Trabalho de Conclusão de Curso

Uso de Lakehouses para um ambiente analítico espacial Big Data

Felipe Ferreira Vasconcelos
ffv@ic.ufal.br

Orientador:
Prof. Dr. Fábio José Coutinho da Silva

Maceió, Junho de 2024

Felipe Ferreira Vasconcelos

Uso de Lakehouses para um ambiente analítico espacial Big Data

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. Fábio José Coutinho da Silva

Maceió, Junho de 2024

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Maria Helena Mendes Lessa – CRB-4 – 1616

V331u Vasconcelos, Felipe Ferreira.
Uso de Lakehouses para um ambiente analítico espacial Big Data / Felipe Ferreira Vasconcelos. – Maceió, 2024.
59 f. : il.

Orientador: Fábio José Coutinho da Silva.
Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2024.

Bibliografia: f. [55]-59.

1. Big data. 2. Banco de dados. 3. Data Lakehouse. 4. Armazenamento de dados. 5. Cidades Inteligentes. 6. Dados – Computação. I. Título.

CDU: 004.6

Agradecimentos

Agradeço à minha família por todo o suporte durante todos os anos, especialmente meus pais Roseane e Lucimar, e minha irmã Raquel.

Agradeço ao professor Fábio Coutinho pela orientação e por todas oportunidades dadas durante a graduação. Estendo os agradecimentos a todos aqueles que fizeram parte do grupo GSD/Laboratório TOCA desde 2021. Agradeço à todos amigos e companheiros de classe, que fizeram a jornada ser mais suportável, especialmente a todos do IFAL e ao meu querido amigo Taígo.

Por fim, mas não menos importante, agradeço a quem esteve ao meu lado, todos os dias e todos os minutos da minha vida desde 2018. Eminha, sou extremamente feliz de ter você ao meu lado, compartilhando todos os momentos e conquistas da vida. Obrigado por me amar. Esse trabalho é dedicado a você, porque *after all, you're my wonderwall.*

Lista de Figuras

Lista de Tabelas

Resumo

A transformação digital impulsionou o surgimento de novas formas de coletar e armazenar dados. Inicialmente, surgiram Data Warehouses e Data Lakes para lidar com análises complexas, altos custos de armazenamento e a necessidade de escalabilidade. No entanto, o crescimento dos dados Big Data revelou limitações nessas arquiteturas, culminando no desenvolvimento de soluções como os Data Lakehouses, que combinam características dos Data Lakes e dos Data Warehouses.

Os Lakehouses oferecem uma abordagem unificada para lidar com a diversidade e o grande volume de dados, mantendo baixos custos, alto desempenho analítico, flexibilidade e escalabilidade. Este trabalho analisa as limitações das arquiteturas de armazenamento atuais, destacando o suporte oferecido para dados Big Data e dados geoespaciais. A partir disso, constrói-se um ambiente Lakehouse baseado em tecnologias como o Delta Lake, para explorar o armazenamento e a análise de dados.

Apresenta-se um estudo de caso real, onde é implementado um ambiente Lakehouse para armazenar e analisar dados geoespaciais do transporte público em quatro cidades brasileiras. Este trabalho contribui para entender as tendências em arquiteturas de dados, especialmente em cenários de análise de dados espaciais Big Data, facilitando a aplicação prática desses conceitos.

Palavras-chave: big data; banco de dados; data lakehouse; data lakes; armazenamento de dados; cidades inteligentes

Abstract

Digital transformation has driven the emergence of new ways of collecting and storing data. Initially, Data Warehouses and Data Lakes emerged to deal with complex analyses, high storage costs and the need for scalability. However, the growth of Big Data revealed limitations in these architectures, culminating in the development of solutions such as Data Lakehouses, which combine characteristics of both Data Lakes and Data Warehouses.

Lakehouses offer a unified approach for dealing with diversity and large volumes of data while maintaining low costs, high analytical performance, flexibility and scalability. This work analyzes the limitations of current storage architectures, highlighting the support offered for Big Data and geospatial data. From this, a Lakehouse environment is built based on technologies such as Delta Lake, to explore data storage and analysis.

It's presented a real case study, where a the Lakehouse environment is implemented to store and analyze geospatial data from public transport in four Brazilian cities. This work contributes to understanding trends in data architectures, especially in Spatial Big Data Analytics scenarios, facilitating the practical application of these concepts.

Key-words: big data; database; data lakehouse; data lakes; data storage; smart cities

Conteúdo

1

Introdução

A utilização de tecnologias para geração e análise de dados não é algo recente, sendo amplamente acolhida por diversos tomadores de decisão, obtendo concretização com a disseminação de ferramentas de análise empresarial (*Business Intelligence*). Com o passar dos anos, novas fontes de dados e novas demandas latentes nas empresas levaram à introdução de novos dispositivos, novas características e novas formas de coletar dados, elevando a dimensão do processo de extração, tratamento, armazenamento e análise de dados.

De acordo com o portal Statista, mais de 75 bilhões de dispositivos de Internet das Coisas (IdC) estarão conectados até o ano de 2025 Statista [2016]. Além disso, existe a expectativa de que 175 *zettabytes* [1] de dados sejam gerados por dia em 2025, sendo aproximadamente metade gerados a partir de dispositivos IdC Rydning et al. [2018]. Essas estatísticas evidenciam o crescimento e a escala de geração de dados existentes atualmente, destacando a consagração do conceito de dados Big Data. Porém, com esse crescimento exponencial, em conjunto com as características heterogêneas e volumosas dos dados, soluções de armazenamento e análise de dados historicamente utilizadas, como SGBDs implementando Data Warehouse (DW), começaram a demonstrar pouca eficácia em sua utilização, além de altos custos de operação e de manutenção Nambiar and Mundra [2022].

Dada a necessidade de lidar com as novas características do Big Data, novas arquiteturas e soluções foram propostas, por exemplo, os Data Lakes (DL) surgiram com uma proposta de armazenamento de baixo custo e versátil, armazenando os dados em formato cru. Os DL foram inicialmente considerados como sucessores do Data Warehouse, solucionando os desafios de lidar com dados massivos e heterogêneos. Porém, percebeu-se que os DL, ao solucionarem esses desafios, acabavam apresentando limitações como: pouca eficiência de consultas analíticas; falta de governança de dados e outros quesitos Harby and Zulkernine [2022].

Para solucionar essa demanda de armazenamento e análise de dados massivos, foi criada a arquitetura 2-Tier, que utilizava como base um Data Lake, onde os dados seriam armazenados

[1] um *zettabyte* equivale a 10^6 *petabytes*

em formato bruto, e a partir dele operações ETL eram realizadas, movendo os dados para um Data Warehouse que ficava responsável pelas consultas analíticas. Porém, esse tipo de arquitetura apresentava pontos fracos como o alto custo de operação e a baixa confiabilidade, sendo propenso a erros devido aos diversos fluxos de dados, além da quebra da premissa de um único ponto de verdade.

O atual estado-da-arte busca ofertar uma solução única, capaz de atender as necessidades Big Data, ofertando fatores como: baixo custo; desempenho em consultas analíticas; flexibilidade e escalabilidade; entre outros. Para isso, foi criada a arquitetura Data Lakehouse (DLH), fruto de uma intersecção de funcionalidades de um Data Lake e de um Data Warehouse (Lake + House). Os DLH vieram com o objetivo de prover uma plataforma única para todos os dados existentes, entregando custo-benefício, desempenho, governança. Também buscavam atender a demandas como volume, variedade e velocidade de dados, além de outras análises avançadas.

Um desafio comum para as arquiteturas de armazenamento são os dados espaciais, que apresentam uma característica de espacialidade, necessitando de soluções e ferramentas específicas para a realização de consultas. Os dados espaciais são fundamentais em diversas áreas, como urbanização, planejamento ambiental, logística. Na urbanização, ajudam a mapear o crescimento das cidades e otimizar a infraestrutura. Na logística, melhoram a eficiência de rotas de transporte. Sua importância está na capacidade de fornecer uma compreensão detalhada das dinâmicas espaciais, possibilitando decisões informadas e estratégias eficientes.

Os Lakehouses se adequam as necessidades de ambiente unificado para armazenamento e análise de dados espaciais, de modo que esses ambientes podem beneficiar diversos tomadores de decisão e diversos casos de uso, como, por exemplo, Cidades Inteligentes, temática que vem ganhando aderência e um maior engajamento no contexto Big Data. Cidades inteligentes utilizam-se de dispositivos IdC para gerar dados em larga escala de diferentes fontes, tais como: administração; saúde; meteorologia; mobilidade urbana; segurança; poluição; entre outros. A natureza desses dados requer sistemas capazes de prover flexibilidade, escalabilidade, confiabilidade e análises de dados eficientes, tendo em vista a escala de dados e as diferentes partes interessadas envolvidas no processo. Em se tratando da Mobilidade Urbana, por exemplo, dados geospaciais produzidos por diferentes dispositivos, tais como veículos, sensores, semáforos e radares, oferecem um grande potencial analítico, sendo relevante para a tomada de decisões que impactam a qualidade de vida em cidades inteligentes Alablani and Alenazi [2020].

Entretanto, dados espaciais apresentam outras características, como a heterogeneidade e velocidade dos dados. A partir disso, existe a necessidade de ambientes e arquiteturas que sejam capazes de suportar não apenas suas características espaciais, mas que também forneçam integração com diferentes soluções de metadados espaciais e de análise e visualização espacial. Os Lakehouses conseguem suprir parte dessas demandas, mas é necessário a construção de um ambiente que consiga prover questões como flexibilidade, escalabilidade, desempenho e interoperabilidade para todos os tipos de dados e operações.

1.1 Objetivos

Este trabalho tem como objetivo principal construir e implementar um ambiente Data Lakehouse eficiente e de baixo custo, que seja capaz de armazenar, processar e analisar dados geoespaciais Big Data. Este trabalho também irá discutir as arquiteturas de armazenamento existentes, as soluções disponíveis para Lakehouses, a interação entre as arquiteturas e os dados geoespaciais, e os benefícios da utilização de Lakehouses.

1.1.1 Objetivos Específicos

Para auxiliar no atendimento aos objetivos gerais deste trabalho, os seguintes objetivos específicos foram definidos:

- Identificar os pontos positivos e negativos das arquiteturas Data Warehouse, Data Lakes e Data Lakehouses.
- Identificar se as arquiteturas suportam o armazenamento e análise de dados geoespaciais.
- Propor um ambiente genérico de Lakehouse utilizando o Delta Lake e outras tecnologias.
- Construir um ambiente experimental com coleta, armazenamento e análise de dados geoespaciais provenientes do transporte público de quatro cidades brasileiras.

1.1.2 Perguntas de Pesquisa

Para nortear a construção deste trabalho e atingir os seus objetivos, estruturou-se as seguintes perguntas de pesquisa:

- Q1: As arquiteturas de armazenamento atuais permitem o armazenamento eficiente de dados Big Data?
- Q2: As arquiteturas de armazenamentos atuais suportam o uso de dados geoespaciais?
- Q3: Quais os benefícios da utilização de Lakehouses como solução de armazenamento?
- Q4: Quais são as ferramentas necessárias para a implementação de um Lakehouse?

1.2 Metodologia

Para atingir os objetivos específicos, a metodologia empregada neste trabalho está dividida em quatro etapas. A primeira etapa se refere a identificar, na literatura, as possíveis limitações e principais desafios na utilização de arquiteturas como Data Warehouses e Data Lakes. Nesta

etapa serão identificados trabalhos que relatam os pontos positivos e negativos das arquiteturas, além do suporte oferecido para dados geoespaciais.

A segunda e terceira etapa referem-se à proposta de implementação de um ambiente DLH para dados geoespaciais. A segunda etapa envolve a escolha da ferramenta que irá implementar o Data Lakehouse, que será realizada seguindo estudos que comparam características como desempenho, custo e integração das principais soluções disponíveis no mercado.

A terceira etapa trata da definição da composição do ambiente, onde serão discutidos quais módulos farão parte da implementação. Esses módulos serão propostos de acordo com as recomendações de trabalhos da literatura e com o objetivo de atender aos requisitos técnicos das arquiteturas Data Lakehouse. Nesta etapa, também será realizada uma discussão baseada em estudos da literatura sobre as principais ferramentas para cada módulo do ambiente.

A quarta etapa envolve a construção de um experimento com o objetivo de demonstrar as capacidades analíticas do ambiente proposto, validando o seu uso na análise de dados espaciais Big Data. O experimento envolverá o armazenamento, processamento e análise de dados geoespaciais. Para subsidiar esse experimento, será construído um conjunto de dados contendo dados geoespaciais reais do transporte público municipal brasileiro, permitindo que casos reais de uso sejam realizados no ambiente.

1.3 Organização do Trabalho

Os seguintes capítulos deste trabalho estão organizados da seguinte forma: A seção 2 se refere a fundamentação teórica sobre os principais conceitos, como Big Data, Data Lakes e Data Warehouses. A seção 3 apresenta o conceito de Lakehouse, suas características e a proposta de implementação original. A seção 6 irá apresentar a proposta de arquitetura para um ambiente Lakehouse que esse trabalho propõe construir, indicando soluções e métodos que devem ser utilizados. A seção 7 aborda um experimento com dados geoespaciais de quatro cidades brasileiras, utilizando-se do ambiente proposto na seção 6. Por fim, a seção 8 conclui o trabalho e apresenta as considerações finais e os trabalhos futuros, além das contribuições realizadas a partir desta monografia.

2

Fundamentação

Nesta capítulo são abordados alguns dos principais tópicos e conceitos que foram utilizados para a construção deste trabalho. A seção 2.1 apresenta dados espaciais e geoespaciais, enquanto a seção 2.2 discute o conceito de dados Big Data. As posteriores seções 2.3, 2.4 e 2.5 apresentam os conceitos de Data Warehouse, Data Lakes e comparam as duas arquiteturas. O conceito de Data Lakehouse e as particularidades de sua arquitetura não serão abordados neste capítulo, de modo que serão apresentados no Capítulo 3.

2.1 Dados espaciais

Dados espaciais são dados que contêm informações acerca de uma referência geométrica de objeto espacial Clarindo et al. [2021]. Conforme a Figura 2.1, dados espaciais podem ser representados como pontos, linhas ou polígonos, o que introduz um nível de complexidade para o seu armazenamento. A partir de dados espaciais é possível realizar diversas análises envolvendo a espacialidade dos objetos, como consultas de intersecção, consultas de vizinhos mais próximos, consultas de junção espacial Clarindo et al. [2021]. A complexidade do armazenamento e das possíveis análises que podem ser realizadas afloram a necessidade de ferramentas específicas para realizar operações nesses dados, visto que a dimensionalidade espacial requer um tratamento diferente comparado a dados não-espaciais.

2.1.1 Dados Geoespaciais

Dados geoespaciais são dados espaciais que têm sua referência na superfície do planeta, contendo informações como latitude e longitude. Por exemplo, pontos podem representar uma posição de latitude e longitude de um objeto na Terra, como uma casa, enquanto linhas podem representar elementos como rodovias, rios, entre outros objetos espaciais.

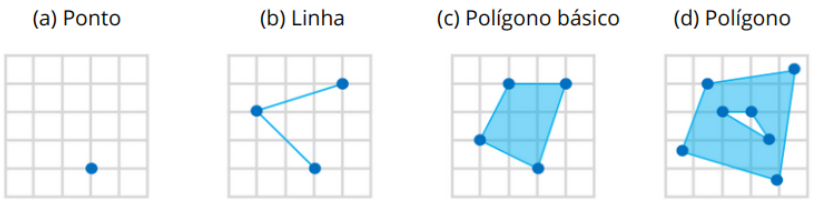

Figura 2.1: Representação dos formatos geométricos de dados espaciais. Fonte: Adaptado de Clarindo et al. [2021]

Trabalhos como o de Alablani and Alenazi [2020] evidenciam a importância do uso de dados geoespaciais, principalmente no contexto emergente de cidades inteligentes, de modo que os autores afirmam que esses dados oferecem um vasto potencial de informações acerca do espaço físico urbano. A disponibilidade desse tipo de informação pode ampliar as análises realizadas e subsidiar a tomada de decisões voltadas ao aumento da qualidade de vida urbana em cidades inteligentes.

2.2 Big Data

Com o desenvolvimento tecnológico, o uso de dispositivos IdC, multimídias, computação em nuvem, entre outros, aumentou progressivamente, o que resultou em um grande processo de geração de dados Nambiar and Mundra [2022]. De acordo com Reinsel et al. [2017], a quantidade de dados gerados em 2019 foi de 40 *zettabytes*. Em 2024, é prevista a geração de mais de 120 *zettabytes* de dados.

O processo de gerar, coletar e processar grandes volumes de dados não é algo recente, tendo em vista que os Data Warehouses foram concebidos para lidar com consultas e análises em grandes volumes de dados, que não eram mais possíveis de serem feitas eficientemente por humanos. Contudo, o conceito de Big Data foi definido formalmente somente em 2016, no trabalho de De Mauro et al. [2016], visando evidenciar as novas tendências na geração de dados. Nele, Big Data é definido da seguinte forma: "Big Data é o ativo de informação caracterizado por volume, velocidade e variedade tão elevados que requerem tecnologia e métodos analíticos específicos para sua transformação em valor" De Mauro et al. [2016].

A mudança de paradigma para dados Big Data evidenciou novos atributos dos dados, indicando a necessidade de novas ferramentas capazes de lidar com a coleta, armazenamento e processamento de grandes volumes de dados. Com o crescimento de volume, variedade e velocidade dos dados, bancos de dados relacionais e SGBDs tradicionais se tornaram incapazes de ofertar um armazenamento de baixo custo e eficiente. Como resultado, ferramentas como Apache Hadoop, com o sistema de armazenamento de baixo custo HDFS, e o Apache Spark, para processamento paralelo de dados, foram desenvolvidas e ganharam espaço na indústria e na academia.

2.2.1 Os V's de Big Data

O trabalho de De Mauro define que os dados Big Data apresentam características que podem ser resumidas em 5 "V's", de modo que uma solução Big Data deve ser capaz de prover soluções para, pelo menos, algumas dessas propriedades. Essas características são: Volume; Variedade; Velocidade; Valor; Veracidade.

Volume refere-se à grande quantidade de dados que são coletados pelas empresas. Com custos mais baixos para geração, coleta e armazenamento de dados, as empresas passaram a armazenar cada vez mais dados com o objetivo de expandir os processos de mineração e análise. No entanto, as soluções existentes na época não eram capazes de lidar com a magnitude dos dados, levando ao surgimento de ferramentas que fossem capazes de abarcar essa quantidade de dados.

Variedade refere-se à heterogeneidade dos formatos dos dados. Dados Big Data podem estar em diferentes formatos, como tabelas relacionais, documentos JSON, imagens, entre outros. Por exemplo, em uma cidade inteligente é possível armazenar dados de trafégo, dados climáticos e dados de poluição. Tais dados podem estar nos mais diversos formatos, no entanto, devem ser armazenados no mesmo ambiente.

Velocidade refere-se à capacidade de coletar dados gerados com alta frequência. Essa propriedade instigou a criação de sistemas de armazenamento que fossem capazes de receber dados gerados em altas velocidades, além de sistemas de processamento que fossem capazes de processar os dados em tempo real.

Valor refere-se às análises que podem ser construídas a partir dos dados e o valor que representam para as partes interessadas. Em um contexto Big Data, é prejudicial para os cientistas de dados armazenarem grandes quantidades de dados que não agreguem algum valor analítico.

Veracidade refere-se à confiabilidade dos dados armazenados, de modo que exista uma garantia de que as análises que serão realizadas a partir deles forneçam informações confiáveis para os cientistas de dados.

2.3 Data Warehouses

O conceito de Data Warehouse (DW) surgiu em um período no qual as empresas administravam diversas fontes de dados espalhadas na organização, os chamados silos de dados. Estes, no entanto, dificultavam o processo de extração de informações valiosas, pois não permitiam a integração dos dados. Para abordar esse desafio, os silos de dados foram substituídos por sistemas de armazenamento unificados, com o foco na análise e mineração de dados.

Os Data Warehouses surgem, portanto, como repositórios centrais de dados que têm como principais características a oferta de um armazenamento não volátil, unificado e orientado a assunto. A finalidade dessa arquitetura é proporcionar análises e consultas rápidas em grandes quantidades de dados que estejam relacionadas a um tópico ou contexto. Sua implementação

geralmente se dá em cinco camadas:

1. **Camada de fontes de dados**, que irão gerar os dados que serão armazenados, como outros bancos ou sistemas transacionais;
2. **Camada de preparo**, onde os dados gerados são coletados e armazenados temporariamente;
3. **Camada ETL *(Extract, Load, Transform)***, onde os dados irão passar por operações de transformação e limpeza para se adequarem ao Data Warehouse;
4. **Camada do Data Warehouse**, onde deverá existir um esquema definido pelo engenheiro de dados, funcionando em alguma solução de armazenamento;
5. **Camada de serviço**, local por onde os usuários devem acessar o Data Warehouse e realizar as consultas desejadas.

2.3.1 Modelagem

A literatura apresenta diversas propostas para realizar a modelagem de dados em um Data Warehouse, sendo essas abordagens guiadas pela natureza dos dados ou pelo tipo de consulta que será realizada. As duas principais abordagens utilizadas na literatura são: esquema estrela e esquema floco de neve. Ambas têm como base dois conceitos importantes para a construção de um Data Warehouse: Fatos e Dimensões. De modo que todas as tabelas constantes em um Data Warehouse serão ou tabela de fatos, ou tabela de dimensões. Sendo assim, tais conceitos são definidos da seguinte forma:

- **Fatos**: indicam eventos centrais a serem representados no armazenamento. Por exemplo, em uma loja, o evento principal é uma venda. Logo, a tabela de fato irá representar uma venda, contendo informações acerca da mesma. As tabelas de fatos são compostas por uma chave primária e diversas chaves estrageiras, referentes as tabelas de dimensões. A Figura 2.2 ilustra uma tabela de fato de uma venda, que contém informações como Produto, Vendedor, entre outros.
- **Dimensões**: representam informações relacionadas a um fato central. No exemplo de um Data Warehouse de vendas, as tabelas dimensões poderiam ser: Produto, contendo informações como valor, validade e lote, e vendedor, contendo informações de quem realizou a venda. A Figura 2.3 ilustra tabelas de dimensões de um DW de vendas, contendo as entidades Produtos, Vendedor, Loja e Comprador.

O esquema estrela recebe este nome por apresentar o formato de estrela, contendo a tabela de fatos centralizada e as tabelas de dimensões nas pontas. É a modelagem mais utilizada

Figura 2.2: Representação de uma tabela fato de uma venda. Fonte: Autor [2024]

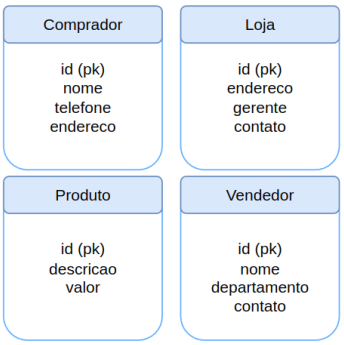

Figura 2.3: Representação de tabelas de dimensões num contexto de vendas. Fonte: Autor [2024]

na literatura, sendo inspiração para diversas outras Golfarelli and Rizzi [2017]. A Figura 2.4 representa uma modelagem para uma loja, onde existe como fato central uma Venda e como dimensões: Produtos; Vendedor; Comprador; Loja.

O esquema estrela não requer normalização dos dados, nem apresenta uma hierarquia entre os dados. Isso implica que podem existir dados duplicados dentro do Warehouse, aumentando o custo de armazenamento. Todavia, a duplicação de dados pode favorecer o desempenho, evitando que sucessivas junções sejam realizados.

O esquema floco de neve é baseado no esquema estrela, porém, ele apresenta formato de um floco de neve por abranger mais dimensões. Ele se baseia na criação de hierarquias entre as dimensões, com o objetivo de evitar a redundância de dados e diminuir o tamanho do Data Warehouse. Entretanto, o esquema floco de neve pode apresentar um alto custo de performance caso sucessivas operações como as junções precisem ser realizadas. A Figura 2.5 apresenta um esquema floco de neve, contendo hierarquias com as tabelas Departamento e Endereço, que tem maior granularidade.

2.3.2 Data Warehouses Espaciais

Data Warehouses Espaciais envolvem a adaptação de DW tradicionais para suportar o armazenamento e a consulta de dados geoespaciais. Dada a característica espacial desses dados, são necessárias mudanças no projeto do DW para que esses novos atributos sejam suportados.

Os atributos espaciais de um DW Espacial podem ser manipulados por meio de operações

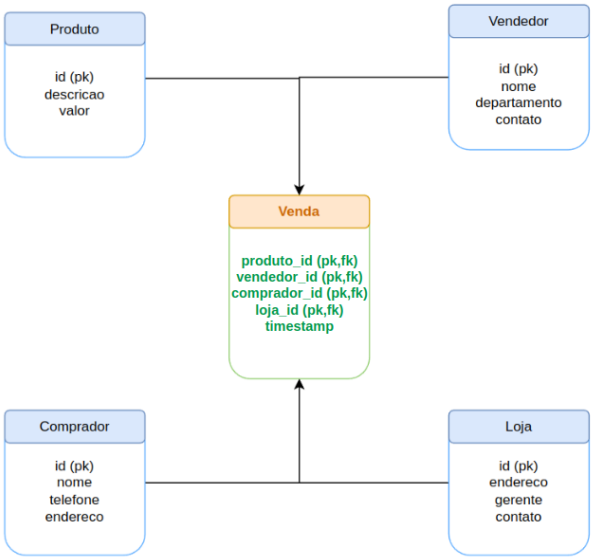

Figura 2.4: Representação de uma modelagem estrela para uma venda. Fonte: Autor [2024]

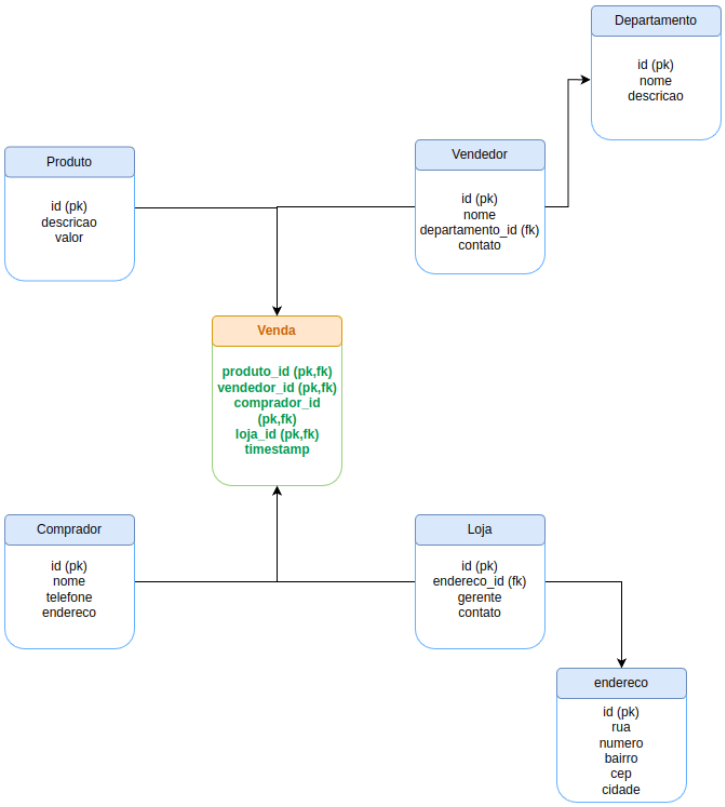

Figura 2.5: Representação de esquema floco de neve com hierarquias. Fonte: Autor [2024]

chamadas SOLAP (Spatial Online Analytical Processing), que funcionam de maneira similar as OLAP (Online Analytical Processing), mas requerem ferramentas e soluções capazes de processar a dimensão espacial dos dados. Essa nova dimensão expande a possibilidade de análises, de modo que fatores como a espacialidade de um acontecimento podem ser integrados aos dados. Uma das ferramentas mais populares é o PostGIS[1], que adiciona o suporte à dados geoespaciais e a consultas espaciais no PostgreSQL, permitindo a construção de Data Warehouses Espaciais.

2.4 Data Lakes

Os Data Lakes foram propostos como uma alternativa pra solucionar alguns dos desafios enfrentados pelos Data Warehouse diante da nova dimensão de dados Big Data, como: custo; flexibilidade e escalabilidade. Ramchand and Mahmood [2022] definem Data Lakes como: "Data lake é um sistema centralizado que pode ser usado para armazenamento, processamento de dados e análise de dados brutos. Ele consegue manter os dados em seu formato original e permite ser consultado somente quando há necessidade. Além disso, possui capacidade de armazenar diversos formatos de dados, incluindo fontes de dados não estruturadas, semiestruturadas a estruturadas."

Os Data Lakes buscam substituir o formato centralizado dos Data Warehouses por uma arquitetura mais flexível, que possibilite as empresas escalarem e modificarem seus sistemas de armazenamentos de acordo com as suas necessidades Nambiar and Mundra [2022]. De um ponto de vista arquitetural, Walker and Alrehamy [2015] definem o Data Lake como:

> "Um data lake usa uma arquitetura plana para armazenar dados em seu formato bruto. Cada entidade de dados no lago está associada a um identificador exclusivo e um conjunto de metadados, e os consumidores podem usar esquemas criados especificamente para acessar dados relevantes, o que resultará em um conjunto menor de dados que pode ser analisado para ajudar a responder à pergunta do consumidor.- Traduzido pelo autor

Essas definições destacam a existência do Data Lake como uma forma de solucionar alguns dos desafios e limitações que surgiram nos DW, como a integração entre diferentes tipos e formatos de dados, e a possibilidade de escalabilidade, e de flexibilidade, de armazenamento e processamento. Todavia, o desenvolvimento dos DL trouxe novas demandas, como a existência de um catálogos de metadados, um conjuntos de políticas de governança para evitar a transformação do lago em um pântano, além do desenvolvimento de arquiteturas que realizassem a organização lógica e física dos dados.

Uma das abordagens mais comuns de implementação de Data Lakes é a arquitetura em zona, de modo que cada zona se refere a um nível de refinamento nos dados, sendo responsável por

[1] https://postgis.net/

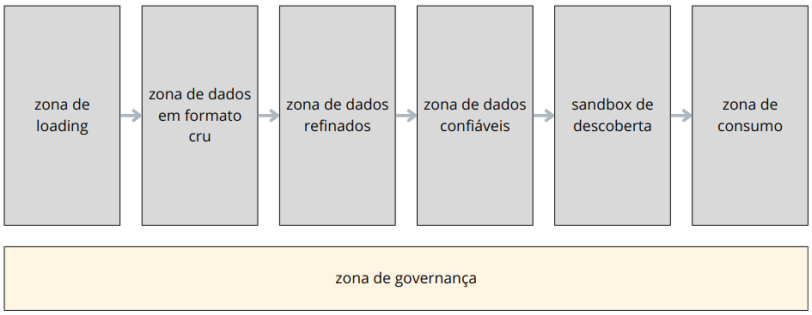

Figura 2.6: Arquitetura de zonas em um Data Lake. Fonte: LaPlante [2016]

Comparação	Data Warehouse	Data Lake
Dados	Dados estruturados	Dados estruturados, semi-estruturados, não estruturados
Esquema	*Schema on-read*	*Schema-on-write*
Processamento	Dados já processados	Dados crus ou processados
Armazenamento	Alto custo; confiável	Baixo custo, pode apresentar pouca confiabilidade
Flexibilidade	Pouca flexibilidade, consultas definidas a priori	Alta flexibilidade, podendo ser reconfigurado
Usuários	Corporativos	Cientistas de dados
Ferramentas	Principais ferramentas são proprietárias	Soluções de código aberto são difundidas
Escalabilidade	Vertical	Horizontal

Tabela 2.1: Comparação entre Data Warehouses e Data Lakes. Fonte: Adaptado de Nambiar and Mundra [2022] e Khine and Wang [2018]

uma parte do fluxo ETL. Na Figura 2.6, observa-se uma arquitetura baseada em 6 zonas que é apresentada por LaPlante [2016], onde as zonas irão envolver desde uma área de carregamento dos dados, até um ambiente para descoberta dos dados.

2.5 Data Warehouses contra Data Lakes

Ambas soluções são arquiteturas comprovadamente eficiente para repositório de dados. Porém, cada arquitetura favorece casos de uso, com pontos fortes e fracos, por exemplo, DW apresentam melhor governança de dados, enquanto DL apresentam um custo menor. Cabe ao usuário definir qual arquitetura se encaixa melhor no seu contexto. Na Tabela 2.1, adaptada de Nambiar and Mundra [2022] e Khine and Wang [2018], comparam-se algumas das características dos DW e dos DL, com a finalidade de evidenciar em quais contextos cada uma das abordagens devem ser aplicadas.

Uma das principais distinções entre os DW e DL está na adoção de *schema-on-read* ou *schema-on-write*, ou seja, se o sistema irá definir o esquema apenas na leitura dos dados ou

durante a gravação dos dados. Essa definição serve para guiar a sequência de operações que devem ser realizadas para extrair, carregar e transformar os dados, ditando se o procedimento será "ETL"ou "ELT".

Os Data Warehouses utilizam o processo "ETL"(Extração, Transformação e Carregamento [2]), de modo que os dados são extraídos das bases de dados, processados para tratamento dos dados, e por fim carregados no Data Warehouse. Esse processo implica na necessidade de um esquema ser definido previamente, o que possibilita boa performance na leitura dos dados, mas pode dificultar a flexibilidade do armazenamento Khine and Wang [2018].

Os Data Lakes adotam o processo "ELT"(Extração, Carregamento e Transformação[3]), no qual os dados são extraídos de diversas fontes e carregados em seu formato original no Data Lake. Devido ao armazenamento em formato cru, os Data Lakes não requerem esquemas definidos previamente, sendo os mesmos definidos apenas no momento da leitura e uso dos dados. Essa abordagem permite que diversos formatos e tipos dados possam ser utilizados numa mesma análise, mas pode ocasionar em uma diminuição na qualidade de dados Khine and Wang [2018].

[2]Do inglês: *Extract, Transform, Load*
[3]Do inglês: *Extract, Load, Transform*

3

Data Lakehouses para ambientes analíticos Big Data

Este capítulo busca apresentar as definições de Data Lakehouses encontradas na literatura, além de discutir os requerimentos técnicos de implementação. A seção 3.1 apresenta a definição do autor original do Data Lakehouse, enquanto a seção 3.2 apresenta os requerimentos técnicos que devem ser implementados por um Data Lakehouses. A seção 3.3 descreve o modelo original de uma solução DLH apresentado por Armbrust et al. [2021].

3.1 Definições de Lakehouse

Diante da disseminação das arquiteturas 2-*Tiers* e dos diferentes desafios de lidar com grande volumes de dados, surgiram problemas práticos que resultaram no aumento dos esforços necessários para implementar e manter esse tipo de arquitetura. Por exemplo, Vakharia et al. [2023] apresenta que a Meta[1] foi levada a reestruturar seu conjunto de ferramentas de dados devido à falta de padronização e de componentes reutilizáveis, o que culminou em despesas para produzir uma solução integrada e centralizada.

Pesquisadores começaram a discutir a possibilidade de integrar os pontos positivos das arquiteturas existentes em um único ambiente, não com o objetivo de gerar inovações tecnológicas como o *MapReduce/HDFS* foram para o Data Lake, mas sim obter avanços na organização dos dados e das próprias arquiteturas, tornando possível o reúso de conceitos e soluções existentes para, assim, oferecer as funcionalidades desejadas. Nesse sentido, Armbrust et al. [2021], autor do artigo original de Lakehouse, levanta a seguinte questão técnica que culminou no desenvolvimento dos Lakehouses:

[1] Empresa responsável pelo Facebook, Instagram e WhatsApp

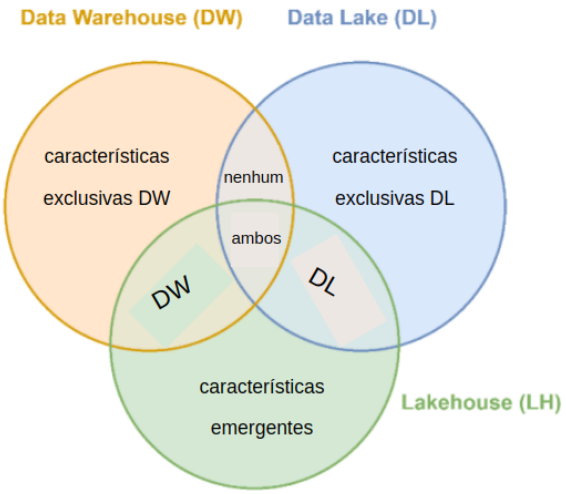

Figura 3.1: Diagrama de Venn sobre a composição do Lakehouse. Fonte: Adaptado de Schneider et al. [2023]

> "É possível transformar data lakes baseados em formatos de dados abertos padronizados, como Parquet e ORC, em sistemas de alto desempenho que podem fornecer recursos de desempenho e gerenciamento de data warehouses e I/O rápida e direta de cargas de trabalho de análise avançada?"Armbrust et al. [2021] - Traduzido pelo autor

Com base nessa questão técnica, buscou-se criar um sistema de armazenamento que integrasse os pontos positivos de ambas arquiteturas. Essa concepção pode ser representada por meio de um diagrama de Venn, como mostrado na Figura 3.1. Com base nessas propostas e ideais, chegou-se ao conceito de Data Lakehouse, que é definido por Armbrust como:

> "Sistema de gerenciamento de dados baseado em armazenamento de baixo custo e acesso direto, que também fornece recursos analíticos tradicionais de gerenciamento e desempenho de SBGD, como: transações ACID, controle de versão de dados, auditoria, indexação, cache e otimização de consultas."Armbrust et al. [2021] - Traduzido pelo autor

Além da definição original de Armbrust, é possível encontrar na literatura outros autores que buscam evidenciar o propósito de integração de ambientes do Data Lakehouse. Por exemplo, Harby and Zulkernine [2022] apresenta a seguinte definição para Data Lakehouses:

> "Integração de características como: O componente flexível de armazenamento Big Data dos DLs; O armazenamento integrado, limpo e estruturado dos DW. Componentes do DL para facilitar a integração, descoberta, gerenciamento de metadados e análise de dados; Uma série de ferramentas de Big Data Analytics, inteligência artificial e aprendizado de máquina que possam ser aplicadas aos dados em diferentes estágios de processamento; Componentes do DW para permitir consultas

Requerimentos		Cargas de trabalho		
#	**Nome**	**Análises/OLAP**	**Mineração/AM**	**Streaming**
R1	Mesmo tipo de armazenamento e formato de dados	●	●	●
R2	CRUD para todos os tipos de dados	●	●	●
R3	Coleção de dados relacionais	●	◐	◐
R4	Linguagem de consulta	●	○	○
R5	Garantia de consistência	●	◐	○
R6	Isolamento e atomicidade	●	○	●
R7	Acesso direto de leitura	○	●	◐
R8	Processamento stream e lote unificado	○	○	●

●Influência forte ◐Influência média ○Influência baixa

Tabela 3.1: Requerimentos técnicos para um Lakehouse e as contribuições para as cargas de trabalho - Fonte: Adaptado de Schneider et al. [2023]

> de Processamento Analítico Online (OLAP) para análises empresariais."Harby and Zulkernine [2022] - Traduzido pelo autor

Com base nessas definições, é possível conceituar Data Lakehouses como uma arquitetura voltada para a construção de ambientes de armazenamento de baixo custo e confiáveis, seguindo princípios dos Data Lakes. Esses ambientes devem ser capazes de lidar com dados batch e em streaming, além de implementar conceitos de Data Warehouse como: gerenciamento; propriedades ACID; versionamento e outros. A utilização de formatos abertos, como Parquet, também é recomendada, buscando evitar a dependência técnica de serviços proprietários.

3.2 Requerimentos técnicos de um Data Lakehouse

As definições propostas por ambos Armbrust e Harby realizam uma abordagem conceitual dos Lakehouses. Porém, alguns autores realizam uma caracterização técnica mais próxima da implementação, detalhando os requerimentos técnicos que possibilitam a construção de um Data Lakehouse. Por exemplo, o trabalho de Schneider et al. [2023] identifica sete requerimentos técnicos que devem ser cumpridos por um ferramenta Data Lakehouse, além de considerar a contribuição desses requisitos para três tipos de cargas de trabalho: Análises/OLAP; Mineração de Dados/Aprendizado de Máquina (AM); Streaming.

A Tabela 3.1, adaptada de Schneider, apresenta a descrição dos requerimentos técnicos e avalia suas influências em cada tipo de carga de trabalho. Esta análise demonstra que os Lakehouses são capazes de oferecer suporte a todas as cargas de trabalho.

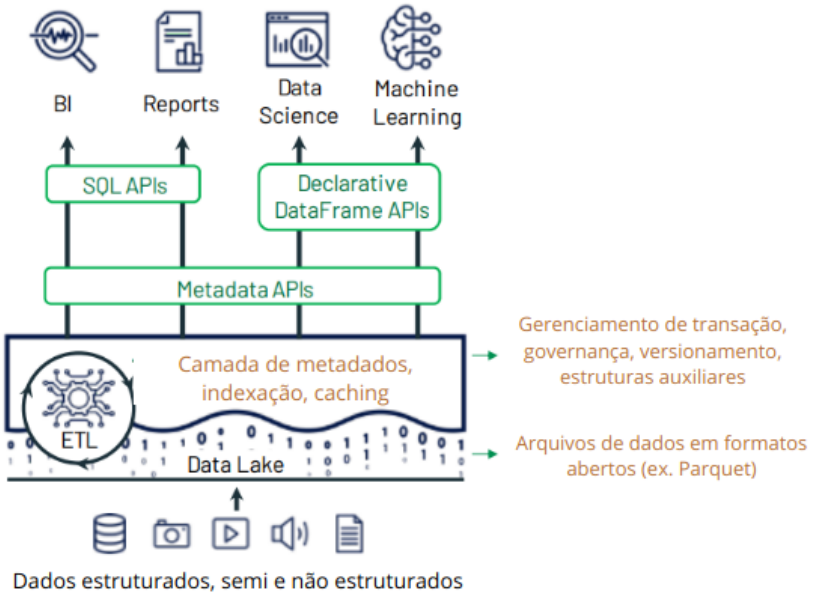

Figura 3.2: Modelo genérico de Data Lakehouse proposto por Armbrust et al. [2021]. Fonte: Adaptado pelo autor [2024]

3.3 Modelo de Lakehouse de Armbrust

Tanto na literatura quanto na indústria, encontram-se diversas abordagens de Lakehouses, como, por exemplo, Vakharia et al. [2023] Vompa [2023]. Todavia, a proposta inicial de implementação de um Data Lakehouse delineada por Ambrust inclui três principais componentes: Camada do Data Lake, Camada de Metadados, APIs de Interação. A Figura 3.2 demonstra como esses componentes interagem e coexistem.

Cada ferramenta que se propõe a implementar uma arquitetura Lakehouse deverá viabilizar a construção desses componentes, buscando resolver os desafios propostos por cada um deles. Os componentes propostos por Armbrust serão detalhados a seguir.

Camada do Data Lake

A essência dos Data Lakehouses está em construir um Data Lake confiável, incorporando capacidades existentes nos Data Warehouses. Para isso, a primeira ação necessária é a construção de uma camada de armazenamento que abrigará o Data Lake principal, hospedado prioritariamente em armazenamentos em nuvem de baixo custo, como o Amazon S3. Esse Data Lake irá armazenar os dados físicos, que devem adotar um formato aberto padrão, como o Apache Parquet[2].

O Apache Parquet é um formato aberto, orientado a colunas, que apresenta suporte a métodos eficientes de compressão e codificação. Os arquivos Parquet contêm metadados acerca das informações armazenadas, oferecendo informações sobre o esquema dos dados e estatísticas,

[2]https://parquet.apache.org/

como valores mínimos e máximos das colunas. Esse tipo de arquivo proporciona um desempenho superior para dados Big Data em comparação com concorrentes, como CSV (*Comma-Separated Values*) Begoli et al. [2016].

Entretanto, armazenar dados puramente em um Data Lake com formato aberto pode resultar em problemas de corretude dos dados e também de desempenho, exigindo a implementação de camadas auxiliares para corrigir essas limitações Armbrust et al. [2020].

Camada de Metadados

A segunda camada necessária para implementar um Data Lakehouse é a camada de metadados. Essa camada se posiciona acima do Data Lake e é responsável pelo gerenciamento e governança dos dados. Ela controla quais objetos fazem parte de quais tabelas, e é onde algumas das principais funcionalidades dos Lakehouses devem ser implementadas, como propriedades ACID, versionamento, cacheamento e indexação Armbrust et al. [2021].

APIs de Interação

O último componente de um Lakehouse são as APIs de interação, que desenvolvem métodos otimizados para acesso, interação e consumo dos dados disponíveis no Data Lakehouse. Os Data Warehouses conseguem entregar consultas otimizadas devido a diversas técnicas existentes, como a utilização de formatos proprietários de dados. Todavia, algumas dessas técnicas não podem ser implementadas em Data Lakehouses, logo, é necessário desenvolver outras estratégias de otimização Armbrust et al. [2021].

A partir disso, motores SQL eficientes, como modificações do SparkSQL, e APIs declarativas de *DataFrames* devem ser construídas com o objetivo de trazer para os Lakehouses uma performance equivalente ao estado-da-arte dos Data Warehouses Vakharia et al. [2023]. Algumas das técnicas utilizadas em soluções de Lakehouses existentes envolvem a utilização de fatores como cacheamento, indexação de dados, otimização das estruturas dos dados, entre outros Armbrust et al. [2021].

3.3.1 Soluções disponíveis

Não existe, claramente difundido na literatura e na indústria, uma única ferramenta que seja a materialização de toda a arquitetura Lakehouse, restando diversas soluções que atacam os desafios impostos por arquiteturas 2-*Tiers* de diferentes formas Belov and Nikulchev [2021]. Para ser considerada uma implementação de um Data Lakehouse, uma ferramenta deve seguir, ou ter semelhanças com, o modelo proposto por Armbrust (seção 3.3) e atender os requisitos técnicos relatados por Schneider (seção 3.2).

Schneider analisa oito ferramentas como possíveis soluções de Data Lakehouse, das quais

apenas três ferramentas conseguem atender todos os requisitos para serem consideradas soluções Lakehouse, sendo elas:

- **Delta Lake**, plataforma criada pela Databricks, desenvolvedora do conceito de Lakehouse. É utilizada na plataforma oferecida pela empresa, tendo uma adoção por mais de 50% das empresas Fortune 500.[3]
- **Apache Hudi**, projeto Apache criado pela Uber com foco em streaming de dados.
- **Apache Iceberg**, projeto Apache criado pela Netflix para garantir corretude e ACID nos processos de análise de dados.

Trabalhos como Hellman [2023] e Belov and Nikulchev [2021] também avaliam as mesmas três ferramentas, demonstrando fortes indícios de que essas ferramentas constituem o estado-da-arte e são as principais implementações de código aberto de Data Lakehouses disponíveis na indústria e na literatura.

[3] https://www.databricks.com/company/newsroom/press-releases/announcing-delta-lake-30-new-universal-format-offers-automatic

4

Limitações das arquiteturas de armazenamento

Este capítulo discute as arquiteturas existentes e suas limitações, além de apresentar o conceito de Data Lakehouse como possível solução para análise de dados espaciais Big Data. A seção 4.1 contextualiza a evolução das arquiteturas de armazenamento de dados, desde os DW até os Data Lakehouses. A seção 4.2 abordam as principais limitações das arquiteturas encontrados na literatura e como esse cenário culminou na criação do conceito de Data Lakehouse, buscando responder às perguntas de pesquisa Q1 e Q2. A seção 4.3 demonstra como o Lakehouse supera as limitações discutidas nas seções anteriores, com o objetivo de responder à pergunta de pesquisa Q3.

4.1 Contexto

O trabalho de Armbrust et al. [2021] categorizou a evolução das arquiteturas de armazenamento em três gerações, cada uma refletindo novas abordagens para superar os desafios e atender às demandas tecnológicas que estavam surgindo. A Figura 4.1 ilustra essa evolução, que tem início com os Data Warehouses e finaliza com a arquitetura mais recente, Lakehouse.

Como evidenciado no Capítulo 2, os DW e DL possuem características que os tornam eficientes para diferentes casos de uso. Os Data Lakes apresentam baixo custo de operação e oferecem alta flexibilidade, porém, não são adequados para soluções de análise empresarial. Por outro lado, os DW apresentam solidez e desempenho analítico, mas apresentam limitações no que tange à escalabilidade e flexibilidade.

Com o aumento da capacidade de geração de dados, tornou-se evidente que a utilização exclusiva de DW ou DLs não seria suficiente para prover o baixo custo e as capacidades analíticas necessárias. Com isso, soluções baseadas na combinação de Data Warehouse e Data Lakes

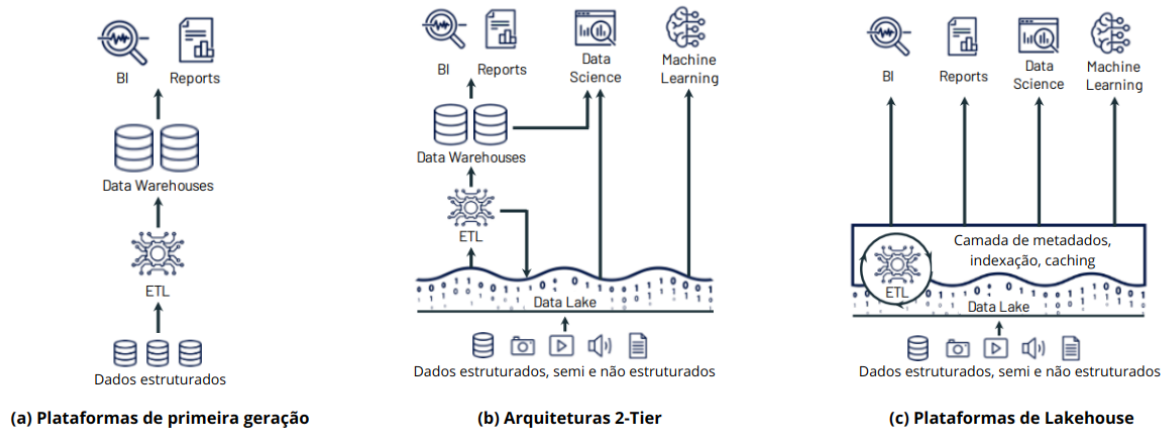

Figura 4.1: Evolução das arquiteturas de armazenamento. Fonte: Adaptado de Armbrust et al. [2021]

começaram a surgir. Essas soluções usufruíam do suporte a dados em tempo real e a dados heterogêneos dos Data Lakes, e ofereciam as capacidades de governança e de análise dos Data Warehouses Harby and Zulkernine [2022].

Soluções que utilizam ambas arquiteturas em um único sistema de armazenamento são denominadas de Arquitetura 2-*Tiers* e foram amplamente adotadas pela indústria. No entanto, algumas das tradicionais limitações apresentadas pelos Data Warehouses e Data Lakes ressurgiram, de modo que fatores como confiabilidade, custo e usabilidade da solução voltaram a ser discutidos. A seção a seguir destaca alguns dos desafios da utilização de Data Warehouses, Data Lakes e Arquiteturas 2-*Tiers*.

4.2 Limitações das arquiteturas

4.2.1 Data Warehouse

A primeira geração, que apresentou o Data Warehouse como solução para as necessidades analíticas de dados, revelou fragilidades com o advento dos dados Big Data. Os sistemas existentes não tinham a flexibilidade necessária para suportar diferentes formatos de dados, como imagens, documentos de texto e outros dados não estruturados. Essa falta de flexibilidade culminava na impossibilidade de realizar consultas com esses dados, o que acarretava perda de possíveis informações valiosas. Os sistemas também apresentavam baixa escalabilidade horizontal, o que resultava em consultas pouco performáticas ou em altos custos com escalabilidade vertical. Esses fatores poderiam inviabilizar a operação Harby and Zulkernine [2022].

Dentro desse cenário, é possível listar os seguintes pontos como limitações dos Data Warehouses:

- **L1**: Baixa flexibilidade de esquema, dado o *schema-on-write*.

- **L2:** Falta de suporte a análises avançadas, como aprendizado de máquina.
- **L3:** Baixa escalabilidade de armazenamento e computação.

4.2.2 Data Lakes

Os Data Lakes representam a segunda geração de arquitetura, desenvolvida com o objetivo de suprir as demandas de escalabilidade, flexibilidade e dados Big Data existentes nos Data Warehouses. Os Data Lakes partem da premissa de aceitar quaisquer formatos de dados em seu estado original, utilizando-se do princípio de *schema-on-read* Armbrust et al. [2021]. A proposta dos Data Lakes foi impulsionada pelo advento do *Hadoop File System* (HDFS), que era capaz de prover um armazenamento escalável de baixo custo Schneider et al. [2023]. Essa combinação permitiu que dados de qualquer natureza, como dados estruturados ou não estruturados, pudessem ser armazenados conjuntamente, ampliando a capacidade de extração de informações dos dados.

Porém, com essa abordagem, novas restrições começaram a surgir nos Data Lakes, como a ausência de componentes de governança e qualidade de dados, fortes funcionalidades dos Data Warehouses. Essa ausência pode culminar na inutilização dos dados do lago, transformando-os em um pântano de dados[1] Harby and Zulkernine [2022]. Diante deste cenário, listam-se os seguintes pontos como limitações dos Data Lake:

- **L4:** Falta de processos de governança e gerenciamento de dados.
- **L5:** Falta de processos que garantam a qualidade dos dados.
- **L6:** Baixa performance em consultas SQL comparados a SGBDs.

4.2.3 Arquitetura 2-Tier

Uma combinação da primeira e segunda geração surgiu a partir da decisão de adotar tanto Data Warehouses quanto Data Lakes como uma solução integrada de armazenamento, cada um desempenhando papéis específicos no processo. Os dados são extraídos das fontes de dados e armazenados no Data Lake, que serve como porta de entrada para o sistema de armazenamento. Em seguida, são aplicados fluxos de ETL para popular as tabelas necessárias no Data Warehouse. Esta abordagem permite o armazenamento de dados estruturados e não estruturados nos Data Lakes e a realização de consultas utilizando-se da capacidade de governança e de gerenciamento dos Data Warehouses Armbrust et al. [2021] Schneider et al. [2023].

A utilização de dois sistemas de armazenamento trouxe algumas limitações, como, por exemplo, a existência de dois pontos de verdade, de modo que, o DL e o DW podem, em determinados momentos, apresentar informações discrepantes para uma mesma premissa. Além

[1] "*Data Swamp*"

dessas limitações, a necessidade de executar diferentes fluxos de dados para cada ambiente aumenta a probabilidade de erros e falhas. Tais falhas podem acabar se propagando na linhagem dos dados e diminuindo sua confiabilidade Schneider et al. [2023].

Diante desse cenário, existem os seguintes pontos como limitações das arquiteturas 2-Tier:

- **L7**: Possível obsolescência nos dados.
- **L8**: Baixa confiabilidade dos dados devido aos múltiplos fluxos de dados.
- **L9**: Custo operacional de manter dois sistemas de armazenamento funcionando independentemente.

4.2.4 Dados Geoespaciais

Além das limitações citadas nas seções anteriores, uma limitação comum para todos os sistemas de armazenamento é a capacidade de lidar com dados geoespaciais. Como citado na seção 2.1, os dados geoespaciais requerem uma abordagem de armazenamento, consultas e análises diferentes de tipos não-espaciais de dados. Pode-se tratar alguns dados espaciais como textuais, porém, perde-se a principal característica desse formato de dado: sua espacialidade. Sem esse fator, consultas como interseccção, união, cálculo de área de polígonos não podem ser realizadas.

Para que os Data Warehouses ofereçam suporte a dados espaciais, são necessárias mudanças na modelagem dos dados e a utilização de soluções para consultas espaciais. Essas adaptações transformam o Data Warehouse em um Data Warehouse espacial[2]. Todavia, mesmo com o suporte adicionado por extensões, os dados geoespaciais apresentam uma característica de heterogeneidade devido às diferentes representações geométricas existentes, de modo que a falta de flexibilidade (L1) dos DW pode ser uma limitação no uso Errami et al. [2023].

Data Lakes resolvem essa limitação de flexibilidade, porém, continuam válidas para os dados geoespaciais limitações como a falta de propriedades ACID, falta de validações de esquemas e outras ferramentas de governança e qualidade de dados para evitar que se tornem inutilizáveis Errami et al. [2023].

A partir desses fatores, surge a necessidade de um ambiente capaz de lidar com a diversidade de formatos de dados geoespaciais, mantendo a qualidade dos dados e oferecendo uma boa interoperabilidade com ferramentas de análise de dados geoespaciais. Além desses quesitos, fatores como o volume e a velocidade de geração de dados geoespaciais a partir de dispositivos IdC são importantes questões a serem solucionadas.

[2] *Spatial Data Warehouse* (SDW)

4.3 Solução para as limitações existentes

Os requerimentos técnicos citados na seção 3.2, juntamente com as definições encontradas na seção 3.1, conferem aos Lakehouses a capacidade de oferecer soluções para as restrições presentes nas arquiteturas DW e DL. Por exemplo, ao utilizar um Data Lake como camada de armazenamento, os Lakehouses superam a baixa flexibilidade dos Data Warehouses. Da mesma forma que, implementando propriedades ACID, os Lakehouses superam as limitações de governança de dados dos Data Lakes.

A Tabela 4.1 apresenta uma análise de como os desafios identificados podem ser superados por meio do uso de Lakehouses. Além disso, discute quais requerimentos técnicos propostos por Schneider garantem a implementação das funcionalidades.

Arquitetura	Limitações	Soluções
Data Warehouse	L1: Baixa flexibilidade de esquema, dado o *schema-on-write*.	Lakehouses aceitam todos os formatos de dados por meio da camada de Data Lake. Os requerimentos técnicos que garantem essa solução são: R1 e R2.
Data Warehouse	L2: Falta de suporte a análises avançadas, como AM	Lakehouses possibilitam acesso direto aos arquivos, favorecendo a utilização em mineração de dados. Os requerimentos técnicos que garantem essa solução é: R7
Data Warehouse	L3: Baixa escalabilidade, tornando consultas em dados de larga escala pouco eficientes.	Lakehouses proporcionam escalabilidade por meio do Data Lake, além de ofertar uma separação entre armazenamento e computação.
Data Lake	L4: Falta de processos de governança e de gerenciamento de dados.	Lakehouses implementam camadas que proporcionam propriedades ACID, validações de esquemas e outras funcionalidades. Os requerimentos técnicos que garantem essa solução são: R5 e R6
Data Lake	L5: Falta de processos que garantam a qualidade dos dados.	Lakehouses implementam camadas que proporcionam propriedades ACID, validações de esquemas e outras funcionalidades. Os requerimentos técnicos que garantem essa solução são: R5 e R6
Data Lake	L6: Baixo desempenho em consultas SQL comparadas a SGBDs.	Os Lakehouses apresentam motores adaptados e se utilizam de metadados e de técnicas de estrutura de dados para diminuir o custo computacional. Os requerimentos técnicos que garantem essa solução são: R3 e R4
2-Tier	L7: Possível obsolescência nos dados	Lakehouses unificam os ambientes, proporcionando um único ponto de verdade para diferentes formatos e tipos de arquivos. Os requerimentos técnicos que garantem essa solução são: R1; R2 e R5.
2-Tier	L8: Baixa confiabilidade dos dados devido as múltiplas pipelines de dados	Ocorre uma diminuição do número de fluxos, tendo em vista a característica unificada dos Lakehouses. A existência menos fluxos de dados leva a uma menor chance de erros. Os requerimentos técnicos que garantem essa solução são: R1; R5; R6 e R8.
2-Tier	L9: Custo operacional de manter dois sistemas de armazenamento funcionando independentemente	Lakehouses tem a premissa de ambiente unificado, não sendo mais necessário manter diversos ambientes. Os requerimentos técnicos que garantem essa solução são: R7 e R8.

Tabela 4.1: Limitações das arquiteturas e a solução proposta pelo uso dos Lakehouses. Fonte: Autor [2024]

5

O uso de Lakehouses na literatura

Este capítulo apresenta o estado da arte e o estado da prática de ambientes Lakehouse, apresentando e discutindo trabalhos da literatura. A seção 5.1 discute trabalhos encontrados na literatura referentes ao design e construção de ambientes Lakehouse de domínio específico. Também serão discutidos trabalhos que abordem os Lakehouses de um ponto de vista conceitual. A seção 5.2 apresenta e discute as principais ferramentas para implementação de Data Lakehouses, Apache Iceberg, Apache Hudi e Delta Lake.

5.1 Trabalhos Relacionados

Data Lakehouses são um tema recente na literatura, de modo que é possível encontrar diversos trabalhos recentes que buscam definir e caracteriza-los. Os autores buscam identificar a evolução histórica dos sistemas de armazenamento, apontando suas falhas e sucessos, e apresentando arquiteturas de implementação. Harby and Zulkernine [2022] caracterizam o que são Data Warehouses, Data Lakes e Data Lakehouses e propõem a implementação de um Lakehouse utilizando o Snowflake. Vompa [2023] discute a arquitetura Lakehouse e busca implementar um ambiente propício para mensageria, de modo a resolver problemas de eficiência em *streaming* existentes, por meio do Apache Hudi. Cherradi [2024] afirma que utilizar o Lakehouse como gerenciamento de dados está se tornando a norma na indústria, e também apresenta uma arquitetura baseada em seis módulos, porém, não apresenta a implementação da mesma.

Outros trabalhos exploram a mesma perspectiva, porém, focam na discussão sobre dados espaciais Big Data, como o trabalho de Errami et al. [2022], que aponta alguns dos principais desafios de lidar com dados espaciais em larga escala. Os autores também utilizam uma arquitetura Lakehouse para realizar testes de performance em diferentes métodos de particionamento e de indexação de dados espaciais. No trabalho de Errami et al. [2023], o foco é detalhar como o Lakehouse pode ser utilizado para gerenciar dados espaciais Big Data, além de definir alguns

dos componentes necessários e quais as melhores práticas para esse tipo de arquitetura. Porém, os autores não oferecem uma implementação base da arquitetura, apenas sua proposta. Mete [2023] realiza a comparação entre formatos de arquivos como GeoJSON e Shapefile, além de comparar motores de processamento espacial como Dask GeoPandas e Sedona, com o objetivo de prover discussões sobre a infraestrutura técnica necessária para a criação de um Lakehouse de cidades inteligentes.

Além disso, podem ser encontradas implementações de Lakehouses focadas em casos de uso específicos, como o Deep Lake, implementação de Lakehouse focada em cargas de trabalho de redes neurais com interoperabilidade com diversos *frameworks* como PyTorch, TensorFlow e JAX Hambardzumyan et al. [2023]. Begoli et al. [2021] apresenta um Lakehouse focado no gerenciamento e análise de dados de biomedicina e saúde, adequando a arquitetura para os conceitos FAIR[1] e apresentando diversos casos de uso.

O presente trabalho se diferencia por abordar tanto as características e limitações das arquiteturas clássicas, como também por explorar a natureza espacial dos dados e suas peculiaridades. Além disso, é proposto um ambiente Big Data com ferramentas técnicas que suportem análise de dados espaciais, e são realizados experimentos a partir de dados reais, com o objetivo de simular um caso de uso real.

5.2 Estado da Prática

O primeiro passo para implementar um Lakehouse envolve escolher a ferramenta utilizada, visto que as ferramentas oferecem diferentes abordagens e técnicas que devem ser levadas em consideração para o restante da construção de um ambiente. Como citado na seção 3.3.1, trabalhos como o de Scheinder se referem a três ferramentas como o estado-da-arte de Data Lakehouses: Delta Lake, Apache Hudi e Apache Iceberg. Outros autores, como Errami et al. [2023] também realizam avaliações sobre as três ferramentas, comparando-as em quesitos como existência de propriedades ACID e cacheamento. A Tabela 5.1 apresenta a comparação realizada por Errami, enquanto as seções a seguir discutem as ferramentas.

O **Apache Iceberg**[2] é um sistema de alto desempenho para dados em formato tabular, criado em 2017 pela Netflix com o objetivo de superar algumas das limitações existentes no Apache Hive Machado et al. [2022]. Criado com foco em ambientes de nuvem, ele suporta a criação e uso de tabelas com mais de um *petabyte* de dados, além de oferecer soluções para problemas de consistência e desempenho existentes em ferramentas como o Hive. O Iceberg utiliza uma estrutura em árvore para manter os arquivos de dados, construindo "*snapshots*"das tabelas, contendo metadados e informações sobre os "*manifest files*", que irão indicar quais os arquivos pertencentes às tabelas e o caminho dos mesmos. O Iceberg se caracteriza como solução Lakehouse principalmente por implementar uma camada transacional em cima de armazenamento

[1] *Findable, Accessible, Interoperable, Reusable*.
[2] https://iceberg.apache.org/

	Delta Lake	*Apache Iceberg*	*Apache Hudi*
ACID	Sim	Sim	Sim
Controle de concorrência	Otimista	Otimista	Otimista
Formato de armazenamento	Parquet	Parquet, ORC	Parquet, AVRO
Compactação	Sim	Não	Sim
Data Skipping	Sim	Sim	Sim
Upsert/Delete	Sim	Não	Sim
Cacheamento	Sim	Não	Não
Viagem no tempo	Sim	Sim	Sim
Evolução de esquemas	Sim	Sim	Sim
Suporte para armazenamento na nuvem	Sim	Sim	Não

Tabela 5.1: Tabela de comparação entre as três ferramentas. Fonte: Errami et al. [2023]

de dados na nuvem, apresentando capacidades como: evolução de esquemas; *rollback*; suporte transacional; suporte ao Apache Spark; entre outros.

O **Apache Hudi**[3] é uma solução criada pela Uber com o objetivo de realizar o gerenciamento de dados Big Data em Data Lakes, ofertando características como operações transacionais, *upserts* e exclusões eficientes. Ele foi criado com foco em ingestão de dados *streaming*, com soluções de alto desempenho para processamentos incrementais. Ele utiliza um sistema de gerenciamento baseado em diretórios, o que lhe concede integração com sistemas como HDFS Belov and Nikulchev [2021]. Cada tabela é armazenada em diretórios que se referem a partições, e cada partição é dividida em grupos de arquivos Parquet. Algumas das funcionalidades que caracterizam o Hudi como solução Lakehouse são: ACID, otimizações de leiaute de dados; formato aberto de dados; *rollback*; entre outros.

O **Delta Lake** foi criado pela Databricks em 2017 e é definido por Armbrust et al. [2020] como "uma camada de armazenamento de tabela ACID sobre armazenamentos de objetos em nuvem"(Traduzido pelo autor). O objetivo do Delta Lake é manter informações sobre objetos do armazenamento dos dados em tabelas Delta, utilizando-se de registros transacionais que irão prover funcionalidades como propriedades ACID, o que leva a uma melhora na consistência e na confiabilidade dos dados Hellman [2023]. Ele também apresenta uma versão modificada do Apache Spark com objetivo de entregar um melhor desempenho em computações, por meio do processamento paralelo. O Delta Lake se caracteriza como uma ferramenta Lakehouse tendo em vista características como: propriedades ACID; otimizações de leiaute de dados; formato aberto de dados; validação de esquemas, entre outros Armbrust et al. [2020].

5.3 Análise das Ferramentas de Lakehouse

Para instanciar o ambiente de análise de dados geoespaciais foi utilizado a ferramenta Delta Lake. A escolha se deu a partir dos seguintes critérios: (i) desempenho; (ii) integração com

[3]https://hudi.apache.org/

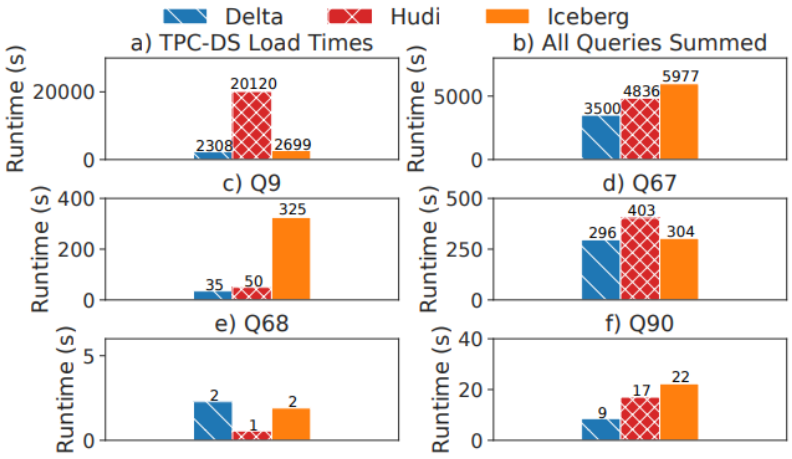

Figura 5.1: Comparação entre os tempos de carregamento e consulta (*load and query times*) do Delta Lake, Hudi e Iceberg 3TB TPC-DS. Fonte: Jain et al. [2023]

ferramentas; (iii) indicação de trabalhos relacionados.

O **critério de desempenho (i)** se demonstra importante para dados em larga escala, de modo que trabalhos como Jain et al. [2023] realizam uma análise de design, funcionalidades e desempenho entre os três frameworks supracitados. Os testes foram realizados em duas suítes de referência, a TPC-DS e a LHBench, desenvolvida pelos autores. Os autores relatam que o Delta Lake oferece um melhor desempenho geral em tempos de carregamento e consulta (*load and query times*) pelo TPC-DS, como pode ser visualizado na Figura 5.1. Os autores também realizaram testes nas estratégias de acesso aos metadados, comparando o Delta Lake e o Iceberg. Nesse teste, o Delta Lake apresenta um melhor desempenho em operações básicas realizadas nas tabelas de metadados, como pode ser visto na Figura 5.2.

O **critério de integração (ii)** com ferramentas se torna importante no contexto de um ambiente com diversas outras soluções. Todas as ferramentas utilizadas devem se integrar para gerar um ambiente fluído e sem gargalos. O Delta Lake apresenta integração com o Apache Spark e com armazenamento em nuvem como o Amazon S3, fatores que facilitam o processo de escalabilidade dos ambientes de armazenamento e processamento.

Para o **critério de indicação de trabalhos relacionados (iii)**, encontram-se diversos autores que realizam estudos acerca das principais soluções de Data Lakehouses. Por exemplo, Errami et al. [2023] discorre sobre as três ferramentas e utiliza o Delta Lake em seus experimentos. Hellman [2023] afirma que o Delta Lake é uma opção mais equilibrada, demonstrando um desempenho mais consistente em diferentes casos e cargas de trabalho.

5.3.1 Delta Lake

O Delta Lake também é capaz de superar todas as limitações citadas na seção 4.2 e na Tabela 4.1, se tornando uma opção recomendada para a implementação de um ambiente Lakehouse. A

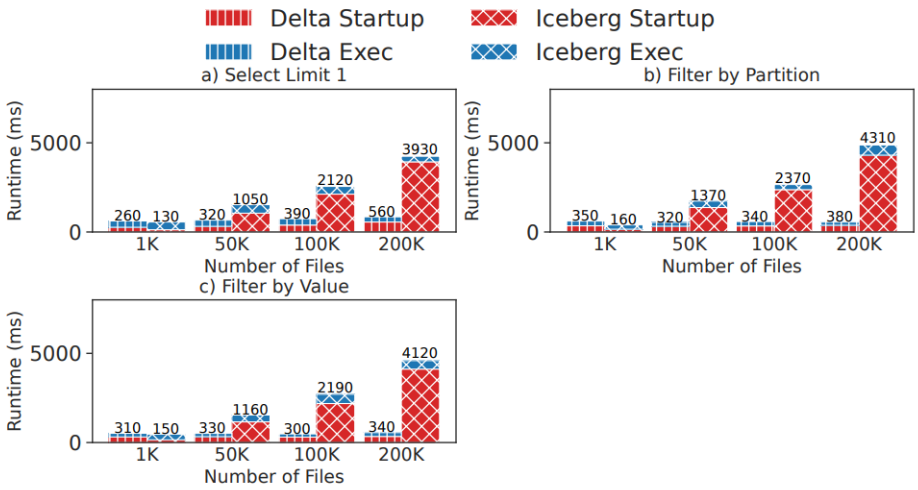

Figura 5.2: Tempos de inicialização e execução de consulta de operações básicas de tabela em uma quantidade variável de dados armazenados no Delta Lake e no Iceberg. Fonte: Jain et al. [2023]

seguir, será descrito de forma detalhada quais são as soluções propostas pela ferramenta para superar as limitações citadas.

- O Delta Lake utiliza-se de uma camada de armazenamento Data Lake que armazena os dados em arquivos Parquet. O Parquet suporta diferentes tipos de dados, que aliados ao *schema-on-write* do Data Lake, conferem flexibilidade, superando a limitação L1.

- O Delta Lake permite o acesso direto aos arquivos em formato Parquet, o que auxilia no processo de mineração de dados, superando a limitação L2.

- O Delta Lake permite a escalabilidade por meio de sua camada de Data Lake, que possibilita a separação entre armazenamento e processamento. Ele também oferece um armazenamento de baixo custo e utiliza o Apache Spark como motor de processamento, permitindo o paralelismo e distribuição, superando a limitação L3.

- O Delta Lake utiliza-se de uma camada de metadados que é capaz de conferir governança e gerenciamento de dados por meio de propriedades ACID, versionamento, entre outras funcionalidades que suprem a limitação L4.

- O Delta Lake busca garantir a qualidade dos dados por meio da utilização de propriedades ACID, que garantem que operações mantenham o estado constante. Ele também se utiliza de validações de esquema para garantir a qualidade das tabelas, na busca de suprir a L5.

- O Delta Lake utiliza uma versão modificada do Spark como motor de computação, além de se aproveitar das estatísticas dos arquivos Parquet, como valores mínimos e máximos das colunas, para otimizar as consultas. Essas modificações tornam o Delta Lake capaz de

entregar desempenho semelhante ou superior à Data Warehouses clássicos, consolidando a L6.

- O Delta Lake, ao ofertar um ambiente unificado, busca suprir a limitação de dados obsoletos constantes na L7, já que existirá apenas um único ponto de verdade. Esse ambiente unificado leva a uma diminuição da quantidade de fluxos de dados, que aliados aos princípios ACID, entregam uma maior confiabilidade, referente a L8. A diminuição da quantidade de fluxos de dados, em conjunto com a existência de um único ambiente, leva a uma diminuição dos custos de armazenamento, referentes a L9.

6

Implementação de Ambiente Lakehouse para dados Geoespaciais

Este capítulo apresenta a implementação de uma arquitetura Data Lakehouse, composta por diferentes módulos, para a criação de um ambiente voltado para a análise de dados geoespaciais. Ele discute algumas das ferramentas e métodos existentes para extração, transformação, armazenamento e análise de dados espaciais no contexto Big Data.

A proposta de implementação, representada na Figura 6.1, apresenta um ambiente para análise de dados espaciais composto por quatro módulos: Módulo de Ingestão de Dados; Módulo do Lakehouse; Módulo de Processamento; Módulo de Visualização. Cada módulo ficará responsável por partes do processo de ingestão, armazenamento e análise dos dados. Os capítulos a seguir discutem e abordam os diferentes desafios e ferramentas específicas para cada um dos módulos.

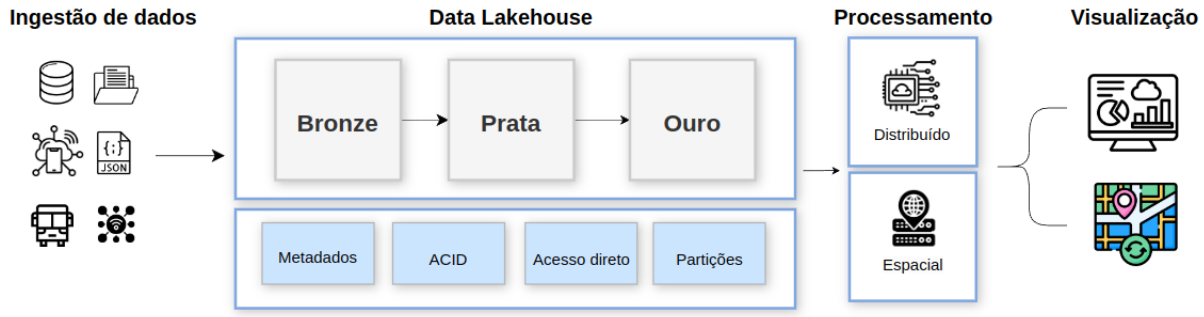

Figura 6.1: Ambiente de análise de dados baseada em Data Lakehouse. Fonte: Autor [2024]

6.1 Módulo de Ingestão de Dados

O módulo de ingestão de dados é a porta de entrada para o ambiente de armazenamento, sendo responsável por suportar a ingestão de dados em *streaming* e em lote. O processo de ingestão de dados não é trivial, tendo em vista que o contexto em que os dados estão inseridos pode trazer diversos desafios. Por exemplo, uma cidade inteligente produz dados volumosos e heterogêneos a partir de múltiplas fontes provenientes de diferentes sistemas como monitoramento do trafégo de veículos, previsão de tempo, alerta de segurança, entre outros. Logo, o processo de ingestão de dados torna-se complexo principalmente devido ao volume, a heterogeneidade e a velocidade de geração dos dados. Ao buscar uma solução, deve-se considerar a possível adição de novas fontes de dados com diferentes formatos; a capacidade de ingestão de dados em tempo real via *streaming*; a tolerância às falhas, dado que erros na transmissão dos dados podem causar distorções nas análises Meehan et al. [2017]; a segurança, de modo a proteger dados sensíveis como a privacidade dos cidadãos, entre outros fatores Mătăcuţă and Popa [2018].

Trabalhos como Vasconcelos et al. [2023] realizam estudos sobre as soluções para análise de dados Big Data, onde comparam duas ferramentas para ingestão de dados: Apache Kafka e Apache Nifi. Os autores ressaltam o Apache Kafka como uma solução mais abrangente, permitindo a integração com uma maior diversidade de ferramentas e dispositivos. Esse fator é relevante para um contexto de Lakehouse, que busca integrar diversas bases e fontes de dados. Seguindo a recomendação do autor, o Kafka é indicado como solução para o ambiente proposto neste trabalho.

6.2 Módulo Lakehouse - Armazenamento

Como citado na seção 5.3.1, a ferramenta escolhida para a implementação do Lakehouse foi o Delta Lake, por meio das bibliotecas disponíveis para a linguagem Python e de soluções como o Databricks Community. Para arquitetura dos dados, escolheu-se a arquitetura *Medallion*, por sua semelhança com as arquiteturas de zona dos Data Lakes, citadas na seção 2.4.

6.2.1 Arquitetura *Medallion*

A arquitetura *Medallion* é um padrão de engenharia de dados que envolve dividir os dados em diferentes estágios, onde cada estágio se refere ao nível de refinamento dos dados. Os estágios mais comuns são: Bronze, Prata e Ouro. Martín [2023]

O estágio Bronze é o inicial dos dados, onde eles serão armazenados em seu formato cru, com a menor quantidade de processos ETL possíveis. O objetivo desse estágio é manter os dados originais, de modo que os dados possam ser recuperados. O estágio Prata armazena os dados que passaram por processos de limpeza e tratamento, possibilitando a utilização dos mesmos em consultas. No estágio Ouro estarão tabelas que irão materializar um caso de uso

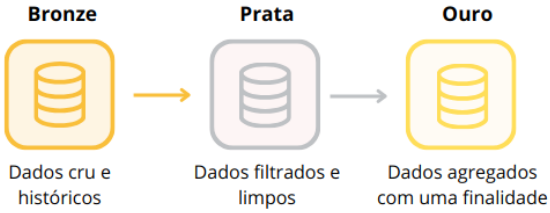

Figura 6.2: Representação da arquitetura *Medallion*. Fonte: Autor [2024]

específico, agregando e analisando os dados Prata para gerar informações ou popular um painel de análises empresariais, por exemplo. Martín [2023]

Essa separação permite que diversos usuários utilizem apenas as tabelas no estágio de refinamento necessário. Por exemplo, modelos de aprendizado de máquina precisam dos dados limpos, logo, utilizariam os dados Prata. Enquanto analistas de dados precisam das tabelas com finalidades específicas, logo, utilizariam dos dados Ouro. A Figura 6.2 a seguir representa a arquitetura *Medallion*

6.2.2 Armazenamento de Dados Espaciais

Como citado na seção 5.3.1, o Delta Lake utiliza-se de arquivos Parquet para armazenar os dados. Esse formato entrega um melhor desempenho, porém, não suporta tipos de dados espaciais nativos. Para acatar essa demanda, propõe-se a utilização do formato GeoParquet, projeto do *Open Geospatial Consortium* (OGC) que adiciona tipos geoespaciais como pontos, linhas e polígonos para o Parquet[1]. A utilização do GeoParquet permite uma melhor interoperabilidade entre sistemas, aumentando a possibilidade de uso de ferramentas de análise de dados espaciais. Ele também apresenta capacidade de distribuição, ao contrário do GeoJSON, além de permitir uma melhor compressão e a existência de arquivos maiores do que em formatos como o Shapefile.

A utilização do Geoparquet como formato para armazenar dados espaciais é amplamente vista na indústria e na literatura, como nos trabalhos Ballantyne and Berragan [2023], Medina et al. [2023]. É possível encontrar na literatura outros formatos que buscam trazer a dimensão espacial para o formato dos arquivos, como o SpatialParquet, que consegue apresentar um melhor desempenho que o Geoparquet, mas não entrega a mesma compatibilidade com ferramentas, fator considerado importante neste trabalho Saeedan and Eldawy [2022]. Existem autores que buscam criar formatos específicos para serem trabalhados em Lakehouses, como o GeoLake Zhang et al. [2023], porém, novamente, não apresentam boa compatibilidade e uso comprovado na indústria e na literatura.

[1] https://geoparquet.org/

6.2.3 Metadados

Metadados apresentam um papel importante no contexto de Data Lakes, sendo responsáveis por identificar e categorizar os dados existentes, facilitando o processo de integração dos dados. Dados espaciais requerem uma maior atenção, visto que armazenam atributos adicionais, relacionados a referência espacial e aos objetos geométricos Errami et al. [2023].

A utilização correta de metadados pode evitar a transformação de Data Lakes em pântanos (*"Data Swamps"*) Sawadogo and Darmont [2021]. Trabalhos como Sawadogo and Darmont [2021] realizam discussões sobre os tipos de metadados e quais os requerimentos que sistemas de metadados para Data Lakes devem atender. Os autores também realizam uma comparação entre dezoito sistemas de metadados para DL.

A utilização de sistemas de metadados pode levar a outros desafios, como problemas de latência e volume dos metadados. Os metadados crescem de acordo com a quantidade de dados e precisam ser consultados com baixa latência, logo, é necessário que os sistemas sejam capazes de lidar com essas questões. Soluções de Lakehouse como o Delta Lake, armazenam os metadados em arquivos JSON, que se compactam em arquivos Parquet, sendo armazenados junto com os dados em ambientes escaláveis. O Delta Lake também realiza a consulta dos metadados em *jobs* paralelos do Spark, diminuindo a latência.

Este trabalho não irá discutir soluções de metadados, deixando em aberto qual a ferramenta e sistema deverá ser utilizado pelos engenheiros de dados. Todavia, existem diversos trabalhos que se apresentam como soluções para o gerenciamento de metadados em Data Lakes, como Constance, que busca descobrir, extrair e anotar os metadados estruturais de dados crus ingeridos, e o GEMMS, solução que foca em ser genérica e extensível Hai et al. [2016] Quix et al. [2016]. Para metadados espaciais, trabalhos como Rustad et al. [2023] e Vogt et al. [2019] apontam o STAC[2] (*SpatioTemporal Asset Catalogs*) como sendo uma das principais formas de lidar com metadados espaciais, atendendo as específicações da *Open Geospatial Consortium* (OGC).

6.3 Módulo de Processamento

O motor de processamento utilizada pelo Delta Lake é uma versão modificada do Apache Spark, construída para entregar um melhor desempenho nas operações. O Apache Spark é uma ferramenta de processamento paralelo e distribuído que oferta bibliotecas de suporte como SparkSQL, para consultas SQL e MLLib para aprendizado de máquina. Tanto o Spark quanto o Delta Lake foram criados pela Databricks, de modo que as mudanças realizadas pelo Delta Lake não apresentam problemas de compatibilidade.

O Spark não é capaz de oferecer consultas espaciais como intersecção ou união de polígonos. Para isso, existem soluções que entregam essas funcionalidades, como o GeoPandas,

[2]https://stacspec.org/

Apache Sedona, GeoMesa, entre outros. O Apache Sedona e o GeoMesa são construídos em cima do Apache Spark, oferecendo computação paralela e distribuída.

O trabalho de de Carvalho Castro et al. [2020] compara algumas das principais ferramentas de análise de dados espaciais, de modo que define seis diretrizes que as ferramentas de análises espaciais (*Spatial Analysis Software* (SAS)) devem seguir para prover uma manipulação de dados espaciais massivos apropriada. De acordo com o autor, o Sedona[3] é o único sistema a seguir todas as diretrizes, sendo a opção que entrega melhor desempenho em consultas espaciais. Mete [2023] também realiza a comparação entre ferramentas de processamento paralelo geoespaciais, comparando o Apache Sedona com o Dask-GeoPandas, demonstrando que o Apache Sedona apresenta um desempenho superior nos quesitos de leitura de arquivos e de análises espaciais. Seguindo os resultados e discussões encontradas na literatura, o Apache Sedona foi escolhido para compor essa arquitetura.

6.3.1 Particionamento e Indexação

As análises e consultas de dados espaciais são complexas e apresentam alto custo computacional, se tornando necessário a adoção de técnicas como particionamento e indexação de dados para diminuir o custo e o tempo necessário para realizar essas operações. Dada a proposta de separação do armazenamento e do processamento dos Lakehouses, eles se tornam ambientes ideais para lidar com esse tipo de dado, visto que se torna possível a criação de clusters específicos para o processamento desse tipo de dado.

Existem diferentes formas de realizar o particionamento e indexação dos dados espaciais, como R-Tree, Octree, GeoHash, H3, entre outros. O particionamento no Delta Lake é realizado de forma semelhante ao Apache Hive, que se baseia na criação de diretórios com os valores das colunas escolhidas. De acordo com a documentação do Delta Lake, esse tipo de particionamento não é adequado para o motor do Delta, existindo apenas para prover compatibilidade com outros sistemas de armazenamento. Todavia, essa abordagem pode trazer um melhor desempenho para tabelas consideradas muito grandes, com dezenas de *terabytes* de dados[4].

Como forma recomendada de otimizar o *data skipping*, o Delta Lake utiliza a técnica *Z-Order*, que funciona em conjunto com metadados para ofertar um melhor desempenho em consultas. *Z-Order* é uma técnica de localidade dos dados que agrupa informações relacionadas nos mesmos grupamentos de dados, que são utilizados pelo motor do Delta Lake e do Spark com o objetivo de reduzir a quantidade de arquivos lidos, otimizando as consultas, da mesma forma que o particionamento de dados[5]. Este método tem um alto custo computacional, porém, a documentação indica que ele deverá ser executado com pouca frequência, apenas quando grandes volumes de novos dados forem adicionados as tabelas.

[3] GeoSpark é o nome do projeto original do Apache Sedona
[4] https://docs.databricks.com/en/tables/partitions.html
[5] https://docs.databricks.com/pt/index.html

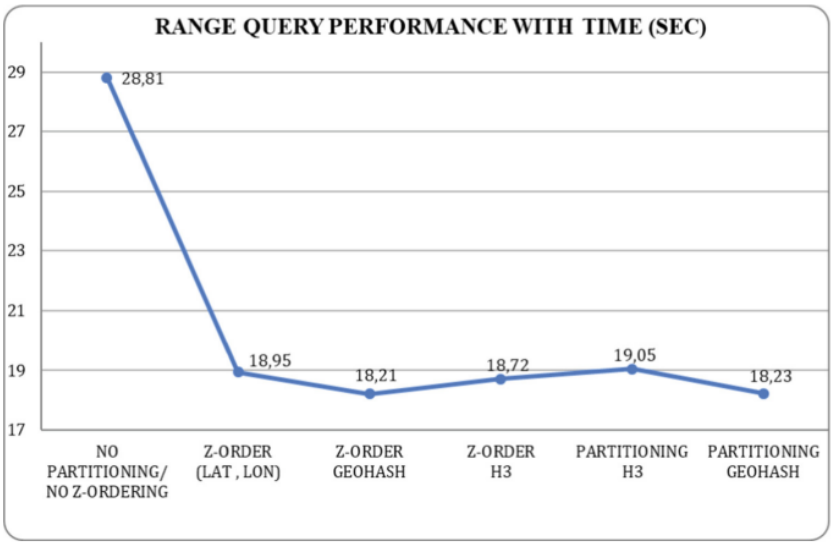

Figura 6.3: Resultados dos testes de performance numa consulta de intervalo. Fonte: Errami et al. [2022]

Para comparar o desempenho entre os métodos de particionamento Hive e o método *Z-Order* do Delta, Errami et al. [2022] construiu um ambiente para analisar a performance de seis tipos de abordagens de particionamento em dados espaciais: Sem partição e Sem *Z-Order*; *Z-Ordering* em colunas de Latitude e Longitude; *Z-Order* no Geohash; *Z-Order* no H3; Particionamento Geohash; Particionamento H3. Os autores chegaram a conclusão de que o *Z-Order* no Geohash apresenta um melhor desempenho, como pode ser ilustrado na Figura 6.3. Porém, eles alertam de que o *Z-Order* é uma tarefa de computação intensiva, de modo que, dependendo da frequência de utilização do *Z-Ordering*, pode ser que apenas o particionamento por Geohash seja mais ideal, apresentando um desempenho próximo ao *Z-Ordering*.

A partir do trabalho de Errami et al. [2022], define-se que o método principal de otimização para essas etapas será o de *Z-Ordering* em conjunto com GeoHash para dados espaciais. Todavia, nos experimentos que serão realizados no Capítulo 7, não serão utilizados métodos de otimização, tendo em vista a menor escala dos testes realizados.

6.3.2 Módulo de Visualização

A visualização de dados é uma etapa de extrema importância, de modo que irá tornar acessível para as partes interessadas as informações coletadas e analisadas nas etapas anteriores, fornecendo informações valiosas. Todavia, a visualização de dados está muito ligada com os próprios dados em utilização e o objetivo da utilização dos mesmos. Os motores de processamento como Spark e Sedona conseguem entregar uma diversidade de consultas e análises, todavia, se limitam as APIs de *DataFrame* e SQL para demonstrar os resultados. Para consultas que não estejam dentro desse escopo, como visualização de gráficos ou de mapas, existem fer-

Figura 6.4: Exemplo de mapa de calor gerado com o QGIS. Fonte: Autor [2024]

ramentas que são amplamente utilizadas, como as bibliotecas Matplotlib[6] e Folium[7] do Python, e programas como QGIS[8] e ArcGIS[9], especializados em dados geoespaciais e amplamente utilizados por especialistas. A Figura 6.4 representa uma visualização de mapa de calor gerada a partir do QGIS, demonstrando sua capacidade de geração de visualização de dados espaciais.

Para um ambiente ser bem-sucedido, ele precisa ter uma boa interoperabilidade com outras ferramentas. O ambiente proposto neste trabalho, ao utilizar o GeoParquet como formato de dados para dados espaciais, permite uma melhor integração com ferramentas como QGIS e ArcGIS. Esses softwares são capazes de interagir com arquivos CSV, porém, como os dados do Delta Lake estão em Parquet, se fazem necessários processos de conversão. Ao utilizar o Geoparquet, as tabelas Ouro podem ser criadas com esse formato, quando destinadas às análises específicas no QGIS, possibilitando a leitura direta desses arquivos, sem a necessidade de fluxos adicionais de transformação em CSV.

[6]https://matplotlib.org/
[7]https://python-visualization.github.io/folium/latest/
[8]https://qgis.org/
[9]https://www.arcgis.com/index.html

7

Experimento

Este capítulo apresenta o experimento realizado neste trabalho, demonstrando a viabilidade de utilizar Data Lakehouses para a construção de um ambiente de armazenamento e análise de dados espaciais. A seção 7.1 apresenta a pipeline de dados utilizada neste experimento. As seções 7.2 e 7.3 apresentam o dataset utilizado e a metodologia para sua construção.

Para subsidiar o experimento e como contribuição adicional deste trabalho, foi construído um dataset de dados geospaciais, proveniente de ônibus municipais de quatro cidades brasileiras. As seções 7.2 e 7.3 apresentam o dataset utilizado e o processo de sua construção.

A utilização desse dataset representa casos de uso reais, explorando utilização de dados de geolocalização de ônibus municipais, conforme abordado na literatura em trabalhos como da Cruz et al. [2016] e Queiroz et al. [2019].

A seção 7.4 apresenta a reprodução de casos de uso que seriam possíveis em Data Warehouses, como a modelagem de dados usando um esquema estrela, além de casos de uso de Data Lakes.

7.1 Ferramentas do fluxo utilizado

Tendo como base as ferramentas discutidas no capítulo anterior, a Figura 7.1 representa a proposta de implementação da arquitetura, utilizando-se de soluções de código aberto e seguindo recomendações encontradas na literatura.

A partir disso, foram utilizadas as seguintes tecnologias para todo o processo de ELTL (Extração, Carregamento, Transformação e Carregamento)[1] dos dados: Airflow; Delta Lake; PySpark; Sedona; QGIS; Matplotlib. Os códigos-fonte referentes às análises e processos estão disponíveis no repositório a seguir: github.com/felipevsc/lakehouse_onibus.

[1] Do inglês: *Extract, Load, Transform, Load*

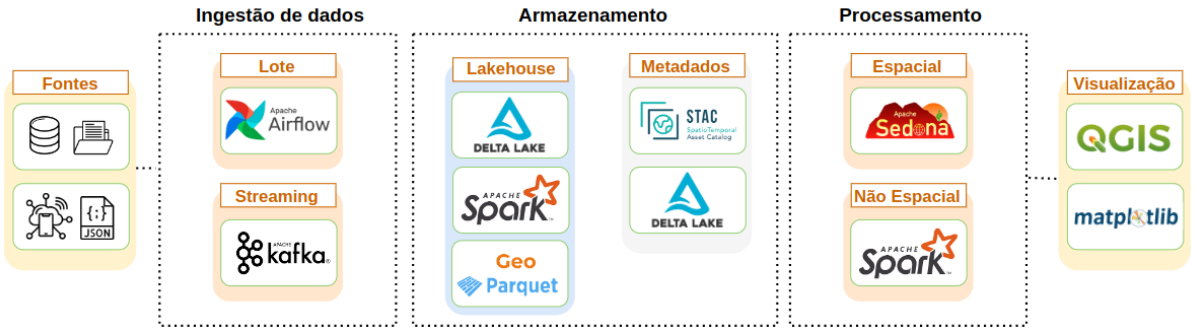

Figura 7.1: Fluxo de ferramentas código aberto para a arquitetura proposta. Fonte: Autor [2024]

7.2 Conjunto de dados utilizado no experimento

O conjunto de dados construído é denominado "BRBus"e envolve dados de geolocalização de ônibus municipais de quatro cidades brasileiras. A construção do BRBus foi uma contribuição adicional realizada durante a escrita desta monografia, sendo uma versão do conjunto de dados relatada por Melo et al. [2023][2]. Para esta monografia, os processos de construção foram adaptados e uma nova coleta foi realizada.

É possível encontrar trabalhos na literatura que propõem a construção e/ou manipulação de conjunto de dados contendo dados geoespaciais referentes ao monitoramento de tráfego de ônibus Cruz Junior [2020], Queiroz et al. [2019], da Cruz et al. [2016]. Entretanto, esses trabalhos não têm como finalidade principal a construção de um conjunto de dados para uso de terceiros e englobam poucas cidades brasileiras.

O BRBus se diferencia por abranger dados de um maior número de cidades brasileiras, realizando o processo de coleta, limpeza e integração dos dados. Essa abordagem abrangente, integrada e compartilhada oferece uma oportunidade de realizar análises exploratórias intercidades, fornecendo insights valiosos para os tomadores de decisão.

7.3 Construção do conjunto de dados BRBus

O fluxo de construção do BRBus seguiu a arquitetura *Medallion*, com o objetivo de se utilizar do ambiente proposto neste trabalho para o processo de análise desses dados. Foram criadas as camadas Bronze, Prata e Ouro, onde cada uma dessas camadas se refere a um nível de refinamento dos dados. Bronze se refere ao menor nível de refinamento, com os dados em seu formato cru; Prata se refere aos dados limpos e tratados, prontos para serem utilizados, e o Ouro apresenta os dados em seu mais alto nível de refinamento, sendo utilizados em casos de uso específicos, como dashboards.

Durante a construção do conjunto foram realizadas as seguintes etapas: (i) extração - obtenção dos dados das APIs; (ii) limpeza - que consiste na remoção de arquivos e ônibus inoperantes;

[2]O autor desta monografia é autor no trabalho de Melo et al. [2023].

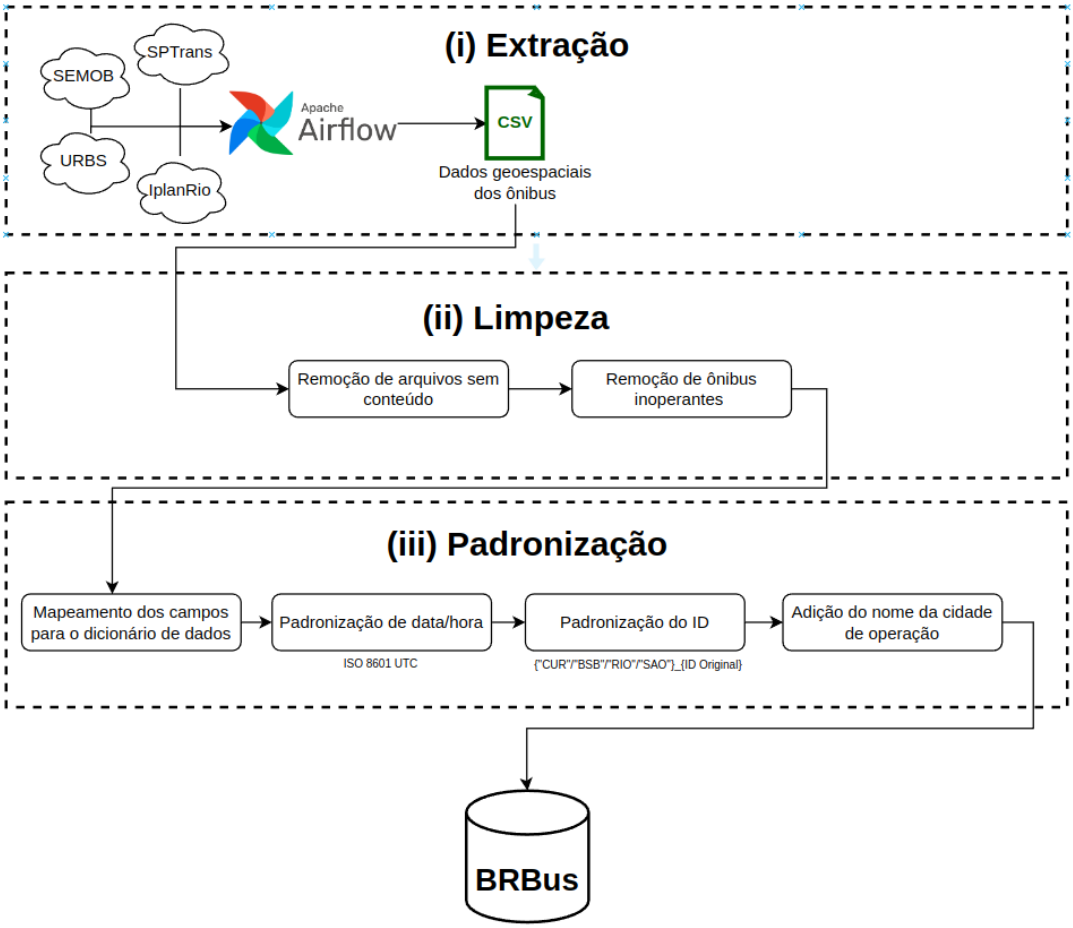

Figura 7.2: Diagrama do fluxo de construção do BRBus. Fonte: Adaptado de Melo et al. [2023]

(iii) padronização - que compreende a criação de um dicionário de dados comum a todas as fontes de dados e na formatação padrão de campos. A Figura 7.2 demonstra o fluxo das atividades e as ferramentas utilizadas na construção do conjunto de dados.

A Figura 7.3 representa como essas etapas se encaixam na arquitetura *Medallion*. Após o processo de extração, os dados são armazenados no Bronze, onde há a seleção de colunas e os dados são armazenados em Parquet. O processo de limpeza e padronização dos dados é realizado na tabela Prata, onde também constará um esquema com o dicionário de dados integrado do BRBus.

Essas etapas são necessárias visto que cada cidade apresenta seu próprio esquema, fonte e formato de dados. Em decorrência disso, uma abordagem de curadoria manual existe com a finalidade de ajustar os dados originais ao padrão criado pelo BRBus.

7.3.1 Extração - Coleta de Dados

A etapa inicial de construção do conjunto de dados consiste na coleta a partir de requisições a quatro diferentes *APIs*, as quais são disponibilizadas pelos departamentos ou empresas responsáveis pela mobilidade urbana das respectivas cidades, conforme descrito a seguir:

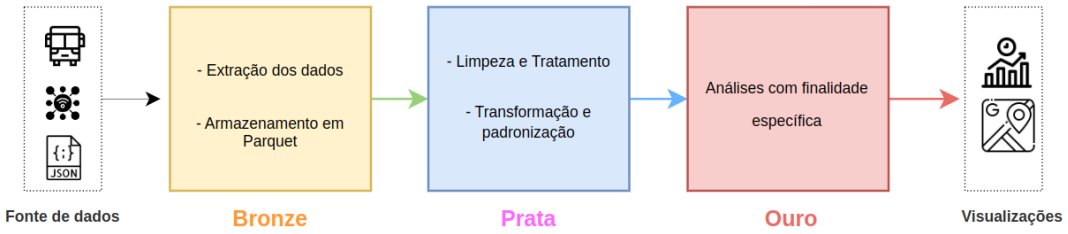

Figura 7.3: Representação das etapas de construção do BRBus e do experimento na arquitetura *Medallion*. Fonte: Autor [2024]

- São Paulo Transporte S/A - SPTrans[3] (São Paulo).
- Empresa Municipal de Informática - IplanRio[4] (Rio de Janeiro).
- Secretaria de Transporte e Mobilidade de Brasília[5] (Brasília).
- Urbanização de Curitiba S/A - URBS[6] (Curitiba).

7.3.2 Extração - Procedimentos

A extração dos dados das APIs foi realizada por meio da ferramenta Apache Airflow, que realizava as requisições a cada 60 segundos, de modo automatizado. O Airflow estava configurado em um ambiente de máquina virtual no DigitalOcean, construído com o objetivo de prover alta disponibilidade.

Foi realizado uma coleta durante o período de 31/01/2024 a 14/02/2024, onde os dados foram inicalmente armazenados localmente no formato *Parquet*. O conjunto de dados contém 7.4GB de arquivos, distribuídos da seguinte forma:

- Curitiba: 18 mil arquivos (770MB)
- São Paulo: 18 mil arquivos (3.5GB)
- Rio de Janeiro: 18 mil arquivos (1.9GB)
- Distrito Federal: 17.5 mil arquivos (1.3GB)

As quatro APIs totalizaram mais de 200 milhões de registros, demonstrando o quesito Big Data e a necessidade de ferramentas específicas para lidar com esse volume de dados. Na Tabela 7.1, pode-se visualizar as informações disponibilizadas por cada cidade.

[3]https://www.sptrans.com.br
[4]https://iplanrio.prefeitura.rio
[5]https://semob.df.gov.br
[6]https://www.urbs.curitiba.pr.gov.br

Informações Disponíveis	Cidades			
	São Paulo	Rio de Janeiro	Brasília	Curitiba
Coordenadas	X	X	X	X
Identificador do ônibus	X	X	X	X
Velocidade instantânea		X	X	
Horário de atualização	X	X	X	X
Código da linha	X	X	X	X
Situação de atraso				X
Sentido da rota	X		X	X
Tipo do veículo			X	
Acessibilidade a deficientes	X			
Informação do letreiro	X			
Quantidade de veículos na linha	X			
Número de ciclos sem atualização				X

Tabela 7.1: Informações disponibilizadas pelas cidades analisadas. Fonte: Autor [2024]

Exemplo das Requisições

No código-fonte 7.1, visualiza-se a resposta de uma requisição realizada para a *API* da cidade de Curitiba, que retorna uma listagem com os identificadores dos ônibus e suas informações. Exemplos das demais *APIs* podem ser encontradas no repositório de código-fonte deste trabalho.

Código 7.1: Exemplo da resposta da API de Curitiba

```
{"JB604":
    {"COD":"JB604",
    "REFRESH":"15:51",
    "LAT":"-25.419448",
    "LON":"-49.348316",
    "CODIGOLINHA":"040",
    "ADAPT":"1",
    "TIPO_VEIC":"4",
    "TABELA":"1",
    "SITUACAO":"ADIANTADO",
    "SITUACAO2":"REALIZANDO ROTA",
    "SENT":"VOLTA",
    "TCOUNT":1}
}
```

Mudança de nomenclatura

Durante a extração, foram realizados processos de desaninhamento de objetos JSON e a mudança de nomenclatura de alguns campos, com o objetivo de melhorar o processo de identificação dos dados. A seguir é possível visualizar alguns exemplos de mudanças na nomenclatura dos campos:

- Em São Paulo, os campos "py"e "px"se tornaram, respectivamente, os campos "latitude"e "longitude".
- No Rio de Janeiro, os campos "datahora"e "ordem"se tornaram, respectivamente, os campos "tempo_captura"e "id_onibus".
- No Distrito Federal, o campo aninhado "velocidade[’valor’]"se tornou apenas "velocidade".
- Em Curitiba, os campos "REFRESH", "LAT"e "LON"se tornaram "tempo_captura", "latitude"e "longitude"

7.3.3 Limpeza

Problemas identificados nos dados originais

Considerando as fontes de dados utilizadas para a construção do BRBus, foram identificadas divergências no dicionário de dados de cada operadora, visto que cada uma registrava os dados em esquema próprio. Além disso, foram encontradas outras questões relacionadas à ausência de dados, esses problemas são descritos em detalhes a seguir.

- Nos dados de Curitiba, o campo “REFRESH”, referente à última atualização da posição do ônibus, consta apenas com o horário no formato "HH:mm", não sendo possível identificar a data.
- Cada fonte utiliza um dicionário de dados próprio.
- As fontes de cada cidade utilizam diferentes formatos para representar o tempo em seus respectivos campos, como, por exemplo, os dados de São Paulo utilizam o formato “YYYY-MM-DD HH:mm:ssZ”, enquanto o Rio de Janeiro utiliza “MM-DD-YYYY HH:mm:ss”.
- A API de Curitiba apresenta todos os campos no formato *String*, fato que diverge das outras APIs, que atribuem tipagem aos campos de acordo com os valores.
- Ausência de dados referentes às operadoras e aos valores das tarifas das linhas de ônibus.
- As fontes de dados podem reportar ônibus fora de operação ou com defeitos no dispositivo de geolocalização.

Seleção inicial de campos

Durante o processo de extração, foram selecionadas apenas informações que fossem pertinentes a qualquer tipo de análise, enquanto outros campos foram descartados. Por exemplo, na API de Curitiba, existe o campo "codigo_imei", utilizado para identificar o IMEI do dispositivo responsável por transmitir os dados do ônibus. Tendo em vista que não existem possíveis análises a serem realizadas com esse campo, ele foi descartado. A seguir, demonstram-se outros campos que foram descartados durante o processo de extração:

- **"datahoraenvio"** (Rio de Janeiro): Horário em que a posição do GPS é comunicada ao servidor. Foi removido dado que pode não corresponder ao horário do GPS, que é representado no campo "datahora"
- **"unidade"** (Distrito Federal): Referente a qual unidade a velocidade estava sendo disponibilizada. O campo sempre será em quilomêtros por hora, logo, foi removido.
- **"qv"** (São Paulo): Referente a quantidade de veículos daquela linha que estão ativos naquele momento. O campo foi descartado visto que pode ser gerado a partir de uma agregação dos dados existentes.

Além da seleção dos campos e da mudança de nomenclatura, os procedimentos a seguir foram realizados com o objetivo de tratar o *dataset*:

- Durante o processo de coleta foram identificados e removidos arquivos que não apresentavam conteúdo, seja por erros no servidor de API ou por instabilidades de conexão.
- Nas respostas das APIs, ônibus que possuíam uma diferença de 5 minutos entre horário da requisição e o horário do último ping de atualização dos dados pela empresa responsável foram descartados.
- Foi identificado que as respostas da API de Curitiba apresentavam dados vazios quando o campo original “CODIGOLINHA” tinha valor “REC”, logo, foram removidas do arquivo da requisição.
- Ausência de valor no campo “LINHA” na API de Brasília indicava que o ônibus não estava em operação. Sendo assim, os dados destes veículos foram removidos.

Exemplos das requisições extraídas

Após os procedimentos de seleção de colunas, desaninhamento de objetos e mudança de nomenclatura, os arquivos estavam prontos para serem armazenados. Nas Figuras 7.4, 7.5, 7.6 e 7.7 são representados arquivos de cada cidade após os procedimentos citados.

	id_onibus	latitude	longitude	tempo_captura	tipo_veiculo	linha	situacao	situacao_2	sentido	tabela
9	GC003	-25.534576	-49.269411	17:26	COMUM	542	NO HORÁRIO	REALIZANDO ROTA	VOLTA	4-2
10	GC016	-25.551326	-49.266953	17:26	COMUM	542	NO HORÁRIO	REALIZANDO ROTA	IDA	2-2
11	GA180	-25.534445	-49.268253	17:25	COMUM	REC				
12	BI017	-25.350753	-49.232223	17:26	COMUM	232	NO HORÁRIO	REALIZANDO ROTA	IDA	1
13	BI014	-25.377315	-49.224608	17:25	COMUM	232	ATRASADO	REALIZANDO ROTA	IDA	3
14	BA036	-25.363871	-49.230336	17:26	COMUM	232	NO HORÁRIO	REALIZANDO ROTA	VOLTA	2-2

Figura 7.4: Exemplo de um arquivo Parquet da cidade de Curitiba. Fonte: Autor [2024]

	latitude	longitude	tempo_captura	id_onibus	c	lt0	lt1
1	-23.539707000000003	-46.631569375	2024-02-06T11:00:01Z	52034	5141-10	PÇA. DO CORREIO	TERM. SAPOPEMBA
2	-23.605799	-46.508037	2024-02-06T10:59:39Z	52125	5141-10	PÇA. DO CORREIO	TERM. SAPOPEMBA
3	-23.5610755	-46.5751625	2024-02-06T10:59:31Z	52911	5141-10	PÇA. DO CORREIO	TERM. SAPOPEMBA
4	-23.584544	-46.534061	2024-02-06T10:59:30Z	52788	5141-10	PÇA. DO CORREIO	TERM. SAPOPEMBA
5	-23.515379250000002	-46.605199	2024-02-06T10:59:56Z	21359	2127-10	LIBERDADE	PQ. EDU CHAVES
6	-23.478293	-46.56974099999999	2024-02-06T10:59:54Z	21723	2127-10	LIBERDADE	PQ. EDU CHAVES

Figura 7.5: Exemplo de um arquivo Parquet da cidade de São Paulo. Fonte: Autor [2024]

	latitude	longitude	tempo_captura	id_onibus	velocidade	linha
1	-22,90928	-43,18872	1707864953000	A72045	0	410
2	-22,92481	-43,26186	1707864946000	A72041	16	422
3	-22,89001	-43,29205	1707864952000	A72089	0	422
4	-22,94213	-43,17946	1707864953000	A72047	42	410
5	-22,95532	-43,1979	1707864952000	A72034	38	410
6	-22,92342	-43,20871	1707864953000	A72019	8	410

Figura 7.6: Exemplo de um arquivo Parquet da cidade do Rio de Janeiro. Fonte: Autor [2024]

	latitude	longitude	tempo_captura	id_onibus	velocidade	sentido	direcao	linha
1	-48.10842	-15.92126	1701355514000	340260	0.28	VOLTA	80.9000015	
2	-48.12325	-15.89951	1706128074000	339784	0	VOLTA	0	
3	-48.12391	-15.90059	1706315254000	340073	0	VOLTA	0	372.9
4	-47.9946	-15.87532	1706551160000	337846	0	IDA	333.434948	
5	-48.02361	-15.87496	1706636961000	339105	0	IDA	90	
6	-47.93532	-15.78702	1706644931000	335771	8.61	VOLTA	35.2276684	0.844

Figura 7.7: Exemplo de um arquivo Parquet da região metropolitana do Distrito Federal. Fonte: Autor [2024]

7.3.4 Padronização

Esta etapa compreende a criação de um dicionário de dados por meio da identificação dos diferentes esquemas utilizados por cada cidade para representar campos similares e a conversão dos dados para um formato padrão, com o objetivo de manter a homogeneidade dos dados. As seguintes informações foram utilizadas no dicionário de dados descrito na seção 7.3.5 e afetadas pela etapa de padronização:

- **Horário de atualização da informação do ônibus:**
  - Cada API utilizava um formato para representar o horário de atualização da informação do ônibus. Todos os dados foram convertidos para o formato ISO 8601 com fuso horário UTC.
- **Sentido de operação do ônibus**
  - As APIs de São Paulo e Curitiba informam o sentido de operação da linha, porém divergem na representação do dado. Estes campos foram convertidos e padronizados para números inteiros conforme descrito na Tabela 7.2, representando os sentidos de ida e volta.
- **Identificadores dos ônibus**
  - Os identificadores dos ônibus foram convertidos para o formato *string*, sendo composto pelo prefixo da cidade - representado pelas 3 primeiras letras do nome da cidade - e o identificador original, separados por underscore "_". Como exemplo desta mudança, o ônibus com identificador "BC941" da cidade de Curitiba teve seu identificador alterado para "CUR_BC941".

7.3.5 Dicionário de dados

O dicionário de dados do BRBus, exibido na Tabela 7.2, foi elaborado a partir dos esquemas fornecidos pelas *APIs*. Para auxiliar comparações com conjuntos de dados internacionais e a utilização por usuários não falantes da língua portuguesa, a nomenclatura dos campos presentes no BRBus foi redigida em inglês.

7.4 Análises

7.4.1 Data Warehouse

O objetivo deste experimento é utilizar o Data Lakehouse para simular um ambiente de Data Warehouse, criando uma modelagem estrela com o BRBus e outros dados, permitindo assim

Tabela 7.2: Dicionário de dados do BRBus, onde os campos em vermelho referem-se a campos adicionados pelo autor. Fonte: Autor [2024]

Campo	Descrição	Tipo
bus_id	Identificador do ônibus	String
city	Cidade de operação do ônibus	String
line_code	Código da linha do ônibus	String
is_adapted	Verificador de acessibilidade no ônibus	Boolean
is_eletric	Indicador se ônibus é elétrico ou não	Boolean
latitude	Latitude do ônibus no momento da requisição	Float
longitude	Longitude do ônibus no momento da requisição	Float
bus_speed	Velocidade do veículo em Km/h	Float
bus_direction	Sentido da linha que o ônibus está operando	1 = Ida 2 = Volta
updated_at	Data/Hora de obtenção da informação	String (ISO 8601)

consultas via SQL. Para isso, foram criadas as tabelas que aparecem na Figura 7.8, que buscam responder à seguinte pergunta: "Qual é a linha de ônibus com maior velocidade em Brasília?". A tabela de fato é a "Ônibus", enquanto as demais são tabelas de dimensões, contendo um campo de informação espacial na coluna "geom"da tabela "Registros", referente aos registros de posição dos veículos.

No contexto da arquitetura, essas tabelas podem ser consideradas tabelas Ouro, tendo em vista que têm um objetivo específico. Foram construídas as tabelas Ouro e realizadas as junções e outras operações necessárias, por meio do Spark, para criar a tabela de fatos. A partir da mesma, torna-se possível realizar a análise de que, na cidade de Brasília, a linha com código "207.3"é a que tem a maior velocidade média, obtendo uma velocidade aproximada de 37 km/h, como é possivel visualizar na Figura 7.9.

7.4.2 Análise dos dados dos ônibus

As próximas análises consistem na utilização exclusiva dos dados dos ônibus, visando obter consultas e análises que gerentes de cidades inteligentes realizariam. Novamente, já que cada tabela necessária para as consultas tem uma finalidade específica, as tabelas podem ser armazenadas na camada Ouro da arquitetura *Medallion*.

Velocidade média

As Figuras 7.10 e 7.11 representam gráficos da velocidade média nas cidades de Brasília e Rio de Janeiro, nos dias 04/02/2024, domingo, e 05/02/2024, segunda-feira, respectivamente. Os gráficos consideram apenas os veículos que estavam em movimento, ou seja, com uma

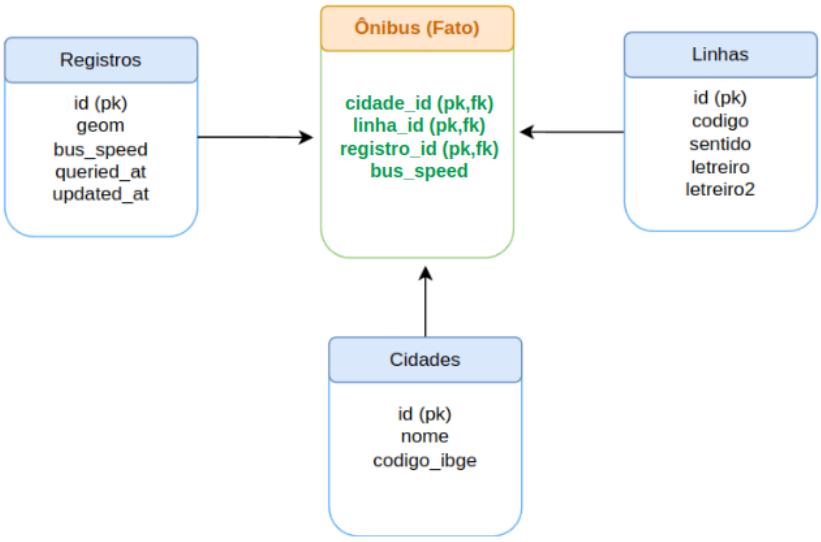

Figura 7.8: Esquema Estrela implementado no Data Lakehouse. Fonte: Autor [2024]

```
+------+-----------------+
|codigo|  media_bus_speed|
+------+-----------------+
| 207.3|37.04830188679245|
+------+-----------------+
```

Figura 7.9: Resultado da consulta por linha com maior velocidade média na cidade de Brasília. Fonte: Autor [2024]

velocidade maior que zero.

Percebe-se que o domingo a velocidade média maior do que a segunda-feira, devido à diminuição da quantidade de veículos trafegando. Também é possível notar uma menor variação da velocidade, visto que o horário de pico leva à diminuição da velocidade média em ambas as cidades.

Mapa de calor com PyDeck

Esta análise utiliza a biblioteca do Python PyDeck[7], integrada ao Sedona, para gerar um mapa de calor dos ônibus da cidade de São Paulo em 05/02/2024 às 12:00 horas. A análise será realizada para demonstrar a interoperabilidade com diversas bibliotecas e ferramentas que a arquitetura proposta permite.

A Figura 7.12 apresenta o mapa de calor da região central da cidade de São Paulo, centralizado na Estação da Sé. É possível perceber a região da Estação da Sé com a maior aglomeração, sendo uma das principais localidades na região central da cidade. Também é possivel visualizar a diferença entre o fluxo de ônibus em avenidas principais e em ruas secundárias, tendo em vista que os ônibus realizam mais trajetos em vias principais.

A Figura 7.13 representa um zoom maior na mesma região, demonstrando uma maior concentração nos terminais rodoviários próximos as estações de metrô da Sé e Anhangabau, que

[7]https://deckgl.readthedocs.io/en/latest/

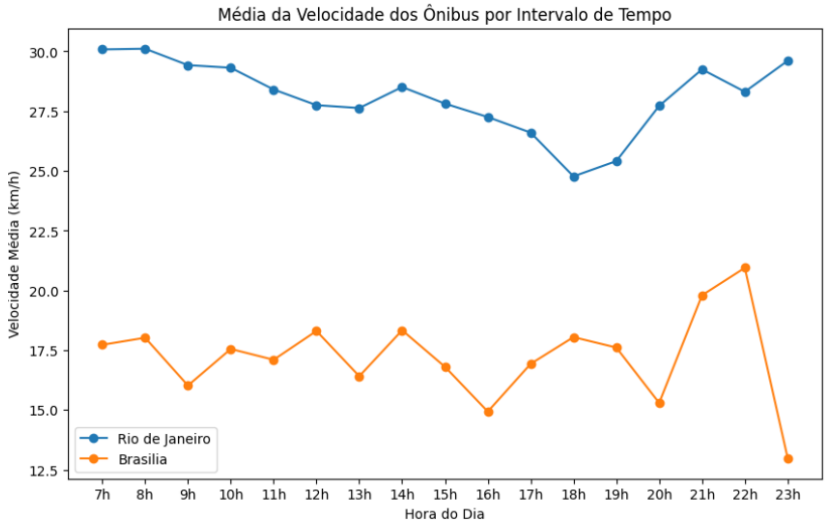

Figura 7.10: Gráfico da velocidade média das cidades de Brasília e Rio de Janeiro no dia 04/02/2024 (Domingo). Fonte: Autor [2024]

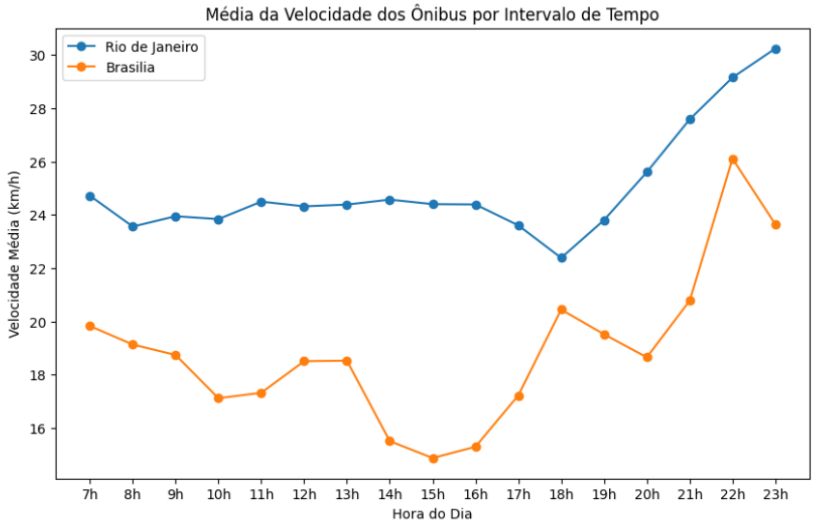

Figura 7.11: Gráfico da velocidade média das cidades de Brasília e Rio de Janeiro no dia 05/02/2024 (Segunda-feira). Fonte: Autor [2024]

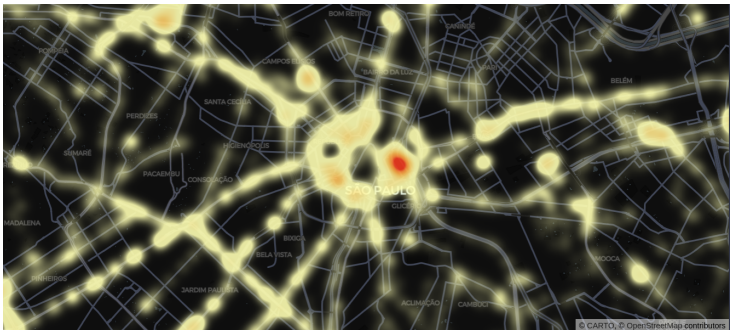

Figura 7.12: Mapa de calor da região central da cidade de São Paulo no dia 05/02/2024 as 12:00 horas. Fonte: Autor [2024]

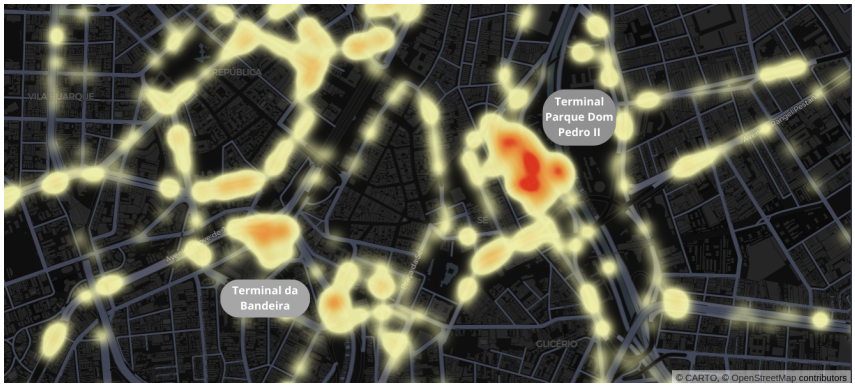

Figura 7.13: Mapa de calor ampliado da região central da cidade de São Paulo no dia 05/02/2024 as 12:00 horas. Fonte: Autor [2024]

são os terminais Parque Dom Pedro II e Bandeira, respectivamente.

A partir de análises de mapas de calor, os tomadores de decisão conseguem visualizar questões como a concentração de veículos em certas regiões em detrimento a outras, o fluxo de movimentação dos congestionamentos, entre outros fatores.

Trajetória de ônibus com o QGIS

O QGIS aceita nativamente o formato Geoparquet, utilizado na arquitetura proposta. Logo, para demonstrar essa capacidade de interoperabilidade, que evita a necessidade de mais processos para esse caso de uso, realizou-se a plotagem de três linhas de ônibus da cidade de Curitiba. A Figura 7.14 se refere à plotagem das tuplas de pontos (LAT,LONG) de três linhas de ônibus da cidade de Curitiba: 030, 507, 203. Esse tipo de visualização permite todos os pontos que foram emitidos pelos veículos, possibilitando a construção de suas trajetórias e também a análise de valores anormais, como momentos em que o ônibus teve que desviar sua rota por algum motivo específico.

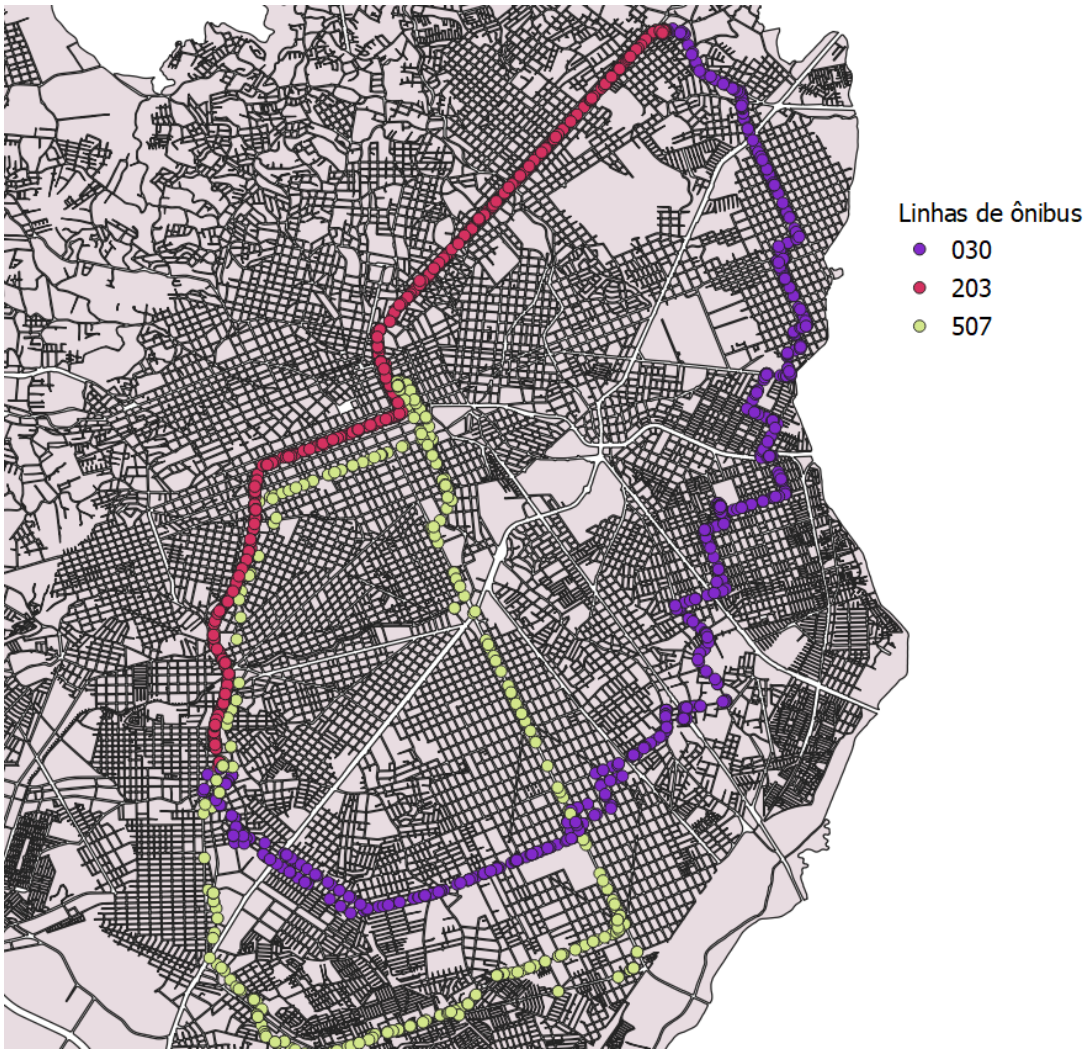

Figura 7.14: Plotagem com o QGIS dos pontos de três linhas de ônibus na cidade de Curitiba.
Fonte: Autor [2024]

8

Conclusão

8.1 Considerações Finais

Os Data Warehouses marcaram o início da era de arquiteturas de armazenamento de dados, visando a integração das informações empresariais. Em contrapartida, os Data Lakes surgiram para proporcionar flexibilidade e escalabilidade aos sistemas, fatores que faltavam ao DW. As arquiteturas 2-*Tiers* combinaram essas abordagens para otimizar os benefícios de ambas; no entanto, surgiram desafios como possíveis discrepâncias nos dados entre os sistemas, aumento da complexidade operacional e custos associados à manutenção das soluções. Essas questões ressaltam a importância contínua da inovação e da busca por soluções integradas que harmonizem flexibilidade, escalabilidade e confiabilidade na gestão e análise de dados corporativos.

Este trabalho discutiu e apresentou as principais limitações da utilização de Data Warehouse e Data Lakes como solução para um ambiente de análise de dados Big Data. Evidenciaram-se as dificuldades de lidar com governança de dados, custo de operação, dados geoespaciais, entre outros fatores. A partir disso, foi proposta a implementação de um ambiente baseado no conceito de Data Lakehouse, que busca ofertar um Data Lake com governança e features de um Data Warehouse.

O ambiente e fluxo propostos servem como linha de base para projetos de análise de dados envolvendo dados espaciais e não espaciais, possibilitando um armazenamento eficiente, de baixo custo e com capacidades analíticas semelhantes à Data Warehouses. A partir de experimentos realizados, foi demonstrada a capacidade da arquitetura de Lakehouse em prover armazenamento e análises para o setor de transporte de uma cidade inteligente, ofertando informações valiosas para os tomadores de decisões.

8.2 Trabalhos Futuros

Tendo em vista que a utilização de Lakehouses no contexto de dados espaciais, cidades inteligentes e num contexto geral é recente, existem avanços que podem ser realizados em versões futuras deste trabalho, como, por exemplo:

- Exploração extensiva das soluções de gerenciamento de metadados, com o objetivo de auxiliar a descoberta de dados no Lakehouse.
- Adaptação da arquitetura para incorporar os princípios FAIR para melhorar a explorabilidade e encontrabilidade dos dados.
- Exploração de diferentes formatos de arquivos, como formatos para dados de trajetória, importantes para dados geoespaciais.
- Utilização de soluções de particionamento e indexação espaciais especializadas para dados geoespaciais.

8.3 Contribuições

Como contribuições que resultaram do processo de construção dessa monografia, foram publicados dois artigos durante o Simpósio Brasileiro de Banco de Dados (SBBD) 2023. Os artigos foram publicados nas seguintes categorias: Workshop de Trabalhos de Alunos de Graduação (WTAG) e Dataset Showcase Workshop (DSW). O trabalho publicado no DSW recebeu a premiação de melhor artigo em sua categoria.

- F. VASCONCELOS, Felipe; T. RAMOS, Vinicius; J. COUTINHO, Fábio. **Os desafios e soluções para a implementação de Big Data Analytics em cidades inteligentes.** In: WORKSHOP DE TRABALHOS DE ALUNOS DA GRADUAÇÃO (WTAG) - SIMPÓSIO BRASILEIRO DE BANCO DE DADOS (SBBD), 38. , 2023, Belo Horizonte/MG. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023 . p. 50-56. DOI: https://doi.org/10.5753/sbbd_estendido.2023.233368.
- MELO, Ruan T.; VASCONCELOS, Felipe F.; SILVA, Rafael Luciano L.; SANTOS, Pedro Victor; RAMOS, Vinicius T.; COUTINHO, Fabio J..**BRBus - construindo um dataset para monitoramento geoespacial dos ônibus de cidades brasileiras.** In: DATASET SHOWCASE WORKSHOP (DSW), 5. , 2023, Belo Horizonte/MG. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023.

Referências bibliográficas

Apache Hudi. https://hudi.apache.org/. Acessado em 9 de abril de 2024.

Apache Iceberg. https://iceberg.apache.org/. Acessado em 9 de abril de 2024.

STAC: SpatioTemporal Asset Catalogs. https://stacspec.org/en/. Acessado em 17 de abril de 2024.

Ibtihal Alablani and Mohammed Alenazi. Edtd-sc: An iot sensor deployment strategy for smart cities. *sensors*, 20(24):7191, 2020.

Michael Armbrust, Tathagata Das, Liwen Sun, Burak Yavuz, Shixiong Zhu, Mukul Murthy, Joseph Torres, Herman van Hovell, Adrian Ionescu, Alicja Łuszczak, et al. Delta lake: high-performance acid table storage over cloud object stores. *Proceedings of the VLDB Endowment*, 13(12):3411–3424, 2020.

Michael Armbrust, Ali Ghodsi, Reynold Xin, and Matei Zaharia. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In *Proceedings of CIDR*, volume 8, page 28, 2021.

Patrick Ballantyne and Cillian Berragan. Overture poi data for the united kingdom: a comprehensive, queryable open data product. *arXiv preprint arXiv:2310.18415*, 2023.

Edmon Begoli, Pragneshkumar Patel, and J Blair Christian. Storage and read-optimized data placement structures for high-performance analysis. In *Optimization Challenges in Complex, Networked and Risky Systems*, pages 171–184. INFORMS, 2016.

Edmon Begoli, Ian Goethert, and Kathryn Knight. A lakehouse architecture for the management and analysis of heterogeneous data for biomedical research and mega-biobanks. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 4643–4651, 2021. **DOI** 10.1109/BigData52589.2021.9671534.

Vladimir Belov and Evgeny Nikulchev. Analysis of big data storage tools for data lakes based on apache hadoop platform. *International Journal of Advanced Computer Science and Applications*, 12(8), 2021.

Mohamed Cherradi. Data lakehouse: Next generation information system. In *Seminars in Medical Writing and Education*, volume 3, pages 67–67, 2024.

João Paulo Clarindo, João Pedro de Carvalho Castro, and Cristina Dutra de Aguiar. Combining fog and cloud computing to support spatial analytics in smart cities. *Journal of Information and Data Management-JIDM*, 12(4):342–360, 2021.

José Ivan Silva da Cruz Junior. Métodos de análise de dados no transporte público urbano. 2020. Monografia (Trabalho de Conclusão de Curso) - Ciência da Computação - Universidade Federal de Campina Grande, Campina Grande, 2020.

Sérgio Manuel Serra da Cruz, Luan Soares Andrade, and Jonice Oliveira Sampaio. Explorando dados abertos governamentais sobre a mobilidade urbana na cidade do rio de janeiro. In *Anais do XLIII Seminário Integrado de Software e Hardware*, pages 1645–1655. SBC, 2016.

Joao Pedro de Carvalho Castro, Anderson Chaves Carniel, and Cristina Dutra de Aguiar Ciferri. Analyzing spatial analytics systems based on hadoop and spark: A user perspective. *Software: Practice and Experience*, 50(12):2121–2144, 2020.

Andrea De Mauro, Marco Greco, and Michele Grimaldi. A formal definition of big data based on its essential features. *Library review*, 65(3):122–135, 2016.

Soukaina Ait Errami, Hicham Hajji, Kenza Ait El Kadi, and Hassan Badir. Managing spatial big data on the data lakehouse. In *International Conference on Networking, Intelligent Systems and Security*, pages 323–331. Springer, 2022.

Soukaina Ait Errami, Hicham Hajji, Kenza Ait El Kadi, and Hassan Badir. Spatial big data architecture: from data warehouses and data lakes to the lakehouse. *Journal of Parallel and Distributed Computing*, 176:70–79, 2023.

Matteo Golfarelli and Stefano Rizzi. From star schemas to big data: 20 years of data warehouse research. *A comprehensive guide through the Italian database research over the last 25 years*, pages 93–107, 2017.

Rihan Hai, Sandra Geisler, and Christoph Quix. Constance: An intelligent data lake system. In *Proceedings of the 2016 international conference on management of data*, pages 2097–2100, 2016.

Sasun Hambardzumyan, Abhinav Tuli, Levon Ghukasyan, Fariz Rahman, Hrant Topchyan, David Isayan, Mark McQuade, Mikayel Harutyunyan, Tatevik Hakobyan, Ivo Stranic, et al. Deep lake: a lakehouse for deep learning. 2023.

Ahmed A Harby and Farhana Zulkernine. From data warehouse to lakehouse: A comparative review. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 389–395. IEEE, 2022.