UNIVERSIDADE FEDERAL FLUMINENSE

**LORENNA CHRIST'NA ARAÚJO NASCIMENTO**

# Análise de Abordagens de Gerência de Dados para Apoiar a Visualização Interativa de Dados Pluviométricos

NITERÓI

2022

UNIVERSIDADE FEDERAL FLUMINENSE

**LORENNA CHRIST'NA ARAÚJO NASCIMENTO**

# Análise de Abordagens de Gerência de Dados para Apoiar a Visualização Interativa de Dados Pluviométricos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação.

Orientador:
Prof. D.Sc. DANIEL CARDOSO MORAES DE OLIVEIRA

Coorientador:
Prof. D.Sc. MARCOS DE OLIVEIRA LAGE FERREIRA

NITERÓI

2022

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

A658a Araújo nascimento, Lorenna Christ'na
Análise de Abordagens de Gerência de Dados para Apoiar a Visualização Interativa de Dados Pluviométricos / Lorenna Christ'na Araújo nascimento ; Daniel Cardoso Moraes de Oliveira, orientador ; Marcos De Oliveira Lage Ferreira, coorientador. Niterói, 2022.
51 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2022.

DOI: http://dx.doi.org/10.22409/PGC.2022.m.16208549701

1. Gerência de Dados. 2. Dados Espaço-Temporais. 3. Dados Pluviométricos. 4. Produção intelectual. I. Cardoso Moraes de Oliveira, Daniel, orientador. II. De Oliveira Lage Ferreira, Marcos, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

LORENNA CHRIST'NA ARAÚJO NASCIMENTO

Análise de Abordagens de Gerência de Dados para Apoiar a Visualização Interativa de Dados Pluviométricos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação.

Aprovada em abril de 2022.

BANCA EXAMINADORA

Prof. D.Sc. Daniel Cardoso Moraes de Oliveira - Orientador, IC/UFF

Prof. D.Sc. Marcos de Oliveira Lage Ferreira - Coorientador, IC/UFF

Profa. D.Sc. Marta Lima de Queirós Mattoso, COPPE/UFRJ

Prof. Ph.D. Nivan Roberto Ferreira Júnior, CIn/UFPE

Profa. D.Sc. Aline Marins Paes Carvalho, IC/UFF

Niterói

2022

# Agradecimentos

A Deus, por todas as oportunidades e momentos que me proporciona.

Aos professores Daniel de Oliveira e Marcos Lage, pelos ensinamentos, apoio, conselhos e compreensão durante essa jornada. Por muitas vezes não foi fácil conciliar o trabalho e o mestrado, mas tive a grande sorte de ter dois professores tão humanos ao meu lado.

Aos meus pais, Margarete Araújo e Paulo Nascimento, por acreditarem que sou capaz de conquistar todos os meus objetivos e me darem todo o apoio que preciso para que isso seja possível. Tudo que faço é por vocês.

À minha avó Ziza, aos meus tios Josiane e Fidélis (em memória), e às minhas primas Amanda e Anna Clara, pelo carinho especial que têm por mim e por sempre fazerem de tudo pela minha felicidade.

Ao meu namorado Mikael Schirru, por me ajudar a ser uma pessoa melhor todos os dias, por me incentivar a lutar pelos meus objetivos e não me deixar desistir dos meus sonhos, e à sua família pelo apoio e carinho.

Ao meu querido professor e amigo Fábio Santos, que foi um divisor de águas na minha vida. Seu apoio e incentivo sempre me impulsionaram a voar cada vez mais alto. Essa conquista é sua também, professor!

À Ellen Beatriz, Júlia Paixão e Suyam Almeida, por me ensinarem que irmãs nem sempre compartilham do mesmo sangue. Ao Francesco Comunale, por ser um amigo tão presente em minha vida e por me apoiar em todos os momentos.

# Resumo

Técnicas de Visualização de Dados têm sido amplamente utilizadas não somente para apresentar resultados e/ou análises realizadas *a priori*, mas também para compor sistemas interativos de apoio à tomada de decisão. Estudos mostram que tais sistemas, chamados de sistemas de Visualização Interativa, precisam responder interações em até 0,5 segundo (500ms) para não interferir na experiência cognitiva do usuário, ou seja, em seu processo de análise e observação dos dados. A análise de um grande conjunto de dados requer a construção e/ou atualização dinâmica de visualizações a partir de seleções, filtragens, junções, *etc*, definidas pelo usuário. Entretanto, consultas envolvendo regiões geográficas e grandes séries temporais são complexas e, consequentemente, podem demorar para serem processadas. Desta forma, atingir os requisitos de tempo de resposta esperados em um sistema de Visualização Interativa é desafiador. Nesta dissertação realizamos um estudo comparativo do desempenho de cinco abordagens de gerência de dados, sendo elas o Apache Drill (um sistema *Polystore*), Apache Spark (um *framework* para *Big Data*), Elasticsearch (um mecanismo de busca), MonetDB (um SGBD colunar) e PostgreSQL (um SGBD relacional). Para tornar este estudo possível, desenvolvemos a ferramenta TEMPO, que captura dados pluviométricos de múltiplas fontes, aplica transformações dos dados e os integra em um único repositório. Ainda, a ferramenta conta com uma aplicação *web* que permite que usuários façam o envio de consultas espaço-temporais e visualizem os índices pluviométricos obtidos em um mapa. Dentre as consultas que podem ser submetidas pelos usuários através da ferramenta, 14 foram selecionadas para a realização deste estudo. Os resultados mostraram que o Apache Spark e o MonetDB são as soluções com melhor desempenho para as consultas selecionadas.

**Palavras-chave**: Gerência de Dados, Dados Espaço-Temporais, Dados Pluviométricos.

# Abstract

Data Visualization techniques have been widely used not only to present results and/or analyses performed *a priori*, but also to compose interactive systems to support decision making. Studies show that such systems, called Interactive Visualization systems, need to respond to interactions within 0.5 seconds (500ms) so as not to interfere with the user's cognitive experience, *i.e.*, in their data analysis and observation process. Analyzing a large dataset requires building and/or updating views dynamically from selections, filters, joins, etc, defined by the user. However, queries that involve geographic regions and large time series are complex and, hence can take time to process. Thus, meeting the response time requirements expected in an Interactive Visualization system is challenging. In this dissertation we carry out a comparative study of the performance of five data management approaches, which are Apache Drill (a Polystore system), Apache Spark (a framework for Big Data), Elasticsearch (a search engine), MonetDB (a columnar DBMS), and PostgreSQL (a relational DBMS). To support this research, we developed the tool `TEMPO`, which captures rainfall data from multiple sources, applies data transformations and integrates them into a single repository. Furthermore, the tool has a web application that allows users to send spatiotemporal queries and view the rainfall obtained on a map. Among the queries that can be submitted by users through the tool, 14 were selected for this study. The results showed that Apache Spark and MonetDB are the solutions with the best performance for the selected queries.

**Keywords**: Data Management, Spatiotemporal Data, Rainfall Data.

# Lista de Figuras


1.1 Cidade de Petrópolis durante eventos de fevereiro de 2022 (Fonte: Portal g1). . . . . 3

1.2 Avenida Marques de Paraná inundada durante um forte temporal (Fonte: Portal g1. . . . . 3

2.1 Representação de pontos no espaço. . . . . 9

2.2 Representação de uma seleção de pontos no espaço. . . . . 9

2.3 Representação de uma junção. . . . . 10

2.4 Representação de uma agregação. . . . . 11

2.5 Representação de uma consulta de . . . . . 11

2.6 Diagrama de Voronoi. . . . . 12

3.1 Arquitetura da `TEMPO`. . . . . 18

3.2 Análise dos índices pluviométricos na região de Niterói em múltiplas janelas de tempo: (a) 15 minutos, (b) 30 minutos, (c) 1 hora, (d) 24 horas. . . . . 19

3.3 Modelagem do DW-TEMPO. . . . . 21

3.4 Número de Tuplas por Estação na Cidade de Niterói . . . . . 22

3.5 Região do Bairro do Ingá na Cidade de Niterói. . . . . 24

# Lista de Tabelas

3.1 Total de Tuplas por Tabela . . . . . 21

3.2 Consultas geradas através da interação do usuário com a `TEMPO`. . . . . 23

4.1 Desempenho das Consultas Q1 a Q14 na máquina virtual m5dn.large. . . . 29

4.2 Desempenho das Consultas Q1 a Q14 na máquina virtual c5.xlarge. . . . . 29

4.3 Desempenho das Consultas Q1 a Q14 na máquina virtual r4.xlarge. . . . . 30

# Sumário

**1 Introdução** **1**

1.1 Motivação . . . . . 1

1.2 Definição do Problema . . . . . 4

1.3 Objetivo . . . . . 6

1.4 Organização da Dissertação . . . . . 6

**2 Referencial Teórico** **7**

2.1 Visualização Interativa . . . . . 7

2.2 Classificação de Consultas Espaço-Temporais . . . . . 8

2.3 Abordagens de Gerência de Dados . . . . . 11

2.4 OLAP e *Data Warehousing* . . . . . 14

**3 Estudo de Caso: Monitoramento de Dados Pluviométricos na Cidade de Niterói** **16**

3.1 Definição do Problema . . . . . 16

3.2 A Ferramenta `TEMPO` . . . . . 17

3.3 `DW-TEMPO` . . . . . 19

3.4 Estatísticas do `DW-TEMPO` . . . . . 20

3.5 Consultas Geradas na `TEMPO` . . . . . 22

**4 Estudo Comparativo das Abordagens de Gerência de Dados** **25**

4.1 *Setup* do Ambiente . . . . . 25

4.2 *Setup* do Experimento . . . . . 26

4.3 Discussão dos Resultados . . . . . 27

**5 Trabalhos Relacionados 31**

5.1 Gerência de Dados Pluviométricos . . . . . 31

5.2 Estudo Comparativo de Abordagens de Gerência de Dados . . . . . 32

**6 Conclusão e Trabalhos Futuros 34**

**Referências 37**

# Capítulo 1

# Introdução

## 1.1 Motivação

Nos últimos anos, o uso de técnicas de visualização tem ganho muita importância com o avanço da área de Ciência de Dados [53]. A área de Ciência de Dados envolve múltiplos assuntos como Gerência de Dados e Aprendizado de Máquina, além da área de Visualização (daqui em diante referenciada somente como *Vis*). Esta última, especialmente, desempenha um papel fundamental, já que apoia a compreensão e análise dos dados de entrada, dos resultados de consultas e de modelos de Aprendizado de Máquina. Tal processo de exploração e análise visual requer, em muitos casos, o uso de *Vis* interativas [39], *i.e.*, capazes de reagir de forma "imediata" a uma determinada ação realizada no sistema (*e.g.*, um filtro temporal ou a seleção de uma área no espaço). Mais precisamente, para produzir *Vis* interativas, os dados devem ser acessados, filtrados e agregados de forma eficiente para que uma nova representação visual seja produzida com o mínimo possível de latência, idealmente abaixo de 0,5 segundos [30].

Sendo assim, um dos desafios de implementar *Vis* interativas é que os filtros e agregações comumente envolvem consultas no espaço e no tempo, o que pode impactar diretamente no tempo de resposta do resultado da consulta. Os dados espaciais envolvidos nessas operações consistem em coleções de pontos, linhas, regiões, retângulos, *etc.* [52], e podem ser usados para representar estruturas mais complexas como bairros e cidades. Dessa forma, diversos sistemas de *Vis* fazem uso de estruturas especializadas para representar dados espaço-temporais [29]. Essas estruturas já se mostraram efetivas para reduzir a latência nas consultas, mas possuem algumas desvantagens.

Paralelamente, pesquisas e estudos relacionados à climatologia têm ganhado cada

vez mais relevância e destaque ao longo dos últimos anos, principalmente em razão do número crescente de eventos climáticos e meteorológicos que vêm ocorrendo ao redor do planeta, inclusive no Brasil [12, 13, 26, 40]. Os eventos climáticos possuem diferentes origens, sendo elas hidrológica (*e.g.*, inundações, alagamentos, enchentes, *etc*), geológica (*e.g.*, processos erosivos, deslizamentos, *etc*), meteorológica (*e.g.*, ciclones tropicais, raios, tornados, *etc*), ou climatológica (*e.g.*, queimadas, incêndios florestais, chuvas de granizo, *etc*) [1]. Nos centros urbanos, o impacto dessas ocorrências pode ser ainda mais severo. Como resultado de um crescimento comumente não ordenado, as cidades tendem a possuir extensas áreas impermeabilizadas, que fazem com que o fluxo de água da chuva corra rapidamente para locais como rios e baixadas. Esses locais comumente não possuem a capacidade e estrutura de escoamento necessária para apoiar o volume de água recebido, o que ocasiona transbordamentos e inundações [58].

De acordo com Mizutori e Guha-Sapir [36], entre 1980 e 1999, 4.212 ocorrências se encontram relacionados a desastres naturais no mundo inteiro, o que custou aproximadamente 1,19 milhões de vidas e afetou 3,25 bilhões de pessoas, resultando em aproximadamente US$ 1,63 trilhão em perdas econômicas. No período de 2000 a 2019, por sua vez, foram registrados 7.348 ocorrências, custando 1,23 milhões de vidas, afetando 4,2 bilhões de pessoas (muitas delas sofreram as consequências de mais de uma ocorrência), resultando em aproximadamente US$ 2,97 trilhões em perdas econômicas globais. O número de ocorrências aumentou significativamente de um período para o outro considerando o mesmo espaço de tempo. Os casos de inundações, particularmente, mais que dobraram, subindo de 1.389 para 3.254 registros no mesmo período [36].

Tais eventos têm acontecido com certa frequência no Brasil. No fim de 2021, fortes enchentes atingiram o estado da Bahia. Estima-se que 53.900 pessoas foram desalojadas de suas casas e cerca de 132 municípios declararam situação de emergência. A intensidade das enchentes foi tão forte que, infelizmente, acabou provocando dezenas de mortes [2]. Em fevereiro de 2022, fortes chuvas atingiram a cidade de Petrópolis (Figura 1.1), no estado do Rio de Janeiro. Em apenas 6 horas choveu o equivalente a todo o mês de fevereiro, causando diversos estragos na cidade e vitimizando mais de 100 pessoas[3]. Uma outra tragédia que ganhou repercussão nacional devido ao impacto das chuvas foi o caso do deslizamento no Morro do Bumba, ocorrido em 2010, na cidade de Niterói, no estado do Rio de Janeiro [4], em que 47 pessoas vieram a óbito.

[1] `https://climaesaude.icict.fiocruz.br/tema/eventos-extremos-0`
[2] `https://t.ly/8Lx8`
[3] `http://shorturl.at/gDKY0`
[4] `https://glo.bo/2PmfyBJ`

Figura 1.1: Cidade de Petrópolis durante eventos de fevereiro de 2022 (Fonte: Portal g1).

Mesmo em casos em que desastres não ocorrem, as chuvas extremas também são responsáveis por causar transtornos grandes no trânsito em diversas cidades ao redor do país. Em Niterói, vias importantes da cidade sofrem comumente com inundações, como a Avenida Marquês de Paraná (Figura 1.2). Apesar dos esforços da prefeitura para realizar intervenções nesses locais, a capacidade de escoamento ainda está longe do suficiente. Com a ocorrência de tempestades repentinas ou pancadas de chuva intermitentes, essas ruas se tornam intransitáveis e impactam o trânsito de toda a cidade (já que são as principais saídas em direção à cidade do Rio de Janeiro).

Figura 1.2: Avenida Marques de Paraná inundada durante um forte temporal (Fonte: Portal g1.

Monitorar o volume de água das chuvas é uma tarefa prioritária para os órgãos governamentais [42], sejam eles municipais, estaduais ou federais. A fim de possibilitar

tal monitoramento, diversas estações pluviométricas (sistemas de obtenção de dados por meio de sensores que fornecem o índice pluviométrico em uma região) se encontram instaladas por todo o país. Ainda que essas estações representem um avanço no que se refere ao monitoramento das chuvas, fornecendo dados praticamente em tempo real, elas sozinhas não resolvem o problema, uma vez que tais dados são disponibilizados por diferentes fontes e em diferentes formatos (*e.g.*, CSV, Arquivos de Texto), além de não existir uma ferramenta de visualização para que seja possível monitorar o índice pluviométrico. Sistemas especializados, como a `TEMPO` [42], capturam esses dados, os tratam de forma apropriada e os disponibilizam para os usuários tanto de forma tabular quanto através de sistemas de análise visual interativos. De fato, a área de *Visualização* desempenha um papel fundamental nesse cenário, já que apoia a compreensão e análise dos dados de brutos e processados, além de ajudar na compreensão de modelos de aprendizado de máquina.

Em especial, no cenário de análise de dados pluviométricos, o processo de exploração e análise visual requer o uso de *Vis* interativas [39], uma vez que os dados podem ser filtrados de acordo com o espaço (*e.g.*, bairros, cidades) e com o tempo (*e.g.*, ano = 2018). Além disso, órgãos governamentais (*e.g.*, Defesa Civil) podem precisar tomar decisões rapidamente para evitar que tragédias ocorram. Dessa forma, é importante que os dados sejam acessados, filtrados e agregados de forma eficiente para que uma nova representação visual seja produzida com o mínimo possível de latência. Assim, a abordagem escolhida para a gerência dos dados possui um papel chave no processo, pois se os dados estiverem mal organizados (e.g., modelados de forma ineficiente ou armazenados em seu grão mais fino somente), o tempo de consulta aos mesmos será maior que o valor esperado.

## 1.2 Definição do Problema

A área de *Vis* interativa tem se destacado com aplicações em múltiplos domínios, *e.g.*, saúde [8], transportes [44], planejamento urbano [34, 35]. Para atender o requisito de latência máxima de 0,5 segundos [30], estruturas como os *Nano Cubes* são utilizadas no desenvolvimento de sistemas de *Vis* interativa para representar dados espaço-temporais [29]. Apesar de serem estruturas especializadas para esse tipo de questão, existem desvantagens como a indisponibilidade do dado em seu grão mais fino e o custo de armazenamento exponencial em relação ao número de atributos. Além disso, construir um *Nano Cube* pode ser um processo laborioso dependendo da quantidades de atributos e os tipos de agregações considerados.

Outra solução bastante utilizada para a gerência de dados espaço-temporais é o PostGIS, uma extensão do Sistema de Gerência de Banco de Dados (daqui em diante referenciado somente como SGBD) PostgreSQL. Apesar de representar um avanço no que tange à facilidade na elaboração de consultas espaço-temporais, muitos usuários detectam problemas de desempenho no PostGIS ao trabalhar com dados espaço-temporais em larga escala [54]. De fato, o PostGIS não foi projetado para apoiar mecanismos de *Vis* interativa que consomem dados espaço-temporais volumosos e detalhados, e as consultas podem demorar muito para retornar resultados, o que não é desejado. Aliadas a tais estruturas, ainda podem ser utilizadas técnicas de processamento aproximado de consultas [62], já tradicionais na área de Banco de Dados.

Concorrentemente, a comunidade de Banco de Dados vem propondo uma série de soluções agnósticas de domínio/problema para gerência de dados nos últimos anos. Tais soluções vão desde SGBDs NoSQL (*e.g.*, MonetDB [7]) até *frameworks* de consulta e análise de dados de larga escala como o Spark SQL [2], além dos SGBDs relacionais tradicionais, como o PostgreSQL. Apesar de não terem sido projetadas para apoiar especificamente consultas de *Vis* interativa, tais soluções têm se mostrado cada vez mais eficientes e podem ser usadas neste contexto, substituindo soluções especializadas como os *Nano Cubes*.

Dessa forma, o problema de pesquisa tratado nessa dissertação é "*Como escolher a abordagem de gerência de dados mais adequada para as consultas de Vis interativa, buscando atingir o requisito da latência mínima de 0,5 segundos?*". Toda a pesquisa deve ser guiada por dados reais, e por isso, escolhemos como estudo de caso a ferramenta de análise de dados pluviométricos da cidade de Niterói denominada `TEMPO` [42]. Apesar de existirem *benchmarks* voltados para *Vis* interativa, o que mais se aproximou do ideal para o estudo realizado nesta dissertação foi o *benchmark* proposto por Battle *et al.*, porém eles se concentram em bancos de dados relacionais. Como nesta dissertação foram exploradas outras categorias de abordagens de gerência de dados, desenvolvemos a ferramenta `TEMPO` para utilizá-la como estudo de caso. A `TEMPO` permite ao usuário submeter consultas espaço-temporais com o objetivo de analisar o comportamento das chuvas no município de Niterói. O sistema possui dados de precipitação medidos em múltiplas estações pluviométricas ao longo de seis anos (mais detalhes na Seção 3).

## 1.3 Objetivo

Nesta dissertação realizamos um estudo comparativo do desempenho de cinco abordagens de gerência de dados de diferentes categorias. As abordagens selecionadas foram (i) Apache Drill (um sistema *Polystore*), (ii) Apache Spark (um *framework* para Big Data), (iii) Elasticsearch (um mecanismo de busca), (iv) MonetDB (um SGBD colunar), e (v) PostgreSQL (um SGBD relacional). Para isso, selecionamos 14 consultas submetidas pelos usuários na ferramenta `TEMPO` e calculamos o tempo de resposta médio de um total de 20 execuções para cada consulta em ambientes de desenvolvimento com diferentes características. Dessa forma, os objetivos específicos dessa dissertação são:

- Desenvolver uma ferramenta (*i.e.*, `TEMPO`) que capture, armazene e consulte dados pluviométricos capturados de múltiplas fontes;
- Modelar os dados capturados nas diferentes abordagens de gerência de dados;
- Selecionar consultas de diferentes categorias e executá-las em cada abordagem de gerência de dados para calcular o tempo de resposta médio;
- Realizar um estudo comparativo das abordagens de gerência de dados no que tange o processamento de dados espaço-temporais;

## 1.4 Organização da Dissertação

Esta dissertação é composta por cinco capítulos além da introdução. O Capítulo 2 apresenta as definições teóricas necessárias para fins de uma melhor compreensão do estudo comparativo, e também as tecnologias utilizadas para apoiar a solução e o estudo desenvolvidos. No Capítulo 3 é explicada a ferramenta `TEMPO`, que foi o estudo de caso escolhido para realizar os estudos desta dissertação. O Capítulo 4 apresenta o ambiente utilizado para a realização dos experimentos e discute os resultados obtidos. O Capítulo 5 apresenta e analisa os trabalhos relacionados. Finalmente, o Capítulo 6 conclui esta dissertação.

# Capítulo 2

# Referencial Teórico

Este capítulo tem como objetivo fornecer a fundamentação teórica acerca do domínio de visualização interativa e os tipos de consultas importantes em sistemas de análise de dados espaço-temporais. Além disso, serão apresentadas as tecnologias utilizadas no desenvolvimento da dissertação. Este capítulo está organizado da seguinte forma: a Seção 2.1 apresenta os requisitos para visualização interativa. A Seção 2.2 discute os padrões de consultas espaço-temporais definidas por Doraiswamy e Freire [11]. A Seção 2.3 apresenta as abordagens de gerência de dados que serão avaliadas nesta dissertação. Finalmente, a Seção 2.4 apresenta os conceitos básicos de *Online Analytical Processing* (OLAP), necessários para o entendimento do estudo de caso.

## 2.1 Visualização Interativa

A área de Ciência de Dados tem se tornado cada vez mais popular e, com isso, o uso de técnicas de visualização tem ganhado muita importância [53]. Apesar da área de Ciência de Dados ser multidisciplinar e envolver assuntos que variam desde Gerência de Dados a Aprendizado de Máquina, a área de Vis desempenha um papel essencial, uma vez que apoia a compreensão e análise dos dados de entrada, dos resultados de consultas e de modelos de Aprendizado de Máquina. Em muitos casos, o processo de exploração e análise visual requer o uso de *Vis* interativas [39] capazes de reagir de forma praticamente "imediata" a uma determinada ação realizada no sistema, como por exemplo um filtro temporal ou a seleção de uma área no espaço.

Sistemas de exploração e análise visual de dados têm sido usados com sucesso para estudar problemas mal-postos (problemas para os quais não é possível encontrar soluções utilizando técnicas puramente computacionais) ou para investigar temas que exigem o

julgamento e/ou a *expertise* de um especialista de domínio [39]. Sistemas de *Vis* têm sido utilizados com sucesso em diversas áreas, como em aplicações relacionadas a saúde [8], gerenciamento urbano [61], segurança [22], mobilidade [59] e educação [10]. Um dos requisitos fundamentais destes sistemas é que eles sejam capazes de atingir tempos de resposta interativos. Liu *et al.* [30] demonstram que latências superiores a 0,5 segundos (500ms) são capazes de interferir na capacidade de observação, generalização e construção de hipóteses durante a exploração visual de dados. Portanto, um dos requisitos comuns a todos sistemas de *Vis* Interativa é obter um tempo de resposta das ações (tempo decorrido desde o momento em que o usuário realiza uma ação até o momento em que recebe a resposta visual) inferior a 0,5 segundos sempre que possível. Os jogos, apesar de não serem considerados como Vis interativa, são um exemplo que deixa claro a importância do tempo de resposta. Beigbeder *et al.* [6] discutem em seu artigo que latências entre 100ms-300ms podem aumentar significativamente a precisão de disparo do usuário no jogo *Unreal Tournament*. Por outro lado, latências maiores que essa faixa degradam significativamente o desempenho do jogador. Esse requisito da latência se torna ainda mais crítico em jogos *online*, como o *Fortnite*[1]. Nesse jogo, sessões com 100 jogadores *online* são criadas e um deve aniquilar o outro em um jogo de tiro. Problemas recentes de latência tem atrapalhado significativamente a experiência no jogo. A fabricante do jogo inclusive emitiu recomendações para os usuários conseguirem reduzir a latência[2].

Entretanto, uma vez que tais ações dependem do acesso aos dados, o tempo de resposta de uma ação se encontra diretamente ligado ao tempo de resposta das consultas ao dados. O grande problema para alcançar o resultado no tempo de resposta esperado, é que tais consultas em sistemas de *Vis* costumam envolver múltiplas junções espaciais, agregações em diversas granularidades e buscas no espaço e no tempo, o que faz com que as consultas se tornem complexas e potencialmente demoradas. Para otimizar a execução de tais consultas, se torna fundamental então que possamos classificá-las. Na Seção 2.2 são detalhadas cinco classes de consultas usadas em análises de dados espaço-temporais.

## 2.2 Classificação de Consultas Espaço-Temporais

Doraiswamy e Freire [11] descrevem uma série de consultas espaço-temporais comumente utilizadas em sistemas de análise de dados. Estas consultas são agrupadas em cinco classes: (C1) *Seleção*, (C2) *Junção*, (C3) *Agregação*, (C4) *Vizinho Mais Próximo* e (C5) *Consultas*

[1] `https://www.epicgames.com/fortnite/pt-BR/home`

[2] `https://www.epicgames.com/help/en-US/fortnite-c75/technical-support-c118/understanding-latency-or-ping-in-fortnite-a7617`

*Geométricas*. Para definir cada classe, consideremos que $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ são as coordenadas de cada ponto no espaço (representações de $X$ na Figura 2.1).

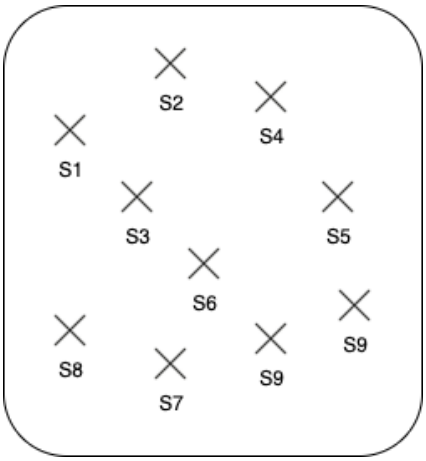


Figura 2.1: Representação de pontos no espaço.

As consultas da classe *Seleção* são aquelas nas quais pontos e polígonos são selecionados de acordo com um predicado de seleção específico. Este tipo de consulta também pode retornar uma série temporal baseada em uma seleção espacial (*e.g.*, retornar a série temporal de chuva em um período de tempo de estações pluviométricas selecionadas) e é apresentado na Figura 2.2. Consideremos $D_p$ um conjunto de pontos de dados (no contexto desta dissertação, um conjunto de estações pluviométricas em uma determinada cidade). Uma consulta do tipo *Seleção* pode ser expressa como a seguinte sintaxe em uma linguagem similar ao SQL: `SELECT * FROM` $D_P$ `WHERE` $A_i$ $\theta$ `valor;` onde $A_i$ é um atributo de cada ponto de dados, $\theta = \{=, <, \leq, >, \geq, \neq\}$, $valor \in Dom_i$ e $Dom_i$ é o domínio $A_i$.

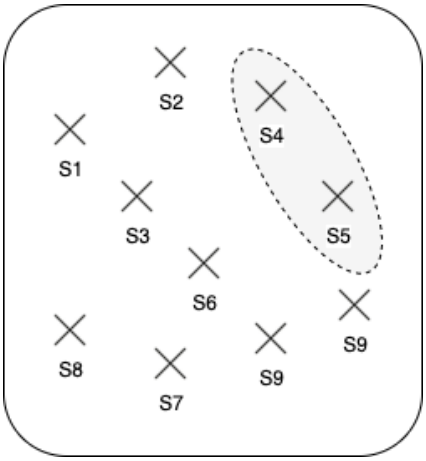


Figura 2.2: Representação de uma seleção de pontos no espaço.

As consultas da classe *Junção*, ilustrada na Figura 2.3, são aquelas em que são necessárias junções ($\bowtie$), seja entre polígonos ou entre polígonos e pontos. As consultas desse tipo podem ser utilizadas, por exemplo, para descobrir quais estações pluviométricas (pontos) estão em uma determinada área do mapa (polígono). Consideremos $D_p$ um conjunto de pontos e $D_Y$ um conjunto de pontos que formam um polígono fechado (*i.e.*, quando a extremidade do último segmento de uma linha poligonal liga-se à extremidade do primeiro). Uma consulta do tipo *Junção* pode ser expressa como a seguinte sintaxe em SQL: `SELECT * FROM` $D_P$`,`$D_Y$ `WHERE` $D_P$`.Location INSIDE` $D_Y$`.Geometry;`

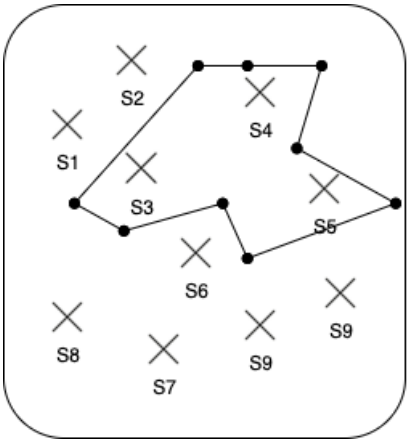


Figura 2.3: Representação de uma junção.

As consultas de *Agregação*, ilustradas na Figura 2.4, são as que normalmente requerem a aplicação de uma função de agregação sobre uma *Seleção* ou uma *Junção*, como por exemplo uma consulta para descobrir a precipitação média nos últimos dois meses de uma determinada área do mapa. Uma *Agregação* sobre uma *Seleção* pode ser representada como a seguinte sintaxe em SQL: `SELECT` $aggreg_f$ `FROM` $D_P$ `WHERE` $A_i$ $\theta$ `valor;` onde $aggreg_f$=`{COUNT,MIN,MAX,AVG,SUM}`, $A_i$ é um atributo de cada ponto de dados, $\theta = \{=, <, \leq, >, \geq, \neq\}$, $valor \in Dom_i$, e $Dom_i$ é o domínio $A_i$. Uma *Agregação* sobre uma *Junção*, por sua vez, pode ser representado como a seguinte sintaxe em SQL: `SELECT` $agreg_f$ `FROM` $D_P$`,`$D_Y$ `WHERE` $D_P$`.Location INSIDE` $D_Y$`.Geometry;` onde $agreg_f$=`{COUNT,MIN, MAX,AVG,SUM}`, $D_p$ um conjunto de pontos no espaço e $D_Y$ um conjunto de dados do polígono.

As consultas da classe *Vizinho Mais Próximo* (Figura 2.5) retornam um conjunto de $k$ pontos no espaço mais próximos de outro ponto especificado de acordo com algum critério, *e.g.*, dada uma coordenada específica, encontre as $k$ estações pluviométricas mais próximas. Uma consulta *Vizinho Mais Próximo* pode ser representada da seguinte forma em SQL: `SELECT * FROM` $D_P$ `WHERE` $D_P$`.Location` $\in$ `KNN(X,K);` onde $X$ é o ponto de

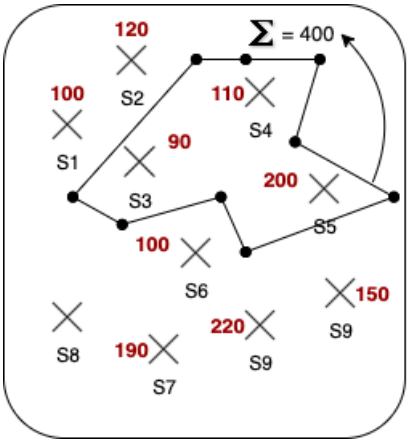


Figura 2.4: Representação de uma agregação.

consulta e $K$ define o número de vizinhos que são retornados.

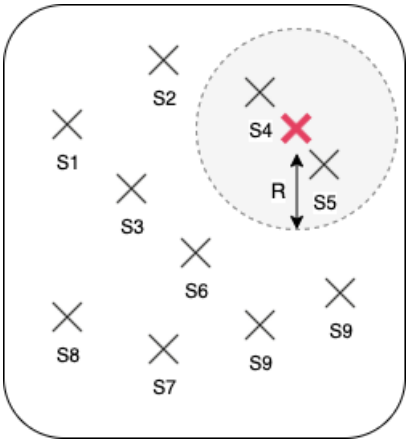


Figura 2.5: Representação de uma consulta de vizinho mais próximo.

Por fim, as consultas *Geométricas* são aquelas que envolvem a decomposição do espaço de dados determinada pela distância para uma dada família de objetos (subconjuntos) no espaço, *e.g.*, diagrama de Voronoi (representado na Figura 2.6). Neste trabalho não serão utilizadas as consultas dessa classe para estudo.

## 2.3 Abordagens de Gerência de Dados

Nesta dissertação, foram selecionadas cinco abordagens de gerenciamento de dados diferentes, cada uma com características próprias. O objetivo é explorar a heterogeneidade das soluções em relação ao modelo de dados adotado, aos mecanismos de processamento de consultas, ao uso de processamento em memória, *etc.* As abordagens selecionadas fo-

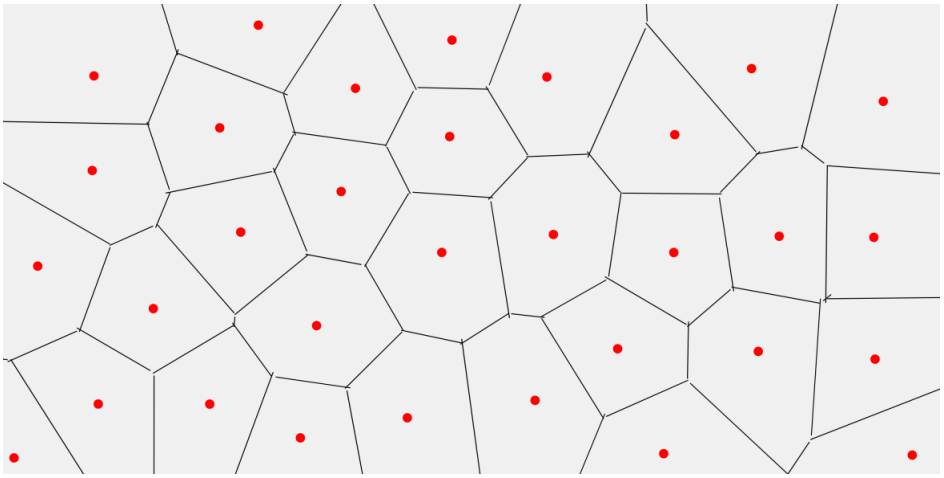

Figura 2.6: Diagrama de Voronoi.

ram (i) um SGBD relacional - PostgreSQL, (ii) um SGBD orientado a colunas - MonetDB, (iii) um arcabouço de processamento de grandes volumes de dados - Apache Spark, (iv) um sistema *Polystore* - Apache Drill e (iv) um mecanismo de busca - Elasticsearch. A seguir, é apresentada a lista de abordagens selecionadas com uma breve explicação sobre cada uma delas.

O *Apache Drill* é um sistema distribuído de código aberto para consulta de grandes volumes de dados. O Drill foi projetado para lidar com até petabytes de dados espalhados por uma grande quantidade de servidores [21]. Alguns dos principais recursos do Apache Drill são (i) Consultas de baixa latência, o que significa que para uma consulta simples o resultado é retornado em poucos milissegundos; (ii) Capacidade de acessar múltiplas fontes de dados em uma única consulta, como tabelas Hive, arquivos JSON e sistemas de arquivos (locais ou distribuídos); e (iii) trabalha com ferramentas de *Business Intelligence*, o que permite a integração direta com ferramentas especializadas de visualização de dados [14]. O Apache Drill é considerado um sistema *Polystore* [28], pois pode consultar vários SGBDs que seguem múltiplos modelos de dados.

O *Apache Spark* é um arcabouço de processamento de grandes volumes de dados capaz de executar tarefas em conjuntos de dados muito grandes e também distribuir tarefas de processamento de dados em vários computadores. O Spark melhora o desempenho das aplicações realizando uma movimentação de dados na memória (diferentemente do Apache Hadoop, onde as operações sempre fazem uso de disco, ou do inglês *disk to disk*) explorando o paralelismo de forma automática. Apesar do Spark ser um arcabouço de uso geral, ele oferece apoio ao processamento de consultas em dados estruturados. Uma das principais vantagens do Spark em comparação com outros arcabouços de processamento de

grandes volumes de dados são suas estruturas na memória, como o *Resilient Distributed Dataset* (*RDD*) [60] e os *DataFrames* [1], que são essencialmente coleções na memória de dados particionadas que podem ser processadas em paralelo. Enquanto RDDs são conjuntos de objetos que representam dados, os *DataFrames* são coleções distribuídas de dados com colunas nomeadas, *i.e.*, os DataFrames atuam como tabelas em bancos de dados relacionais (*e.g.*, PostgreSQL, Oracle) Além disso, o Spark possui uma técnica de otimização chamada Execução de Consulta Adaptável (AQE), que se baseia em estatísticas de tempo de execução para estimar a estratégia de execução mais eficiente [55]. O Spark é capaz de acessar e ler diferentes formatos de dados, como arquivos JSON, CSV e ORC, tabelas Hive, *etc.* [56].

O *Elasticsearch* é uma abordagem que faz parte do *Elastic Stack*, ou pilha ELK. Essa pilha é composta pelas ferramentas *Elasticsearch*, *Logstash* e *Kibana*. O *Elasticsearch* é responsável por indexar, pesquisar e analisar conjuntos de dados. O *Logstash* é responsável por coletar e agregar dados a serem indexados pelo *Elasticsearch*. O *Kibana*, por sua vez, oferece maneiras de realizar a visualização e exploração dos dados [17]. O *Elasticsearch* oferece apoio a diferentes tipos de dados e fornece um índice para cada tipo. Tais índices são otimizados para garantir buscas rápidas e eficientes. Um índice é uma coleção de documentos no formato JSON, e um documento é uma coleção de campos representados como pares chave-valor (*key-value*). Cada campo do documento é indexado usando uma estrutura de dados otimizada para seu tipo de dado específico, e isso torna as consultas mais eficientes [16]. O Elasticsearch oferece apoio a consultas estruturadas, como as que podem ser elaboradas usando SQL, e consultas de texto, nas quais uma pesquisa é realizada usando uma *string* (um conjunto de caracteres) em todos os documentos e o resultado é retornado por relevância [18].

O *MonetDB* é um SGBD de código aberto orientado a colunas amplamente utilizado para processamento de consultas que envolvem grandes volumes de dados em diferentes setores, como saúde, telecomunicações, e outros [23]. Inicialmente, foi projetado para aplicações OLAP, que são amplamente utilizados para fornecer apoio à processos de tomada de decisão [23]. O MonetDB foi desenvolvido para apoiar processamento distribuído com o objetivo de reduzir o tempo de resposta para consultas complexas [37]. Ele também apoia consultas escritas em SQL e fornece uma solução moderna e escalável sem exigir investimentos consideráveis em *hardware* [37]. Em termos de índices, embora o usuário possa definir um índice usando comandos SQL, o MonetDB não cria um índice secundário físico conforme definido no comando SQL. Ele decide qual acelerador de pesquisa de coluna criar, persistir e usar durante a execução da consulta SQL. O MonetDB oferece

apoio a dois tipos de índice: *Imprints* e *Ordered*. Tais índices podem ser associados a uma única coluna e somente os atributos numéricos podem ser indexados.

Finalmente, o *PostgreSQL* é um SGBD de código aberto que segue o modelo relacional e usa a linguagem SQL combinada com recursos que armazenam e dimensionam cargas de trabalho de dados complexas [50]. Existem muitas extensões disponíveis para uso no PostgreSQL, incluindo PostGIS, que é uma extensão de banco de dados espacial para PostgreSQL. O PostGIS adiciona apoio nativo para objetos geográficos e habilita funções que podem ser facilmente interpretadas e executadas pelo SQL [48]. Apesar de tais funções adicionarem semântica às consultas e facilitarem o trabalho do usuário, elas comumente não são eficientes para grandes volumes de dados. O PostgreSQL provou ser escalável em termos da quantidade de dados que pode manipular e do número de usuários simultâneos [50]. Os tipos de índice passíveis de uso no PostgreSQL são B-tree, Hash, GiST, SP-GiST, GIN e BRIN [49]. O tipo de índice utilizado nesse trabalho foi a B-tree.

## 2.4 OLAP e *Data Warehousing*

OLAP (*Online Analytical Processing*) é um tipo de consulta que é capaz de manipular e analisar um grande volume de dados sob múltiplas perspectivas, ou também chamadas de dimensões. As aplicações OLAP são comumente utilizadas para tomada de decisão, uma vez que permitem de forma otimizada uma série de análises comparativas. Os *Data Warehouses* (DW) são repositórios para grandes volumes de dados coletados de múltiplas fontes por meio de processos apropriados de Extração, Transformação e Carregamento (ETL, do inglês *Extract, Transform and Load*) [27], estando diretamente associados ao conceito de OLAP. De acordo com [24], os DWs devem lidar com os dados de modo variável no tempo, não volátil e orientada ao assunto, o que permite que os usuários possam consultar os dados em seu grão mais fino, mas também agregados por níveis distintos de granularidade. Os DWs possuem características diferentes dos bancos de dados tradicionais uma vez que são desenvolvidos para otimizar a inserção e a consulta aos dados. Além disso, dependem fortemente da *modelagem dimensional* [27] para manter sua estrutura lógica compreensível, assim como para otimizar o desempenho das consultas.

O modelo dimensional é baseado no conceito de *tabelas fato* e *tabelas de dimensões* para armazenar os dados. As tabelas fato contêm os valores de interesse de consulta (*e.g.*, os índices pluviométricos no contexto dessa dissertação), enquanto que as dimensões qualificam esse dado quantitativo. Um modelo dimensional tradicionalmente segue um dos

seguintes padrões: (i) esquema estrela (*star schema*) e (ii) floco de neve (*snowflake*) [20]. O esquema estrela baseia-se no projeto de uma tabela fato central, enquanto as dimensões restantes são projetadas como tabelas de dimensão que se associam com a fato por meio de chaves estrangeiras. Já no esquema floco de neve, as tabelas de dimensão são normalizadas com relacionamentos "um-para-muitos" entre os atributos normalizados. Dessa forma, o esquema estrela e o floco de neve são padrões concorrentes a serem escolhidos de acordo com a disponibilidade de disco e poder de processamento computacional [27].

Um DW pode ser implementado de três formas diferentes [20]: (i) *Relational* OLAP (ROLAP), (ii) *Multidimensional* OLAP (MOLAP) e (iii) *Hybrid* OLAP (HOLAP). No ROLAP, o modelo multidimensional é implementado em um SGBD relacional, como por exemplo o mySQL e o PostgreSQL. No MOLAP, o modelo multidimensional é implementado em estruturas vetoriais que permitem uma rápida indexação de dados pré-agregados. No HOLAP, existe a combinação do armazenamento do ROLAP com os cálculos rápidos do MOLAP. Uma vez modelados, os DWs devem ser carregados com os dados de múltiplas fontes. Essa carga é realizada por meio de um processo chamado de ETL (Extração, Transformação e Carga). Após a ingestão de dados, os DWs são capazes de oferecer consultas multidimensionais em níveis distintos de granularidade por meio de relatórios dinâmicos e interativos [20].

# Capítulo 3

# Estudo de Caso: Monitoramento de Dados Pluviométricos na Cidade de Niterói

Este capítulo detalha o desenvolvimento da ferramenta `TEMPO`, aplicação escolhida como *benchmark* para gerar as consultas espaço-temporais e realizar o estudo comparativo proposto nesta dissertação. Na Seção 3.1 é explicado o Programa de Desenvolvimento de Projetos Aplicados (PDPA) e a sua relação com o estudo de caso escolhido. A Seção 3.2 apresenta a arquitetura da ferramenta `TEMPO` e explica a função de suas camadas. A Seção 3.3 detalha como o `DW-TEMPO` foi proposto por Nascimento *et al.* [42]. A Seção 3.4 apresenta os dados estatísticos do `DW-TEMPO`. Por fim, a Seção 3.5 apresenta as consultas comumente submetidas pelo usuário por meio da interação com a ferramenta `TEMPO`.

## 3.1 Definição do Problema

Monitorar o acumulado de chuvas das cidades é fundamental para órgãos governamentais (*e.g.*, Defesa Civil) [42]. Tal monitoramento permite que seja possível analisar cenários, tomar decisões e agir de antemão para evitar que tragédias ocorram (*e.g.*, deslizamentos, alagamentos). A disponibilização de dados de chuva é feita por estações pluviométricas que estão localizadas em múltiplas cidades ao redor do país. Na cidade de Niterói, por exemplo, um conjunto de estações pluviométricas são gerenciadas pela prefeitura, enquanto um outro subconjunto é gerenciado pelo CEMADEN (Centro Nacional de Monitoramento e Alertas de Desastres Naturais) [9]. Apesar dos dados pluviométricos serem disponibilizados publicamente, obter tais dados e integrá-los em uma única fonte não é uma tarefa trivial. A fim de facilitar o processo de coleta, integração e análise dos dados,

foi desenvolvida a ferramenta TEMPO (sisTema dE Monitoramento PluviométricO).

A ferramenta TEMPO, apresentada na Seção 3.2, se encontra no contexto do projeto "*Niterói Organizada e Segura: Estudo do Impacto das Chuvas*", realizado no contexto do PDPA. O Programa de Desenvolvimento de Projetos Aplicados (PDPA) é resultado de uma parceria entre a Prefeitura Municipal de Niterói (PMN), a Universidade Federal Fluminense (UFF) e a Fundação Euclides da Cunha (FEC). O objetivo do PDPA é incentivar o desenvolvimento de projetos aplicados a fim de promover soluções para os principais desafios enfrentados pela cidade de Niterói em diferentes áreas[1]. O PDPA torna possível utilizar conhecimento da UFF para solucionar problemas públicos da cidade de Niterói, de modo a contribuir para o desenvolvimento sustentável do município.

## 3.2 A Ferramenta TEMPO

A ferramenta TEMPO tem por objetivo captar dados pluviométricos de múltiplas fontes com diferentes granularidades e integrá-los em um repositório único para possibilitar consultas e análises posteriores. Essas consultas são consultas analíticas (*i.e., Online Analytical Processing* ou OLAP) e comumente demandam uma grande necessidade de processamento. Na Figura 3.1 é apresentada a arquitetura da TEMPO, e esta é composta por três camadas principais: (i) Fontes de Dados, (ii) Camada de Armazenamento e (iii) Camada de Análise.

A camada de Fontes de Dados é onde os dados brutos são obtidos para processamento e análise. Nessa camada se encontram as principais fontes que fornecem informação sobre os índices pluviométricos de uma determinada região em um determinado instante de tempo [42]. No que se refere aos formatos de dados, esses provedores disponibilizam seus dados de variadas formas. Por exemplo, o CEMADEN fornece arquivos textuais em formato CSV, enquanto o Alerta Rio fornece dados em formato de texto próprio (que necessita de um dicionário de dados para compreensão). Outro ponto crítico se refere a granularidade dos dados. Enquanto que o Alerta Rio e a Prefeitura de Niterói disponibilizam dados a cada 15 minutos (com chuva ou sem chuva), o CEMADEN disponibiliza dados a cada hora, em caso de tempo sem chuva, ou a cada 10 minutos quando há precipitação. Dessa forma, para sermos capazes de capturar os dados em formatos e granularidades distinhas, cada fonte possui um *Adaptador* associado dentro da TEMPO. O componente *Crawler* realiza o *download* dos dados brutos invocando os Adaptadores (passo ❶), que por sua vez acessam APIs públicas, servidores FTP, e outros possíveis

[1] https://somosfec.org.br/projetos-aplicados/

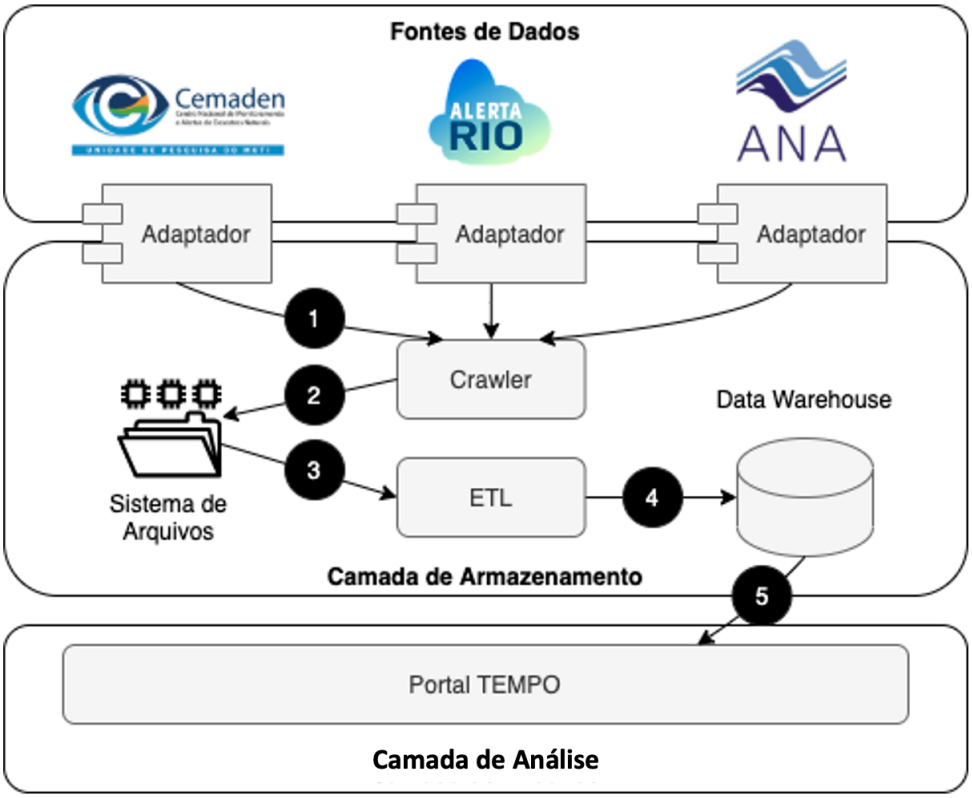


Figura 3.1: Arquitetura da TEMPO.

mecanismos dos provedores para obter os dados. Tais dados coletados são armazenados no *Sistema de Arquivos* da Camada de Armazenamento em seu estado bruto (passo ❷). Além de armazenar os dados brutos (*e.g.*, em formato CSV, JSON), são criadas tabelas *Staging* para cada fonte de dados consideradas. Essas tabelas são temporárias e possuem os mesmos atributos que constam em cada um dos arquivos captados nas fontes de dados, porém já com um tratamento inicial de formato.

Assim que os dados brutos são coletados nas fontes e as tabelas de *Staging* se encontram disponíveis, o componente *ETL* realiza as devidas transformações nos dados, como, por exemplo, o ajuste de granularidade, a padronização de endereços e localizações, agregações, *etc* (passo ❸). Em sua versão atual, a TEMPO utiliza a ferramenta *Pentaho Data Integration* versão 9.1 para realizar o carregamento dos dados no DW, que chamamos de DW-TEMPO (passo ❸).

Por fim, na Camada de Análise, o Portal TEMPO fornece mecanismos de visualização de dados espaciais para o usuário. Apesar do usuário poder submeter consultas diretamente ao DW-TEMPO, a representação tabular retornada pela consulta pode dificultar a análise e visualização dos dados [42]. Tomemos como exemplo um usuário que deseja

comparar a evolução dos índices pluviométricos em um período de tempo de interesse. Se o usuário puder, ao invés de inspecionar uma planilha contendo uma série de atributos (*e.g.*, índice pluviométrico, a localização da estação e a data), visualizar um conjunto de mapas com valores acumulados de chuva na região de interesse, conseguirá identificar padrões de precipitação potencialmente relevantes de forma mais efetiva.

A Figura 3.2 ilustra um exemplo de visualização de índices pluviométricos geradas pelo `Portal TEMPO` [42]. Na ilustração é possível observar os acumulados de chuva dos últimos 15 min (janela a), 30 min (janela b), 1 hora (janela c) e 24 horas (janela d). Embora o `DW-TEMPO` forneça apenas índices pluviométricos em pontos específicos do espaço (onde existem estações pluviométricas), utilizando técnicas de interpolação baseadas em distância [31], é possível construir uma grade com valores acumulados de chuva cobrindo toda a região geográfica de interesse. Ao usar estes valores interpolados, pode-se visualizar índices pluviométricos por meio de mapas de calor, como os apresentados na Figura 3.2.

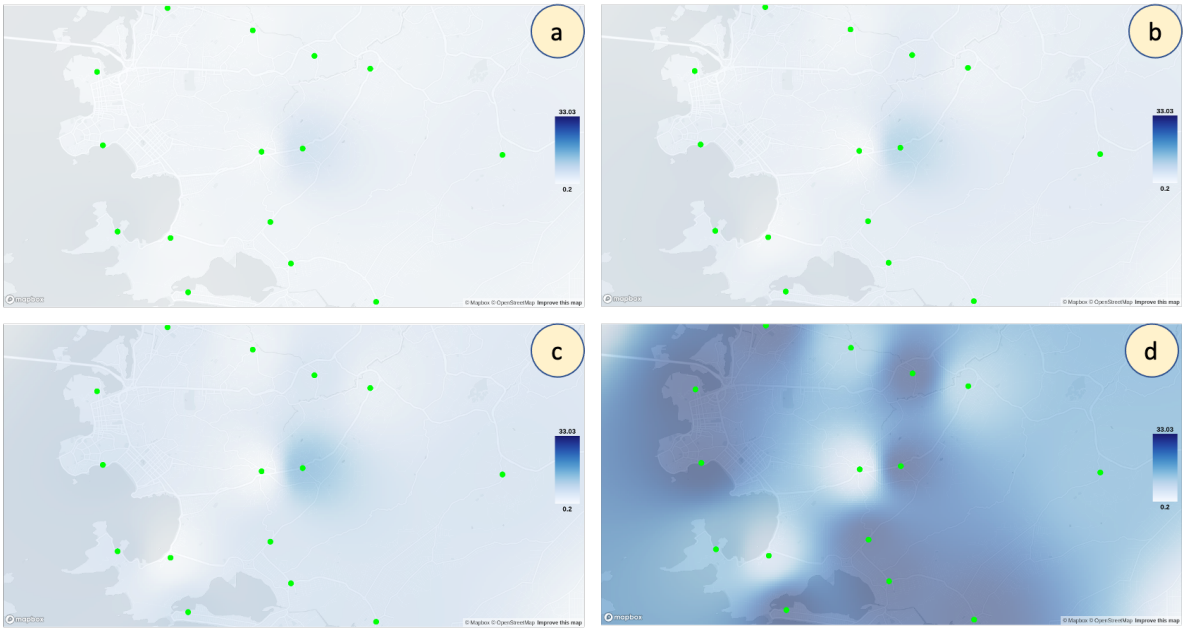


Figura 3.2: Análise dos índices pluviométricos na região de Niterói em múltiplas janelas de tempo: (a) 15 minutos, (b) 30 minutos, (c) 1 hora, (d) 24 horas.

## 3.3 DW-TEMPO

A modelagem do `DW-TEMPO` foi realizada seguindo o esquema estrela proposto por Kimball e Ross [27]. Esse esquema consiste em uma *tabela fato*, que fica localizada em uma posição central do modelo, ligada por meio de múltiplos relacionamentos *1:n* com as *tabelas de dimensão*. A tabela fato é a responsável por armazenar os dados quantitativos de interesse

enquanto que as dimensões focam nos dados qualitativos que contextualizam a quantidade armazenada na fato. Esse tipo de modelagem facilita o apoio a consultas analíticas uma vez que permite que dados sejam pré-agregados nas múltiplas dimensões. O esquema estrela foi escolhido uma vez que as dimensões são desnormalizadas, o que facilita o processo de ETL e a própria agregação dos dados [42]. Nessa proposta, a abordagem utilizada foi a ROLAP (*i.e.*, *Relational* OLAP, onde um banco de dados dimensional é instanciado em um SGBD relacional), uma vez que o DW-TEMPO foi instancializado no SGBD PostgreSQL versão 4.26 [42].

O *schema* do DW-TEMPO é ilustrado na Figura 3.3. Esse *schema* possui uma tabela fato e quatro tabelas de dimensão. As tabelas de dimensão possuem uma *primary-key* (PK), chave única de identificação para cada tupla. A tabela fato, por sua vez, está vinculada a todas as tabelas de dimensão por um relacionamento (*1:n*). Dessa forma, haverá uma *foreign-key* (FK) correspondente a cada *primary-key* de cada tabela dimensão. A tabela fato *Chuvas* contém os atributos Id_Localidade (FK), Id_Fonte (FK), Id_Estacao (FK), Id_Tempo (FK) Indice_Pluviometrico (quantidade de interesse para análise), onde todas as *foreign keys* referenciam as tabelas de dimensão associadas. A dimensão *Localidade* representa os locais onde as estações pluviométricas se encontram (onde as medições foram realizadas), e contém os atributos: Id_Localidade (PK), Latitude, Longitude, Cidade, Estado, Pais e Endereço. A dimensão *Fonte* representa a proveniência do dado, ou seja, de qual fonte o dado foi obtido. Essa tabela contém os atributos: Id_Fonte (PK), Fonte e Fonte_URL. A dimensão *Tempo* representa os possíveis intervalos de tempo para consulta, e contém os atributos: Id_Tempo (PK), Dia, Mes, Ano, Semestre, Hora e Minuto. Por fim, a dimensão *Estação* representa as estações que foram consideradas para análise, e contém os atributos: Id_Estacao (PK), Codigo e Nome.

## 3.4 Estatísticas do DW-TEMPO

A seguir são apresentadas algumas estatísticas do DW-TEMPO. O DW-TEMPO contém dados pluviométricos coletados no período de 2013 a 2019. A Tabela 3.1 apresenta o número total de tuplas da tabela fato e de cada tabela de dimensão. Podemos destacar nesse momento que apesar de o DW-TEMPO possuir somente seis anos de dados, a quantidade de tuplas não é desprezível (mais de 18 milhões de leituras).

Além disso, o índice pluviométrico é registrado em momentos diferentes e varia de acordo com a fonte de dados que computa o registro, *e.g.*, os dados baixados do CE-

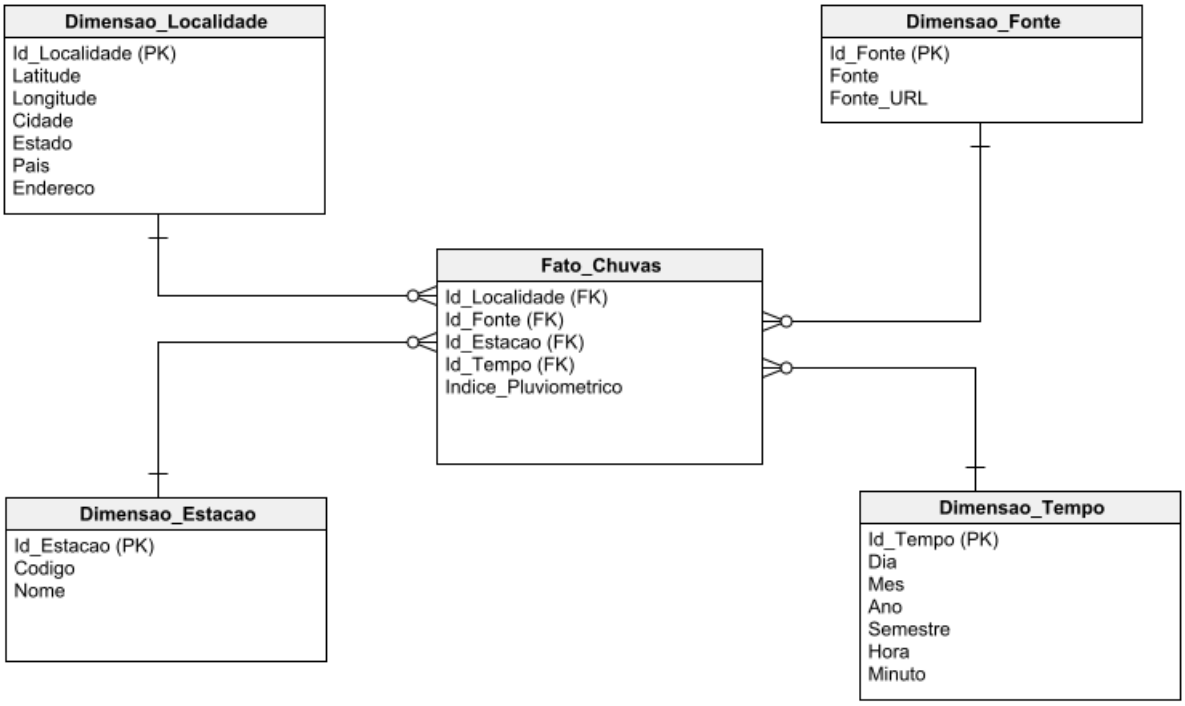


Figura 3.3: Modelagem do DW-TEMPO.

MADEN são capturados a cada hora (quando não está chovendo) e a cada 10 minutos (se estiver chovendo), enquanto no Alerta Rio os registros são feitos sempre a cada 15 minutos [42]. Como a ferramenta TEMPO reúne dados do CEMADEN (que é um órgão federal), o DW-TEMPO armazena dados de chuva de vários municípios ao redor do Brasil. Os números apresentados na Tabela 3.1 englobam todos os dados armazenados. Entretanto, como essa dissertação é voltada para a cidade de Niterói, as análises visuais realizadas pelos usuários na TEMPO são focadas no contexto de tal cidade. A Tabela 3.4, por sua vez, apresenta o número de tuplas na tabela fato por estação pluviométrica no município de Niterói ($\times$1.000). Ao coletar metadados das interações do usuário com a TEMPO foi identificada uma série de consultas comuns e frequentes ao banco de dados, que serão apresentadas na seção a seguir.

Tabela 3.1: Total de Tuplas por Tabela

| Tabela | Número de Tuplas |
|---|---|
| Fato_Chuvas | 18,291,812 |
| Dimensao_Tempo | 201,576 |
| Dimensao_Localidade | 675 |
| Dimensao_Estacao | 472 |
| Dimensao_Fonte | 2 |

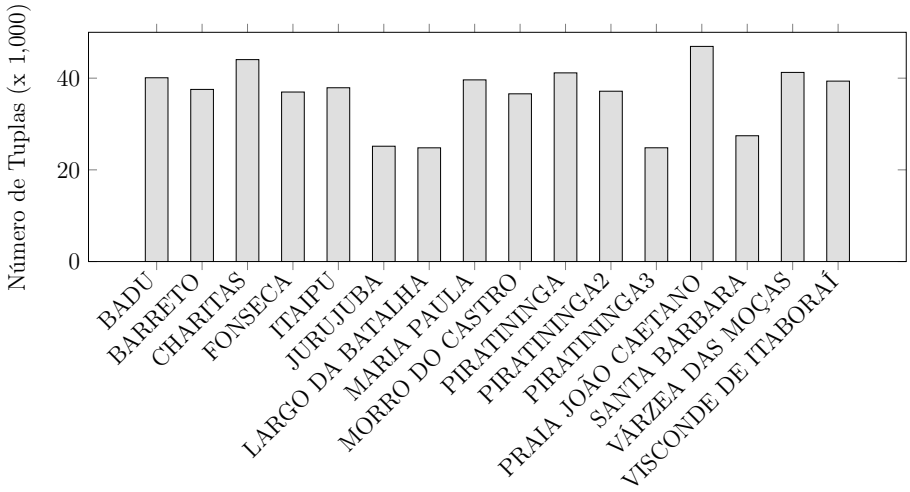


Figura 3.4: Número de Tuplas por Estação na Cidade de Niterói

## 3.5 Consultas Geradas na TEMPO

Por meio da coleta de metadados gerados a partir das interações do usuário com a TEMPO, foi identificada uma série de consultas frequentemente submetidas ao banco de dados. A Tabela 3.2 apresenta as 14 consultas frequentes e que foram utilizadas no estudo comparativo entre as abordagens.

As consultas foram classificadas de acordo com as classes definidas por Doraiswamy e Freire [11] e previamente apresentadas na Seção 2.2 desta dissertação. Embora existam diversas outras consultas submetidas durante a interação dos usuários com a TEMPO, essas 14 foram selecionadas devido à sua heterogeneidade e por englobarem a maioria das classes mencionadas anteriormente.

As consultas Q1, Q2 e Q3 são consultas do tipo C1. A consulta Q1 tem como objetivo executar uma *Seleção* de uma série temporal com base em uma data. As consultas Q2 e Q3 exploram a *Seleção* com base no índice pluviométrico. A diferença entre as consultas Q2 e Q3 é a seletividade de cada uma delas. A Q2 retorna menos tuplas que a Q3, portanto, requer menos *CPU* e Entrada/Saída.

As consultas Q4, Q5 e Q6 são consultas da classe C4 (*Vizinho mais Próximo*). Essa classe tem por objetivo retornar dados de todas as estações dentro de um raio de 10 Km a partir de um ponto específico no espaço. As consultas Q5 e Q6 também retornam dados de todas as estações dentro de um raio de 10 km de um ponto no espaço, mas filtram tais dados com base no índice pluviométrico e data, respectivamente. A consulta Q7 também retorna dados de todas as estações em um raio de 10 km, e, além disso, filtra os dados se

Tabela 3.2: Consultas geradas através da interação do usuário com a TEMPO.

| Consulta | Descrição | Classe | # Tuplas |
|---|---|---|---|
| Q1 | Retornar a série temporal de precipitações completa em 2018 | C1 | 2.211.196 |
| Q2 | Retornar a série temporal de precipitações completa para todas as estações cujo índice pluviométrico é maior que 25.00 (*i.e., outliers*) | C1 | 5.457 |
| Q3 | Retornar a série temporal de precipitações completa para todas as estações cujo índice pluviométrico é maior que 1.50 | C1 | 281.924 |
| Q4 | Retornar a série temporal de precipitações para as estações a uma distância de 10 km de um ponto específico | C4 | 688.653 |
| Q5 | Retornar a série temporal de precipitações para as estações a uma distância de 10 km de um ponto específico cujo índice pluviométrico é maior que 5.00 | C1,C4 | 2.646 |
| Q6 | Retornar a série temporal de precipitações para as estações a uma distância de 10 km de um ponto específico em 08/04/2017 | C1,C4 | 408 |
| Q7 | Retornar a série temporal de precipitações para as estações a uma distância de até 10 km de um ponto específico e a data do registro está entre 11/15/2018 e 11/30/2018 | C1, C4 | 5.105 |
| Q8 | Retornar o índice pluviométrico mais recente para a estação mais próxima de um ponto específico | C1, C3, C4 | 1 |
| Q9 | Retornar o índice pluviométrico para as 10 estações mais próximas de um ponto específico | C1, C4 | 10 |
| Q10 | Retornar o índice pluviométrico para as 20 estações mais próximas de um ponto específico | C1, C4 | 20 |
| Q11 | Retornar a média do índice pluviométrico para as 10 estações mais próximas de um ponto específico | C3, C4 | 10 |
| Q12 | Retornar a série temporal de precipitações para todas as estações que se encontram no polígono da região do Ingá, em Niterói | C1, C2 | 46.928 |
| Q13 | Retornar a série temporal de precipitações para todas as estações que se encontram no polígono da região do Ingá, em Niterói, e a data é entre 01/01/2019 e 06/30/2019 | C1, C2 | 4.303 |
| Q14 | Retornar a série temporal de precipitações para todas as estações que se encontram no polígono da região do Ingá, em Niterói, e o índice pluviométrico é igual ou maior que 2.00 | C1, C2 | 536 |

baseando em um intervalo de tempo.

A consulta Q8 seleciona o índice pluviométrico mais recente (que envolve uma ordenação dos dados) armazenado para a estação mais próxima de um ponto específico no espaço. As consultas Q9 e Q10 também são consultas do tipo *Vizinho Mais Próximos*, e a diferença entre essas consultas é seu valor $K$.

A consulta Q9 retorna dados relacionados às 10 estações mais próximas de um ponto específico no espaço, a Q10 retorna dados relacionados às 20 estações mais próximas. A consulta Q11 é formada por meio de uma combinação entre as características de uma consulta da classe *Vizinho Mais Próximo* e uma consulta de *Agregação*, pois considera as 10 estações mais próximas, mas também calcula a médica do índice pluviométrico.

As consultas Q12, Q13 e Q14 são consultas da classe *Junção*. Nessas três consultas são realizadas junções espaciais, e consideramos a região do bairro do Ingá (uma região perto do *campus* da UFF) na cidade de Niterói, que está representada na Figura 3.5. A Q12 retorna a série temporal completa de chuvas das estações que se encontram dentro do polígono que representa o bairro do Ingá. As consultas Q13 e Q14 também retornam a série temporal de chuvas das estações pluviométricas dentro do polígono que representa o bairro do Ingá, mas também filtram os dados com base em uma data e índice pluviométrico, respectivamente.

Figura 3.5: Região do Bairro do Ingá na Cidade de Niterói.

# Capítulo 4

# Estudo Comparativo das Abordagens de Gerência de Dados

Este capítulo tem como objetivo apresentar o estudo comparativo das diferentes abordagens de gerência de dados no processamento das 14 consultas previamente apresentadas na Seção 3. A Seção 4.1 descreve as configurações do ambiente utilizado nos testes. A Seção 4.2 descreve as configurações do experimento. Finalmente, a Seção 4.3 discute os resultados obtidos.

## 4.1 *Setup* do Ambiente

Para a realização dos experimentos, cada abordagem de gerência de dados (*i.e.*, PostgreSQL, MonetDB, Apache Spark, Apache Drill e Elasticsearch) foi instalada no ambiente da Amazon AWS. A Amazon AWS é um dos provedores de nuvem mais populares e muitas aplicações científicas e comerciais já se encontram implantadas nele. A Amazon AWS fornece diversos tipos diferentes de máquinas virtuais para os usuários instanciarem e utilizarem. Cada uma delas possui características únicas, tais como tipo de CPU, quantidade de memória RAM e capacidade de armazenamento. Existem vários tipos de máquinas virtuais, como a a1.medium (1 vCPU, 2 GiB de RAM), a c6g.2xlarge (8 vCPU, 16 GiB de RAM), a r4.2xlarge (8 vCPUs, 64 GiB de RAM) e a d2.4xlarge (16 vCPUs, 122 GiB de RAM). Nos experimentos apresentados nesta dissertação foram considerados os tipos m5dn.large, c5.xlarge e r4.xlarge da Amazon.

As máquinas virtuais da categoria M5 oferecem recursos balanceados de CPU, memória e rede para diferentes tipos de *workloads*, incluindo bancos de dados médios e grandes. Uma máquina virtual do tipo m5dn.large possui 8 vCPUs, 16 GiB de RAM e é alimen-

tada com CPU Intel(R) Xeon(R) Platinum 8259CL @ 2,50 GHz (2) e 8 GB de RAM. As máquinas virtuais da classe R4 são otimizadas para aplicações com uso intensivo de memória, bancos de dados de alto desempenho, mineração e análise de dados, entre outros. Uma máquina virtual do tipo r4.xlarge possui 4 vCPUs, 30,5 GiB de RAM e é alimentada com CPU Intel Xeon E5-2686 v4 de 2.3 GHz.

As máquinas virtuais da categoria C5 são otimizadas para *workloads* com uso computacional intenso e oferecem um alto desempenho devido ao tipo de processador utilizado. Uma máquina virtual do tipo c5.xlarge possui 4 vCPUs, 8 GiB de RAM e é alimentado com CPU escalável Intel Xeon de 2ª geração (*Cascade Lake*) com uma frequência turbo de todos os núcleos de 3.6GHz e frequência turbo de núcleo único de até 3,9 GHz. Todas as máquinas virtuais possuem 105 GB de disco e utilizam o CentOS Linux 7 (Core) como Sistema operacional. De acordo com a Amazon, todas as máquinas virtuais foram instanciadas no Leste dos EUA (Norte da Virgínia) e seguem as regras de preços dessa localidade.

## 4.2 *Setup* do Experimento

Os experimentos foram realizados nas cinco abordagens mencionadas na Seção 2.3 usando as seguintes versões: Apache Drill versão 1.19.0, Apache Spark versão 3.2.0, Elasticsearch versão 7.15.1, MonetDB versão 11.41. 11 e PostgreSQL versão 9.2.24. Para todas as abordagens o uso de cache foi desabilitado. Todos os índices possíveis foram criados em cada abordagem. Especialmente no Apache Spark, transformamos o *DataFrame* em um *Resilient Distributed Dataset* (RDD), que é um conjunto particionado de elementos que podem ser processados em paralelo pelo Spark [57]. Essa transformação em um RDD permite utilizar o método *RDD.ZipWithIndex* para criar o índice sobre o RDD. Depois de criar o índice, convertemos o RDD em um *DataFrame* novamente. Cada consulta apresentada na Tabela 3.2 foi executada 21 vezes, e a primeira execução não é considerada na avaliação. Os dados usados nos experimentos apresentados neste artigo podem ser obtidos em `https://osf.io/pa2zn/`. Os resultados apresentados nesse capítulo são descritos por Nascimento *et al.* [41, 43].

## 4.3 Discussão dos Resultados

As Tabelas 4.1, 4.2 e 4.3 apresentam o tempo médio de execução (*i.e.*, $\overline{x}$), o desvio padrão (*i.e.*, $\sigma$) e a variância (*i.e.*, $\hat{V}$) de cada consulta descrita na Tabela 3.2 para as máquinas virtuais m5dn.large, c5.xlarge e r4.xlarge, respectivamente. É possível observar que o Apache Spark apresentou o melhor desempenho em 11 das 14 consultas, seguido pelo MonetDB que, por sua vez, apresentou o melhor desempenho em 3 das 14 consultas, independente do tipo de máquina utilizada.

A fim de entender o comportamento do Apache Spark e seu bom desempenho na grande maioria das consultas, é necessário entender como o Spark processa as consultas em SQL. Inicialmente, a consulta SQL é traduzida para uma Árvore Sintática Abstrata (*Abstract Syntax Tree (AST)*). As regras de otimização podem ser aplicadas às operações usando o *Catalyst*, o otimizador do Spark. Cada expressão da AST gerada (e eventualmente otimizada) é transformada em uma AST implementada em Scala. Em seguida, o código é compilado em *Bytecode* Java, que é enviado para os Spark *Executors* (*i.e.*, os processos que de fato processam as consultas). Quando o código está disponível, o Spark cria um plano lógico de *Resilient Distributed Datasets* (RDDs). O Spark transforma esse plano lógico em um plano de execução física composto por etapas e tarefas paralelas. O plano físico pode ser executado em paralelo em diferentes *Executors*, consumindo partições do RDD. Dessa forma, o Spark pode se beneficiar do paralelismo, mesmo em uma única máquina virtual com vários vCPUS, enquanto as outras abordagens não (embora o MonetDB tenha uma versão com suporte paralelo, essa versão não é utilizada nos experimentos apresentados nesta dissertação).

Também é importante ressaltar que o `DW-TEMPO` possui aproximadamente 4,8 GB (a tabela fato contém 18.291.812 tuplas) e cabe quase inteiramente na memória das máquinas virtuais escolhidas para esse experimento. Assim, o Spark pode realizar praticamente todo o processamento em memória usando *DataFrames*, o que torna a execução mais eficiente em comparação com outras abordagens que precisam acessar o disco. Outra abordagem que procura manter os dados o máximo em memória é o MonetDB. Como esperado, o MonetDB também se mostra uma abordagem interessante para o processamento das consultas.

O MonetDB apresentou os melhores tempos de processamento para as consultas Q2, Q5 e Q6. Apesar de não executar os experimentos de forma paralela, os tempos de execução das consultas Q1 a Q11 (exceto o caso da consulta Q4) em todas as máquinas

virtuais são aceitáveis para o cenário de *Vis* interativa (*i.e.*, resultado da consulta em um tempo inferior a 0,5 segundos). A vantagem do MonetDB não é apenas a leitura mais rápida, mas ele também se beneficia de sua tecnologia de banco de dados colunar, que carrega na memória apenas os atributos necessários para a consulta. O desempenho do MonetDB para as consultas Q12 e Q14 apresentou uma sobrecarga não desprezível devido às múltiplas junções com o polígono que representa a região do bairro do Ingá. O Elasticsearch também apresentou resultados aceitáveis para as consultas Q5 a Q14 (exceto Q12) em todas as máquinas virtuais, mas o desempenho do ElasticSearch para as consultas Q12 também apresentou uma sobrecarga, semelhante ao MonetDB. Além disso, no Elasticsearch, consultas grandes podem consumir muita memória durante a fase de análise e, nesse caso, o mecanismo SQL do Elasticsearch também pode adicionar uma sobrecarga. É importante ressaltar que máquinas virtuais com menos de 4GiB de memória RAM fazem com que o uso do Elasticsearch se torne inviável.

Diferentemente do Apache Spark, do MonetDB e do Elasticsearch, tanto o Apache Drill quanto o PostgreSQL apresentaram tempos de execução superiores a 0,5 segundos para a grande maioria das consultas. Mesmo com a criação de índices sempre que possível, *e.g.*, nas chaves estrangeiras da tabela fato, em atributos que são usados para seleções (*e.g.*, "ano = 2018"), os tempos de execução são maiores que os 0,5 segundos esperados. No caso do Apache Drill, o mesmo interpreta a consulta enviada em um plano distribuído, a envia para todos os processos de *drillbit*, que pesquisam as fontes de dados, acessam os dados usando os conectores, executam a consulta, retornam os resultados para o primeiro nó para agregação e, em seguida, envia o resultado final. Dependendo da fonte de dados, o Drill pode precisar carregar todos os dados e filtrá-los separadamente, o que requer tempo de execução adicional. Uma opção para reduzir as despesas gerais é usar o formato *Parquet* em vez de carregar dados de um arquivo CSV.

Após sumarizar os resultados obtidos, podemos afirmar que o Apache Spark e o MonetDB se mostraram como as abordagens recomendadas para processar consultas na `TEMPO`. Vale ressaltar que esse resultado é justificado uma vez que a `TEMPO` normalmente submete consultas OLAP, e os dados não são atualizados e/ou excluídos após o carregamento (já que na `TEMPO` os dados são sempre carregados após serem capturados pelos sensores). Consultas OLAP são benéficas para o MonetDB e para o Apache Spark. Outro resultado adquirido a partir dos experimentos é que os sistemas *Polystore*, como o Apache Drill, não são adequados para cenários em que o usuário possui um conjunto de dados médio/pequeno armazenado em uma única (ou em poucas) máquina(s). Nesse caso, a maioria das abordagens existentes apresentará um desempenho melhor que o Drill.

Tabela 4.1: Desempenho das Consultas Q1 a Q14 na máquina virtual m5dn.large.

| Query | Apache Drill | Apache Spark | ElasticSearch | MonetDB | PostgreSQL |
|---|---|---|---|---|---|
| | $\overline{x}$ | $\overline{x}$ | $\overline{x}$ | $\overline{x}$ | $\overline{x}$ |
| Q1 | 21664,20 | 36,10 | 37244,85 | 342,15 | 11158,35 |
| Q2 | 6371,30 | 39,10 | 488,45 | 21,50 | 1292,50 |
| Q3 | 7845,55 | 40,80 | 23909,50 | 66,40 | 2629,85 |
| Q4 | 11638,20 | 49,20 | 68089,55 | 2278,70 | 5347,30 |
| Q5 | 5989,95 | 70,75 | 280,55 | 37,05 | 1286,80 |
| Q6 | 5831,15 | 43,70 | 64,00 | 26,20 | 1144,00 |
| Q7 | 5964,90 | 71,65 | 417,05 | 150,15 | 1334,60 |
| Q8 | 6478,50 | 32,70 | 489,30 | 239,15 | 2002,90 |
| Q9 | 6228,65 | 70,40 | 490,60 | 239,50 | 2001,10 |
| Q10 | 6167,25 | 40,55 | 491,15 | 239,90 | 2010,25 |
| Q11 | 7465,00 | 125,25 | 250,12 | 253,80 | 25529,95 |
| Q12 | 8766,50 | 86,85 | 3923,50 | 828,85 | 1540,80 |
| Q13 | 43625,85 | 63,25 | 368,40 | 7201,85 | 1381,00 |
| Q14 | 7281,20 | 47,80 | 54,40 | 760,55 | 2410,10 |

Tabela 4.2: Desempenho das Consultas Q1 a Q14 na máquina virtual c5.xlarge.

| Query | Apache Drill | Apache Spark | ElasticSearch | MonetDB | PostgreSQL |
|---|---|---|---|---|---|
| | $\overline{x}$ | $\overline{x}$ | $\overline{x}$ | $\overline{x}$ | $\overline{x}$ |
| Q1 | 18079,95 | 20,55 | 32739,40 | 249,15 | 9173,60 |
| Q2 | 4449,00 | 20,65 | 428,40 | 12,05 | 1119,15 |
| Q3 | 6300,90 | 13,25 | 20441,50 | 44,45 | 2255,95 |
| Q4 | 10392,75 | 33,20 | 57483,10 | 1054,60 | 4573,35 |
| Q5 | 5025,25 | 21,20 | 243,00 | 20,10 | 1115,80 |
| Q6 | 4748,55 | 33,15 | 61,40 | 15,10 | 989,45 |
| Q7 | 4832,80 | 26,40 | 352,05 | 91,85 | 1153,55 |
| Q8 | 4980,45 | 21,75 | 390,65 | 108,80 | 1749,05 |
| Q9 | 4913,75 | 34,05 | 389,30 | 108,50 | 1752,35 |
| Q10 | 4910,30 | 24,40 | 395,35 | 108,65 | 1756,85 |
| Q11 | 5579,80 | 28,25 | 2,95 | 115,70 | 22128,45 |
| Q12 | 6706,25 | 44,40 | 3404,90 | 451,70 | 1323,95 |
| Q13 | 37689,40 | 39,15 | 308,95 | 4531,20 | 1191,30 |
| Q14 | 5623,00 | 32,75 | 46,05 | 375,40 | 2087,30 |

Tabela 4.3: Desempenho das Consultas Q1 a Q14 na máquina virtual r4.xlarge.

| Query | Apache Drill | Apache Spark | ElasticSearch | MonetDB | PostgreSQL |
|---|---|---|---|---|---|
| | $\overline{x}$ | $\overline{x}$ | $\overline{x}$ | $\overline{x}$ | $\overline{x}$ |
| Q1 | 24164,90 | 28,00 | 42095,35 | 281,55 | 13955,00 |
| Q2 | 6073,05 | 32,25 | 579,10 | 17,95 | 1512,65 |
| Q3 | 8760,55 | 21,30 | 28002,10 | 53,35 | 3160,65 |
| Q4 | 12826,30 | 28,90 | 77948,65 | 1304,60 | 5920,15 |
| Q5 | 6229,30 | 55,75 | 332,45 | 28,70 | 1510,75 |
| Q6 | 6098,15 | 34,95 | 77,75 | 20,85 | 1391,30 |
| Q7 | 6282,95 | 27,45 | 477,65 | 129,85 | 1640,50 |
| Q8 | 6760,90 | 50,65 | 495,45 | 139,50 | 2462,30 |
| Q9 | 6640,50 | 37,05 | 496,25 | 140,30 | 2449,20 |
| Q10 | 6605,25 | 29,65 | 495,95 | 140,70 | 2450,70 |
| Q11 | 7928,80 | 55,50 | 5,05 | 150,10 | 30166,55 |
| Q12 | 9793,80 | 51,80 | 4583,70 | 525,35 | 1850,40 |
| Q13 | 55005,45 | 50,50 | 424,35 | 6402,15 | 1688,25 |
| Q14 | 8149,50 | 42,00 | 67,05 | 454,95 | 2809,00 |

# Capítulo 5

# Trabalhos Relacionados

Este capítulo tem como objetivo apresentar os trabalhos relacionados ao estudo comparativo realizado nesta dissertação. A Seção 5.1 apresenta os trabalhos existentes que propõem o uso de técnicas OLAP e de visualização para dados pluviométricos. A Seção 5.2 apresenta estudos comparativos de abordagens de gerência de dados em geral.

## 5.1 Gerência de Dados Pluviométricos

Na literatura são encontradas outras abordagens que propõem o uso de técnicas OLAP e de visualização para dados pluviométricos. Salas *et al.* [51] propõem uma infraestrutura que tem como objetivo realizar a fusão de múltiplas fontes de dados heterogêneos através de uma arquitetura de dados abertos. A abordagem coleta dados hidrológicos de múltiplas fontes e realiza o processamento por meio de um sistema de *workflow* (VisTrails), no qual o usuário é capaz de personalizar *scripts* para processamento dos dados. Diferentemente da `TEMPO`, a abordagem proposta por Salas *et al.* [51] não considera o uso de *Data Warehouses* para manter dados pré-agregados nem disponibiliza estratégias de visualização de dados espaciais.

A abordagem proposta por Morais e Ferreira [38], por sua vez, apresenta um banco de dados para armazenar informações sobre o comportamento do índice pluviométrico no Estado de Goiás. Os dados pluviométricos foram coletados na ANA (Agência Nacional de Águas) e os dados de imagens foram coletados através do *Tropical Rainfall Measuring Mission* (TRMM), controlado pela NASA. Embora os autores explicitem a metodologia utilizada para elaborar o trabalho, o *schema* do banco de dados não foi apresentado. Além disso, não foi explicado se a abordagem proposta considera dados advindos de múltiplas fontes. Esplugues *et al.* [19] propõem a aplicação de técnicas de mineração de dados

pluviométricos para analisar os períodos de seca na Espanha. Os dados pluviométricos utilizados nessa abordagem foram coletados através da AEMET (Agência Estatal de Meteorología) e foi utilizado o SPI (*Standard Precipitation Index*), comumente utilizado para o monitoramento de condições associadas a secas e excesso de chuva. Apesar dos autores terem explicado os métodos utilizados para elaboração do trabalho, assim como no caso da abordagem de Morais e Ferreira, o banco de dados e a ferramenta de ETL utilizados para o desenvolvimento não foram discutidos, o que limita o entendimento da proposta apresentada.

## 5.2 Estudo Comparativo de Abordagens de Gerência de Dados

Comparar o desempenho de SGBDs e soluções de gerência de dados com objetivos específicos é uma tarefa conhecida na comunidade de Banco de Dados. Existem diversos *benchmarks* que podem ser usados para avaliar SGBDs novos e existentes, *e.g.*, TPC [47] (que é especializado em TPC-C para aplicações OLTP, TPC- H [46] para aplicações OLAP, TPCx-HS [32] para aplicações *Big Data*, TPC-DS [45] para aplicações de apoio à decisão, TPC-E para aplicações OLTP, TPC-DI para integração de dados, TPCx-V para bancos de dados virtualizados) SPEC [5], *etc.* No entanto, *benchmarks* como SPEC e TPC não são projetados para avaliar soluções de gerência de dados para sistemas de *Vis* interativa, pois não consideram consultas de banco de dados geradas por meio de interações do usuário.

Recentemente, alguns esforços estão sendo feitos para gerar *benchmarks* para Vis interativa [4]. Embora o *benchmark* proposto por Battle *et al.* [4] represente um passo à frente, eles se concentram em bancos de dados relacionais. O trabalho de Makris *et al.* [33] também compara dois SGBDs diferentes para aplicações espaço-temporais, mas considera apenas dois tipos de SGBDs (relacional e orientado a documentos) e não considera consultas geradas por meio de interações do usuário. Eichmann *et al.* [15] propõem um *benchmark* chamado IDEBench para avaliar sistemas de Vis interativa. Os autores propõem uma série de consultas sintéticas a serem executadas no SGBD MonetDB orientado a colunas.

Em paralelo, outras abordagens se preocupam em fornecer estruturas para realizar consultas em Vis interativas de maneira eficiente. Muitas dessas estruturas são especializadas em apoiar esse tipo de Vis, como os *Nano Cubes* [29]. Os *Nano Cubes* são estruturas

em memória que podem ser usadas para gerar visualizações conhecidas como mapas de calor, histogramas, e gráficos de coordenadas paralelas. A grande vantagem dos *Nano Cubes* em comparação com os *Data Cubes* tradicionais é que eles podem armazenar até bilhões de dados sem usar o disco, enquanto os *Data Cubes* [3] gerenciam um conjunto de dados muito menor. No entanto, tais estruturas devem ser reconstruídas para cada coleção de atributos e tipos de agregação exigidos pela aplicação (*ex.*, *SUM*, *AVG*, *etc.*), o que pode ser muito trabalhoso. Além disso, como os dados já estão pré-agregados em memória, qualquer consulta que requeira acesso aos dados em uma granularidade mais fina não pode ser realizada. Por fim, o tamanho da estrutura cresce exponencialmente de acordo com o número de atributos considerados. Diferente de Lins *et al.* [29] e Battle *et al.* [3] que optaram por estruturas especializadas em memória para apoiar sistemas de Vis interativa, o trabalho de Jiang *et al.* [25] propõe que tais sistemas usem SGBDs como uma abordagem de gerenciamento de dados. No artigo, os autores analisam o tempo de resposta de consultas espaço-temporais no SGBD PostgreSQL.

# Capítulo 6

# Conclusão e Trabalhos Futuros

Com a popularização da área de Ciência de Dados, a área de Vis Interativa tem ganhado muito destaque. Diversas técnicas de Vis Interativas têm sido utilizadas no que diz respeito a grandes quantidades de dados (*Big Data*). Essas técnicas já foram aplicadas em múltiplos segmentos, *e.g.*, planejamento urbano, saúde, entre outros. Entretanto, o desafio é aplicar tais técnicas de modo que o tempo de resposta da aplicação de Vis Interativa não interfira na experiência cognitiva do usuário. Segundo Liu *et al.* [30], os dados devem ser acessados, filtrados e agregados no máximo em 0,5 segundos (500ms). Tempos de resposta acima do valor esperado já é suficiente para interferir na experiência do usuário e, consequentemente, na tomada de decisão. A abordagem de gerência de dados que está acoplada ao mecanismo de Vis interativa pode desempenhar um papel chave no desempenho da visualização, uma vez que as consultas submetidas podem ser complexas e demorar mais que os 0,5 segundos esperados.

Uma aplicação que se beneficia fortemente de técnicas de Vis interativa é o acompanhamento e análise de índices pluviométricos em grandes centros urbanos. Os desastres naturais têm ocorrido com uma frequência cada vez maior ao redor do planeta, inclusive em diversos estados do Brasil. Tais desastres podem ter diferentes origens, e dependendo de sua intensidade, são capazes de causar tragédias e provocar inúmeras mortes. Nos centros urbanos as consequências desses eventos podem ser extremamente severas devido à falta de estrutura nas cidades para suportar tal impacto. Para reduzir o impacto das chuvas, um monitoramento constante dos índices pluviométricos é aconselhável para que seja possível tomar decisões antes que a situação se agrave. Esse monitoramento é possível de ser realizado já que dados sobre as chuvas são computados e disponibilizados publicamente por múltiplas organizações no Brasil. Apesar disso, o processo de captura e transformação dos dados não é trivial, conforme mencionado anteriormente. Além disso,

somente capturar os dados e visualizá-los em formato tabular pode não ser muito eficiente, principalmente com o aumento constante da quantidade de dados. Entretanto, mesmo que seja desenvolvido um sistema para apoio que use técnicas de Vis interativa, caso a abordagem de gerência de dados acoplada não seja a ideal, o mesmo pode sofrer com *delays* desnecessários.

Nesta dissertação realizamos um estudo comparativo do desempenho de cinco abordagens de gerência de dados e utilizamos o cenário de análise de dados pluviométricos como estudo de caso. Para que fosse possível realizar o estudo comparativo e avaliar o desempenho das abordagens, desenvolvemos a ferramenta `TEMPO` [42]. A `TEMPO` tem como objetivo capturar dados pluviométricos de múltiplas fontes, aplicar as transformações necessárias para deixá-los com a mesma granularidade e integrá-los em um único repositório. O `Portal TEMPO`, por sua vez, fornece mecanismos de visualização de dados espaciais para o usuário. Para a realização dos testes, 14 consultas foram selecionadas e os critérios de escolha foram sua heterogeneidade e por englobarem a maioria das classes propostas por Doraiswamy e Freire [11].

O estudo comparativo realizado considerou cinco soluções de gerência de dados para escolha a partir da submissão da consulta: (i) Apache Spark, (ii) Apache Drill, (iii) MonetDB, (iv) ElasticSearch e (v) PostgreSQL. Observamos que das 14 consultas consideradas, o Apache Spark e o MonetDB foram as abordagens de gerência de dados que apresentaram melhor desempenho, sendo capazes de processar grande parte das consultas em menos de 0,5 segundos em todos os tipos de máquinas virtuais utilizadas. Uma das vantagens do Spark SQL e que justifica esse resultado é o fato dele realizar o processamento em memória de forma paralela, aproveitando a estrutura de *DataFrames* do Spark. O MonetDB também foi capaz de processar a maioria das consultas em menos de 0,5 segundos, mas apresentou uma perda de desempenho nas consultas envolvendo junções espaciais.

Como trabalhos futuros, pretendemos incluir no estudo comparativo a avaliação de novas abordagens de gerência de dados, como sistemas *Polystore (e.g.,* BigDAWG) e estruturas especializadas para representar dados espaço-temporais (*e.g., Nano Cubes*). Pretendemos estudar o comportamento do Apache Spark, particularmente, para entender como será o seu comportamento quando a base de dados não couber somente em memória e for necessário utilizar parte do disco para realizar o processamento. Além disso, queremos explorar melhor as classes de consultas descritas por Doraiswamy e Freire [11], principalmente a classe de Consultas Geométricas (C5), que não foi abordada nesta disser-

tação. Por fim, pretendemos utilizar outros estudos de caso para observar o desempenho de cada abordagem de gerência de dados com diferentes tipos de dados.

# Referências


[1] Armbrust, M.; Bateman, D.; Xin, R.; Zaharia, M. Introduction to spark 2.0 for database researchers. In *SIGMOD '16* (2016), p. 2193–2194.

[2] Armbrust, M.; Xin, R. S.; Lian, C.; Huai, Y.; Liu, D.; Bradley, J. K.; Meng, X.; Kaftan, T.; Franklin, M. J.; Ghodsi, A.; Zaharia, M. Spark sql: Relational data processing in spark. In *SIGMOD* (New York, NY, USA, 2015), p. 1383–1394.

[3] Battle, L.; Chang, R.; Stonebraker, M. Dynamic prefetching of data tiles for interactive visualization. In *SIGMOD* (2016), ACM, pp. 1363–1375.

[4] Battle, L.; Eichmann, P.; Angelini, M.; Catarci, T.; Santucci, G.; Zheng, Y.; Binnig, C.; Fekete, J.-D.; Moritz, D. Database benchmarking for supporting real-time interactive querying of large data. In *SIGMOD* (New York, NY, USA, 2020), p. 1571–1587.

[5] Bays, W.; Lange, K. SPEC: driving better benchmarks. In *Third Joint WOSP/-SIPEW International Conference on Performance Engineering, ICPE'12, Boston, MA, USA - April 22 - 25, 2012* (2012), D. R. Kaeli, J. Rolia, L. K. John, and D. Krishnamurthy, Eds., ACM, pp. 249–250.

[6] Beigbeder, T.; Coughlan, R.; Lusher, C.; Plunkett, J.; Agu, E.; Claypool, M. The effects of loss and latency on user performance in unreal tournament 2003®. In *Proceedings of the 3rd Workshop on Network and System Support for Games, NETGAMES 2004, Portland, Oregon, USA, August 30, 2004* (2004), W. Feng, Ed., ACM, pp. 144–151.

[7] Boncz, P. A.; Manegold, S.; Rittinger, J. Updating the pre/post plane in monetdb/xquery. In *XIME-P* (2005), D. Florescu and H. Pirahesh, Eds.

[8] Caban, J. J.; Gotz, D. Visual analytics in healthcare–opportunities and research challenges, 2015.

[9] CEMADEN. Centro nacional de monitoramento e alertas de desastres naturais - http://www.cemaden.gov.br/apresentacao/, 2021.

[10] Ciesielska, M.; Rizun, N.; Janowski, T. Interdisciplinarity in smart sustainable city education: exploring educational offerings and competencies worldwide. In *54th Hawaii International Conference on System Sciences, HICSS 2021, Kauai, Hawaii, USA, January 5, 2021* (2021), ScholarSpace, pp. 1–10.

[11] Doraiswamy, H.; Freire, J. A gpu-friendly geometric data model and algebra for spatial queries. In *SIGMOD* (New York, NY, USA, 2020), p. 1875–1885.

[12] dos Reis, J. B. C.; Rennó, C. D.; Lopes, E. S. S. Validation of satellite rainfall products over a mountainous watershed in a humid subtropical climate region of brazil. *Remote. Sens. 9*, 12 (2017), 1240.

[13] dos Santos, J. R.; Franco, E. F.; Carvalho, H. C.; Armenia, S.; Pompei, A.; Medaglia, C. M. Water used to be infinite: a brazilian tale of climate change. *Kybernetes 48*, 1 (2019), 143–162.

[14] Drill, A. Apache drill - https://drill.apache.org/docs/drill-introduction/, 2022.

[15] Eichmann, P.; Zgraggen, E.; Binnig, C.; Kraska, T. Idebench: A benchmark for interactive data exploration. In *SIGMOD* (New York, NY, USA, 2020), p. 1555–1569.

[16] Elastic. Elasticsearch docs - https://bit.ly/3l7oz4l, 2021.

[17] Elastic. Elasticsearch intro - https://bit.ly/3fkokj4, 2021.

[18] Elastic. Elasticsearch search and analyze - https://bit.ly/3lbnmnp, 2021.

[19] Esplugues, F. B.; Gramaje, M. d. C. P.; García-Haro, F. J. Técnicas de minería de datos para el análisis de periodos de sequía en españa. *Revista Tiempo y Clima 5*, 30 (2013).

[20] Han, J.; Kamber, M.; Pei, J. *Data mining concepts and techniques.* Elsevier, 2012.

[21] Hausenblas, M.; Nadeau, J. Apache drill: interactive ad-hoc analysis at scale. *Big data 1*, 2 (2013), 100–104.

[22] Hochstetler, J.; Hochstetler, L.; Fu, S. An optimal police patrol planning strategy for smart city safety. In *18th IEEE International Conference on High Performance Computing and Communications; 14th IEEE International Conference on Smart City; 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, Sydney, Australia, December 12-14, 2016* (2016), J. Chen and L. T. Yang, Eds., IEEE Computer Society, pp. 1256–1263.

[23] Idreos, S.; Groffen, F.; Nes, N.; Manegold, S.; Mullender, S.; Kersten, M. Monetdb: Two decades of research in column-oriented database. *IEEE Data Engineering Bulletin* (2012).

[24] Inmon, W.; Welch, J.; Glassey, K. *Building the Data Warehouse.* Sons Inc, New York, 2005.

[25] Jiang, L.; Rahman, P.; Nandi, A. Evaluating interactive data systems: Workloads, metrics, and guidelines. In *SIGMOD* (New York, NY, USA, 2018), p. 1637–1644.

[26] Jr., A. L. X.; Bonatti, D.; Celaschi, S. Rain gauge simulator and first tests with a new mobile climate alert system in brazil. *J. Braz. Comput. Soc. 22*, 1 (2016), 2:1–2:14.

[27] Kimball, R.; Ross, M. *The Data Warehouse Toolkit: The complete guide to dimensional modeling.* Wiley, New York, 2002.

[28] Kranas, P.; Kolev, B.; Levchenko, O.; Pacitti, E.; Valduriez, P.; Jiménez-Peris, R.; Patiño-Martínez, M. Parallel query processing in a polystore. *Distributed Parallel Databases 39*, 4 (2021), 939–977.

[29] Lins, L. D.; Klosowski, J. T.; Scheidegger, C. E. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE TVCG 19*, 12 (2013), 2456–2465.

[30] Liu, Z.; Heer, J. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics 20*, 12 (2014), 2122–2131.

[31] Lu, G. Y.; Wong, D. W. An adaptive inverse-distance weighting spatial interpolation technique. *Computers & geosciences 34*, 9 (2008), 1044–1055.

[32] Magdon-Ismail, T. Tpcx-hs. In *Encyclopedia of Big Data Technologies*, S. Sakr and A. Y. Zomaya, Eds. Springer, 2019.

[33] Makris, A.; Tserpes, K.; Spiliopoulos, G.; Zissis, D.; Anagnostopoulos, D. Mongodb vs postgresql: A comparative study on performance aspects. *GeoInformatica 25*, 2 (2021), 243–268.

[34] Miranda, F.; Doraiswamy, H.; Lage, M.; Zhao, K.; Gonçalves, B.; Wilson, L.; Hsieh, M.; Silva, C. T. Urban pulse: Capturing the rhythm of cities. *IEEE Trans. Vis. Comput. Graph. 23*, 1 (2017), 791–800.

[35] Miranda, F.; Hosseini, M.; Lage, M.; Doraiswamy, H.; Dove, G.; Silva, C. T. Urban mosaic: Visual exploration of streetscapes using large-scale image data. In *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020* (2020), R. Bernhaupt, F. F. Mueller, D. Verweij, J. Andres, J. McGrenere, A. Cockburn, I. Avellino, A. Goguey, P. Bjøn, S. Zhao, B. P. Samson, and R. Kocielnik, Eds., ACM, pp. 1–15.

[36] Mizutori, M.; Guha-Sapir, D. Human cost of disasters 2000-2019. Tech. rep., United Nations Office for Disaster Risk Reduction, 2020.

[37] MonetDB. Monetdb - https://www.monetdb.org/documentation-jul2021/user-guide/introduction-to-monetdb/key-concepts/, 2021.

[38] Morais, L. d.; Ferreira, N. C. Banco de dados pluviométricos integrados por dados do sensor trmm e estações pluviométricas no estado de goiás. *Anais Eletrônicos 17* (2015).

[39] Munzner, T. *Visualization analysis and design.* CRC press, 2014.

[40] Nascimento, J. G.; Althoff, D.; Bazame, H. C.; Neale, C. M. U.; Duarte, S. N.; Ruhoff, A. L.; Gonçalves, I. Z. Evaluating the latest IMERG products in a subtropical climate: The case of paraná state, brazil. *Remote. Sens. 13*, 5 (2021), 906.

[41] Nascimento, L.; Chagas, R.; Lage, M.; de Oliveira, D. Beyond click-and-view: a comparative study of data management approaches for interactive visualization. *Journal of Information and Data Management - JIDM* (2022), Submitted.

[42] Nascimento, L. C.; Knust, L.; Santos, R.; Sá, B.; Moreira, G.; Freitas, F.; Moura, N.; Lage, M.; Oliveira, D. Análise de dados pluviométricos multi-fonte baseada em técnicas olap e de visualização: uma abordagem prática. In *WCAMA* (2021), pp. 1–10.

[43] Nascimento, L. C.; Lage, M.; de Oliveira, D. Um estudo sobre o uso de abordagens de gerência de dados em sistemas de análise visual de dados espaço-temporais. In *Anais do XXXVI Simpósio Brasileiro de Bancos de Dados* (Porto Alegre, RS, Brasil, 2021), SBC, pp. 361–366.

[44] Ota, M.; Vo, H. T.; Silva, C. T.; Freire, J. Stars: Simulating taxi ride sharing at scale. *IEEE Trans. Big Data 3*, 3 (2017), 349–361.

[45] Poess, M. TPC-DS. In *Encyclopedia of Big Data Technologies*, S. Sakr and A. Y. Zomaya, Eds. Springer, 2019.

[46] Poess, M. TPC-H. In *Encyclopedia of Big Data Technologies*, S. Sakr and A. Y. Zomaya, Eds. Springer, 2019.

[47] Poess, M.; Nambiar, R. TPC. In *Encyclopedia of Big Data Technologies*, S. Sakr and A. Y. Zomaya, Eds. Springer, 2019.

[48] PostGIS. Postgis - https://postgis.net/, 2021.

[49] PostgreSQL. Index types - https://www.postgresql.org/docs/9.5/indexes-types.html, 2022.

[50] PostgreSQL. Postgresql - https://www.postgresql.org/about/, 2022.

[51] Salas, D.; Liang, X.; Navarro, M.; Liang, Y.; Luna, D. An open-data open-model framework for hydrological models' integration, evaluation and application. *Environ. Model. Softw. 126* (2020), 104622.

[52] Samet, H. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[53] Schmidt, J. Usage of visualization techniques in data science workflows. In *Proc. of the VISIGRAPP* (2020), pp. 309–316.

[54] Sevim, A.; Mahin, M. T.; Vu, T.; Maxon, I.; Eldawy, A.; Carey, M.; Tsotras, V. A brief introduction to geospatial big data analytics with apache asterixdb. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on APIs and Libraries for Geospatial Data Science* (2021), pp. 1–2.

[55] Spark, A. Apache spark aqe - https://spark.apache.org/docs/latest/sql-performance-tuning.html#adaptive-query-execution, 2021.

[56] Spark, A. Spark sql data sources - https://spark.apache.org/docs/latest/sql-data-sources.html, 2021.

[57] Spark, A. Apache spark - rdd - https://bit.ly/3wop6cv, 2022.

[58] Thorndahl, S.; Willems, P. Probabilistic modelling of overflow, surcharge and flooding in urban drainage using the first-order reliability method and parameterization of local rain series. *Water Research 42*, 1 (2008), 455–466.

[59] Wang, A.; Zhang, A.; Chan, E. H. W.; Shi, W.; Zhou, X.; Liu, Z. A review of human mobility research based on big data and its implication for smart city development. *ISPRS Int. J. Geo Inf. 10*, 1 (2021), 13.

[60] Zaharia, M.; Chowdhury, M.; Das, T.; Dave, A.; Ma, J.; McCauly, M.; Franklin, M. J.; Shenker, S.; Stoica, I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012* (2012), S. D. Gribble and D. Katabi, Eds., USENIX Association, pp. 15–28.

[61] Zheng, Y.; Wu, W.; Chen, Y.; Qu, H.; Ni, L. M. Visual analytics in urban computing: An overview. *IEEE Transactions on Big Data 2*, 3 (2016), 276–296.

[62] Zimbrao, G.; de Souza, J. M. A raster approximation for processing of spatial joins. In *VLDB* (1998), Morgan Kaufmann, pp. 558–569.