (COOPER, 2015) para processamento de testes de longa duração).

#### 2.4.2.2 Propriedades de tempo de execução (*Runtime*)

Embora os arquivos de classes e parâmetros das cargas de trabalho definam um *Workload* específico, há configurações adicionais que podem ser necessárias para um processamento particular do *benchmark*. Essas configurações são fornecidas na linha de comando quando o YCSB “*client*” é executado. Essas configurações são:

- *Threads*: o número de conexões de clientes. Como configuração padrão, o YCSB “*client*” usa um único *Thread* de trabalho, mas valores adicionais podem ser especificados. Isso é muitas vezes feito para aumentar a quantidade de carga oferecida sobre a base de dados (utilizado 50, conforme proposto por (SOUZA; SANTOS, 2015)).
- *S*: status. Se trata do relatório YCSB “*client*”, utilizado para uma carga de trabalho de longa duração, possibilitando o acompanhamento do progresso da operação. Ao especificar "-s" na linha de comando, o “*client*” informará o estado da operação a cada 10 segundos.

## 2.5 TESTES ESTATÍSTICOS DE KRUSKAL WALLIS

O Teste de Kruskal Wallis (ou teste H) se trata de um teste não paramétrico utilizado para comparar três ou mais populações (CHAN; WALMSLEY, 1997). Ele é usado para testar a hipótese nula de que todas as populações possuem funções de distribuição iguais contra a hipótese alternativa de que ao menos duas das populações possuem funções de distribuição diferentes. Ou seja, ele determina se três ou mais grupos independentes são o mesmo ou diferente em alguma variável de interesse para um nível ordinal de dados ou um nível intervalar ou de razão de dados disponível (CHAN; WALMSLEY, 1997).

O teste de Kruskal-Wallis é o análogo ao teste F utilizado na ANOVA 1 fator. Enquanto a análise de variância depende da hipótese de que todas as populações em confronto são independentes e normalmente distribuídas, o teste de Kruskal-Wallis não coloca nenhuma restrição sobre a comparação (CHAN; WALMSLEY, 1997).

A aplicação do teste utiliza os valores numéricos transformados em postos e agrupados em um só conjunto de dados. Isto é, todos os escores de todas as k amostras combinadas estão dispostos em uma única série de postos. Ao menor escore atribui-se o posto q, ao seguinte o posto 2 e assim sucessivamente até o maior posto que é n, onde n é o número total de observações independentes nas k amostras. A comparação dos grupos é realizada por meio da soma dos postos, o teste verifica se estas somas são tão diferentes entre si que não seja provável que tenham sido retiradas de uma mesma população (VARGHA; DELANEY, 1998).

Pode-se mostrar que se as k amostras forem efetivamente de uma mesma população, isto é, se  $H_0$  é verdadeira, então H (estatística de Kruskal-Wallis calculada abaixo) tem uma distribuição qui-quadrado com graus de liberdade (gl) = k-1, desde que os tamanhos das amostras não sejam muito pequenos (VARGHA; DELANEY, 1998). Isto é:

$$H = \frac{12}{n(n+1)} \sum_{j=1}^k \frac{R_j^2}{n_j} - 3(n+1), \text{ onde} \quad (1)$$

- K=número de amostras;
- $N_j$ =número de elementos na amostra “j”

- $R_j$ =soma dos postos na amostra (coluna) “j”
- $N = \sum n_j$  = número total de elementos em todas as amostras combinadas.

O teste de Kruskal-Wallis pressupõe as seguintes condições para o seu adequado uso (BRESLOW, 1970):

- Comparação de três ou mais amostras independentes;
- Não pode ser usado para testar diferenças em uma única amostra de respondentes mensurados mais de uma vez;
- Dados cujo nível de mensuração seja no mínimo ordinal e que possam ser ordenados e aos quais seja possível atribuir postos ou ordens;
- O tamanho mínimo de cada amostra deve ser de 6 para se poder recorrer ao  $\chi^2$ .

Quando  $n_j > 6$ , e há mais do que 5 amostras, a significância de H pode ser determinada por recorrência à Tabela do Qui-quadrado. Para testar diferenças entre amostras de tamanho inferior a 6 e número de amostras  $\leq 5$ , deve-se recorrer à tabelas especiais (Tabela de valores críticos da distribuição H de Kruskal-Wallis) (VARGHA; DELANEY, 1998).

## Capítulo 3

# MATERIAL E MÉTODOS

Este trabalho envolve a manipulação do armazenamento de dados meteorológicos, utilizados por pesquisas em ambientes atmosféricos naturais e urbanos, sobre o comportamento das variáveis do solo, balanço de energia e evapotranspiração.

Nesse sentido buscou-se estudar o comportamento de dois diferentes bancos de dados (Relacional e NoSQL) na manipulação desses dados científicos, considerando o desempenho dos mesmos quanto a aspectos como: tempo de espera entre operações requisitadas (Latência) e velocidade para realização de determinadas operações (Fluxo de dados).

### 3.1 MATERIAIS

Os materiais escolhidos seguiram alguns critérios de escolha, sendo o principal deles, no caso dos *softwares*, a utilização de *softwares* livres, considerando que os mesmos possibilitam o acesso ao seu código fonte, quando necessário, além de estarem descritos e discutidos em diversos fóruns, artigos e dissertações *on line*.

#### 3.1.1 *Hardware*

De forma a possibilitar o entendimento sobre a sensibilidade dos testes ao *hardware* e variações geradas pela repetição dos testes, os experimentos foram executados em um laboratório da UFMT contendo 30 máquinas. Os computadores que foram utilizados na execução das tarefas de servidor de banco de dados eram caracterizados pela seguinte configuração:

- Placa-mãe - Hewlett-Packard - 1850;

- Processador AMD A10-5800B 1.4 GHZ - 64 Bits;
- 2 X 2 Gigabytes (GB) - DIMM Synchronous 800 Megahertz (MHz) (1.0 ns) (4 Gb no total);
- Sistema Operacional - Ubuntu Release 14.04 64 bits - Kernel Linux 4.2.0-18-generic;
- Disco Rígido Seagate 500 Gb.

### 3.1.2 Sistemas de Gerenciamento de Banco de Dados

Para a análise do BD Relacional foi utilizado o PostgreSQL 9.4 (Utilizado na implementação do sistema Mimi, desenvolvido por (OLIVEIRA, 2011)) e, para o BD NoSQL, o Cassandra na versão 2.1.11, utilizado por grandes empresas, tais como a Spotify, Netflix, Apple, etc.

#### 3.1.2.1 PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados objeto relacional, por apresentar características de um SGBD relacional e ainda, funcionalidades de orientação a objetos (GROUP, 2015). É um sistema de código aberto que concorre com produtos comerciais como SQL Server e Oracle.

Por se tratar de um código aberto, com licença BSD (*Berkeley Software Distribution*), o sistema possibilita o acesso, alterações e adaptações no código por seus próprios usuários (OBE; HSU, 2011).

O PostgreSQL segue a padronização SQL, uma linguagem de interface para SGBD. Possui uma ferramenta de texto chamada psql, para receber os comandos SQL, e uma interface gráfica chamada PGAdmin para tornar mais fácil e intuitivo o uso do SGBD. O sistema possibilita também a manipulação dos dados armazenados através de camadas ODBC e JDBC ou com acesso nativo na linguagem C, tornando possível desenvolver aplicações complexas usando linguagem de programação, como C/C++, Delphi, Java, Perl, PHP, Python, Visual Basic, e outras (LEITE, 2007).

#### 3.1.2.2 Cassandra

A base de dados Cassandra surgiu, inicialmente, como um projeto dos desenvolvedores do Facebook (LAKSHMAN A.; MALIK, 2009), afim de criar um banco de dados que fosse escalável (HAN et al., 2011) e otimizado para o

propósito da rede social. Em 2008, seu código foi aberto para a comunidade e, posteriormente, mantido pela fundação Apache e colaboradores de diversas empresas.

O Cassandra é um banco de dados não-relacional de natureza distribuída (NoSQL) que se utiliza do modelo de dados de linhas particionadas com consistência ajustável (CATTELL, 2011). Atualizações são armazenadas em memória e posteriormente são escritas em disco, e as informações no disco são periodicamente compactadas. O Cassandra faz particionamento, replicação, detecção de falhas e recuperação automaticamente (LAKSHMAN A.; MALIK, 2009).

O Cassandra foi projetado, desde o início, para ser um sistema distribuído. O comportamento padrão do Cassandra, diferentemente dos bancos relacionais, não garante a consistência dos dados em todas as réplicas. Isso é feito com o objetivo de diminuir os tempos de leitura e escrita, de forma a não prejudicar sua escalabilidade e, possivelmente, no entanto, alterar a consistência para cada leitura e escrita em particular (SOUZA; SANTOS, 2015).

### 3.1.3 *YCSB*

O *software* escolhido para a análise de desempenho dos BDs em estudo foi definido conforme os mesmos critérios descritos utilizados para definição dos BDs, considerando a restrição de que um mesmo *Benchmark* precisaria realizar as análises de desempenho nos dois modelos de BDs em estudo (relacional e NoSQL), além de ser adequado para análises específicas em banco de dados.

## 3.2 MÉTODOS

A definição dos procedimentos necessários para estudar o comportamento dos dois diferentes bancos de dados (Relacional e NoSQL) na manipulação de dados climáticos, considerando o desempenho dos mesmos, se fundamentou em 2 (duas) etapas principais: Preparação do ambiente de teste e execução dos testes. Ao final, prosseguiu-se com a análise estatística dos resultados obtidos pelos testes de performance executados.

### 3.2.1 Preparação do ambiente de teste

Quando tratamos do BD relacional PostgreSQL, não existe nenhum pré-requisito de instalação, no entanto, o NoSQL Cassandra demanda a Instalação

do JAVA na versão 6, ou superior. Quanto ao YCDB, observa-se apenas a necessidade de instalação de uma versão JAVA qualquer.

No ambiente físico não foi instalado nenhum programa, a não ser o próprio banco de dados a ser executado, evitando assim excesso de processos consumindo memória e recursos que poderiam interferir nos resultados dos testes.

### 3.2.2 Teste de performance YCSB

Conforme descrito no Capítulo 2, as cargas de trabalho (*Workloads*) utilizadas para realização dos testes no YCSB são caracterizadas por variações de um mesmo tipo de aplicações básicas envolvendo inserção, atualização, leitura e seleção dos dados no BD. Considerando que os procedimentos padrões na manipulação de séries temporais oriundas de torres meteorológicas não incluem a atualização dos registros já inseridos, uma vez que os registros correspondem a descrição dos fenômenos naturais e alterações nos mesmos comprometem a validade dos dados, a operação “*Update*” não será incluída neste estudo.

As definições das cargas de trabalho no *YCSB Core Package* com atribuição das operações a serem executadas e suas respectivas proporções, além das distribuições das escolhas por determinados registros para execução das operações estão descritas na Tabela 2.

Tabela 2: Cargas de trabalho no YCSB desenvolvidas para o estudo

| Carga de Trabalho (Worload) | Operações    | Seleção de registros |
|-----------------------------|--------------|----------------------|
| A - Somente inserção        | Insert: 100% | Zipfian              |
| B - Somente seleção         | Scan: 100%   | Zipfian              |
| C - Somente leitura         | Read: 100%   | Zipfian              |

Algumas propriedades, não descritas na tabela, foram incorporadas no arquivo de propriedade usados com os geradores de carga de trabalho do *YCSB Core Package*. Os valores propostos seguiram algumas recomendações de Cooper (2015) juntamente com adaptações demandadas para avaliação de dados oriundos de séries temporais:

- *FieldCount*: utilizado 20, conforme proposto por (COOPER, 2015) para processamento de testes de longa duração;
- *Fieldlength*: 30 bytes - considerando as demandas provenientes da quantidade de caracteres geradas em uma linha de dados climáticos;
- *Readallfields*: mantido padrão: true, uma vez que o teste busca simular o stress gerado pela utilização do BD em sua totalidade;

- *Operationcount*: foram utilizadas quantidades diferentes de operações, variando de 20.000 à 15.000.000, de forma a possibilitar o entendimento do comportamento dos bancos sob diferentes demandas de operações, conforme proposto por Cooper et al. (2010) e Souza e Santos (2015);
- *Table*: utilizado "data";
- *Recordcount*: utilizado 15.000.000, conforme proposto por (COOPER, 2015) para processamento de testes de longa duração.

### 3.2.2.1 Adaptações no código Java do YCSB

Foram necessárias adaptações no código do YCSB, escrito em Java, para os testes com operações de *Insert* e *Scan*, tanto para o BD Cassandra e JDBC para o PostgreSQL. Primeiramente será apresentada a adaptação realizada no código do YCSB, escrito para os testes com operações de *Insert*, utilizado para o BD Cassandra:

---

**Algorithm 1** CassandraCQLClient.java - Método Insert

---

```

1: String[] opcoes = {"Temperatura", "Umidade", "Radiao"};
2: String selecionada = opcoes[new Random().nextInt(opcoes.length)];
3: String[] unid = {"C", "F", "K"};
4: String unidade = unid[new Random().nextInt(unid.length)];
5: Random r = new Random();
6: c.set(Calendar.DATE, c.get(Calendar.DATE) + 1);
7: insertStmt.value(YCSB_KEY, key);
8: insertStmt.value("data", new Timestamp(c.getTimeInMillis()));
9: insertStmt.value("nomedata", selecionada);
10: insertStmt.value("torre", new Random().nextInt(8));
11: insertStmt.value("unidade", unidade);
12: insertStmt.value("valordata", new Random().nextInt(40));

```

---

Pode-se observar que, no código 1, as linhas foram modificadas e/ou adicionadas para se adaptar ao padrão demandado por uma modelagem de dados climáticos, sendo possível inserir datas, nomes de torres meteorológicas, valores das torres e unidade de medida.

A adaptação realizada no código do YCSB, escrito para os testes com operações de *Insert*, utilizado para o BD PostgreSQL procedeu-se da seguinte forma:

Pode-se observar que, no código 2, as linhas foram modificadas e/ou adicionadas para também se adaptar ao padrão demandado por uma modelagem de dados climáticos.

---

**Algorithm 2** JdbcDBClient.java - Método Insert
 

---

```

1: String[] opcoes = {"Temperatura", "Umidade", "Radio"};
2: String selecionada = opcoes[newRandom().nextInt(opcoes.length)];
3: String[] unid = {"C", "F", "K"};
4: String unidade = unid[newRandom().nextInt(unid.length)];
5: int numFields = values.size();
6: Random r = new Random();
7: c.set(Calendar.DATE, c.get(Calendar.DATE) + 1);
8: StatementType type = new StatementType(StatementType.Type.INSERT, tableName,
9: numFields, getShardIndexByKey(key));
10: PreparedStatement insertStatement = cachedStatements.get(type);
11: if(insertStatement == null){
12: insertStatement = createAndCacheInsertStatement(type, key); }
13: insertStatement.setTimestamp(1, new Timestamp(c.getTimeInMillis()));
14: insertStatement.setInt(2, new Random().nextInt(8));
15: insertStatement.setString(3, selecionada);
16: insertStatement.setFloat(4, new Random().nextInt(40));
17: insertStatement.setString(5, unidade);

```

---

Finalizadas as adaptações no código escrito para os testes com operações de *Insert* nos dois BDs deu-se início as adaptações no código escrito para os testes com operações de *Scan*. As modificações realizadas para o BD Cassandra podem ser visualizadas a seguir:

---

**Algorithm 3** CassandraCQLClient.java - Método Scan
 

---

```

1: Timestampparam1 = newTimestamp(1800, 12, 31, 00, 00, 01, 01);
2: Timestampparam2 = newTimestamp(2500, 12, 31, 23, 59, 59, 59);
3: scanStmt.append(initialStmt.substring(0, initialStmt.length() - 1));
4: scanStmt.append("WHERE");
5: scanStmt.append(QueryBuilder.token(YCSB_KEY));
6: scanStmt.append("» = ");
7: scanStmt.append("token('");
8: scanStmt.append(newSimpleDateFormat("yyyy-MM-ddhh:mm:ss")
9: .format(param1.getTime()));
10: scanStmt.append("'");
11: scanStmt.append("and");
12: scanStmt.append(QueryBuilder.token(YCSB_KEY));
13: scanStmt.append("« = ");
14: scanStmt.append("token('");
15: scanStmt.append(newSimpleDateFormat("yyyy-MM-ddhh:mm:ss")
16: .format(param2.getTime()));
17: scanStmt.append("'");
18: scanStmt.append("LIMIT");
19: scanStmt.append(recordcount);

```

---

Pode observar, da mesma forma como para as operações de *Insert*, que no código 3 as linhas foram modificadas e/ou adicionadas para se obter uma seleção de dados com um intervalo de data, sendo as mesmas passadas por parâmetros, como será demonstrado no comando 3.

Da mesma forma, observa-se a seguir o código escrito para os testes com operações de *Scan* para JDBC para ser utilizado na conexão no BD PostgreSQL:

---

**Algorithm 4** JdbcDBClient.java - Método Scan
 

---

```

1: scanStatement.setTimestamp(1, newTimestamp(newSimpleDateFormat
2: ("yyyy-MM-ddhh:mm:ss").parse(props.getProperty(PARAM1)).getTime()));
3: scanStatement.setTimestamp(2, newTimestamp(newSimpleDateFormat
4: ("yyyy-MM-ddhh:mm:ss").parse(props.getProperty(PARAM2)).getTime()));
5: System.err.println("DataNoInseridaouInvalida");
6: Timestampparam1 = newTimestamp(1800, 12, 31, 00, 00, 01, 01);
7: Timestampparam2 = newTimestamp(2500, 12, 31, 23, 59, 59, 59);
8: scanStatement.setTimestamp(1, param1);
9: scanStatement.setTimestamp(2, param2);
10: scanStatement.setInt(3, recordcount);

```

---

As linhas foram modificadas e/ou adicionadas para o método *Scan*. As

palavras *PARAM1* e *PARAM2* foram incorporadas no código nos locais onde havia a necessidade de inclusão de intervalos de datas.

### 3.2.2.2 Configurações da linha de comando

De forma a possibilitar o entendimento das configurações utilizadas nos comandos dos testes dentro do YCSB são apresentadas a seguir as linhas de comando configuradas para as operações requeridas por cada um dos Workloads em estudo.

O comando 1 a seguir ilustra a linha de comando utilizada para as operações de *insert*:

---

**Demonstração: 1** `./bin/ycsblogdcassandra2 - cql - phosts = 127.0.0.1 - pport = 9042 - ptable = data - precordcount = 15000000 - s - Pworkloads/workloadc > load_cassandra.txt`

---

Esse comando 1 se trata da carga de inserção no banco, será povoado o banco com as parametrizações necessárias, conforme descrito no capítulo 2, sendo a principal delas o *recordcount* para 15.000.000 de dados, de forma a possibilitar um teste que simule demandas futuras de manipulação de grandes volumes de dados meteorológicos.

O comando 2 a seguir ilustra a linha de comando utilizada para as operações de *Scan*:

---

**Demonstração: 2** `./bin/ycsbruncassandra2 - cql - phosts = 127.0.0.1 - pport = 9042 - ptable = data - pfielcount = 20 - poperationcount = 20000 - precordcount = 15000000 - pmeasurementtype = timeseries - ptimeseries.granularity = 2000 - param1'2016 - 02 - 1117 : 11 : 48' - param2'2020 - 02 - 1317 : 12 : 32' - s - Pworkloads/workloadb > run_b_20000_cassandra.txt`

---

Este comando 2 determina a execução exclusiva de operações de seleção dos dados específicos, caracterizados por um intervalo de tempo pré definido. Observa-se o comando *param1* e *param2* que determina o intervalo das datas desejadas para o teste.

O comando 3 a seguir ilustra a linha de comando utilizada para as operações de *Read*. Este comando determina a execução exclusiva de operações de leitura dos dados no BD a partir de um padrão de distribuição de escolha pré definido pelo YCSB (para o *Workload C*, parte da leitura será feita por escolha uniforme e parte por popularidade (*Zipfian*)).

---

**Demonstração: 3** */bin/ycsbruncassandra2-cql-phosts = 127.0.0.1-pport = 9042-pfielcount = 20-poperationcount = 400000-precordcount = 15000000-pmeasurementtype = timeseries - ptimeseries.granularity = 2000 - s - Pworkloads/workloadc > run\_c\_400000\_cassandra.txt*

---

### 3.2.3 Análise estatística

Para comparar estatisticamente os resultados encontrados nos diferentes cenários do estudo será utilizada a estatística de Kruskal-Wallis, que comprova ou não a existência de diferenças significativas nos valores no desempenho obtidos nos diferentes testes e modelos de banco de dados, sem necessidade de pressupostos.

Considerou-se um nível de significância ( $\alpha$ ) igual a 0,05 para verificar a igualdade das amostras. Pela análise de Kruskal-Wallis, nos testes que apresentem *p-value* inferiores ao nível de significância ( $\alpha$ ) estabelecido, deve-se rejeitar a hipótese de que não existe diferença significativa entre os cenários, ou seja confirma-se a existência de variações significativas entre as amostras estudadas (LEE E. T.; WANG, 2003).

## Capítulo 4

# RESULTADOS

### 4.1 *WORKLOAD A* – SOMENTE *INSERT*

Os dois bancos de dados em estudo foram testados com o *Workload A*, que consiste em 100% de operações *Insert* (conforme configuração descrita na tabela 2), para uma quantidade de 20.000, 40.000, 60.000, 80.000, 100.000, 120.000, 140.000, 200.000 e 400.000 operações e os resultados podem ser vistos nas figuras 6 e 7.

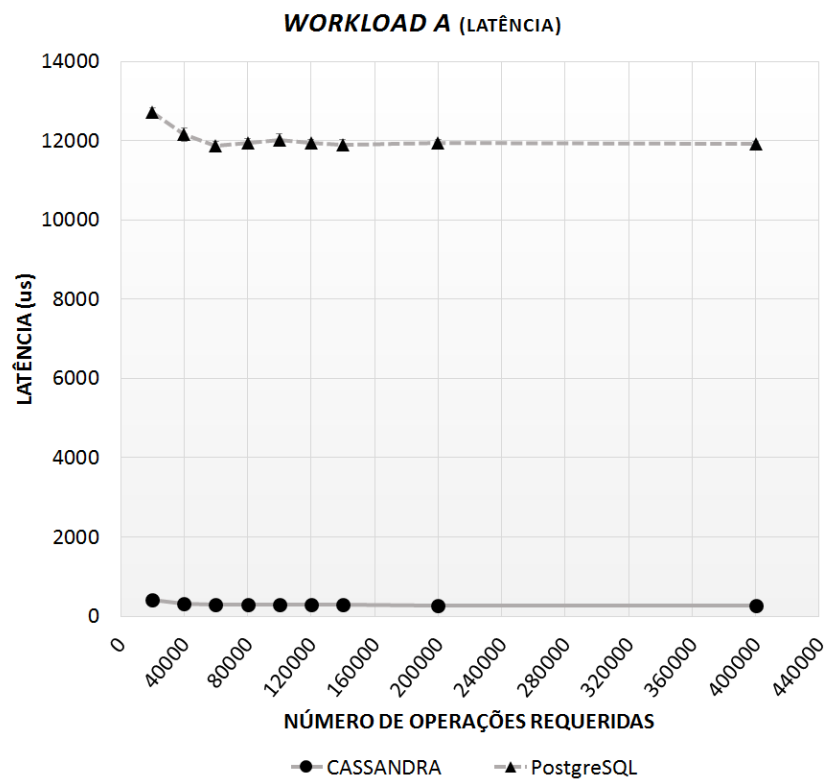


Figura 6: Latência do *Workload A* - Somente *Insert*

A figura 6 apresenta a latência em relação ao número de operações. Observa-se que, para operações de inserção de dados novos, de uma forma geral, o BD PostgreSQL apresenta maior latência em relação do BD Cassandra, ou seja, o tempo de resposta da operação é maior, para todos os números de operações definidos no teste. Estas diferenças são mais expressivas quanto menor a quantidade de operações requeridas pelo teste, sendo observadas diferenças de até 12.311  $\mu$  quando 20.000 operações de inserção são solicitadas, no entanto a partir de 80.000 operações não se observa diferenças acima de 11.740  $\mu$  entre as latências dos dois bancos.

O teste de Kruskal Wallis na tabela 3 revela que existem diferenças significativas entre os valores de Latência obtidos para 20.000 e 100.000 operações (com *p value* variando entre 0,0001096 e 0,01304) no BD Cassandra, no entanto entre os valores de latência obtidos para volumes acima de 100.000 operações o teste já revela a inexistência de diferenças significativas entre as latências obtidas (com *p value* variando de 0,0567 a 0,106) e a igualdade dos dados apenas aumenta a medida que se eleva a quantidade de operações. Quando tratamos do BD PostgreSQL o teste de Kruskal Wallis somente revela diferença entre os dados de latência obtidos entre 20.000 e 40.000 operações (com *p value* igual a 0,0003461), a partir do qual não são observadas diferenças significativas de latência com a elevação do número de operações (com *p value* variando entre 0,1219 e 0,6272).

Tabela 3: Teste de Kruskal-Wallis - Workload A (20.000 a 140.000 operações)

| Banco de Dados | Latência       |          |          |          |         |         |         |
|----------------|----------------|----------|----------|----------|---------|---------|---------|
|                | 20.000         | 40.000   | 60.000   | 80.000   | 100.000 | 120.000 | 140.000 |
| Cassandra      | 0,0001096      | 0,004669 | 0,02092  | 0,01304  | 0,0355  | 0,0567  | 0,106   |
| PostgresQL     | 0,0003461      | 0,1219   | 0,233    | 0,6272   | 0,5078  | 0,4529  | 0,5078  |
| Banco de Dados | Fluxo de Dados |          |          |          |         |         |         |
|                | 20.000         | 40.000   | 60.000   | 80.000   | 100.000 | 120.000 | 140.000 |
| Cassandra      | 0,00004113     | 0,003228 | 0,004669 | 0,005584 | 0,0204  | 0,03767 | 0,04331 |
| PostgresQL     | 0,0003461      | 0,1018   | 0,2692   | 0,6911   | 0,5078  | 0,4529  | 0,5078  |

De forma a verificar se esta igualdade se mantém com maior volume de operações requeridas, realizou-se um teste com 15.000.000 de operações de inserção. Observa-se que a latência dos 2 bancos se comporta como uma reta, reduzindo em média 7 $\mu$ , no BD Cassandra, e 34 $\mu$ , no BD PostgreSQL, a partir da elevação de 14.600.000 no volume de operações, confirmando que a elevação da quantidade de inserções não provoca influência expressiva na latência de nenhum dos bancos quando se trabalha com elevado número de operações de inserção.

O teste de Kruskal Wallis na tabela 4 confirma a inexistência de diferença

significativa entre os resultados obtidos entre os testes com 400.000 e 15.000.000 operações de inserção, com  $p$  value igual a 0,9648 para o BD PostgreSQL e 0,7728 para o BD Cassandra, comprovando que, para elevados números de operações, as latências permanecem constantes, independente do incremento no número de dados inseridos.

Tabela 4: Teste de Kruskal-Wallis - Workload A (400.000 a 15.000.000 operações)

| Teste          | Workload A |            |
|----------------|------------|------------|
|                | Cassandra  | PostgresQL |
| Latência       | 0,7728     | 0,9648     |
| Fluxo de Dados | 0,729      | 0,8946     |

Quando observamos o fluxo dos dados durante as operações solicitadas, apresentado na figura 7, observa-se, para todos volumes de operações testados, maior velocidade na inserção dos dados no BD Cassandra quando comparado ao BD PostgreSQL. Estas diferenças são menores quanto menor a quantidade de operações requeridas pelo teste, sendo observadas diferenças de até 2.080 ops/sec quando 20.000 operações de inserção são solicitadas, no entanto a partir de 120.000 operações não se observa diferenças abaixo de 3.313 ops/sec entre o fluxo de dados dos dois bancos.

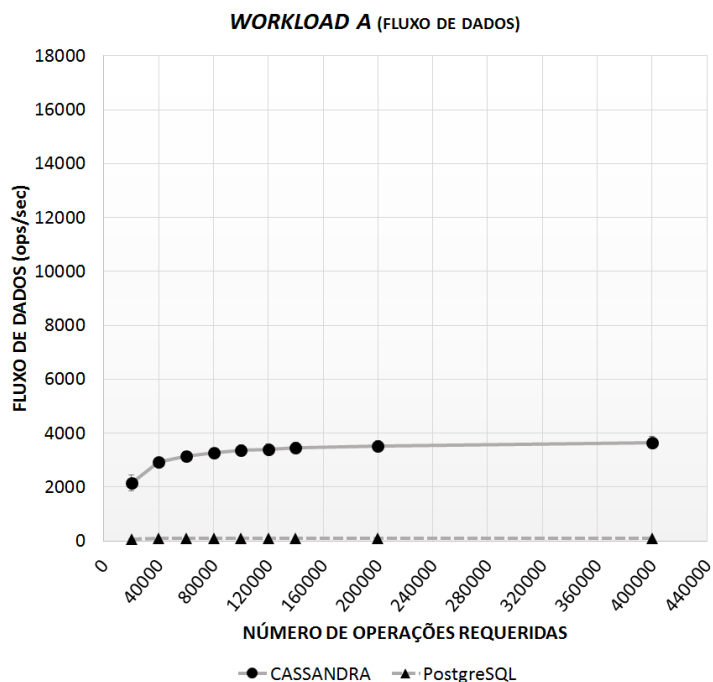


Figura 7: Fluxo de Dados *Workload A* - Somente *Insert*

O teste de Kruskal Wallis na tabela 3 revela que existem diferenças sig-

nificativas entre os valores de Fluxo de dados obtidos no BD Cassandra (com  $p$  value variando de 0,0000413 a 0,04331) em todas as operações requeridas no teste. No BD PostgreSQL, no entanto, esta diferenciação somente é observada entre os valores de Fluxo de dados obtidos entre 20.000 e 40.000 operações (com  $p$  value igual a 0,0003461), a partir do qual não observa-se diferenças significativas entre os valores de Fluxo de dados com o incremento de operações no teste (com  $p$  value variando de 0,1018 à 0,6911).

Quando analisados os resultados com 15.000.000 de operações de inserção, o fluxo de dados se comporta da mesma maneira que a latência, com variações pouco expressivas, confirmado pelo teste de Kruskal Wallis (tabela 4) que apresentou  $p$  value igual a 0,8946 para o BD PostgreSQL e 0,729 para o BD Cassandra, comprovando que a elevação da quantidade de inserções não provoca influência na velocidade de inserção dos dados, em nenhum dos bancos, quando se trabalha com elevado número de operações de inserção.

Tais resultados podem ser justificados pela estrutura utilizada por cada um dos bancos para a operação de inserção. Conforme GROUP (2015) para se povoar uma base de dados no BD PostgreSQL de forma a manter a máxima integridade devem ser realizado processos de verificação de inserção a cada linha de dado inserida, comprometendo assim a eficiência do banco.

## 4.2 **WORKLOAD B – SOMENTE SCAN**

Os dois bancos de dados em estudo foram testados com o *Workload B*, que consiste em 100% de operações *Scan* (conforme configuração descrita na tabela 2), para uma quantidade de 20.000, 40.000, 60.000, 80.000, 100.000, 120.000, 140.000, 200.000 e 400.000 operações e os resultados podem ser vistos nas figuras 8 e 9. Foi acrescentado a esse teste uma passagem de parâmetro para o intervalo de data 4 anos dos dados inseridos.

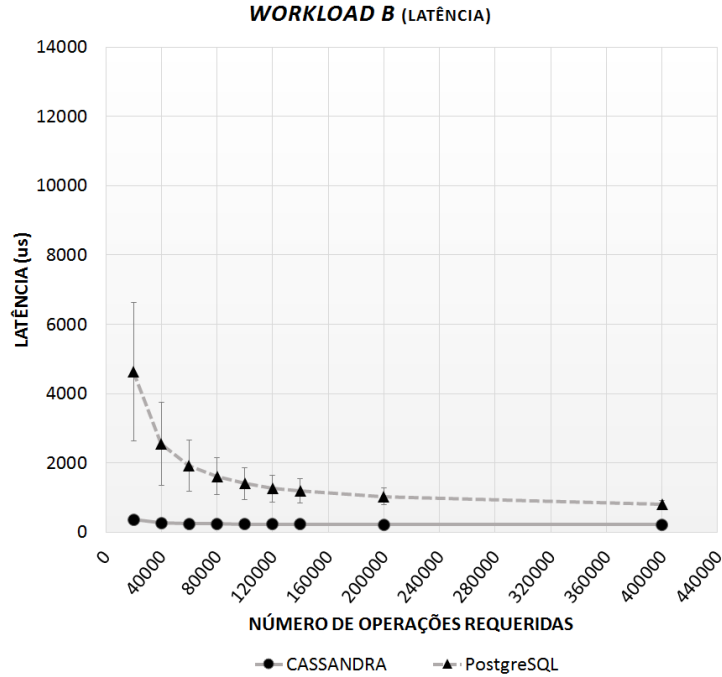


Figura 8: Latência do *Workload B* - Somente *Scan*

A figura 8 apresenta a latência em relação ao número de operações. Observa-se que, para operações de seleção de dados, de uma forma geral, o BD PostgreSQL apresenta maior latência em relação do BD Cassandra, ou seja, o tempo de resposta da operação é maior, para todos os números de operações definidos no teste. Estas diferenças são mais expressivas quanto menor a quantidade de operações requeridas pelo teste, sendo observadas diferenças de até  $4.279 \mu$  quando 20.000 operações de seleção são solicitadas, no entanto a partir de 80.000 operações não se observa diferenças acima de  $1.377 \mu$  entre as latências dos dois bancos.

O teste de Kruskal Wallis na tabela 5 revela que existem diferenças significativas entre os valores de Latência obtidos no BD Cassandra para todas as operações realizadas no teste (com *p value* variando entre 0,00003226 e 0,04965), revelando a existência de interferência na latência pelo incremento no volume de dados. Quando tratamos do BD PostgreSQL o teste de Kruskal Wallis somente revela diferença entre os dados de latência obtidos entre 20.000 e 60.000 operações (com *p value* variando entre 0,01654 e 0,02782), a partir do qual não são observadas diferenças significativas de latência com a elevação do número de operações (com *p value* variando entre 0,05273 e 0,07095).

De forma a verificar de se este comportamento se mantém com maior volume de operações requeridas, realizou-se um teste com 15.000.000 de operações de seleção. Observa-se que a latência dos 2 bancos se comporta como uma reta,

Tabela 5: Teste de Kruskal-Wallis - Workload B (20.000 a 140.000 operações)

| Banco de Dados | Latência       |          |          |         |         |         |         |
|----------------|----------------|----------|----------|---------|---------|---------|---------|
|                | 20.000         | 40.000   | 60.000   | 80.000  | 100.000 | 120.000 | 140.000 |
| Cassandra      | 0,00003226     | 0,004669 | 0,01107  | 0,02824 | 0,04965 | 0,04965 | 0,01793 |
| PostgresQL     | 0,01654        | 0,02782  | 0,06128  | 0,05273 | 0,0658  | 0,07095 | 0,01654 |
| Cassandra      | Fluxo de Dados |          |          |         |         |         |         |
|                | 20.000         | 40.000   | 60.000   | 80.000  | 100.000 | 120.000 | 140.000 |
| PostgresQL     | 0,00003226     | 0,001823 | 0,006657 | 0,02434 | 0,03263 | 0,02821 | 0,01304 |
|                | 0,01654        | 0,02782  | 0,06128  | 0,05273 | 0,0658  | 0,07095 | 0,01654 |

reduzindo em média  $4,2\mu$ , no BD Cassandra, e  $166\mu$ , no BD PostgreSQL, a partir da elevação de 14.600.000 no volume de operações, confirmando que a elevação da quantidade de inserções não provoca influência expressiva na latência de nenhum dos bancos, quando se trabalha com elevado número de operações de inserção.

O teste de Kruskal Wallis (tabela 6) confirma a inexistência de diferença significativa entre os resultados obtidos entre os testes com 400.000 e 15.000.000 operações de inserção, com *p value* igual a 0,6224 para o BD PostgreSQL e 0,6861 para o BD Cassandra, comprovando que, para elevados números de operações, as latências permanecem constantes, independente do incremento no número de dados inseridos.

Tabela 6: Teste de Kruskal-Wallis - Workload B (400.000 a 15.000.000 operações)

| Teste          | Workload B |            |
|----------------|------------|------------|
|                | Cassandra  | PostgresQL |
| Latência       | 0,6861     | 0,6224     |
| Fluxo de Dados | 0,729      | 0,718      |

Quando observamos o fluxo dos dados durante as operações solicitadas, apresentado na figura 9, observa-se, para todos volumes de operações testados, maior velocidade na inserção dos dados no BD Cassandra quando comparado ao BD PostgreSQL. Estas diferenças são menores quanto menor a quantidade de operações requeridas pelo teste, sendo observadas diferenças de até 2.140 ops/sec quando 20.000 operações de seleção são solicitadas, no entanto a partir de 120.000 operações não se observa diferenças abaixo de 3.320 ops/sec entre o fluxo de dados dos dois bancos.

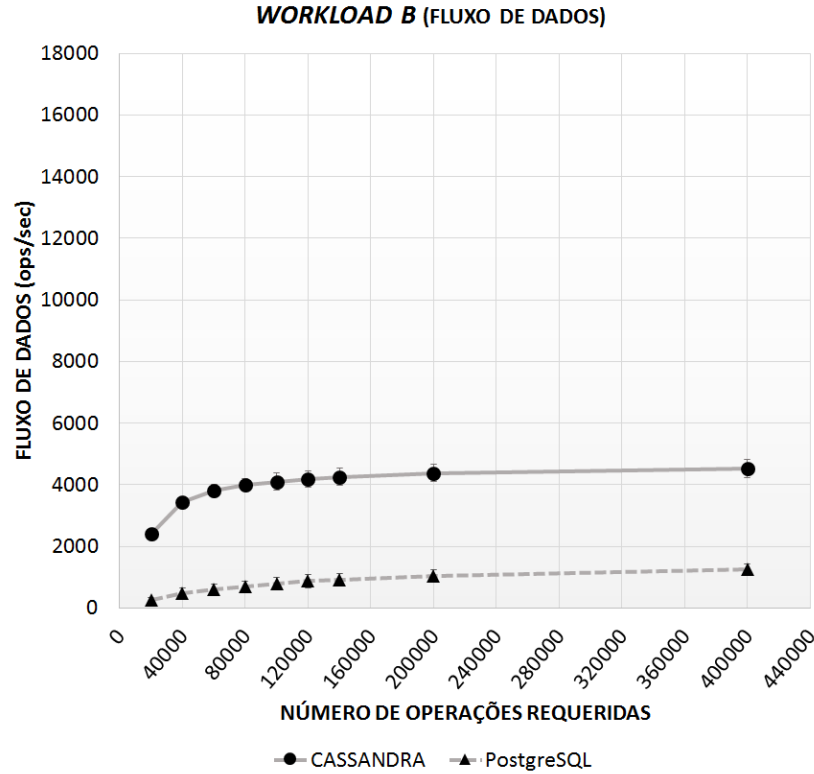


Figura 9: Fluxo de dados do *Workload B* - Somente *Scan*

O teste de Kruskal Wallis na tabela 5 revela que existem diferenças significativas entre os valores de Fluxo de dados obtidos no BD Cassandra (com *p value* variando de 0,00003226 a 0,03263) em todas as operações requeridas no teste. No BD PostgreSQL, no entanto, esta diferenciação somente é observada entre os valores de Fluxo de dados obtidos entre 20.000 e 60.000 operações (com *p value* variando de 0,01654 a 0,02782), a partir do qual não observa-se diferenças significativas entre os valores de Fluxo de dados com o incremento de operações no teste (com *p value* variando de 0,06128 à 0,07095).

Quando analisados os resultados com 15.000.000 de operações de seleção, o fluxo de dados se comporta da mesma maneira que a latência, com variações pouco expressivas, confirmado pelo teste de Kruskal Wallis (tabela 6) que apresentou *p value* igual a 0,718 para o BD PostgreSQL e 0,729 para o BD Cassandra, comprovando que a elevação da quantidade de seleções não provoca influência na velocidade de seleção dos dados, em nenhum dos bancos, quando se trabalha com elevado número de operações de seleção.

Estudos realizados por Souza e Santos (2015), que testaram a performance de 3 diferentes BDs (Cassandra, MySQL e Redis) no Benchmark YCSB, também avaliaram o desempenho dos BDs quanto as operações de *scan* (seleção). Para

os testes, no entanto, foi utilizado somente o BD Redis e o BD Cassandra, pois o BD MySQL não suporta escaneamento de ranges, procedimento necessário para a operação de Scan. Os autores no entanto trabalharam com volumes de dados inferiores aos utilizados neste trabalho e com a implementação padrão do YCSB, desenvolvida por Cooper (2015), impossibilitando uma comparação direta entre os resultados.

No entanto, considera-se a possibilidade de uma comparação qualitativa entre os resultados encontrados para o *Workload B* e as observações apontadas por Souza e Santos (2015). Os valores de latência apresentados por Souza e Santos (2015) para o BD Cassandra foram inferiores aos observados para o *Workload B*, quando são requeridas 20.000 operações ( $140\mu$  e  $340\mu$  respectivamente), no entanto os dois trabalhos revelam uma redução nos valores de latência a medida que se eleva o volume de operações de seleção requeridas. Quanto ao fluxo de dados, Souza e Santos (2015) encontrou valores aproximadamente 10x inferiores aos encontrados nos testes do *Workload B* para o BD Cassandra, resultado este que pode ter relação com a capacidade de processamento impostas pelo *Hardware*, considerando que os computadores utilizados por Souza e Santos (2015) possuíam configurações inferiores as utilizadas nos testes apresentados neste estudo. No entanto os dois trabalhos revelam uma elevação nos valores de fluxo de dados a medida que se eleva o volume de operações de seleção requeridas.

Tais resultados revelam que apesar de a utilização de maior volume de dados e variações na implementação padrão o YCSB melhorarem o desempenho do BD Cassandra, principalmente no que tange a quantidade de operações de seleção realizadas por segundo, o comportamento da latência e do fluxo de dados em relação ao volume de operações requeridas se manteve semelhante nos dois estudos.

### 4.3 *WORKLOAD C – SOMENTE READ*

Os dois bancos de dados em estudo foram testados com o *Workload C*, que consiste em 100% de operações *Read* (conforme configuração descrita na tabela 2, para quantidade de 20.000, 40.000, 60.000, 80.000, 100.000, 120.000, 140.000 e 200.000 e 400.000 operações e os resultados podem ser vistos nas figuras 10 e 11. Neste passo, lembramos que não foi alterado o código do YCSB.

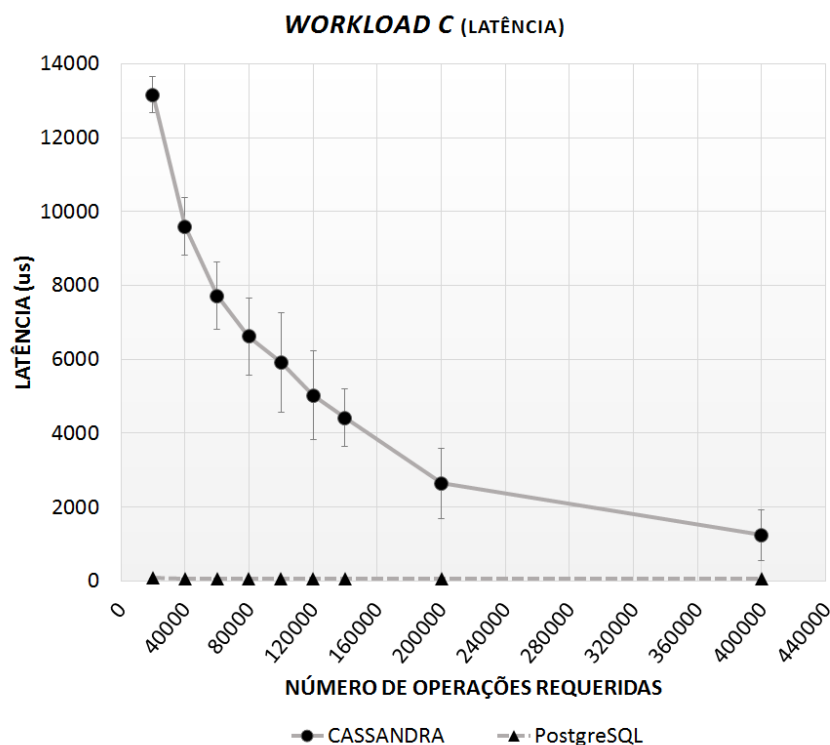


Figura 10: Latência do Workload C - Somente *Read*

A figura 10 apresenta a latência em relação ao número de operações. Observa-se que, para operações de leitura de dados, de uma forma geral, o BD PostgreSQL apresenta menor latência em relação do BD Cassandra, ou seja, o tempo de resposta da operação é menor, para todos os números de operações definidos no teste. Estas diferenças são mais expressivas quanto menor a quantidade de operações requeridas pelo teste, sendo observadas diferenças de até  $13.087 \mu$  quando 20.000 operações de seleção são solicitadas, no entanto a partir de 80.000 operações não se observa diferenças acima de  $6.570 \mu$  entre as latências dos dois bancos.

O teste de Kruskal Wallis revela na tabela 7 que existem diferenças significativas entre os valores de Latência obtidos para 20.000 e 80.000 operações (com *p value* variando entre 0,0001223 e 0,007827) no BD Cassandra, no entanto entre os valores de latência obtidos para volumes acima de 80.000 operações o teste já revela a inexistência de diferenças significativas entre as latências obtidas (com *p value* variando de 0,1228 a 0,4118) e a igualdade dos dados apenas aumenta a medida que se eleva a quantidade de operações. Quando tratamos do BD PostgreSQL, da mesma forma como no BD Cassandra, o teste de Kruskal Wallis somente revela diferença entre os dados de latência obtidos para 20.000 e 80.000 operações (com *p value* variando entre 0,00007105 e 0,007827), a partir do

qual não são observadas diferenças significativas de latência com a elevação do número de operações (com  $p$  value variando entre 0,1228 e 0,341).

Tabela 7: Teste de Kruskal-Wallis - Workload C (20.000 a 140.000 de operações)

| Banco de Dados | Latência       |          |          |        |         |         |         |
|----------------|----------------|----------|----------|--------|---------|---------|---------|
|                | 20.000         | 40.000   | 60.000   | 80.000 | 100.000 | 120.000 | 140.000 |
| Cassandra      | 0,0001223      | 0,001814 | 0,007827 | 0,1228 | 0,08184 | 0,4118  | 0,1449  |
| PostgresQL     | 0,00007105     | 0,001814 | 0,007827 | 0,1228 | 0,08184 | 0,341   | 0,1152  |
| Banco de Dados | Fluxo de Dados |          |          |        |         |         |         |
|                | 20.000         | 40.000   | 60.000   | 80.000 | 100.000 | 120.000 | 140.000 |
| Cassandra      | 0,00007105     | 0,07095  | 0,4905   | 0,2505 | 0,7676  | 0,7676  | 0,6695  |
| PostgresQL     | 0,00007105     | 0,03283  | 0,09395  | 0,1227 | 0,8696  | 0,4905  | 0,4905  |

De forma a verificar de se este comportamento se mantém com maior volume de operações requeridas, realizou-se um teste com 15.000.000 de operações de seleção. Observa-se que a latência dos 2 bancos se comporta como uma reta, reduzindo em média  $468\mu$ , no BD Cassandra, e  $1,87\mu$ , no BD PostgreSQL, a partir da elevação de 14.600.000 no volume de operações, confirmando que o aumento da quantidade de leituras não provoca influência expressiva na latência de nenhum dos bancos, principalmente no BD PostgreSQL, quando se trabalha com elevado número de operações de leitura.

O teste de Kruskal Wallis (tabela 8) confirma a inexistência de diferença significativa entre os resultados obtidos entre os testes com 400.000 e 15.000.000 operações de leitura, com  $p$  value igual a 0,8182 para o BD PostgreSQL e 0,9738 para o BD Cassandra, comprovando que, para elevados números de operações, as latências permanecem constantes, independente do incremento no número de dados lidos.

Tabela 8: Teste de Kruskal-Wallis - Workload C (400.000 a 15.000.000 operações)

| Teste          | Workload C |            |
|----------------|------------|------------|
|                | Cassandra  | PostgresQL |
| Latência       | 0,9738     | 0,8182     |
| Fluxo de Dados | 0,9081     | 0,8182     |

Quando observamos o fluxo dos dados durante as operações solicitadas, apresentado na figura 11, observa-se, para todos volumes de operações testados, maior velocidade na leitura dos dados no BD PostgreSQL quando comparado ao BD Cassandra. Estas diferenças são menores quanto menor a quantidade de operações requeridas pelo teste, sendo observadas diferenças de até 10.267 ops/sec quando 20.000 operações de seleção são solicitadas, no entanto a partir

de 120.000 operações não se observa diferenças abaixo de 16.453 ops/sec entre o fluxo de dados dos dois bancos.

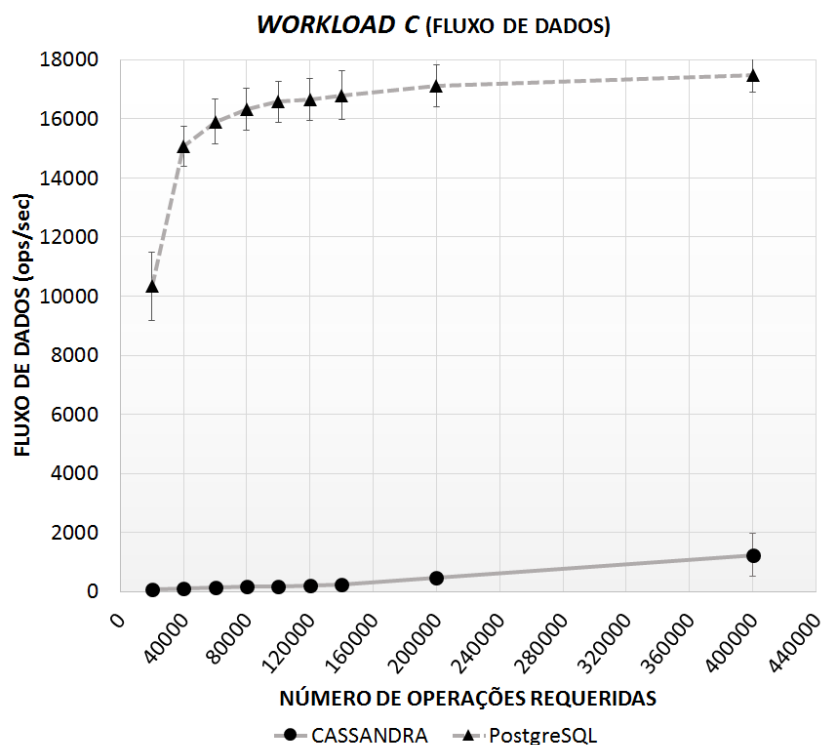


Figura 11: Fluxo de dados do Workload C - Somente *Read*

O teste de Kruskal Wallis na tabela 7 revela que existem diferenças significativas entre os valores de Fluxo de dados obtidos para 20.000 e 60.000 operações (com *p value* de 0,00007105) no BD Cassandra, no entanto entre os valores de Fluxo de dados obtidos para volumes acima de 60.000 operações o teste já revela a inexistência de diferenças significativas entre os fluxos obtidos (com *p value* variando de 0,07095 a 0,7676) e a igualdade dos dados apenas aumenta à medida que se eleva a quantidade de operações. No BD PostgreSQL observa-se comportamento semelhante, com diferenciação somente observada entre os valores de Fluxo de dados obtidos entre 20.000 e 80.000 operações (com *p value* variando de 0,00007105 a 0,03283), a partir do qual não observa-se diferenças significativas entre os valores de Fluxo de dados com o incremento de operações no teste (com *p value* variando de 0,09395 à 0,8696).

Quando analisados os resultados com 15.000.000 de operações de seleção, o fluxo de dados se comporta da mesma maneira que a latência, com variações pouco expressivas, confirmado pelo teste de Kruskal Wallis (tabela 8) que apresentou *p value* igual a 0,8182 para o BD PostgreSQL e 0,9081 para o BD Cassandra, comprovando que a elevação da quantidade de leituras não provoca influência na

velocidade de leitura dos dados, em nenhum dos bancos, quando se trabalha com elevado número de operações de leitura.

Os valores de latência apresentados por Souza e Santos (2015) para as operações de leitura no BD Cassandra foram muito inferiores aos observados para o *Workload C*, quando são requeridas 20.000 operações ( $20\mu$  e  $12.000\mu$  respectivamente). No entanto os dois trabalhos revelam uma redução nos valores de latência a medida que se eleva o volume de operações de seleção requeridas. Quanto ao fluxo de dados, Souza e Santos (2015) encontrou valores muito superiores aos encontrados nos testes do *Workload C* para o BD Cassandra (400ops/sec e 80ops/sec respectivamente), resultado este que pode ter relação com a quantidade de dados inserida no teste, uma vez que Souza e Santos (2015) trabalhou com 1.000 dados e este trabalho desenvolveu os testes em 15.000.000 de dados. No entanto os dois trabalhos revelam uma elevação nos valores de fluxo de dados a medida que se eleva o volume de operações de leitura requeridas.

Tais resultados revelam que apesar de a utilização de maior volume de dados e variações na implementação da tabela padrão do YCSB comprometerem o desempenho do BD Cassandra para operações de leitura, o comportamento da latência e do fluxo de dados em relação ao volume de operações requeridas se manteve semelhante nos dois estudos.

## 4.4 LATÊNCIA *WORKLOAD A*, *B* e *C*

Quando analisa-se a Latência dos dois BDs em estudo, quando submetidos a cada um dos testes, verifica-se de uma forma geral o melhor desempenho do BD Cassandra no *Workload A* e *Workload B*, sendo mais significativa a eficiência no processamento das operações no *Workload A*, conforme pode ser observado na figura 12 (a) e (b) (inserção e seleção dos dados). No entanto, quando tratamos da leitura dos dados, realizada no *Workload C*, o BD PostgreSQL se mostrou mais eficiente, principalmente no processamento de volumes de operações menores que 400.000 (figura 12 (c)).

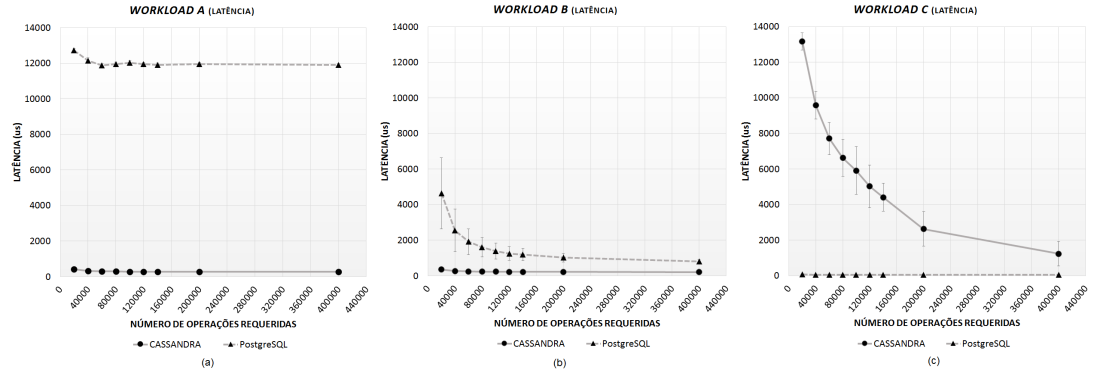


Figura 12: Latência *Workload A*, *B* e *C*

## 4.5 FLUXO DE DADOS *WORKLOAD A*, *B* e *C*

Quando analisa-se o Fluxo de dados dos dois BDs em estudo, quando submetidos a cada um dos testes, verifica-se de uma forma geral o melhor desempenho do BD Cassandra no *Workload A* e *Workload B*, conforme pode ser observado na figura 13 (a) e (b) (inserção e seleção dos dados). No entanto, quando tratamos da leitura dos dados, realizada no *Workload C*, assim como para Latência, o BD PostgreSQL se mostrou mais eficiente, principalmente no processamento de volumes de operações menores que 200.000 (figura 13 (c)).

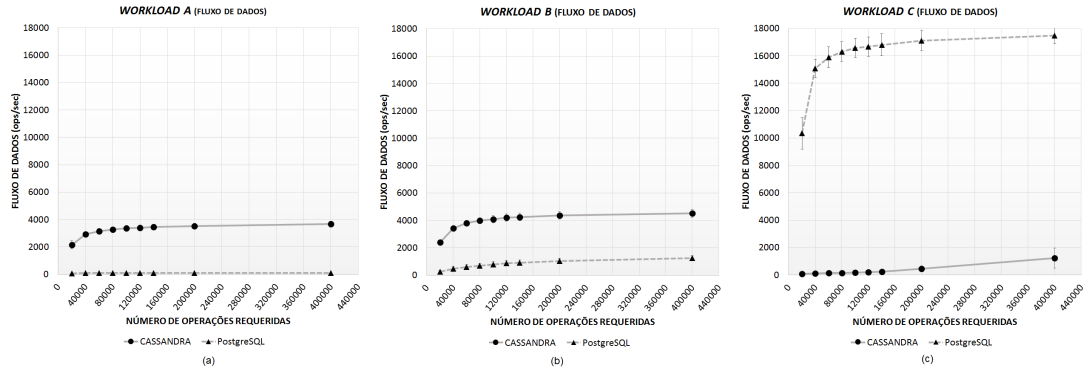


Figura 13: Fluxo de dados *Workload A*, *B* e *C* – Somente *Scan*

## Capítulo 5

# CONCLUSÃO

A partir da análise quantitativa dos resultados apresentados pode-se concluir que o BD Cassandra possui maior eficiência no processamento das operações de Inserção e Seleção dos dados com estrutura equivalente às Séries Temporais obtidos a partir de torres meteorológicas, quando comparado ao Banco de Dados relacional PostgreSQL, com menor latência e maior fluxo de dados para todos os volumes de operações testados. No entanto, quando tratamos das operações de leitura o BD PostgreSQL revelou-se mais eficiente, tanto quanto a latência quanto ao fluxo de dados.

Quanto as diferenças na latência e no fluxo de dados entre os dois BDs em estudo, para requisições de operações abaixo de 400.000, observa-se uma redução nas diferenças de latência e elevação nas diferenças no fluxo de dados a medida que se eleva a quantidade de operações requisitadas. No entanto, quando se trabalha com elevado número de operações (a partir de 400.000), variações no volume de operações requisitadas nos dois BDs estudados não influenciam nos tempos de resposta e processamento dos bancos.

Conclui-se que o Banco de dados NSQL Cassandra foi o que apresentou formas mais eficientes no *Insert* e *Scan* de dados com estrutura equivalente às Séries Temporais obtidas a partir de torres meteorológicas, sendo assim o mais adequado para implantação nas plataformas de gerenciamento de dados climáticos.

### 5.1 CONTRIBUIÇÕES

Estas foram as principais contribuições geradas no desenvolvimento deste trabalho:

- A disponibilização de mais uma ferramenta para análises padronizadas para

testes de eficiência.

- A alteração de uma ferramenta padrão de teste para análise de dados temporais.
- A construção de um resultado como este auxilia na escolha de uma ferramenta adequada para desenvolvimento de sistema no auxílio na manipulação de dados de series temporais obtidos a partir de estudo meteorológicos.

## 5.2 TRABALHOS FUTUROS

Apesar das diversas contribuições científicas e tecnológicas já obtidas até o presente momento, pela análise dos bancos de dados, entende-se que sua evolução ocorrerá com as seguintes abordagens:

Um aspecto importante a ser adicionado nesta pesquisa é a incorporação de mais modelos de banco de dados, reforçando e melhorando a escolha do mais adequado e eficiente para a manipulação de series temporais.

Se fazem necessários testes de desempenho com o YCSB em máquinas servidoras, a partir do qual, teremos aprofundamento no conhecimento das atribuições de memória, memória cache, uso de memória RAM e disponibilidade de disco.

Deverão ser realizados testes com a utilização de *Cluster* (nós), possibilitando estudo sobre escalabilidade, ampliando o conhecimento sobre suas replicações principalmente no desempenho e tempo de resposta do banco de dados para operação de *Read*.

# REFERÊNCIAS

ALVES, H. *Tecnologia da Informação*. Fortaleza, CE: [s.n.], 2014, (Apostila). Citado 2 vezes nas páginas 10 e 11.

ANDERSON, J. C.; LEHNARDT, J.; SLATER, N. *CouchDB: the definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2010. Citado na página 16.

ARRUDA, P. H. Z. *Dinâmica das trocas de massa e energia em região de cerrado na baixada cuiabana*. 2014. 70f. Tese (Doutorado) — Tese (Doutorado em Física Ambiental)-Universidade Federal de mato Grosso, Instituto de Física, Cuiabá, 2014.[Links], 2014. Citado na página 2.

BORGES, K.; DAVIS, C. Modelagem de dados geográficos. *CÂMARA, G.; MONTEIRO, AM; DAVIS, C. Geoprocessamento: teorias e aplicações*. 3v, v. 3, 2002. Citado na página 11.

BRESLOW, N. A generalized kruskal-wallis test for comparing k samples subject to unequal patterns of censorship. *Biometrika*, Biometrika Trust, v. 57, n. 3, p. 579–594, 1970. Citado na página 28.

BREWER, E. A. Towards robust distributed systems. In: *PODC*. [S.l.: s.n.], 2000. v. 7. Citado na página 17.

BRITO, R. W. Bancos de dados nosql x sgbd's relacionais: análise comparativa. *Faculdade Farias Brito e Universidade de Fortaleza*, 2010. Citado 3 vezes nas páginas 13, 15 e 17.

CARDOSO, M. R. A. Análise de séries temporais em epidemiologia: uma introdução sobre os aspectos metodológicos. *Rev. bras. epidemiol*, SciELO Public Health, v. 4, n. 3, 2001. Citado na página 6.

CATTELL, R. Scalable sql and nosql data stores. *ACM SIGMOD Record*, ACM, v. 39, n. 4, p. 12–27, 2011. Citado 2 vezes nas páginas 17 e 31.

CEPED-UFSC. *Chuvas: por que o Brasil não consegue evitar essa tragédia*. 2014. Citado na página 1.

CHAN, Y.; WALMSLEY, R. P. Learning and understanding the kruskal-wallis one-way analysis-of-variance-by-ranks test for differences among three or more independent groups. *Physical therapy*, American Physical Therapy Association, v. 77, n. 12, p. 1755–1761, 1997. Citado na página 27.

CHAUDHURI, S. What next?: a half-dozen data management research goals for big data and the cloud. In: ACM. *Proceedings of the 31st symposium on Principles of Database Systems*. [S.l.], 2012. p. 1–4. Citado na página 9.

CHEN, P. P.-S. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, ACM, v. 1, n. 1, p. 9–36, 1976. Citado na página 11.

CODD, E. F. A relational model of data for large shared data banks. *Communications of the ACM*, ACM, v. 13, n. 6, p. 377–387, 1970. Citado 2 vezes nas páginas 14 e 15.

COOPER, B. F. *Benchmarking Cassandra and other NoSQL databases with YCSB*. 2015. Citado 5 vezes nas páginas 25, 26, 32, 33 e 45.

COOPER, B. F.; RAMAKRISHNAN, R.; SRIVASTAVA, U.; SILBERSTEIN, A.; BOHANNON, P.; JACOBSEN, H.-A.; PUZ, N.; WEAVER, D.; YERNENI, R. Pnuts: Yahoo!’s hosted data serving platform. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 1, n. 2, p. 1277–1288, 2008. Citado na página 21.

COOPER, B. F.; SILBERSTEIN, A.; TAM, E.; RAMAKRISHNAN, R.; SEARS, R. Benchmarking cloud serving systems with ycsb. In: ACM. *Proceedings of the 1st ACM symposium on Cloud computing*. [S.l.], 2010. p. 143–154. Citado 7 vezes nas páginas 21, 22, 23, 24, 25, 26 e 33.

COUTINHO, E. R.; SILVA, R. M.; HUBER, F.; DELGADO, A. R. S.; CORREA, B. S. P. M.; VALE, I. G. Desenvolvimento de uma ferramenta de apoio a processamento e monitoramento de dados meteorológicos. *Revista Edu. Tec.*, v. 2, n. 1, 2015. Citado 2 vezes nas páginas 2 e 8.

CUNHA, A. d. *Parâmetros agrometeorológicos de cultura de pimentão (*Capsicum annuum* L.) em ambientes protegido e campo*. 2001. 128 f. Tese (Doutorado) — Tese (Doutorado em Energia na Agricultura)-Universidade Estadual Paulista, Faculdade de Ciências Agrônômicas, Botucatu, 2001.[Links], 2001. Citado 2 vezes nas páginas 2 e 9.

CUZZOCREA, A.; SONG, I.-Y.; DAVIS, K. C. Analytics over large-scale multidimensional data: the big data revolution! In: ACM. *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP*. [S.l.], 2011. p. 101–104. Citado 2 vezes nas páginas 9 e 15.

DANELICHEN, V. *Conservação no Pantanal Matogrossense por Sensoriamento Remoto*. 2015. 118 f. Tese (Doutorado) — Tese (Doutorado em Física Ambiental)-Universidade Federal de mato Grosso, Instituto de Física, Cuiabá, 2015.[Links], 2015. Citado na página 2.

DATE, C. *Introdução a sistemas de bancos de dados*. Campus, 2004. ISBN 9788535212730. Disponível em: <<https://books.google.com.br/books?id=xBeO9LSIK7UC>>. Citado 4 vezes nas páginas 8, 10, 12 e 15.

DENARO, G.; POLINI, A.; EMMERICH, W. Early performance testing of distributed software applications. In: ACM. *ACM SIGSOFT Software Engineering Notes*. [S.l.], 2004. v. 29, n. 1, p. 94–103. Citado na página 19.

DIAS, P. Mudanças climáticas; como conviver com as incertezas sobre os cenários futuros. In: *10<sup>o</sup> Encontro de Geógrafos da América Latina*. [S.l.]: Universidade de São Paulo, Faculdade de Filosofia, letras e Ciências humanas, 2005. p. 47–48. Citado na página 1.

D'ANDREA, E. Big data. *Revista Informationweek, Coluna Segurança*, Insight 25 anos, p. 38, 2010. Citado 2 vezes nas páginas 3 e 9.

EHLERS, R. S. Análise de séries temporais. *Laboratório de Estatística e Geoinformação. Universidade Federal do Paraná*, 2007. Citado na página 5.

ELMASRI, R.; NAVATHE, S. B. *Fundamentals of database systems*. [S.l.]: Pearson Addison Wesley, 2014. Citado na página 11.

ELMASRI, R.; NAVATHE, S. B.; PINHEIRO, M. G.; CANHETTE, C. C.; MELO, G. C. V.; AMADEU, C. V.; MORAIS, R. de O. Sistemas de banco de dados. Pearson Addison Wesley, 2005. Citado 5 vezes nas páginas 3, 8, 12, 14 e 15.

FERREIRA, S. H. S.; CARVALHO, L. S. M.; FILHO, E. Ó. Banco de dados meteorológicos para previsão de tempo e estudos climáticos. In: *XI Congresso Brasileiro de Meteorologia, Rio de Janeiro*. [S.l.: s.n.], 2000. Citado 2 vezes nas páginas 15 e 16.

FISHER, D.; DELINE, R.; CZERWINSKI, M.; DRUCKER, S. Interactions with big data analytics. *interactions*, ACM, v. 19, n. 3, p. 50–59, 2012. Citado na página 10.

GRAY, J. *Database and Transaction Processing Performance Handbook*. [S.l.]: Morgan Kaufman Publishers, 1993. Citado na página 20.

GROUP, P. G. D. *POSTGRESPL*. 2015. Citado 2 vezes nas páginas 30 e 41.

GUIMARAES, L. Um aplicativo para teste de performances de bancos de dados através dos benchmarks tpc-c e tpc-hl. Laboratórios de Pesquisa em Ciência da Computação Departamento de Computação – UFC, 2007. Citado na página 20.

GUJARATI D. N.; PORTER, D. C. *Econometria Básica-5*. [S.l.]: McGraw Hill Brasil, 2011. Citado na página 5.

HAN, J.; HAIHONG, E.; LE, G.; DU, J. Survey on nosql database. In: *Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on*. [S.l.: s.n.], 2011. p. 363–366. Citado na página 30.

HOUGHTON, J.; DING, Y.; GRIGGS, D.; NOGUER, M.; LINDEN, P. Van der; DAI, X.; MASKELL, K.; JOHNSON, C. Ipcc 2001: Climate change 2001. *The Climate change Contribution of Working Group I to the Third Assessment Report*

of the Intergovernmental Panel on Climate Change, United Kingdom and New York, NY, USA: Cambridge University Press, v. 159, 2001. Citado na página 1.

KORTH, H. F.; SILBERSCHATZ, A.; SUDARSHAN, S. *Sistema de banco de dados*: Tradução de marília guimarães pinheiro e cláudio César canhette. São Paulo: Makron Books, 1999. Citado na página 10.

LAI, E. Twitter growth prompts switch from mysql to ‘nosql’database. *ComputerWorld*, 2010. Citado na página 17.

LAKSHMAN A.; MALIK, P. Cassandra: structured storage system on a p2p network. In: ACM. *Proceedings of the 28th ACM symposium on Principles of distributed computing*. [S.l.], 2009. p. 5–5. Citado 4 vezes nas páginas 17, 18, 30 e 31.

LAKSHMAN A.; MALIK, P. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, ACM, v. 44, n. 2, p. 35–40, 2010. Citado na página 16.

LEE E. T.; WANG, J. W. *Statistical methods for survival data analysis*. [S.l.]: New Jersey: John Wiley & Sons, 2003. v. 3. ed. Citado na página 37.

LEITE, M. *Acessando Bancos de Dados com ferramentas RAD: Aplicações em Visual Basic*. [S.l.]: Brasport, 2007. Citado na página 30.

LEMOES P.H.S.; FIGUEIREDO, P. *Uma Análise dos Novos Sistemas de Bancos de Dados Relacionais Escaláveis*. 2014. Citado na página 21.

LÓSCIO, B. F.; OLIVEIRA, H. R. d.; PONTES, J. C. d. S. Nosql no desenvolvimento de aplicações web colaborativas. In: UNIVERSIDADE FEDERAL DE RIO DE JANEIRO. *VIII Simpósio Brasileiro de Sistemas Colaborativos*. Paraty, RJ, 2011. Citado 3 vezes nas páginas 12, 13 e 14.

MACIEL, C. R. *Condições Microclimáticas de Espaços Abertos: Simulação de Estratégias por meio do Software ENVI-Met. 2014. 96f.* Tese (Doutorado) — Doutorado em Física Ambiental, Instituto de Física, universidade Federal de Mato Grosso, 2014. Citado na página 2.

MADDALA, G. S.; LAHIRI, K. *Introduction to econometrics*. [S.l.]: Macmillan New York, 1992. v. 2. Citado 2 vezes nas páginas 3 e 5.

MORESI, E.; PRADO, H.; ALCANTARA, A. Cenários prospectivos, monitoração ambiental e metadados. *Revista de Ciência da Informação*, v. 11, n. 1, 2010. Citado 2 vezes nas páginas 6 e 7.

OBE, R.; HSU, L. *PostGIS in action*. [S.l.]: Manning Publications Co., 2011. Citado na página 30.

OLIVEIRA, A. *Arquitetura de Dados Para Gerenciamento de Informações Científicas*. 2011. 86p. Dissertação (Mestrado) — Mestrado em Física Ambiental, Universidade Federal de mato Grosso, Instituto de Física, Cuiabá, 2011.[Links], 2011. Citado na página 30.

OLIVEIRA, A. *Plataforma computacional para mineração de dados micrometeorológicos*. 2015. 84f. Tese (Doutorado) — Doutorado em Física Ambiental, Universidade Federal de Mato Grosso, Instituto de Física, Cuiabá, 2015.[Links], 2015. Citado 2 vezes nas páginas 2 e 6.

PEREIRA, O. A. *Estimativas do balanço de energia e fluxo de CO<sub>2</sub> por diferentes métodos em floresta de transição no sudoeste da Amazônia*. 2013. 107f. Tese (Doutorado) — Doutorado em Física Ambiental, Instituto de Física, Universidade Federal de Mato Grosso, 2013. Citado na página 2.

PIRES, C. E. S.; NASCIMENTO, R. O.; SALGADO, A. C. Comparativo de desempenho entre bancos de dados de código aberto. *Escola Regional de Banco de Dados, Anais da ERBD06, Porto Alegre*, 2006. Citado na página 20.

POTTIER, M. Operational databases at météo-france. In: *Fifth Workshop on Meteorological Operational Systems, ECMWF*. [S.l.: s.n.], 1995. Citado 2 vezes nas páginas 2 e 8.

PRESSMAN, R. S. *Engenharia de Software*. 7<sup>a</sup>. ed. New York, EUA: McGraw-Hill Companies, 2011. Citado na página 19.

PRITCHETT, D. Base: An acid alternative. *Queue*, ACM, v. 6, n. 3, p. 48–55, 2008. Citado na página 17.

RAOULT, B. Architecture of the new mars. In: ECMWF. *Sixth Workshop on Meteorological Operational Systems*. [S.l.], 1997. Citado 2 vezes nas páginas 2 e 8.

RODRIGUES, T. *Análise de parâmetros biofísicos que controlam o fluxo de calor latente em área de Cerrado Campo Sujo*. 2014. 94f. Tese (Doutorado) — Tese (Doutorado em Física Ambiental)-Universidade Federal de Mato Grosso, Instituto de Física, Cuiabá, 2014.[Links], 2014. Citado na página 2.

ROSSETI, K. *Efeitos do uso de telhados vegetados em ilhas de calor urbanas com simulação pelo software ENVI-Met*. 2013, 273f. Tese (Doutorado) — Doutorado em Física Ambiental, Instituto de Física, Universidade Federal de Mato Grosso, Cuiabá, 2013. Citado na página 2.

SANDERS, C. Data management in the Australian NMC. In: *Sixth Workshop on Meteorological Operational Systems, ECMWF*. [S.l.: s.n.], 1997. Citado 2 vezes nas páginas 2 e 8.

SANTANNA, F. *Estimativa dos Fluxos de Calor Sensível e Latente Através do Método de Covariância de Vórtices Turbulentos em Área de Campo Sujo na Baixada Cuiabana*. 2013. 73f. Tese (Doutorado) — Tese (Doutorado em Física Ambiental)-Universidade Federal de Mato Grosso, Instituto de Física, Cuiabá, 2013.[Links], 2013. Citado na página 2.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S.; VIEIRA, D. *Sistema de banco de dados*. [S.l.]: Elsevier, 2006. Citado na página 3.

SILBERSCHATZ A.; SUDARSHAM, K. E. *Sistemas de Bancos de Dados*. 5<sup>a</sup>. ed. [S.l.: s.n.], 2006. Citado na página 12.

SILVA, C. A. R. F. O. *Data Modeling with NoSQL : How, When and Why*. Dissertação (Mestrado) — Engenharia Informática e Computação. Faculdade de Engenharia. Universidade do Porto, PORTUGAL, 2011. Citado na página 17.

SILVA, L. B. *Relações entre aporte de serrapilheira, nutrientes e efluxo de dióxido de carbono em floresta inundável de Vochysia divergens Pohl no Pantanal Mato-Grossense*. 2013. 76f. Tese (Doutorado) — Doutorado em Física Ambiental, Instituto de Física, Universidade Federal de Mato Grosso, 2013. Citado na página 2.

SILVEIRA, S. W. G. *Análise de Desempenho de Produtos MODIS para Modelagem da Dinâmica de Inundação do Pantanal Mato-Grossense*. 2015. 100f. Tese (Doutorado) — Doutorado em Física Ambiental, Instituto de Física, Universidade Federal de Mato Grosso, Cuiabá, 2015. Citado na página 2.

SINGH S.; SINGH, N. Big data analytics. In: SARDAR PATEL INSTITUTE OF TECHNOLOGY. *Anais de International Conference on Communication, Information & Computing Technology*. Mumbai, India, 2012. Citado na página 9.

SOUZA, V.; SANTOS, M. Amadurecimento, consolidação e performance de sgbd nosql - estudo comparativo. In: . [S.l.: s.n.], 2015. Citado 9 vezes nas páginas 3, 16, 21, 26, 31, 33, 44, 45 e 49.

STONEBRAKER, M.; MADDEN, S.; ABADI, D. J.; HARIZOPOULOS, S.; HACHEM, N.; HELLAND, P. The end of an architectural era:(it's time for a complete rewrite). In: VLDB ENDOWMENT. *Proceedings of the 33rd international conference on Very large data bases*. [S.l.], 2007. p. 1150–1160. Citado na página 3.

SVOBODOVA, L. *Computer performance measurement and evaluation methods: analysis and applications*. [S.l.]: North Holland, 1976. v. 2. Citado na página 20.

TAKAI, O. K.; ITALIANO, I. C.; FERREIRA, J. E. Introdução a banco de dados. *DCC-IMEUSP: Fevereiro*, 2005. Citado 2 vezes nas páginas 13 e 14.

TIWARI, S. *Professional NoSQL*. [S.l.]: John Wiley & Sons, 2011. Citado na página 16.

VARGHA, A.; DELANEY, H. D. The kruskal-wallis test and stochastic homogeneity. *Journal of Educational and Behavioral Statistics*, Sage Publications, v. 23, n. 2, p. 170–192, 1998. Citado 2 vezes nas páginas 27 e 28.

VIEIRA, M. R.; FIGUEIREDO, J. M. d.; LIBERATTI, G.; VIEBRANTZ, A. F. M. Bancos de dados nosql: conceitos, ferramentas, linguagens e estudos de casos no contexto de big data. *Simpósio Brasileiro de Bancos de Dados*, 2012. Citado 2 vezes nas páginas 3 e 9.

WEBER, S. Nosql databases. *University of Applied Sciences HTW Chur, Switzerland*, 2010. Citado 2 vezes nas páginas 18 e 19.

WEYUKER, E. J.; VOKOLOS, F. I. Experience with performance testing of software systems: issues, an approach, and case study. *IEEE transactions on software engineering*, IEEE, n. 12, p. 1147–1156, 2000. Citado na página 19.