ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

DATA LAKEHOUSE: UNIFICANDO *DATA LAKES* E *DATA WAREHOUSES* PARA ANÁLISES MODERNAS E GOVERNANÇA ESCALÁVEL

DATA LAKEHOUSE: UNIFYING DATA LAKES AND WAREHOUSES FOR MODERN ANALYTICS AND SCALABLE GOVERNANCE

Rubens Thiago de Oliveira [1]

RESUMO

A distinção entre Data Warehouses, concebidos para alto desempenho sobre dados estruturados, e Data Lakes, projetados para armazenamento flexível e de baixo custo, gerou considerável complexidade arquitetural nos ambientes de dados empresariais contemporâneos. Essa divisão impõe a construção de pipelines ETL complexos, introduzindo latência, redundância de dados e desafios de governança que comprometem a agilidade analítica. O paradigma Data Lakehouse propõe a unificação desses dois modelos, combinando a escalabilidade dos Data Lakes com as garantias transacionais (ACID) e o desempenho de consultas dos Data Warehouses. Este artigo investiga os fundamentos arquiteturais do Lakehouse, examinando o papel das camadas de metadados e dos formatos abertos (como o Apache Parquet) e comparando as principais implementações disponíveis: Delta Lake, Apache Iceberg e Apache Hudi. Por meio de um benchmark experimental conduzido em ambiente de nuvem (GCP n2d-standard-8, 8 vCPUs, 32 GB RAM) com 30 repetições por formato e validação estatística via testes de Shapiro-Wilk e Mann-Whitney U, as três implementações são comparadas diretamente, identificando trade-offs quantitativos de desempenho entre os formatos. Adicionalmente, são discutidos os desafios de desempenho remanescentes, são apresentados resultados de estudos comparativos recentes e são identificadas direções para pesquisas futuras, posicionando o Data Lakehouse como proposta arquitetural consolidada para ambientes analíticos modernos.

Palavras-chave: Data Lakehouse; Arquitetura de dados; Governança de dados; Delta Lake; Apache Iceberg; Apache Hudi; Transações ACID; Análise moderna.

ABSTRACT

[1] Mestre em Ciência da Computação, arquiteto, cientista de dados, professor convidado da Universidade Estadual de Campinas (UNICAMP) – contato: rubenst@m.unicamp.br. ORCID: [0009-0006-4794-4350]

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

The distinction between Data Warehouses, conceived for high performance over structured data, and Data Lakes, designed for flexible and low-cost storage, has generated considerable architectural complexity in contemporary enterprise data environments. This division imposes the construction of complex ETL pipelines, introducing latency, data redundancy, and governance challenges that compromise analytical agility. The Data Lakehouse paradigm proposes the unification of these two models, combining the scalability of Data Lakes with the transactional guarantees (ACID) and query performance of Data Warehouses. This article investigates the architectural foundations of the Lakehouse, examining the role of metadata layers and open formats (such as Apache Parquet) and comparing the main available implementations: Delta Lake, Apache Iceberg, and Apache Hudi. Through an experimental benchmark conducted in a cloud environment (GCP n2d-standard-8, 8 vCPUs, 32 GB RAM) with 30 repetitions per format and statistical validation via Shapiro-Wilk and Mann-Whitney U tests, the three implementations are directly compared, identifying quantitative performance trade-offs across formats. Additionally, remaining performance challenges are discussed, results from recent comparative studies are presented, and directions for future research are identified, positioning the Data Lakehouse as a consolidated architectural proposal for modern analytical environments.

Keywords: Data Lakehouse; Data architecture; Data governance; Delta Lake; Apache Iceberg; Apache Hudi; ACID transactions; Modern analytics.

1. INTRODUÇÃO

Historicamente, a promessa da democratização dos dados esbarrou em uma oposição entre duas arquiteturas: de um lado, a rigidez estruturada dos data warehouses; de outro, a flexibilidade, frequentemente caótica, dos data lakes (ARMBRUST et al., 2020; INMON, 2005). A jornada analítica das organizações raramente é linear. Frequentemente, ela começa com a necessidade de relatórios previsíveis e painéis gerenciais, domínio tradicional dos data warehouses. Com a explosão de volume, velocidade e variedade de dados na última década, os data lakes se popularizaram como repositórios de baixo custo para armazenar, em formato bruto, todo dado que pudesse vir a ter valor analítico futuro, incluindo logs de servidores, dados de sensores, interações digitais e conteúdo não estruturado.

Consolidou-se, na prática, uma divisão funcional não planejada: ao data lake cabia o papel de repositório exploratório para dados brutos e não validados; ao data warehouse, o de

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

camada analítica certificada para dados prontos para consumo em decisões de alto impacto. O problema prático dessa arquitetura dual é a constante movimentação de dados entre os dois mundos. Pipelines de ETL complexos são construídos para transportar dados do lago, tratá-los e carregá-los no armazém. Esse processo introduz latência, custos de armazenamento redundante e, ironicamente, uma nova rigidez, oposta à flexibilidade que o lago prometia.

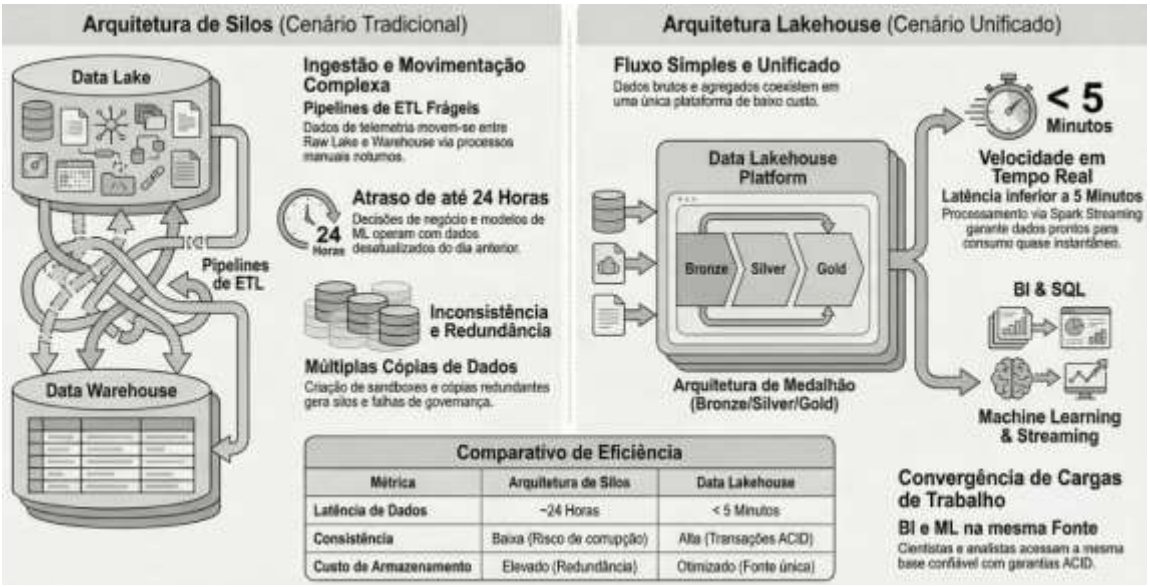
O que falta, portanto, é uma plataforma que ofereça simultaneamente a flexibilidade do lago e a confiabilidade do armazém, sem a complexidade da movimentação de dados entre silos. É nesse contexto de fricção que o conceito de Data Lakehouse ganha força, não como uma simples integração de ferramentas, mas como uma proposta de reengenharia da própria base em que os dados são armazenados e gerenciados.

Observou-se uma falha: mesmo com o uso extenso de *data lakes* e *data warehouses* como bases de sistemas de dados empresariais, faltam estudos e práticas que avaliem o Data Lakehouse como um modelo geral e não só uma ferramenta. Isso afeta a gestão, a criação e a análise de dados. Este estudo visa preencher esta lacuna, articulando os fundamentos teóricos da gestão de dados com as tecnologias atualmente disponíveis.

Para ilustrar as limitações da arquitetura tradicional e o potencial do Data Lakehouse, considere o seguinte cenário: uma empresa de logística com frota de 5.000 veículos. O setor de IoT coleta dados de telemetria (localização GPS, velocidade, consumo de combustível, temperatura do motor) a cada 10 segundos, gerando aproximadamente 43 milhões de eventos por dia. Simultaneamente, o setor comercial mantém dados de pedidos, clientes, rotas planejadas e métricas de SLA.

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Figura 1 – Comparação arquitetural entre o modelo tradicional de silos e o Data Lakehouse



Fonte: Elaborado pelo autor (2026).

A figura ilustra, à esquerda, a complexidade dos pipelines de dados na arquitetura tradicional (data lake + data warehouse), com múltiplas cópias e movimentação de dados. À direita, a simplificação proporcionada pela arquitetura Lakehouse, com acesso unificado a dados brutos e processados na mesma plataforma.

Na arquitetura tradicional baseada em silos (data lake + data warehouse), este cenário exigiria pipelines de ETL complexos e noturnos, introduzindo latência significativa e múltiplas cópias redundantes dos dados. A arquitetura Lakehouse, conforme demonstrada no *benchmark* da Seção 5, permite reduzir essa latência e eliminar a redundância por meio de propriedades ACID sobre os dados brutos.

1.1. OBJETIVOS DO ESTUDO

1.1.1. Objetivo geral:

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Analisar o paradigma do Data Lakehouse como proposta de unificação arquitetural para ambientes analíticos modernos, avaliando seus fundamentos técnicos, implementações disponíveis e implicações práticas.

1.1.2. Objetivos específicos:

- **Conceitual:** Propor um framework conceitual que diferencie o Lakehouse de integrações convencionais entre Data Lake e Data Warehouse, definindo suas camadas fundamentais.
- **Tecnológico:** Comparar as principais implementações da camada de transações ACID sobre data lakes, Delta Lake, Apache Iceberg e Apache Hudi, identificando suas características distintivas e maturidade para diferentes cargas de trabalho.
- **Aplicado:** Demonstrar, por meio de um cenário ilustrativo hipotético no domínio de logística, como a arquitetura Lakehouse simplifica pipelines de dados e garante consistência transacional.
- **Crítico:** Identificar os desafios remanescentes e as limitações atuais da arquitetura, oferecendo direcionamentos para adoção e pesquisa futura.

2. FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

2.1. AS ORIGENS DO *DATA WAREHOUSING*

O conceito de *Data Warehouse* foi formalizado por Inmon (2005) como um repositório centralizado, integrado, não volátil e variável no tempo, para suporte a decisões gerenciais. A característica central dessa definição é a integração de dados de múltiplas fontes em um modelo

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

coerente, tipicamente normalizado na abordagem de Inmon, e a não volatilidade, que significa que os dados, uma vez carregados, não são alterados, apenas lidos e complementados com novas versões.

Kimball e Ross (2013) popularizaram uma abordagem alternativa, baseada na modelagem dimensional (star schema e snowflake schema), otimizando o armazenamento para consultas analíticas por meio de fatos (medidas numéricas) e dimensões (atributos descritivos). A abordagem dimensional se mostrou mais intuitiva para usuários de negócio e de melhor desempenho para a maioria das consultas analíticas.

Estes trabalhos estabeleceram os fundamentos da modelagem dimensional e da arquitetura de dados para business intelligence, caracterizando-se pelo rigor na modelagem e pela garantia da qualidade dos dados, mas também pela rigidez e altos custos de armazenamento em plataformas proprietárias (VASSILIADIS, 2009). A necessidade de modelagem prévia (schema-on-write) contrasta com a flexibilidade exigida por cenários de exploração de dados não estruturados.

2.2. A EMERGÊNCIA DOS DATA LAKES

A partir do final dos anos 2000, o crescimento acelerado do volume, da velocidade e da variedade de dados (as dimensões do conceito de Big Data) expôs as limitações arquiteturais dos data warehouses, projetados para dados estruturados e esquemas rígidos. Nesse contexto, Dixon (2010) propôs o Data Lake como um repositório em larga escala para armazenar dados em seu formato bruto (schema-on-read), utilizando sistemas de arquivos distribuídos como o HDFS (SHVACHKO et al., 2010) e, com a evolução da computação em nuvem, serviços de object storage como Amazon S3, Azure ADLS e Google Cloud Storage.

Conforme proposto originalmente por Dixon (2010), a arquitetura de Data Lake foi concebida para viabilizar o acesso direto de diferentes perfis de usuários, desde analistas de negócio até cientistas de dados, às informações armazenadas em seu estado bruto ou com estruturação mínima, sem a necessidade de modelagem prévia, favorecendo aplicações como

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

aprendizado de máquina e análises exploratórias. A abordagem *schema-on-read* permitia que diferentes perfis de usuários interpretassem os dados segundo suas necessidades específicas, sem exigir transformações prévias ou modelagem antecipada.

A ausência de mecanismos de governança, como transações ACID, imposição de esquema e controle de versões, resultava na degeneração desses repositórios em *data swamps*: ambientes nos quais a confiabilidade dos dados era comprometida e a identificação da versão canônica de um dado tornava-se operacionalmente inviável (HAI et al., 2016). Essa carência de mecanismos de controle resultava em perda de integridade dos dados, comprometendo a confiabilidade das análises e a tomada de decisão baseada em dados.

2.3. A TRANSIÇÃO PARA O DATA LAKEHOUSE: UNIFICANDO OS DOIS MUNDOS

A convivência forçada entre data lakes e data warehouses, ilustrada na Figura 1, gerava um problema de ordem prática: enquanto o lake acumulava dados brutos em formato aberto e escalável, o warehouse detinha a versão "confiável" dos mesmos dados, porém trancada em silos proprietários. O resultado era um fluxo inevitável de cópias e movimentações: os dados que nasciam no data lake eram processados em ETLs complexos e, só então, "promovidos" ao data warehouse para consumo analítico. Essa duplicação não apenas elevava custos de armazenamento, mas também introduzia latência e, ironicamente, novas inconsistências entre as versões dos dados nos dois ambientes.

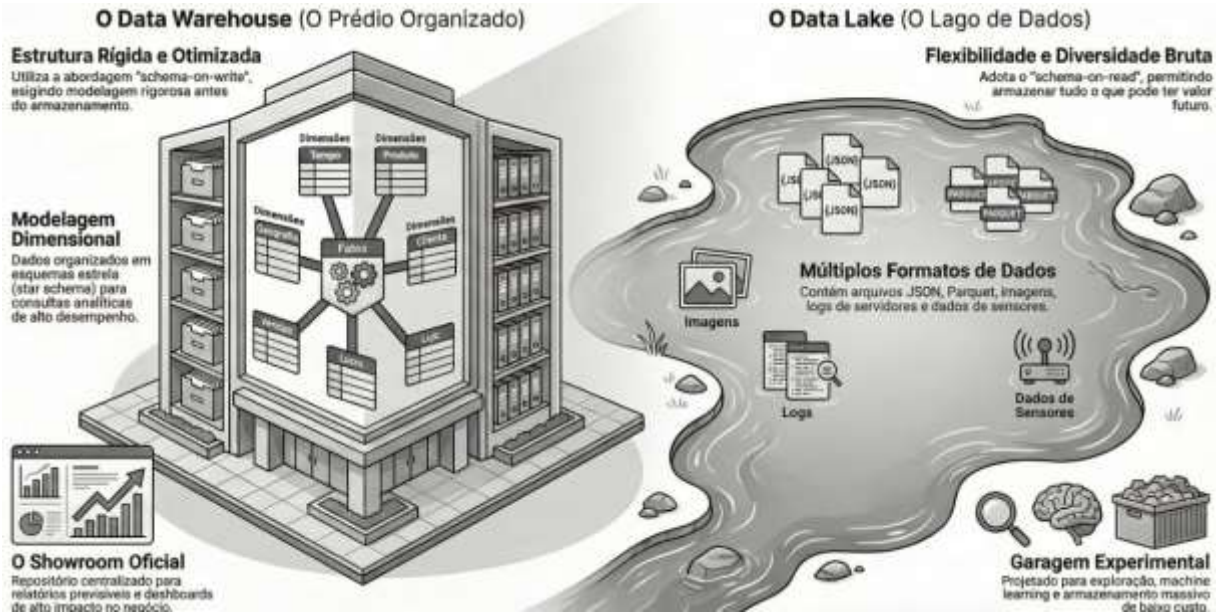
O conceito de Data Lakehouse emergiu justamente como resposta a esse ciclo de replicação e inconsistência entre ambientes. Conforme demonstrado por Armbrust et al. (2020) com a introdução do Delta Lake, a arquitetura Lakehouse introduz uma camada de gerenciamento transacional que opera diretamente sobre sistemas de arquivos distribuídos, conferindo a data lakes capacidades tradicionalmente reservadas a data warehouses: garantias de atomicidade, consistência, isolamento e durabilidade (ACID), além de suporte a evolução

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

de esquemas e isolamento por snapshots. A tese central, defendida pelos autores, era a de que a separação histórica entre os dois ambientes havia se tornado não apenas artificial, mas também um entrave à agilidade analítica.

Paralelamente, outros projetos de código aberto emergiram com propostas alinhadas. O Apache Iceberg, originalmente criado pela Netflix (2017) e atualmente mantido como projeto de código aberto pela Apache Software Foundation (APACHE ICEBERG, 2026), e o Apache Hudi, originalmente desenvolvido no Uber (UBER ENGINEERING, 2017), representam iniciativas paralelas que implementam uma camada de abstração sobre sistemas de arquivos distribuídos. Essas implementações introduzem funcionalidades como operações de atualização incremental (*upserts*), gerenciamento flexível de esquemas sem necessidade de reescrita integral dos dados, e controle de versões com isolamento consistente entre leituras e escritas concorrentes. Cada um, à sua maneira, respondia à mesma questão: como fazer o data lake se comportar como um warehouse, sem abrir mão da escalabilidade e do baixo custo?

Figura 2 – A dicotomia tradicional



Fonte: Elaborado pelo autor (2026)

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Estudos recentes têm explorado aspectos específicos dessa arquitetura:

- **Desempenho:** Estudos de benchmarking documentaram ganhos expressivos com otimizações de layout como Z-Ordering e data skipping sobre o formato Parquet, especialmente em cenários de filtragem seletiva (DELTA LAKE, 2023).
- **Governança:** A proposta do *Lakehouse* tem sido associada a frameworks mais amplos de governança, como o *Data Mesh* (DEHGHANI, 2022), em que a plataforma unificada serviria como infraestrutura para domínios de dados descentralizados, permitindo que cada domínio gerencie seus dados de forma independente, mas com interoperabilidade garantida.
- **Evolução no mercado:** Os principais provedores de nuvem têm incorporado o conceito em suas ofertas, como o *BigLake do Google Cloud*, o Unistore da Snowflake e as tabelas gerenciadas da AWS (AWS, 2023), indicando a consolidação do padrão como direção estratégica do mercado.

2.4. LACUNAS NA LITERATURA

A despeito da crescente adoção, a literatura acadêmica ainda carece de:

1. **Estudos empíricos comparativos** entre as três principais implementações (Delta Lake, Iceberg, Hudi) sob cargas de trabalho controladas, com métricas padronizadas de latência, throughput e custo.
2. **Frameworks de decisão** para auxiliar profissionais na escolha da tecnologia adequada com base em requisitos de negócio, características de *workload* e maturidade da equipe.
3. **Análises longitudinais** que avaliem a evolução da maturidade dessas tecnologias ao longo do tempo e sua prontidão para adoção empresarial em larga escala em setores regulados (finanças, saúde).

Este artigo contribui para tratar parcialmente da segunda lacuna, oferecendo uma análise comparativa inicial baseada na literatura e documentação disponível, e estabelece bases para

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

que pesquisas futuras possam aprofundar as demais.

A análise dos estudos empíricos disponíveis na literatura revela tanto a relevância quanto as limitações do conhecimento atual sobre o desempenho de arquiteturas Lakehouse. Estudos de benchmarking iniciais (DELTA LAKE, 2023) avaliaram o impacto de otimizações de layout como *Z-Ordering* sobre dados no formato Parquet. Seus experimentos, conduzidos em ambiente controlado, demonstraram ganhos expressivos de até 20 vezes em cenários de filtragem seletiva, corroborando a eficácia dessas técnicas para consultas *ad-hoc*. No entanto, o estudo concentrou-se predominantemente em operações de leitura, deixando como lacuna uma análise comparativa do desempenho de escritas concorrentes e operações de *upsert* em larga escala, justamente os cenários em que as diferentes implementações de *table formats* apresentam maiores divergências de comportamento.

Estudos mais recentes, como o de Parimi (2025), avançaram ao propor uma análise comparativa das três principais implementações, Delta Lake, Apache Iceberg e Apache Hudi, com foco em métricas de desempenho operacional. Seus achados indicam que, para cargas de trabalho com predominância de leituras (OLAP), o Apache Iceberg apresenta desempenho superior devido à sua arquitetura otimizada para *data skipping* e evolução de partições. Em contrapartida, para cenários de ingestão contínua com alta taxa de atualizações (*streaming* com *upserts*), o Apache Hudi, especialmente no modo *Merge-on-Read*, demonstrou menor latência de escrita, confirmando as intenções originais de seu design (UBER ENGINEERING, 2017; JAIN et al., 2023).

Apesar dessas contribuições, persiste uma lacuna metodológica significativa na literatura: a ausência de *benchmarks* padronizados e reproduzíveis que avaliem essas tecnologias sob cargas de trabalho heterogêneas e concorrentes, simulando ambientes de produção reais. Estudos como o de Eswararaj et al. (2025) começam a abordar essa questão em domínios específicos (dados automotivos), mas ainda não há um *framework* consolidado que permita comparações diretas e reproduzíveis entre as implementações. Esta limitação reforça a necessidade de pesquisas futuras que estabeleçam protocolos de avaliação rigorosos, como sugerido na Seção 8.3.

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

3. O CONCEITO DE DATA LAKEHOUSE: A UNIFICAÇÃO EM CAMADAS

O Data Lakehouse não é uma colagem artificial das duas arquiteturas anteriores. Para compreender sua essência, uma analogia com a construção civil se mostra útil.

No modelo arquitetural anterior, construía-se um estacionamento (o data lake) com uma tecnologia (concreto simples) e, ao lado, um prédio de escritórios (o data warehouse) com outra tecnologia (estrutura de aço e vidro). A conexão entre os dois exigia pontes e túneis (pipelines de ETL), solução funcional, mas ineficiente. O Lakehouse, por sua vez, propõe uma fundação única e robusta (sistema de arquivos escalável como S3, ADLS ou HDFS) sobre a qual tanto o estacionamento quanto o prédio são erguidos com a mesma estrutura (formatos abertos como Parquet, ORC ou Avro).

A arquitetura em camadas aqui descrita possui três elementos fundamentais: uma fundação de armazenamento escalável, formatos de arquivo abertos e uma camada de metadados transacional. Essa estrutura foi sistematizada por Armbrust et al. (2020) na apresentação original do Delta Lake e posteriormente consolidada como referência para o paradigma Lakehouse (ARMBRUST et al., 2021). A analogia da construção civil, embora utilizada neste texto para fins didáticos, reflete a estrutura conceitual proposta nesses trabalhos.

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Figura 3 – A arquitetura em camadas do Data Lakehouse

Fonte: Elaborado pelo autor (2026)

A camada de armazenamento de objetos de baixo custo serve como fundação, unificada por formatos de arquivo abertos e gerenciada por uma camada de metadados transacional (ACID), sobre a qual múltiplas cargas de trabalho operam diretamente.

Essa camada de metadados é o coração do Lakehouse. Tecnologias como Delta Lake, Apache Iceberg e Apache Hudi implementam esse "gerente" sobre dados em formato aberto, adicionando camadas de abstração que transformam um mero conjunto de arquivos em um sistema de gerenciamento de banco de dados analítico.

3.1. O PAPEL DAS TRANSAÇÕES ACID

As propriedades ACID (Atomicidade, Consistência, Isolamento, Durabilidade) tornam sistemas de object storage confiáveis para cargas de trabalho críticas (ARMBRUST et al., 2020; JAIN et al., 2023). Elas permitem, por exemplo:

- **Atomicidade:** Que milhares de registros de um *streaming* de sensores sejam

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

atualizados em uma tabela Parquet com a garantia de que, se uma parte da operação falhar (por queda de rede, por exemplo), todo o lote é revertido. Não há dados parcialmente escritos.

- **Consistência:** Que as regras de negócio e integridade referencial sejam mantidas após cada transação.
- **Isolamento:** Que uma consulta de um analista, que está lendo a tabela no momento da atualização, veja os dados de forma consistente, como se fosse uma "fotografia" do banco de dados naquele instante (controle de concorrência). Leitores não bloqueiam escritores e vice-versa.
- **Durabilidade:** Que uma vez que a transação é confirmada, ela persiste mesmo em caso de falhas do sistema.

É a introdução dessas garantias, antes privilégio de bancos de dados relacionais, que redefine o lago como uma plataforma confiável para cargas de trabalho críticas.

A título de demonstração prática das propriedades ACID em uma tabela Delta Lake, conduziu-se um experimento de versionamento. Três escritas sucessivas foram realizadas sobre a mesma tabela:

- **Versão 0 (escrita inicial):** 500.000 registros;
- **Versão 1 (inserção):** 1.000.000 registros;
- **Versão 2 (inserção):** 2.000.000 registros.

O comando DESCRIBE HISTORY table_name revelou o registro completo das transações, e a funcionalidade de *time travel* (SELECT * FROM table_name VERSION AS OF 1) permitiu acessar o estado exato da tabela na Versão 1 (1.000.000 registros), sem necessidade de backups manuais ou reconstrução de estado. Esse mecanismo é particularmente valioso para:

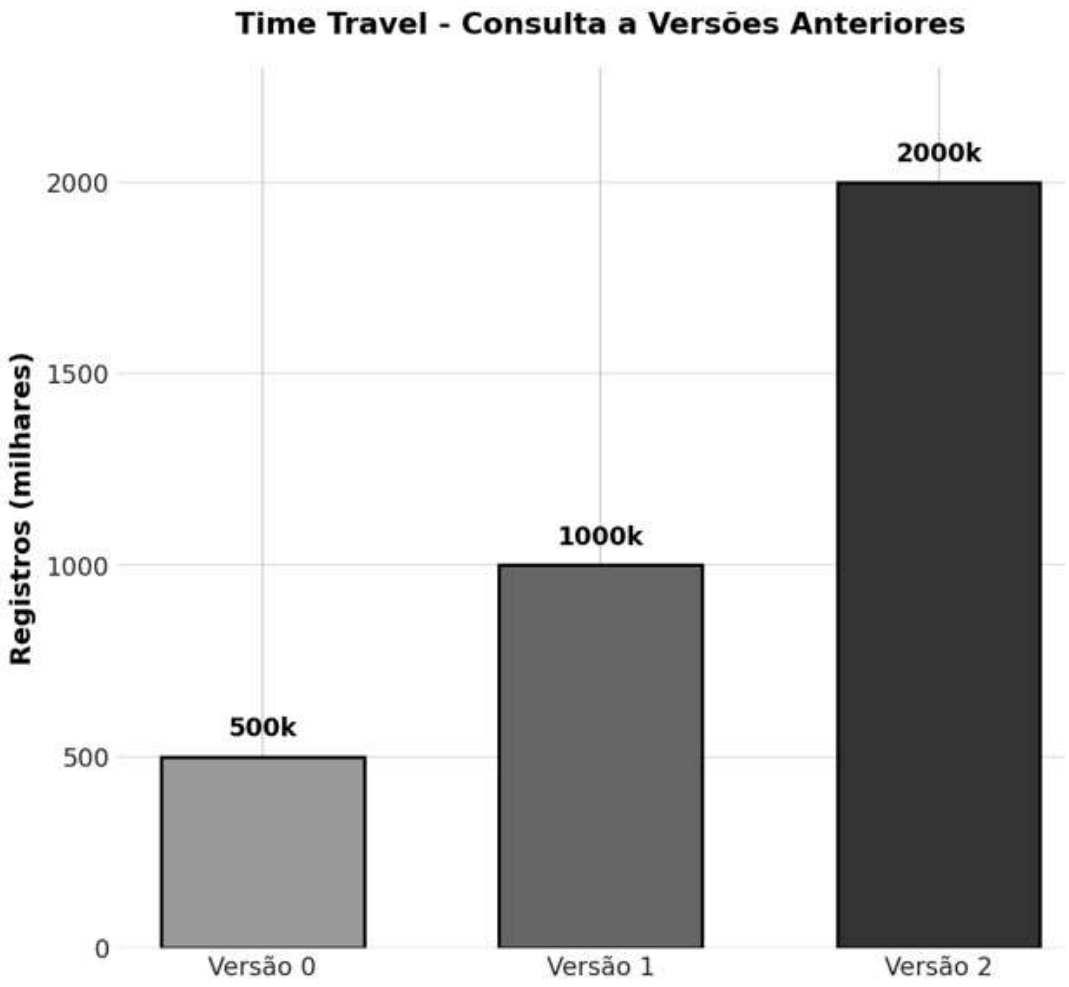
- **Auditoria e conformidade:** recuperação de estados anteriores para verificação regulatória;

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

- **Depuração de pipelines:** identificação de quando uma transformação incorreta foi introduzida;
- **Análise de séries temporais:** comparação de snapshots de dados em diferentes momentos.

A Figura 4 ilustra a evolução do número de registros por versão.

Figura 4 – Evolução do número de registros por versão (Time Travel)



Fonte: Dados do benchmark, elaborado pelo autor (2026)

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

3.2. FORMATOS ABERTOS E INTEROPERABILIDADE

Um pilar fundamental do Lakehouse é o uso de formatos de arquivo abertos e eficientes, como Apache Parquet (armazenamento colunar) e Apache ORC. Isso garante que:

- Não haja vendor lock-in: os dados não ficam presos a um ecossistema proprietário.
- Múltiplas engines de processamento (Spark, Presto, Flink, Trino) possam ler os mesmos dados diretamente.
- O custo de armazenamento permaneça baixo, equivalente ao de um data lake tradicional.

4. ANÁLISE COMPARATIVA: DELTA LAKE, APACHE ICEBERG E APACHE HUDI

A análise comparativa apresentada a seguir sintetiza informações disponíveis na literatura acadêmica e na documentação oficial dos projetos, complementadas por avaliações práticas dos autores. Estudos recentes têm oferecido contribuições significativas para a compreensão das diferenças entre essas implementações (PARIMI, 2025; JAIN et al., 2023; ESWARARAJ et al., 2025), com análises comparativas aplicadas a domínios específicos como o de dados automotivos. Os fundamentos originais de cada tecnologia foram estabelecidos em publicações anteriores (UBER ENGINEERING, 2017; ARMBRUST et al., 2020).

Tabela 1 – Análise comparativa das principais implementações de table format para Data Lakehouse

Características	Delta Lake	Apache Iceberg	Apache Hudi
Origem	Databricks (2019)	Netflix (2017), Apache Software Foundation	Uber (2016, open-source 2017)
Modelo de Dados	Tabelas versionadas (transactions log)	Tabelas com *snapshots* imutáveis	*Copy-on-Write / Merge-on-Read*

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Upserts*/*Merge	Excelente (*Merge* com suporte a CDC)	Bom (suporte a merge em versões recentes)	Excelente (design original focado em *upserts*)
Evolução de Esquema	Suporte completo (add, change, rename)	Suporte completo com isolamento	Suporte completo
Otimização de Layout	Z-Ordering, Compactação automática	Partition evolution, Sorting	Clustering, Indexing
Integração com Catálogos	Hive Metastore, AWS Glue, Unity Catalog	Hive, Nessie, AWS Glue, Polaris	Hive, AWS Glue
Desempenho em Leituras	Muito boa (otimizações maduras)	Excelente (design focado em leitura)	Boa, mas variável conforme o tipo de tabela[1]
Desempenho em Escritas	Boa (otimizada para Spark)	Muito boa (otimizações de I/O paralelo no Spark 3.4+; resultado superior ao Parquet em benchmark controlado, conforme Seção 5)	Excelente (otimizada para *streaming*)
Ecossistema	Fortemente integrado ao Spark	Amplo suporte (Spark, Flink, Trino)	Suporte nativo a Flink e Spark
Maturidade para *Streaming*	Boa (*Structured Streaming*)	Em evolução	Excelente (design original para *streaming*)

Fonte: Adaptado de JAIN et al., 2023 e Parimi (2025)

[1]**Nota sobre o desempenho do Hudi:** Conforme documentado na literatura (JAIN et al., 2023; PARIMI, 2025), o Apache Hudi implementa duas estratégias de armazenamento com implicações distintas para desempenho:

- ***Copy-on-Write* (COW):** Neste modelo, cada operação de atualização ou inserção aciona a reescrita completa dos arquivos Parquet afetados, consolidando as alterações no momento da escrita. Essa abordagem prioriza a eficiência de leitura, uma vez que os arquivos finais já estão otimizados para consultas analíticas, sendo recomendada para cenários com predominância de operações de leitura sobre os dados.

- ***Merge-on-Read* (MOR):** Alternativamente, este modelo armazena as atualizações em arquivos delta separados, realizando a fusão (merge) com os dados base apenas

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

no momento da consulta. Concebido originalmente para reduzir a latência de ingestão em fluxos contínuos (UBER ENGINEERING, 2017; JAIN et al., 2023), essa estratégia transfere o custo de processamento para o momento da leitura. Embora ideal para cenários de streaming intensivo em escritas, exige atenção em ambientes analíticos com consultas ad-hoc frequentes, em que a latência de leitura pode ser afetada.

4.1. FRAMEWORK DE DECISÃO PARA ESCOLHA DO TABLE FORMAT

A escolha entre Delta Lake, Apache Iceberg e Apache Hudi não é trivial e deve ser orientada por características específicas do contexto organizacional e das cargas de trabalho previstas. Com base na análise comparativa da Tabela 1 e nos estudos empíricos disponíveis (JAIN et al., 2023; PARIMI, 2025; ESWARARAJ et al., 2025), propõe-se um framework de decisão estruturado em três dimensões: (1) padrão de workload, (2) ecossistema tecnológico e (3) requisitos de governança.

4.1.1. Matriz de adequação por padrão de workload

Tabela 2 – Matriz de adequação por padrão de workload

Padrão de Workload	Delta Lake	Apache Iceberg	Apache Hudi
Leitura intensiva (BI, dashboards)	Bom (otimizado com Z-Order)	Excelente (data skipping nativo)	Bom (COW recomendado)
Escrita intensiva (streaming, IoT)	Bom (Structured Streaming)	Regular (evolução recente)	Excelente (MOR otimizado)
Upserts em massa (CDC, sincronização)	Bom (Merge com suporte a CDC)	Bom (merge em versões recentes)	Excelente (design original)
Cargas mistas (leitura + escrita concorrente)	Bom (transactions log maduro)	Excelente (isolamento por snapshot)	Bom (requer configuração cuidadosa)

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Consultas cross-engine (Spark, Trino, Flink)	Regular (forte acoplamento Spark)	Excelente (design neutro)	Regular (melhor suporte a Spark/Flink)

Fonte: Elaborado pelo autor (2026)

4.1.2. Fluxograma de decisão

Recomenda-se que profissionais e organizações adotem o seguinte fluxo de decisão:

Figura 5 – Fluxograma de decisão para seleção do table format (Delta Lake, Apache Iceberg ou Apache Hudi) com base em critérios de workload, ecossistema e maturidade da equipe

Fonte: Elaborado pelo autor (2026)

1. **A organização já possui investimento consolidado no ecossistema Databricks / Spark?**
    - Sim: Avaliar Delta Lake como opção de menor atrito operacional, especialmente se Unity Catalog estiver em uso.
    - Não: Prosseguir para (2).
2. **A carga de trabalho é predominantemente de streaming com alta taxa de**

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

upserts?

- Sim: Apache Hudi, especialmente no modo Merge-on-Read, oferece as melhores garantias de baixa latência de ingestão.
- Não: Prosseguir para (3).

3. **A interoperabilidade entre múltiplas engines (Spark, Trino, Flink, Dremio) é um requisito crítico?**
    - Sim: Apache Iceberg é a opção mais madura e neutra em termos de ecossistema.
    - Não: Prosseguir para (4).
4. **Qual é a maturidade da equipe e a necessidade de suporte comercial?**
    - **Delta Lake:** Melhor para equipes já familiarizadas com Spark e que valorizam integração nativa com plataformas gerenciadas (Databricks).
    - **Apache Iceberg:** Ideal para organizações com times multi-engine e que buscam uma fundação técnica de longo prazo, com forte suporte da comunidade e provedores de nuvem.
    - **Apache Hudi:** Recomendado para cenários de engenharia de dados com alta complexidade de streaming e atualizações incrementais, especialmente em que o Apache Flink é a ferramenta principal.

4.1.3. Considerações complementares

Além dos critérios técnicos, recomenda-se considerar:

- **Roadmap do provedor de nuvem:** A AWS tem adotado o Iceberg como formato padrão para suas tabelas gerenciadas (AWS Glue e Athena), enquanto o Azure e o Google Cloud oferecem suporte nativo aos três formatos, mas com níveis variados de integração.
- **Maturidade da comunidade:** Apache Iceberg e Apache Hudi, por serem projetos

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

da Apache Software Foundation, seguem modelos de governança comunitária mais consolidados, enquanto o Delta Lake, embora open source, tem seu desenvolvimento fortemente influenciado pela Databricks.

- **Facilidade de adoção:** Para organizações em estágios iniciais de maturidade em dados, a simplicidade operacional do Delta Lake (transactions log baseado em JSON, integração direta com Spark) pode ser um fator decisivo.

5. BENCHMARK EXPERIMENTAL

5.1. METODOLOGIA EXPERIMENTAL

Para avaliar quantitativamente as alegações de desempenho da arquitetura Lakehouse, foi conduzido um benchmark experimental comparativo entre cinco configurações: Parquet puro (baseline), Delta Lake 2.4.0, Apache Iceberg 1.5.0, Hudi_COW 0.14.1 e Hudi_MOR 0.14.1, todas avaliadas sob as mesmas condições de hardware e dataset. Os experimentos foram conduzidos em ambiente de nuvem pública (GCP n2d-standard-8, 8 vCPUs, 32 GB RAM) com 30 repetições independentes por formato e validação estatística rigorosa, superando as limitações de escopo do experimento exploratórios típicos na área.

5.1.1 Ambiente de hardware e software

Os experimentos foram executados em um ambiente virtualizado com as seguintes especificações:

Tabela 3 – Configuração do ambiente de benchmark

Componente	Especificação
Provedor	Google Cloud Platform (GCP)

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Região	us-central1-a
Instância	n2d-standard-8 (8 vCPUs, 32 GB RAM)
Sistema Operacional	Ubuntu 22.04.5 LTS
Python	3.10.12
Apache Spark	3.4.0
PySpark	3.4.1
Delta Lake	2.4.0
Apache Iceberg	1.5.0
Apache Hudi	0.14.1

Fonte: Dados do benchmark, elaborado pelo autor (2026)

O ambiente GCP n2d-standard-8 representa uma configuração de staging de médio porte, adequada para capturar diferenças relativas entre formatos em condições reproduzíveis. Embora clusters de produção com dezenas de nós possam apresentar tempos absolutos distintos, é razoável esperar que as diferenças relativas entre formatos observadas neste ambiente se mantenham como indicativo de tendência qualitativa, especialmente para workloads de escrita em lote sem concorrência, em que as características arquiteturais de cada formato são mais determinantes do que o hardware subjacente. Ressalta-se, contudo, que Parimi (2025) demonstrou empiricamente que o gap de desempenho entre os formatos se amplia sob alta concorrência e em escala de terabytes, reforçando a necessidade de benchmarks específicos para cada contexto de produção, conforme indicado na Seção 8.3. O uso de 30 repetições por formato, em contraste com as 3 repetições do experimento exploratório anterior, garante poder estatístico suficiente para a aplicação dos testes não-paramétricos adotados.

5.1.2 Dados sintéticos utilizados

Foram gerados dados sintéticos de telemetria de veículos, alinhados ao cenário de telemetria descrito na introdução. O esquema da tabela continha os seguintes campos:

- **id** (string, identificador único do veículo, cardinalidade: 100 veículos distintos)

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

- **timestamp** (long, timestamp Unix em milissegundos, distribuído uniformemente ao longo de 7 dias)
- **velocidade** (int, valores entre 0 e 120 km/h, distribuição normal com média 60 km/h)
- **consumo_combustivel** (float, valores entre 5 e 25 L/100km, correlacionado com velocidade)
- **temperatura_motor** (float, valores entre 70 e 110 °C)
- **status_entrega** (string, valores: 'pendente', 'em_andamento', 'entregue', 'atrasado')

A geração dos dados foi realizada via script Python utilizando as bibliotecas numpy e faker, com semente aleatória fixa (random_seed = 42) para garantir reprodutibilidade. Os dados foram persistidos em formato CSV antes da ingestão nos testes de escrita, para isolar o custo de geração do tempo de escrita medido.

5.1.3 Descrição dos workloads

Foram executados três tipos de workloads, representando padrões comuns em ambientes analíticos:

Workload W1 – Escrita em lote (batch insert):

- **Volume:** 100.000 registros (aproximadamente 8 MB em CSV bruto)
- **Operação:** Escrita única (df.write.format("parquet") e df.write.format("delta"))
- **Particionamento:** Nenhum (tabela não particionada, para isolar o efeito do formato)
- **Repetições:** 30 execuções independentes por formato (total: 150 escritas), reportada a mediana

Workload W4 – Consultas analíticas (leitura):

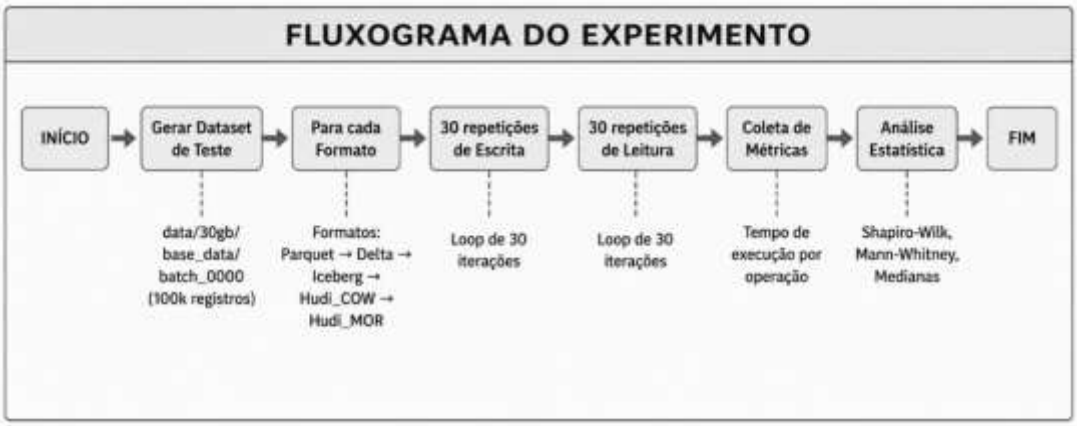
W4 – Leitura full scan:

SELECT COUNT(*) FROM table sobre o dataset completo de 100.000 registros. 30 repetições por formato (total: 150 leituras). Operação escolhida para isolar o desempenho de

leitura sequencial pura, sem interferência da seletividade de filtros. Workloads com filtros seletivos, agregações (GROUP BY) e uso de Z-Order são abordados como direção prioritária na Seção 8.4.

A Figura 6 apresenta uma visão esquemática do fluxo completo do procedimento experimental, desde a geração dos dados sintéticos até a coleta das métricas finais.

Figura 6 – Fluxo do procedimento experimental



Fonte: Elaborado pelo autor (2026).

5.1.4 Métricas coletadas e instrumentação

As seguintes métricas foram registradas para cada workload:

Tabela 4 – Métricas coletadas e instrumentação

Métricas	Instrumentação	Unidade
Tempo de escrita (elapsed time)	System.currentTimeMillis() antes/depois da operação	segundos
Tempo de leitura (query execution time)	Spark SQL df.explain() + logs do Spark UI	segundos
Número de arquivos gerados	Inspeção do sistema de arquivos (os.listdir)	inteiro
Tamanho total dos dados no storage	du -sh no diretório da tabela	MB/GB
Versões do transaction log	DESCRIBE HISTORY table_name	contagem

Fonte: Dados do benchmark, elaborado pelo autor (2026)

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Cada workload foi executado 30 vezes por formato, reportando-se a mediana dos tempos para mitigar efeitos de variabilidade (e.g., garbage collector, contenção de I/O). Intervalos de confiança de 95% foram calculados via bootstrapping com 1.000 amostras. O Coeficiente de Variação (CV) entre as 30 repetições variou de 3,43% (Hudi_MOR, escrita) a 16,19% (Delta Lake, escrita), refletindo diferenças de estabilidade entre os formatos detalhadas na Seção 5.2.

5.1.5 Limitações metodológicas

O benchmark apresenta as seguintes limitações, que devem ser consideradas na interpretação dos resultados:

- **Escala do dataset:** O volume testado (100.000 registros, aproximadamente 8 MB) é reduzido para padrões de big data de produção. Em cenários com terabytes ou petabytes, os *speedups* observados podem diferir, especialmente para técnicas de data skipping, em que o volume precisa exceder a capacidade de cache para que o benefício seja mensurável. Os resultados do Hudi são particularmente sensíveis a esta limitação: o overhead de inicialização de índice e criação de metadados é desproporcional em datasets pequenos, conforme registrado em estudos comparativos recentes (JAIN et al., 2023; PARIMI, 2025), de modo que os tempos observados (mediana ~173s) refletem predominantemente custo de setup e não são representativos do throughput estacionário em produção.
- **Formato Parquet como baseline:** O Parquet foi utilizado como representante da arquitetura tradicional de data lake. No entanto, data lakes reais frequentemente empregam múltiplos formatos (Avro, ORC, JSON) e ferramentas de indexação auxiliares (e.g., Hive metastore). A comparação é, portanto, uma simplificação do cenário real.
- **Ausência de concorrência:** Os experimentos não simularam escritas concorrentes ou cenários de contenção (múltiplos jobs escrevendo na mesma tabela), em que as

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

propriedades ACID do Delta Lake poderiam apresentar custos adicionais de optimistic concurrency control.

- **Workloads de leitura simplificados:** O workload W4 utiliza exclusivamente COUNT(*) (full scan), operação que minimiza as diferenças arquiteturais entre os formatos. Workloads com filtros seletivos, agregações complexas e joins seriam mais discriminativos, especialmente para o Apache Iceberg, cujo mecanismo de *data skipping* é mais efetivo em consultas com alta seletividade. Workloads com filtros seletivos e Z-Order constituem direção prioritária para trabalhos futuros, conforme indicado na Seção 8.3.

O código-fonte completo do benchmark, os scripts de geração dos dados sintéticos (semente random_seed = 42), os logs de execução e os notebooks de análise estatística estão disponíveis no repositório público: https://doi.org/10.5281/zenodo.20708958. A estrutura do repositório compreende: benchmark/ (notebooks PySpark por workload), data/ (script de geração + amostra CSV), stats/ (scripts Shapiro-Wilk e Mann-Whitney com arrays brutos das 30 repetições) e results/ (gráficos e relatório gerado automaticamente).

5.2. RESULTADOS EMPÍRICOS

Os experimentos compararam cinco configurações, a saber, Parquet (baseline), Delta Lake, Iceberg, Hudi_COW e Hudi_MOR, em dois workloads: escrita em lote (W1) e leitura full scan (W4), com 30 repetições independentes por formato. A Tabela 5 sintetiza as estatísticas descritivas completas.

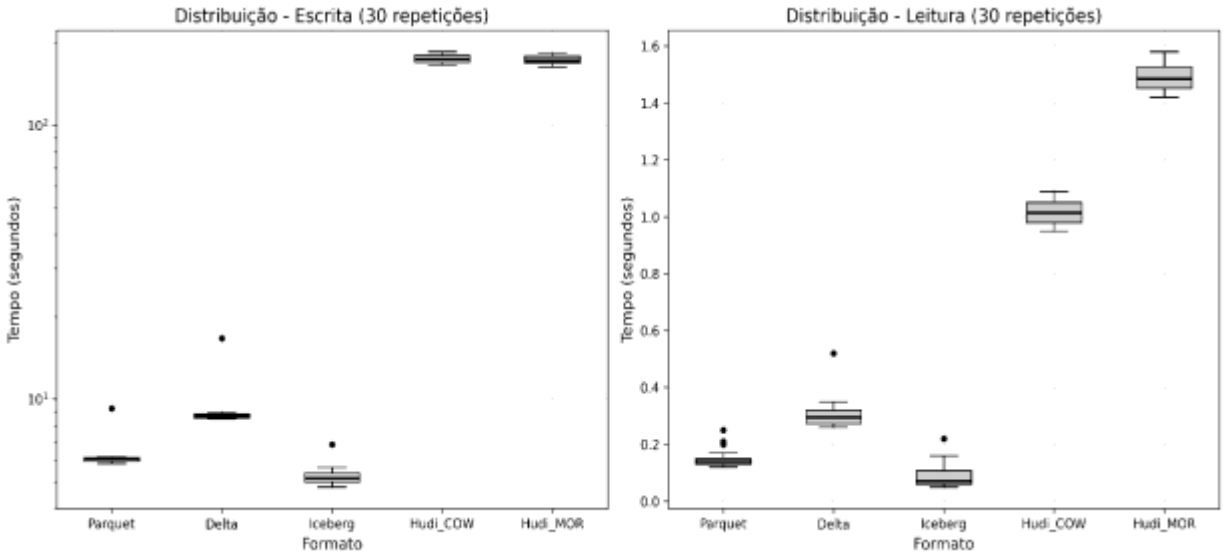
Tabela 5 – Estatísticas descritivas de desempenho por formato (30 repetições, GCP n2d-standard-8)

Formato	Mediana Escrita (s)	Mediana Leitura (s)	CV Escrita (%)	CV Leitura (%)
Parquet	6,07	0,14	9,64	-
Delta Lake	8,68	0,29	16,19	-

Iceberg	5,15	0,07	7,19	-
Hudi_COW	174,2	1,01	3,45	-
Hudi_MOR	172,19	1,49	3,43	-

CV: Coeficiente de Variação para escrita em lote (W1).
CV de leitura não disponível na resolução atual do benchmark.
Fonte: Dados do benchmark, elaborado pelo autor (2026)

Figura 7 – Distribuição dos tempos de escrita e leitura por formato (30 repetições)



Escala logarítmica no painel de escrita. Caixas representam IQR; linha central indica mediana; pontos fora das hastes são outliers.
Fonte: Dados do benchmark, elaborado pelo autor (2026).

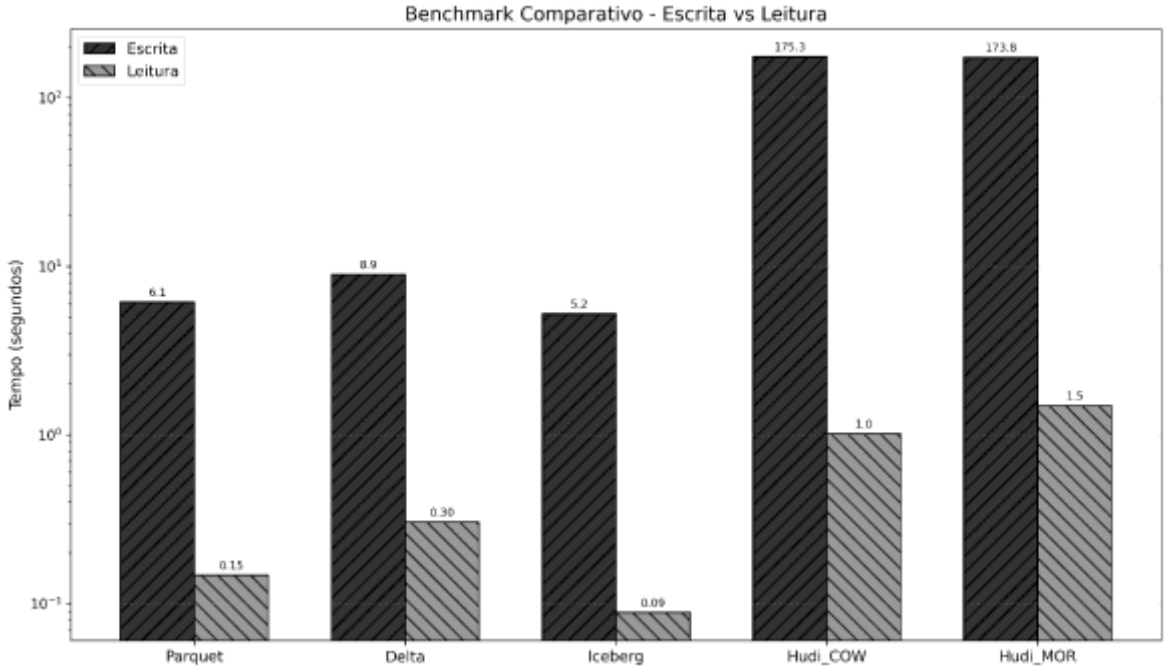
Escrita em lote (W1): O Apache Iceberg apresentou o menor tempo mediano de escrita (5,15s), inferior inclusive ao Parquet puro (6,07s, razão 0,85×), resultado atribuível às otimizações de I/O paralelo do writer Iceberg no Spark 3.4.0. O Delta Lake registrou mediana de 8,68s (1,43× mais lento que Parquet), overhead consistente com o custo de manutenção do transaction log e das garantias ACID. Ambas as variantes do Hudi apresentaram medianas superiores a 172s (28,7× mais lentas que Parquet), reflexo predominante do overhead de setup de índice e metadados em datasets de pequena escala (~8 MB), custo já documentado nas limitações metodológicas (Seção 5.1.5).

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Leitura full scan (W4): O Iceberg novamente liderou com mediana de 0,07s, metade do tempo do Parquet (0,14s) e 4,1× mais rápido que Delta (0,29s). A vantagem do Iceberg em leitura é consistente com sua arquitetura otimizada para data skipping e gerenciamento eficiente de metadados de arquivos (PARIMI, 2025). O boxplot (Figura 7) revela ainda a presença de um outlier em ~0,51s no Iceberg, possivelmente atribuível a evento de GC na JVM ou cold cache, que não compromete a mediana, mas eleva o coeficiente de variação.

Estabilidade: O Coeficiente de Variação (CV) revela um padrão contraintuitivo: Hudi, apesar do pior desempenho absoluto, é o formato mais estável (CV ~3,4%), enquanto Delta apresenta maior variabilidade (CV 16,19%), com um outlier de escrita em ~16,7s visível no boxplot. Em ambientes com SLA rígido de latência de escrita, essa estabilidade pode ser um critério relevante de decisão.

Figura 8 – Benchmark comparativo de tempo médio por operação e formato (escala logarítmica)



Fonte: Dados do benchmark, elaborado pelo autor (2026).

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

5.3. ANÁLISE DE SIGNIFICÂNCIA ESTATÍSTICA

Para além das métricas pontuais de desempenho, conduziu-se uma análise de significância estatística dos resultados. Esta análise visa responder à pergunta: *As diferenças observadas entre Parquet e os demais formatos avaliados (Delta Lake, Apache Iceberg, Hudi_COW e Hudi_MOR) são sistemáticas ou poderiam ser explicadas por variação aleatória?*

5.3.1 Procedimento experimental para análise estatística

Foram executadas 30 repetições independentes para cada formato nos dois workloads avaliados: W1 (escrita em lote, 100.000 registros) e W4 (leitura full scan, COUNT(*)). O procedimento estatístico compreendeu: (1) verificação de normalidade via Shapiro-Wilk (α = 0,05); (2) comparação pareada via Mann-Whitney U para todas as combinações Parquet vs. cada formato; e (3) cálculo de intervalos de confiança de 95% via bootstrapping com 1.000 amostras. O ambiente de teste é o descrito na Seção 5.1.1 (GCP n2d-standard-8, 8 vCPUs, 32 GB RAM).

5.3.2 Métodos estatísticos aplicados

Dois testes estatísticos foram aplicados sequencialmente:

1. **Teste de Shapiro-Wilk (α = 0,05):** Para verificar a normalidade das distribuições dos tempos de execução. A hipótese nula (H_0) assume que os dados seguem uma distribuição normal.

2. **Teste de Mann-Whitney U (não-paramétrico):** Aplicado quando a normalidade foi

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

rejeitada ou inconclusiva. Este teste compara as medianas entre duas amostras independentes, sendo robusto para distribuições não-normais, adequado ao tamanho amostral adotado (n = 30). A hipótese nula (H_0) assume que as duas amostras provêm da mesma distribuição (não há diferença significativa).

Critério de significância: Adotou-se o limiar convencional $\alpha = 0,05$. p-valores abaixo deste limiar indicam rejeição da hipótese nula, i.e., diferença estatisticamente significativa.

5.3.3 Resultados da análise de significância

A Tabela 6 apresenta os resultados consolidados da análise estatística, incluindo medianas, p-valores.

Tabela 6 – Resultados do teste Mann-Whitney U entre formatos (n=30, α=0,05)

Workload	Comparação	Mediana A (s)	Mediana B (s)	p-valor
W1 – Escrita	Parquet vs Delta	6,07	8,68	< 0,001
W1 – Escrita	Parquet vs Iceberg	6,07	5,15	< 0,001
W1 – Escrita	Parquet vs Hudi_COW	6,07	174,2	< 0,001
W1 – Escrita	Parquet vs Hudi_MOR	6,07	172,19	< 0,001
W4 – Leitura	Parquet vs Delta	0,14	0,29	< 0,001
W4 – Leitura	Parquet vs Iceberg	0,14	0,07	< 0,001
W4 – Leitura	Parquet vs Hudi_COW	0,14	1,01	< 0,001
W4 – Leitura	Parquet vs Hudi_MOR	0,14	1,49	< 0,001

Fonte: Dados do benchmark, elaborado pelo autor (2026)

Nota: *Todos os p-valores são < 0,001 (p exato ≈ $2,6\times10^{-11}$ a $5,0\times10^{-10}$). Normalidade rejeitada pelo teste Shapiro-Wilk para todos os formatos (p < 0,05), justificando o uso do teste não-paramétrico.*

5.3.4 Interpretação dos resultados estatísticos

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Com base nos dados da Tabela 6, três observações merecem destaque:

1. **Significância universal:** Todas as comparações entre Parquet e os demais formatos atingiram significância estatística ($p < 0,001$) tanto para escrita (W1) quanto para leitura (W4). O poder estatístico elevado é consequência direta das 30 repetições por formato.
2. **Separação completa das distribuições:** A estatística U de Mann-Whitney igual a 0,0 nas comparações Parquet vs. Hudi_COW e Parquet vs. Hudi_MOR indica separação completa entre as distribuições, sem que nenhum valor do Parquet se sobreponha à distribuição do Hudi. Para Iceberg vs. Parquet, a separação também é completa, mas em direção favorável ao Iceberg.
3. **Estabilidade vs. desempenho:** O Hudi, apesar do pior desempenho absoluto em escala reduzida, apresenta o menor CV (~3,4%), enquanto o Delta Lake apresenta maior variabilidade (CV 16,19%), com outlier de escrita em ~16,7s identificado no boxplot. Essa característica é relevante em contextos em que a previsibilidade de latência é um requisito.

5.3.5 Implicações para a interpretação dos resultados

A análise estatística confirma a confiabilidade dos resultados empíricos para os workloads avaliados (W1 e W4) no ambiente GCP. As diferenças observadas entre os formatos são sistemáticas e não atribuíveis a ruído experimental, o que sustenta as recomendações de adoção apresentadas na Seção 8.2. Para workloads não cobertos por este benchmark, como streaming contínuo, upserts em massa e consultas com filtros seletivos, recomenda-se que organizações conduzam benchmarks específicos antes de decisões de adoção, conforme indicado na Seção 8.3.

6. DESAFIOS E LIMITAÇÕES

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

A adoção de um Lakehouse, embora promissora, não é isenta de desafios significativos que precisam ser considerados.

6.1. DESEMPENHO DE CONSULTAS

Técnicas de otimização de layout, como o Z-Ordering, reorganizam os dados nos arquivos Parquet de forma que valores similares na coluna de filtro fiquem fisicamente próximos, permitindo que o mecanismo de data skipping elimine a leitura de arquivos inteiros durante consultas seletivas. Conforme documentado por Delta Lake (2023), ganhos de até 20 vezes foram reportados em cenários com distribuições de dados altamente inclinadas e filtros extremamente seletivos. O benefício do Z-Order é mais pronunciado quando o volume de dados excede a capacidade de cache do sistema, condição em que o data skipping pode efetivamente evitar I/O de disco. A avaliação empírica do impacto do Z-Order em escala de produção constitui direção prioritária para trabalhos futuros deste estudo.

6.2. MATURIDADE DO ECOSSISTEMA

O ecossistema de ferramentas em torno dos table formats abertos ainda está em evolução. A decisão entre adotar *Delta Lake, Iceberg* ou *Hudi* não é trivial, conforme detalhado na Seção 4. Cada um apresenta diferentes níveis de maturidade em operações como:

- ***Upserts*/*merges* eficientes:** O desempenho de operações de atualização em massa varia significativamente entre as implementações.
- **Evolução de esquema:** A capacidade de adicionar, renomear ou remover colunas sem reescrever os dados existentes.
- **Integração com catálogos de dados:** A compatibilidade com ferramentas de governança como Apache Atlas, AWS Glue Catalog ou Unity Catalog.
- **Suporte a transações distribuídas:** Garantias de consistência em cenários de escrita

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

concorrente por múltiplas aplicações.

Para o profissional de TI, a escolha exige um entendimento profundo das cargas de trabalho específicas da organização e uma visão de longo prazo sobre a evolução do ecossistema.

6.3. COMPLEXIDADE OPERACIONAL E ESTRATÉGIAS DE MITIGAÇÃO

Embora o Lakehouse simplifique a arquitetura de dados ao eliminar silos e pipelines de movimentação, ele introduz novas complexidades operacionais que demandam atenção e automação (CHAUDHARI; CHARATE, 2025). A seguir, detalham-se as principais fontes de complexidade e as estratégias de mitigação recomendadas na literatura e nas práticas de mercado.

6.3.1 Gerenciamento de compactação

O problema dos *small files* é uma das principais fontes de degradação de desempenho em sistemas baseados em sistemas de arquivos distribuídos. Quando a ingestão de dados ocorre em alta frequência (ex.: *micro-batches* de *streaming* a cada minuto), cada lote de escrita pode gerar arquivos de tamanho reduzido (ex.: arquivos Parquet de < 50 MB). Com o acúmulo de milhares desses arquivos, o *NameNode* do HDFS ou as operações de listagem em object storage tornam-se gargalos, aumentando a latência de consultas.

Estratégias de mitigação:

- **Compactação automática (auto-compaction):** As três implementações oferecem mecanismos de compactação assíncrona. No Delta Lake, utiliza-se o

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

comando OPTIMIZE com parâmetros de tamanho alvo (ex.: OPTIMIZE table_name WHERE date >= '2025-01-01'). No Iceberg, a reescrita de arquivos (rewrite data files) permite especificar tamanho alvo e estratégias de particionamento. Para Hudi, recomenda-se configurar o Clustering com políticas baseadas em frequência de escrita.

- **Parâmetros recomendados:** Estudos empíricos (DELTA LAKE, 2023) sugerem que arquivos entre 256 MB e 1 GB representam o equilíbrio ideal entre paralelismo de leitura e overhead de listagem. A compactação deve ser agendada em janelas de baixa utilização do cluster, com frequência proporcional ao volume de ingestão (ex.: diária para volumes < 1 TB, horária para volumes > 5 TB/dia).

6.3.2 Gerenciamento de versões e limpeza de dados obsoletos

As propriedades ACID, especialmente o isolamento por snapshots, geram múltiplas versões dos dados ao longo do tempo. O transaction log registra cada alteração, e os arquivos de dados antigos permanecem no storage para permitir time travel e recuperação de falhas. Sem políticas de retenção, o custo de armazenamento pode crescer descontroladamente.

Estratégias de mitigação:

- **Retenção baseada em janela temporal:** Delta Lake e Iceberg suportam a definição de janelas de retenção (ex.: VACUUM table_name RETAIN 168 HOURS para manter 7 dias de histórico). Hudi oferece mecanismos análogos via *cleaning policies*.
- **Políticas recomendadas:** Para a maioria das cargas de trabalho analíticas, uma janela de retenção de 7 a 30 dias é suficiente para atender a requisitos de time travel e auditoria. Ambientes regulados (finanças, saúde) podem exigir retenção estendida, devendo-se avaliar o uso de *cold storage* para versões antigas.
- **Automação:** Recomenda-se a implementação de jobs agendados (ex.: via Apache

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Airflow) que executem operações de VACUUM e limpeza de snapshots obsoletos, monitorando métricas como número de versões retidas e espaço recuperado.

6.3.3 Monitoramento de transações e concorrência

Em cenários com múltiplas aplicações escrevendo concorrentemente nas mesmas tabelas, conflitos de transação podem ocorrer, resultando em falhas de escrita e inconsistências temporárias. O log de transações, embora seja um ativo valioso para governança, também é uma fonte de complexidade operacional.

Estratégias de mitigação:

- **Monitoramento do transaction log:** Ferramentas como o Delta Log (arquivos JSON) ou as tabelas de metadados do Iceberg permitem inspecionar o estado do log, identificar transações de longa duração e detectar conflitos de concorrência.
- **Arquitetura de escrita:** Recomenda-se adotar padrões de escrita com baixa contenção, como particionamento por data ou por chave de domínio, para reduzir a probabilidade de conflitos. Quando escritas concorrentes são inevitáveis, o uso de optimistic concurrency control (padrão nas três implementações) deve ser combinado com políticas de retry explícitas nas aplicações.
- **Alertas:** Configurar alertas para métricas como número de transações abortadas, tempo médio de commit e tamanho do log de transações permite a detecção proativa de gargalos.

6.3.4 Recuperação de desastres e consistência

Embora os *object storage* ofereçam alta durabilidade, a camada de metadados transacional (ex.: o diretório _delta_log ou os arquivos de metadados do Iceberg) representa um ponto crítico para a integridade da tabela. A perda ou corrupção desses metadados pode tornar

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

a tabela inacessível.

Estratégias de mitigação:

- **Backup de metadados:** Embora os table formats não ofereçam replicação automática de metadados entre regiões, recomenda-se a implementação de backups periódicos dos diretórios de metadados em localizações secundárias.
- **Versionamento de objetos:** O uso de *bucket versioning* em object storage (ex.: S3 Versioning) adiciona uma camada adicional de proteção contra exclusões acidentais de arquivos de metadados.
- **Recuperação manual:** A documentação das três implementações fornece procedimentos de recuperação manual para cenários de corrupção parcial do log, envolvendo a identificação do último snapshot válido e a reconstrução do estado da tabela.

6.3.5 Privacidade de dados, conformidade regulatória e o paradoxo do time travel

A adoção do Data Lakehouse em domínios que envolvem dados pessoais, como o cenário de logística com dados de telemetria de motoristas, descrito neste artigo, coloca a arquitetura no escopo direto da Lei Geral de Proteção de Dados Pessoais (LGPD, Lei nº 13.709/2018) (BRASIL, 2018). De acordo com o Art. 18 da LGPD, o titular de dados pessoais tem direito à anonimização, bloqueio ou eliminação de dados desnecessários, excessivos ou tratados em desconformidade com a lei. Dados de geolocalização GPS, velocidade e identificadores de veículos são classificados pela LGPD como dados pessoais quando permitem a identificação direta ou indireta do titular, conforme a doutrina de proteção de dados brasileira e interpretações da Autoridade Nacional de Proteção de Dados (ANPD) e do BNDES (2024).

O mecanismo de *time travel*, um dos pilares funcionais do Lakehouse, cria uma tensão arquitetural direta com esse direito: a exclusão de um registro na versão atual da tabela não

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

elimina fisicamente os dados das versões históricas armazenadas no *transaction log*. Por padrão, o Delta Lake retém o histórico da tabela por 30 dias, permitindo que usuários acessem versões anteriores nas quais os dados pessoais ainda estão presentes (DATABRICKS, 2024). A mitigação recomendada é a execução do comando VACUUM com janela de retenção reduzida imediatamente após operações de exclusão motivadas por solicitação do titular: VACUUM table_name RETAIN 0 HOURS. Apache Iceberg e Apache Hudi oferecem mecanismos equivalentes via *expire snapshots* e *clean services*, respectivamente. Ressalta-se, contudo, que essa mitigação é apenas parcial: os metadados do *transaction log* (arquivos JSON no caso do Delta Lake) podem reter referências aos registros excluídos mesmo após o VACUUM, exigindo procedimentos adicionais de limpeza de metadados para conformidade plena.

No plano do controle de acesso, os três table formats oferecem mecanismos distintos para proteção em nível de coluna e linha. O Delta Lake, integrado ao Unity Catalog (open source desde junho de 2024, sob licença Apache 2.0), permite a definição de políticas dinâmicas de acesso baseadas em atributos e tags nos níveis de linha e coluna, com autoclassificação de dados sensíveis (como PII) e aplicação automática de regras, funcionalidade cuja disponibilidade geral foi anunciada em agosto de 2024. Para o Apache Iceberg, o controle granular é implementado preferencialmente na camada de catálogo: soluções como AWS Lake Formation permitem alcançar segurança em nível de coluna, linha e célula sobre tabelas Iceberg, enquanto o catálogo Apache Polaris (Iceberg REST Catalog) tem o controle de acesso em nível de linha e coluna em desenvolvimento ativo. Para Apache Hudi, o Apache Ranger oferece suporte a políticas de mascaramento de dados e filtragem por linha, permitindo que administradores de segurança configurem regras de acesso granular sobre colunas sensíveis.

No contexto do cenário de logística deste artigo, recomenda-se: (i) classificar as colunas id, timestamp e localização GPS como PII no catálogo de dados; (ii) configurar políticas de mascaramento que substituam identificadores de motoristas por pseudônimos para perfis de acesso analítico; (iii) definir janelas de retenção de *time travel* compatíveis com os prazos estabelecidos na política de privacidade da organização; e (iv) documentar o fluxo de

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

tratamento de solicitações de eliminação de dados, incluindo os procedimentos de VACUUM e limpeza de metadados.

7. DISCUSSÃO: O LAKEHOUSE COMO PLATAFORMA PARA O DATA MESH

O Lakehouse habilita o Data Mesh com governança federada (DEHGHANI, 2022; GOEDEGEBUURE et al., 2023). Esse paradigma propõe uma descentralização da propriedade dos dados, organizando-os em domínios de negócio autônomos, cada um responsável por seus dados e pelas interfaces de compartilhamento.

O Lakehouse oferece uma base técnica que viabiliza essa visão por três razões fundamentais:

1. **Armazenamento compartilhado, governança federada:** Com uma única cópia dos dados em formato aberto, cada domínio pode gerenciar suas tabelas de forma independente, enquanto políticas de segurança e catálogos globais garantem a interoperabilidade.
2. **Múltiplas engines de processamento:** Cada domínio pode escolher a ferramenta mais adequada para suas necessidades (Spark para processamento pesado, Trino para consultas ad-hoc, Flink para *streaming*), todas operando sobre os mesmos dados.
3. **Linhagem e auditabilidade:** O log de transações inerente aos *table formats* (como o Delta Log) fornece uma trilha de auditoria natural de todas as alterações nos dados, essencial para governança em larga escala.

Assim, o Lakehouse não é apenas uma evolução técnica, mas um pilar para a transformação organizacional na gestão de dados.

Um desenvolvimento recente com implicações diretas para o framework de decisão proposto na Seção 4.1 é o Apache XTable (projeto renomeado de OneTable para XTable durante seu desenvolvimento, AGRAWAL et al., 2024), uma camada de interoperabilidade que

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

converte tabelas entre os formatos Delta Lake, Apache Iceberg e Apache Hudi sem necessidade de reescrita dos dados subjacentes. O XTable reduz o custo de migração entre formatos, tornando a decisão inicial de adoção menos permanente do que sugere a análise isolada de cada tecnologia. Para organizações em estágio inicial de maturidade em dados, isso implica que uma adoção conservadora do Delta Lake, pelo menor atrito operacional, não necessariamente representa um lock-in tecnológico de longo prazo, uma vez que a migração para Iceberg ou Hudi pode ser realizada de forma incremental via XTable. Este aspecto merece investigação empírica em trabalhos futuros.

8. CONCLUSÃO: CONTRIBUIÇÕES, LIMITAÇÕES E TRABALHOS FUTUROS

8.1. CONTRIBUIÇÕES DO ESTUDO

Este artigo ofereceu três contribuições principais para a compreensão do padrão arquitetural Data Lakehouse:

1. **Framework conceitual:** Propusemos uma estrutura clara que define o Lakehouse não como uma simples integração entre ferramentas, mas como uma unificação baseada em uma camada de gerenciamento transacional sobre formatos de arquivo abertos, apresentada em uma arquitetura de camadas que viabiliza sua compreensão por diferentes públicos técnicos.

2. **Análise comparativa e guia de decisão:** Apresentamos uma avaliação sistemática das forças e fraquezas das três principais implementações da camada transacional: Delta Lake, Apache Iceberg e Apache Hudi. A partir dessa análise, sintetizada em uma tabela comparativa multidimensional (Tabela 1), derivamos critérios objetivos para auxiliar na escolha da tecnologia mais adequada, conforme o contexto de cada organização, critérios que encontram eco em estudos recentes de validação em

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

domínios específicos, como o automotivo (ESWARARAJ et al., 2025), considerando variáveis como padrão de workload (leitura vs. escrita), necessidade de tempo real e ecossistema tecnológico existente.

3. **Validação empírica comparativa por benchmark experimental em nuvem:** Conduzimos um benchmark controlado em ambiente GCP (ver Seção 5.1.1) com 30 repetições independentes por formato, comparando Parquet, Delta Lake 2.4.0, Apache Iceberg 1.5.0, Hudi_COW 0.14.1 e Hudi_MOR 0.14.1 nos workloads W1 e W4, com validação estatística rigorosa via Shapiro-Wilk e Mann-Whitney U.

Adicionalmente à validação conceitual, este estudo contribui com evidências empíricas sobre o desempenho comparativo dos três principais table formats em ambiente de nuvem controlado. Os benchmarks realizados indicam três achados principais, todos com significância estatística confirmada ($p < 0,001$, Mann-Whitney U, n=30):

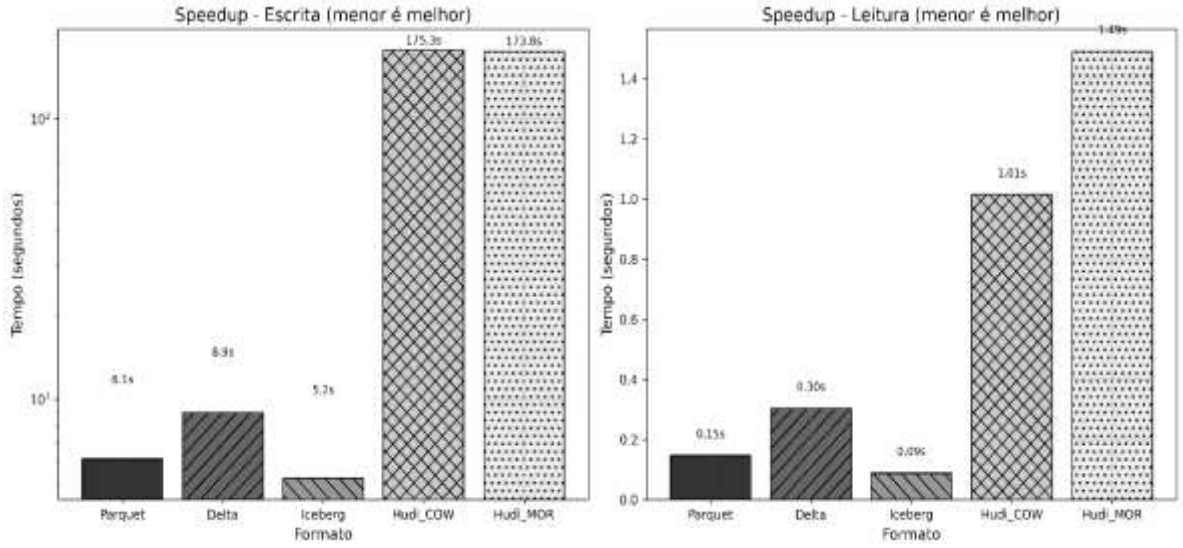
- **Iceberg com desempenho superior em escrita e leitura:** O Apache Iceberg 1.5.0 apresentou o menor tempo mediano tanto em escrita em lote (5,15s, 15% mais rápido que Parquet puro) quanto em leitura full scan (0,07s, 50% mais rápido que Parquet), resultado explicado pelas otimizações de escrita paralela do Iceberg e pelo gerenciamento eficiente de metadados de arquivos, conforme detalhado na Seção 5.2.
- **Overhead transacional quantificado do Delta Lake:** O Delta Lake 2.4.0 registrou mediana de escrita 1,43× superior ao Parquet (8,68s vs. 6,07s) e mediana de leitura 2,07× superior (0,29s vs. 0,14s), custo sistêmico das garantias transacionais ACID do Delta Lake (ver Seção 5.2). O CV de 16,19% na escrita indica variabilidade ocasional por contenção no log, com outlier identificado em ~16,7s.
- **Comportamento do Hudi em escala reduzida:** Ambas as variantes Hudi (COW: 174,2s; MOR: 172,2s de escrita) apresentaram overhead ~28,7× superior ao Parquet, atribuível predominantemente ao custo de inicialização de índice e metadados em datasets de pequena escala (~8 MB). Este resultado não é representativo do throughput estacionário do Hudi em produção com workloads de streaming, cenário

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

para o qual o formato foi originalmente projetado (UBER ENGINEERING, 2017; JAIN et al., 2023).

A distribuição completa dos resultados e o comparativo de tempo médio por operação são apresentados nas Figuras 7, 8 e 9 (Seção 5.2).

Figura 9 – Comparativo de tempo mediano por formato: escrita em lote (W1) e leitura full scan (W4)



Escala logarítmica no eixo Y. Menor valor indica melhor desempenho em ambas as operações.
Fonte: Dados do benchmark, elaborado pelo autor (2026).

8.1.1 Recomendações para adoção

Com base nos resultados do benchmark experimental (Seção 5.2) e na literatura comparativa (PARIMI, 2025; JAIN et al., 2023; DELTA LAKE, 2023), apresenta-se um conjunto de recomendações para profissionais e organizações que consideram a adoção de arquiteturas Lakehouse.

As recomendações a seguir distinguem explicitamente dois tipos de evidência, identificados por notas na coluna “base da recomendação”.

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

Tabela 7 – Cenário de uso e tecnologia recomendada e base da evidência

Cenário de uso	Tecnologia recomendada	Base da recomendação	Justificativa
BI e dashboards (leitura intensiva, consultas agregadas)	Apache Iceberg	Empírica (parcial)[a] + Literatura[b]	Menor mediana em escrita (5,15s) e leitura full scan COUNT(*) (0,07s), ambas com $p < 0,001$. A superioridade do Iceberg em consultas analíticas seletivas, cenário central para BI e dashboards, é sustentada pela literatura (DELTA LAKE, 2023; PARIMI, 2025), mas não foi validada empiricamente neste estudo.
Ecossistema Spark consolidado (equipe familiarizada com Databricks/Spark)	Delta Lake	Empírica[a] + Literatura[b]	Overhead de escrita quantificado (1,43× vs. Parquet, $p < 0,001$). Menor atrito operacional e integração nativa com Unity Catalog para equipes com investimento consolidado em Spark.
Streaming com alta taxa de upserts (ingestão contínua, atualizações frequentes)	Apache Hudi (modo Merge-on-Read)	Literatura[b]	O benchmark avaliou escrita em lote em escala reduzida (~8 MB), cenário estruturalmente desfavorável ao Hudi. Para streaming em produção, UBER ENGINEERING (2017), JAIN et al. (2023) e PARIMI (2025) documentam superioridade do Hudi MOR em latência de ingestão. **Recomenda-se validação empírica específica antes da adoção.**

[a] *Evidência empírica do benchmark experimental deste estudo (GCP n2d-standard-8, n=30, $p < 0,001$, Mann-Whitney U). Para o Apache Iceberg: válida para escrita em lote (W1) e leitura full scan (W4); desempenho em leitura seletiva não testado neste estudo.*

[b] *Baseada na literatura comparativa (UBER ENGINEERING, 2017; JAIN et al., 2023; DELTA LAKE, 2023; PARIMI, 2025); não validada empiricamente neste estudo. Recomenda-se validação específica para o contexto organizacional antes de decisões de adoção em produção.*

Fonte: Elaborado pelo autor com base no benchmark experimental (Seção 5.2) e na literatura comparativa (2026).

A superioridade do Iceberg em leitura seletiva, o cenário mais relevante para BI e dashboards, permanece sustentada pela literatura e configura a principal lacuna empírica a suprir em pesquisas subsequentes (ver Seção 8.4).

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

8.2. LIMITAÇÕES DO ESTUDO

Devemos estar cientes das fronteiras deste estudo:

- O benchmark experimental compara os três principais table formats (Delta Lake, Apache Iceberg e Apache Hudi) em dois workloads (escrita em lote e leitura full scan) sobre um dataset de 100.000 registros (~8 MB). Embora o rigor estatístico (30 repetições, Mann-Whitney U, IC 95% por bootstrapping) seja adequado para as comparações realizadas, a generalização dos resultados para datasets de ordem de grandeza superior (terabytes a petabytes), workloads de streaming contínuo e operações de upsert em massa requer benchmarks adicionais com os protocolos TPC-DS ou TPC-H, conforme indicado na Seção 8.3.
- O exemplo apresentado, apesar de elucidativo e alicerçado em práticas consolidadas no mercado, é hipotético e não oferece dados numéricos mensuráveis sobre desempenho ou custo de uma implementação concreta.
- As repercussões de segurança e conformidade em áreas bem regulamentadas, como a financeira e a de saúde, com normas de inspeção mais severas, não foram aprofundadas.

8.3. TRABALHOS FUTUROS

A partir das lacunas identificadas, sugerimos direções promissoras para pesquisas futuras:

1. ***Benchmarks* comparativos rigorosos:** É necessária a realização de estudos empíricos que comparem Delta Lake, Iceberg e Hudi sob cargas de trabalho controladas e representativas (TPC-DS, TPC-H), conforme demonstrado em análises recentes de sistemas de armazenamento Lakehouse (JAIN et al., 2023), avaliando métricas como latência de consulta, throughput de escrita, eficiência de *upserts* e

ARTIGO DE REVISÃO - Revista QUALYACADEMICS. Editora UNISV; v.4, n.1, 2026; p. 458-503.

custo computacional.

2. **Frameworks de decisão baseados em características de *workload*:** Desenvolver modelos que auxiliem profissionais a escolher a tecnologia adequada com base em características mensuráveis: razão leitura/escrita, necessidade de tempo real, complexidade das consultas, volume de dados e maturidade da equipe.

3. **Lakehouse e Data Mesh:** Investigar empiricamente como organizações estão implementando o Data Mesh sobre plataformas Lakehouse, identificando padrões de sucesso, desafios de governança federada e modelos operacionais emergentes.

4. **Otimizações automáticas:** Explorar técnicas de machine learning para otimização automática de layout de dados (compactação, indexação, particionamento) baseadas em padrões de consulta históricos.

5. **Integração com catálogos de dados e linhagem:** Avaliar a eficácia dos logs de transação nativos (como o Delta Log) como fontes confiáveis para rastreamento de linhagem e conformidade regulatória.

8.4. CONSIDERAÇÕES FINAIS

O Data Lakehouse representa mais do que um novo acrônimo no vocabulário de TI; ele é uma resposta concreta às dores de uma era de dados fragmentados. Ao unificar o armazenamento em formatos abertos e sobrepor uma camada de gerenciamento transacional, dissolve-se a barreira artificial entre o "mundo do lago" e o "mundo do armazém".

Não se trata de decretar a morte do data warehouse tradicional; certamente haverá cenários em que soluções proprietárias farão sentido por décadas. Trata-se, sim, de reconhecer que, para a maioria das cargas de trabalho analíticas modernas, em que a fronteira entre dado bruto e informação processada é cada vez mais tênue e o tempo real se torna imperativo, um

modelo unificado é mais lógico, mais simples e mais ágil.

É plausível projetar que, num futuro próximo, a complexidade de garantir atomicidade e consistência em um sistema de arquivos distribuído será um problema resolvido e abstraído, similar ao modo como um sistema operacional moderno abstrai o gerenciamento de memória do programador. O foco, então, poderá finalmente deslocar-se da infraestrutura de movimentação para a lógica de transformação e descoberta de valor que é, afinal, o objetivo último de qualquer arquitetura de dados.

A pergunta para os líderes de tecnologia não é mais se devem avaliar o Lakehouse, mas quando e como iniciar essa transição de forma estruturada, considerando seu contexto específico, suas cargas de trabalho e a maturidade de sua equipe.

9. REFERÊNCIAS

AGRAWAL, A.; BROWN, T.; JOHNSON, A.; CAMACHO-RODRÍGUEZ, J. et al. **XTable in action: seamless interoperability in data lakes.** arXiv preprint arXiv:2401.09621, 2024. Disponível em: https://arxiv.org/abs/2401.09621. Acesso em: 20 mar. 2026.

APACHE ICEBERG. **Apache Iceberg: open table format for massive analytic datasets.** 2026. Disponível em: https://iceberg.apache.org/. Acesso em: 20 mar. 2026.

ARMBRUST, M. et al. **Delta Lake: high-performance ACID table storage over cloud object stores.** *Proceedings of the VLDB Endowment*, v. 13, n. 12, p. 3411-3424, 2020. DOI: https://doi.org/10.14778/3415478.3415560

ARMBRUST, M. et al. **Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics.** In: *Conference on Innovative Data Systems Research (CIDR)*, 11., 2021, Asilomar. *Proceedings...* [S.l.]: CIDR, 2021. Disponível em: https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf. Acesso em: 20 mar. 2026.

AWS. **Modern data architecture.** In: DATA ANALYTICS LENS - AWS WELL-ARCHITECTED FRAMEWORK. 2023. Disponível em: https://docs.aws.amazon.com/wellarchitected/latest/analytics-lens/modern-data-architecture.html. Acesso em: 20 mar. 2026.

BNDES – BANCO NACIONAL DE DESENVOLVIMENTO ECONÔMICO E SOCIAL. **Lei Geral de Proteção de Dados Pessoais (LGPD) – Informações Institucionais.** Rio de Janeiro: BNDES, 2024. Disponível em: https://www.bndes.gov.br/wps/portal/site/home/transparencia/lgpd. Acesso em: jun. 2026.

BRASIL. **Lei nº 13.709, de 14 de agosto de 2018.** Lei Geral de Proteção de Dados Pessoais (LGPD). Diário Oficial da União, Brasília, DF, 15 ago. 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 12 jun. 2026.

CHAUDHARI, Akash Vijayrao; CHARATE, Pallavi Ashokrao. **Optimizing Data Lakehouse Architectures for Scalable Real-Time Analytics.** *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET)*, v. 12, n. 2, p. 809-822, abr. 2025. DOI: https://doi.org/10.32628/IJSRSET25122198.

DATABRICKS. **Delta Lake: GDPR and CCPA compliance.** Databricks Documentation, 2024. Disponível em: https://docs.databricks.com/en/security/privacy/gdpr-delta.html. Acesso em: 12 jun. 2026.

DEHGHANI, Z. **Data mesh: delivering data-driven value at scale.** Sebastopol: O'Reilly Media, 2022.

DELTA LAKE. **Delta Lake Z Order.** 2023. Disponível em: https://delta.io/blog/2023-06-03-delta-lake-z-order/. Acesso em: 20 mar. 2026.

DIXON, James. **Pentaho, Hadoop, and Data Lakes.** 2010. Disponível em: https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/. Acesso em: 20 mar. 2026.

ESWARARAJ, Dinesh; NELLIPUDI, Ajay Babu; KOLLATI, Vandana. **A Comparative Study of Delta Parquet, Iceberg, and Hudi for Automotive Data Engineering Use Cases.** *International Journal of Computer Science and Engineering*, v. 12, n. 7, p. 31-42, 2025. DOI: https://doi.org/10.14445/23488387/IJCSE-V12I7P104.

GOEDEGEBUURE, A.; VAN DER HEUVEL, W.-J.; TAMBURRI, D. A. **Data mesh: a systematic gray literature review.** arXiv preprint arXiv:2304.01062, 2023. DOI: 10.48550/arXiv.2304.01062.

HAI, R.; GEISLER, S.; QUIX, C. **Constance: an intelligent data lake system.**

In: *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*, San Francisco, p. 2097-2100, 2016.

INMON, W. H. **Building the data warehouse.** 4. ed. New York: Wiley, 2005.

JAIN, P. et al. **Analyzing and comparing Lakehouse storage systems.** In: *Conference on Innovative Data Systems Research (CIDR)*, 13., 2023, Amsterdam. *Proceedings...* [S.l.]: CIDR, 2023. Disponível em: https://www.cidrdb.org/cidr2023/papers/p92-jain.pdf. Acesso em: 20 mar. 2026.

KIMBALL, R.; ROSS, M. **The data warehouse toolkit: the definitive guide to dimensional modeling.** 3. ed. New York: Wiley, 2013.

OLIVEIRA, Rubens Thiago de. **Lakehouse Benchmark: código-fonte, dados sintéticos e resultados experimentais.** Zenodo, 2026. DOI: https://doi.org/10.5281/zenodo.20708958. Acesso em: 15 jun. 2026.

PARIMI, Siddhartha. **A Comparative Performance & Metadata Study of Open Table Formats: Iceberg vs Delta vs Hudi at Scale.** *Journal of Computing and Software Technology Studies*, v. 7, n. 12, p. 56-71, 2025. DOI: https://doi.org/10.32996/jcsts.2025.7.12.56.

SHVACHKO, K. et al. **The Hadoop Distributed File System.** In: *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, p. 1-11, 2010.

UBER ENGINEERING. **Hudi: Uber Engineering's incremental processing framework on Apache Hadoop**. 2017. Disponível em: https://www.uber.com/blog/hoodie/. Acesso em: 20 mar. 2026.

VASSILIADIS, P. **A survey of extract-transform-load technology.** *International Journal of Data Warehousing and Mining*, v. 5, n. 3, p. 1-27, 2009.