

How does the Data Lakehouses compare to Data Warehouses and Data Lakes in terms of scalability, governance, and performance?

Benjamin Casey
HES-SO
Lausanne, Switzerland
benjamin.casey@master.hes-so.ch

Bruno Fournier
HES-SO
Lausanne, Switzerland
bruno.fournier@master.hes-so.ch

Antoine Ottiger
HES-SO
Lausanne, Switzerland
antoine.ottiger@master.hes-so.ch

ABSTRACT

The diversity of data management technologies offers complementary approaches that meet various requirements, whether it be query efficiency or the capacity to store diverse data types. This article explores the emerging architecture of Data Lakehouses (LHs), rigorously comparing them to traditional Data Warehouses (DWs) and modern Data Lakes (DLs). We propose an analysis that focuses on three key aspects: scalability, governance, and performance. DWs, characterized by structured storage and robust governance, enable efficient querying. However, they may face scalability challenges when dealing with surges in incoming data.

In contrast, DLs offer scalable and flexible storage for a variety of data formats, but they pose significant challenges in terms of governance and performance. Poorly managed Data Lakes often devolve into Data Swamps, rendering the data unusable.

A hybrid solution may take the form of a LH, combining the scalable and cost-effective storage of Data Lakes with the rigorous governance mechanisms and efficient querying capabilities traditionally associated with Data Warehouses. Using open and transactional file formats such as Apache Iceberg, Delta Lake, and Apache Hudi, LHs enable strong data governance, support schema evolution, and facilitate real-time analytics. Our analysis suggests that LHs represent a promising approach, effectively addressing the limitations of previous architectures and fully meeting the current needs in analytics and Business Intelligence (BI).

CCS CONCEPTS

• **Information systems** → **Database management system; Data management systems engines.**

KEYWORDS

Data Lakehouse, Data Warehouse, Data Lake, Scalability, Governance, Performance

ACM Reference Format:

Benjamin Casey, Bruno Fournier, and Antoine Ottiger. 2025. How does the Data Lakehouses compare to Data Warehouses and Data Lakes in terms of

scalability, governance, and performance?. In *Proceedings of New Technologies in Data Engineering (NTDE) (NTDE 2025)*. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/0000000.0000000>

1 INTRODUCTION

Data management has undergone a profound evolution over the past four decades, driven by explosive growth in data volume, variety, and velocity. Traditional Data Warehouses emerged in the 1990s to provide business users with clean, integrated, and historical views of structured data for reporting and decision support [13, 20]. With the advent of web, mobile, and IoT applications in the 2000s, organizations began generating massive streams of semi-structured and unstructured data, giving rise to the concept of the Data Lake: a low-cost, schema-on-read repository capable of storing raw data in its native form [4, 8]. However, poorly governed Data Lakes often devolve into “Data Swamps,” undermining data quality and usability. More recently, the Data Lakehouse paradigm has been proposed to combine the strengths of both DWs and DLs—bringing ACID guarantees, fine-grained governance, and high-performance querying into the flexible storage model of the lake. In this paper, we aim to trace this architectural trajectory, identify the drivers and shortcomings of each approach, and demonstrate how LHs address the pressing needs of modern analytics and Business Intelligence.

In the sections that follow, we first provide a detailed overview of each architecture: section 2 defines and characterizes Data Warehouses, highlighting their schema-on-write governance and OLAP-style performance; section 3 examines Data Lakes, with emphasis on their scalability, schema-on-read flexibility, and governance challenges; and section 4 introduces Data Lakehouses, unpacking their layered design, transactional table formats (e.g., Delta Lake, Iceberg, Hudi), and unified governance framework. We then systematically compare these three paradigms across the dimensions of scalability, governance, and performance (section 5), present key findings (section 6), and conclude with future research directions (section 7).

2 DATA WAREHOUSES

2.1 Definition

Bill Inmon, widely recognized as the father of Data Warehouses [7], defines this data management system as follows: “A data warehouse is a subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management’s decision-making process.” [14]. Subject-oriented means that the system is organized around a core business theme (such as Customers, Products, Sales, or Finance) rather than by operational function. The data, which can come from multiple sources, is “integrated,” meaning it is transformed to ensure both semantic and structural consistency. The

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

NTDE 2025, 2025, Lausanne, Switzerland

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-x-xxxx-xxxx-x/25/05

<https://doi.org/10.1145/0000000.0000000>

term time-variant indicates that data must be associated with a temporal value: recording the date of entry into the system is mandatory, and, in some cases, an expiration date for data validity may also be added. This feature enables historical analyses (e.g., sales by month, year, etc.). Finally, "non-volatile" means that once data is entered into the system, it cannot be modified or deleted, only new data can be added. The figure 1 shows a data warehouse pipeline.

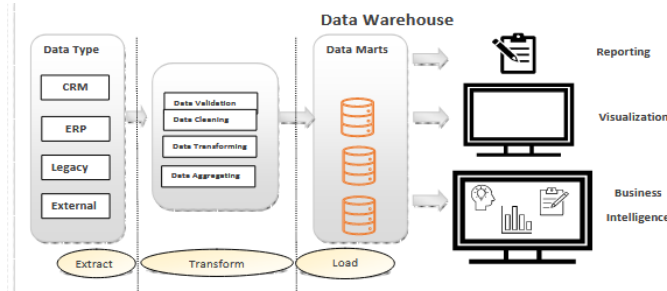


Figure 1: Data warehouse pipeline [9]

2.2 Architecture and Characteristics of the DW

In a study conducted by Ahmed A. Harby in 2022 comparing different data management systems, the architecture of a Data Warehouse is described as follows [9]: First, data is integrated through the following stages: extraction (Extract), transformation (Transform), and loading (Load) — a process known as ETL. Extraction involves retrieving data from various sources, which may have different formats and structures. In practice, this could mean collecting data from different retail outlets, e-commerce sites, mobile applications, or CRM (Customer Relationship Management) systems. At this stage, the data is heterogeneous and must be processed before analysis. During the transformation phase, the data is cleaned, filtered, organized, and structured. This may involve removing duplicates, standardizing formats, or enriching the data (for example, calculating a duration based on two dates). It may also involve merging data from different sources into a single table to simplify access. Once transformed, the data is loaded into various data marts to enable specialized analyses. The data is thus integrated in a way that supports visualization, reporting, or business intelligence activities. In essence, a DW is a data management system that preprocesses heterogeneous data from multiple sources into structured and formatted datasets for specific analyses. The schema for ingested data is predetermined and must be strictly adhered to; this is referred to as schema-on-write. A DW also applies the ACID principles: each transaction is Atomic (all-or-nothing), maintains Consistency of data, executes in Isolation from others (preventing two users from performing the same operation simultaneously), and ensures Durability of changes even in the event of a failure.

2.3 Users

Data Warehouses are primarily designed for business professionals (executives, managers, supervisors) or business analysts [17]. These users require reliable, consistent, historical, and easily queryable data. DW systems can be used in conjunction with Online Analytical Processing (OLAP) tools [5] to perform complex analyses across

multiple dimensions (such as time, region, product, etc.). The main purpose of a DW in this context is to centralize and structure data from multiple sources to support decision-making processes.

3 DATA LAKES

Data Lakes have emerged as a fundamental component of modern big data architectures, offering a flexible and scalable repository for heterogeneous data. Unlike traditional data warehouses, which require pre-defined schemas and structured data, Data Lakes are designed to store raw data in its native format, accommodating structured, semi-structured, and unstructured data types. This chapter presents an in-depth review of Data Lakes, focusing on their architecture, scalability, governance, and performance characteristics.

3.1 Definition and conceptual framework

A Data Lake (figure 2) is a centralized repository that allows organizations to store all types of data at any scale. Data is ingested in its raw form, thereby supporting a "schema-on-read" approach. This flexibility enables data scientists and engineers to extract value by applying various processing techniques only when the data is accessed for analytics or machine learning purposes. The inherent design promotes agility and cost efficiency, particularly when dealing with high-volume, diverse data streams [12].

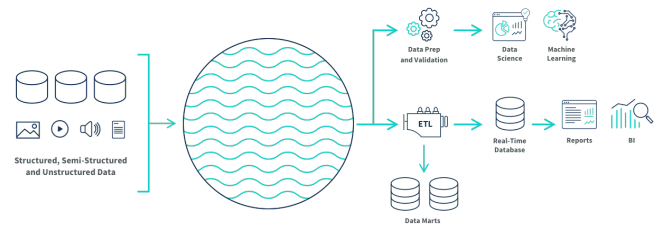


Figure 2: Data lake operating diagram (source : Qlik)

3.2 Architectural considerations

The architecture of a Data Lake typically integrates distributed storage systems with scalable parallel processing frameworks, as mentioned in the following section. This separation between storage and compute layers enables the scaling of each component independently to meet varying workload demands. Data Lakes are designed for horizontal scalability, accommodating massive data volumes with minimal performance degradation. Moreover, the integration of metadata management and indexing services helps mitigate some of the performance issues associated with unstructured data [17].

3.3 Technologies

Data Lakes rely on a diverse ecosystem of technologies that enable efficient data ingestion, storage, processing, and governance. On the storage layer, distributed systems such as the Hadoop Distributed

File System (HDFS), Amazon S3, Google Cloud Storage, and Azure Blob Storage provide scalable and cost-effective foundations [9]. For data processing, engines like Apache Spark, Apache Flink, and Presto support batch and real-time workloads across petabyte-scale datasets. These engines enable transformation, querying, and analytics directly on raw data without the need for prior schema enforcement [3].

Metadata and governance are handled through specialized tools such as Apache Atlas, AWS Glue Data Catalog, and Amundsen, which facilitate data discovery, lineage tracking, and policy enforcement. For data ingestion, technologies like Apache Kafka and Apache NiFi are commonly employed to capture and stream real-time or batch data from various sources into the lake [9]. Furthermore, open table formats like Apache Iceberg (section 4.3.2) and Delta Lake (section 4.3.1) are increasingly integrated into Data Lakes to introduce features such as ACID transactions, schema evolution, and time travel, which were traditionally absent in early Data Lake implementations [3]. These technological advancements collectively enhance the performance, reliability, and manageability of Data Lake environments.

4 DATA LAKEHOUSES

The Data Lakehouse represents a new paradigm in data architecture, emerging to address the limitations of both Data Lakes and Data Warehouses [19]. While Data Lakes offer flexibility and cost-effective storage for diverse data types, they often lack the robust governance and ACID (Atomicity, Consistency, Isolation, Durability) properties essential for traditional analytics. In contrast, data warehouses, while providing strong governance and performance for structured data, can struggle with the scalability and variety of modern big data [2]. The Data Lakehouse seeks to bridge this gap by providing a unified platform that combines the strengths of both architectures.

4.1 Bridging the Gap Between Data Lakes and Data Warehouses

The Data Lakehouse architecture (figure 3) is designed to bridge the gap between Data Lakes and Data Warehouses, primarily by enabling an open data architecture. As Mazumdar highlights [16], this architecture stores data in open file formats (e.g. Parquet) and table formats (e.g., Apache Iceberg) within an organization's cloud storage, allowing multiple analytical engines to access data directly. This approach is crucial because it allows organizations to use different compute engines for specific workloads and avoids locking data into vendor-specific formats, unlike traditional RDBMS-OLAP systems. Furthermore, the Data Lakehouse architecture reduces the need for data copies, which simplifies ETL processes, improves data governance, and mitigates the risks associated with ungoverned data copies.

4.2 Key Architectural Components

Unlike conventional architectures that separate the storage of data and analytical processing into two distinct systems, a data lakehouse provides a layered architectural strategy that adds transactional reliability, governance, and performance optimizations

directly to the flexible and scalable environment of a data lake [19]. At the heart of a data lakehouse is an open and shared storage layer, typically built on top of cloud object storage systems (e.g., Amazon S3 or Azure Data Lake Storage), with columnar file formats such as Apache Parquet. The storage layer is decoupled from the computation, enabling independent scaling and cost-effective data persistence. What then makes this data lake a lakehouse is the addition of a transactional table format such as Delta Lake, Apache Iceberg, or Apache Hudi [10]. These systems introduce ACID transactions, schema enforcement, and versioning, which provide consistent and concurrent reads/writes without the cost of data inconsistency.

The metadata and governance layer in a data lakehouse architecture plays the central role of enabling the link between raw and curated data. It manages table schemas, partitioning, data lineage, and access control policies. The layer natively handles schema evolution, which enables users to make changes to the data structure over time without downstream application breakage, as required for handling diverse and evolving data sources [19].

Another defining feature is the processing layer, which bridges batch and streaming workloads [9]. Lakehouses typically leverage distributed compute engines, such as Apache Spark, Trino, or Presto, which can query the data in the lake directly through SQL or other APIs. This eliminates traditional duplication of ETL between lakes and warehouses and enables real-time analytics, machine learning, and reporting from the same source of truth. In addition, a data lakehouse also provides fine-grained access control, data auditing, and governance capabilities that are not usually available in traditional data lakes [9]. Together with time travel and rollback capabilities, users have full control of their data, which is necessary for compliance and reproducibility in enterprise environments.

By unifying these components : open storage, transactional layers, metadata and governance services, and multi-purpose compute engines. A data lakehouse offers the flexibility of a lake, the reliability of a warehouse, and the agility that modern data workloads require in a single unified platform.

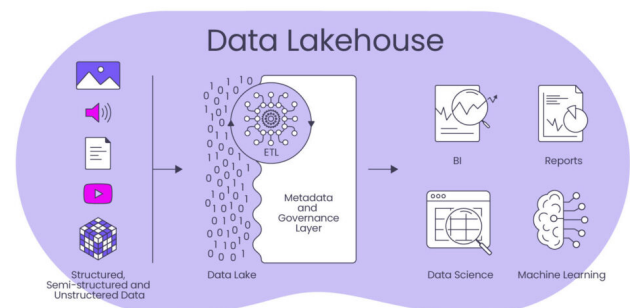


Figure 3: Data lakehouse architecture (source : datanomiq)

4.3 Technologies in Data Lakehouse Architecture

4.3.1 Delta Lake. is an open storage layer built by Databricks that brings ACID transactional capabilities to data lakes. It offers data reliability and consistency by applying a transaction log (also known as the Delta Log), which keeps a record of all the modifications to a dataset over time [10]. This enables concurrent reads and writes without compromising data integrity. Delta Lake is designed to process batch and streaming data within the same unified pipeline and therefore is ideal for modern data architectures where real-time insights are critical. It also provides features such as schema enforcement, schema evolution, and time travel so that users can query previous versions of a dataset in the event of auditing or rollback requirements. Because it is strongly coupled with Apache Spark, Delta Lake offers clear performance and scalability to handle a wide range of analytics and machine learning workloads [3].

4.3.2 Apache Iceberg. is an open-source table format developed by Netflix and currently governed by the Apache Software Foundation, designed to be optimal for high-performance analytics of big data. Unlike Hive-based formats, Iceberg was architected from the ground up to address petabyte-scale tables with consistent and fast querying [15]. It includes one of its key innovations as hidden partitioning, which simplifies and accelerates query logic by abstracting the partition management complexity. Iceberg supports full schema evolution, by which users have the ability to add, remove, or rename columns with safety without being required to rewrite entire data sets. Iceberg also provides time travel and snapshot isolation features, which facilitate constant reading and reproducibility among multi-step pipelines. Iceberg is capable of supporting a multitude of engines such as Apache Spark, Trino, Presto, Flink, and Hive and is thereby appropriate for different data processing organizations[10].

4.3.3 Apache Hudi. (Hadoop Upserts Deletes and Incrementals) is a data lake layer intended for real-time data ingestion and incremental processing [2]. Developed at Uber, Hudi makes it easy for users to perform record-level inserts, updates, and deletes on data lake storage, something traditional formats cannot [15]. This is especially useful for highly changing datasets such as IoT telemetry, logs, and CDC (Change Data Capture) use cases. Hudi provides two core types of tables: Copy-on-Write (COW) for read-optimized queries and Merge-on-Read (MOR) for low-latency updates. It features in-built rollback, versioning of data, and indexing that allows for the execution of efficient upserts and secure management of data. With native support within Apache Spark, and engines like Presto, Hive, and Flink, Apache Hudi is one such tool of choice to build near real-time data pipelines on top of data lakes[10].

To illustrate the performance characteristics of these foundational Data Lakehouse technologies, Jain et al. [15] from UC Berkeley, Stanford University and Databricks conducted a benchmark analysis using the TPC-DS (Transaction Processing Performance Council - Decision Support) suite (figure 4). The 3 TB TPC-DS dataset represents a significant volume of data designed to simulate complex decision support queries typical in business intelligence workloads.

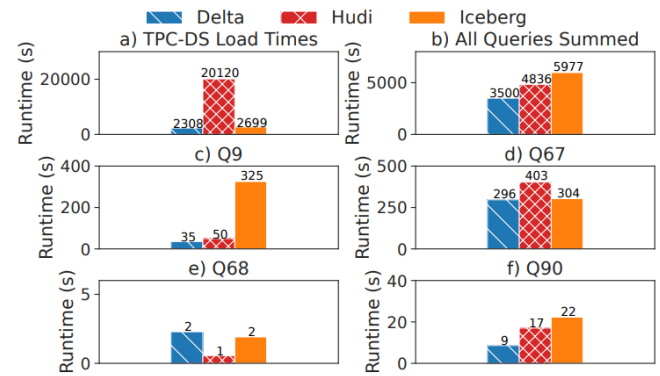


Figure 4: Comparison of Delta Lake, Hudi and Iceberg 3 TB TPC-DS load and query times, including four queries with large differences [15]

Their comparison of Delta Lake, Hudi, and Iceberg on this benchmark revealed notable differences in both data loading times and subsequent query performance, providing empirical evidence for the trade-offs associated with each storage format.

5 SCALABILITY, GOVERNANCE, AND PERFORMANCE

5.1 Data Warehouses

5.1.1 Challenges of Data Warehouses. A 2017 study conducted by Pravin Chandra and his colleagues highlights the various challenges of data warehouses mentioned in different papers [5]. Regarding governance-related issues, the first point to address is the extraction and cleaning of data. This step must be meticulously carried out to ensure the integrity of the information. Transformation and integration are also crucial steps that must guarantee coherent governance: a consistent format, uniform definitions, and correct semantics. In terms of performance, the main challenge for a data warehouse is to achieve efficient retrieval of results. Optimizing the query cache improves DW performance by storing the results of frequent queries directly in memory cache [11]. Furthermore, query optimization plays a key role: by dynamically adapting execution plans based on data characteristics and estimated costs, it is possible to significantly enhance system responsiveness [21].

A 2011 article written by Chaudhary [6] discussed the issue of WH maintenance. The development of new tools to improve maintenance is considered necessary in order to scale a WH effectively. Cloud systems, which are currently widely adopted, help address this issue notably by enabling automatic resource allocation.

5.1.2 Cloud Data Warehouse. In the context of DW scalability, the concept of elasticity can be introduced, which involves the use of cloud systems [1]. Elasticity is defined as the ability of a DW to dynamically adjust its resources (storage or computing capacity) according to demand. It is the DW's capability to adapt to increasing concurrency, the volume of stored data, the number of queries, and the rate of incoming data. To ensure the elasticity capacity of DWs, cloud technology solutions are adopted, allowing flexible adaptation

of allocated resources. Examples include Amazon Redshift, Azure Synapse Analytics, Google BigQuery, and Snowflake. However, the use of a cloud system can present risks such as security breaches, cost overruns, or dependence on a single cloud provider. These risks are not inevitable and can be mitigated through the adoption of an appropriate strategy.

5.2 Data Lakes

One of the primary advantages of Data Lakes is their ability to scale out efficiently [2]. The use of distributed file systems and object storage solutions allows organizations to expand storage capacity in a cost-effective manner. Additionally, the decoupled architecture between compute and storage ensures that processing power can be augmented without necessitating simultaneous changes in storage infrastructure. This flexibility is essential for handling the unpredictable nature of big data workloads, which often require dynamic resource allocation. Such scalability is a key factor when comparing Data Lakes to traditional data warehouses, where scaling typically involves higher costs and architectural rigidity [18].

Despite their flexibility, Data Lakes pose significant challenges in terms of data governance and quality control [12]. The “schema-on-read” paradigm can lead to issues related to data consistency, security, and discoverability if not properly managed. Effective governance frameworks are necessary to prevent the so-called “data swamp” scenario, where data becomes unusable due to poor management practices. To address these challenges, organizations deploy metadata catalogs and automated data quality tools. These governance mechanisms help enforce policies regarding data access, lineage, and compliance with regulatory requirements. For instance, integrating tools like Apache Atlas or cloud-native governance solutions has proven critical in managing the vast and varied datasets in Data Lakes [17].

The performance of Data Lakes is influenced by several factors including the efficiency of data ingestion, processing frameworks, and query engines. While Data Lakes offer high scalability, the lack of predefined schema can result in slower query performance compared to traditional data warehouses optimized for analytical processing. To overcome these limitations, various optimization techniques such as data partitioning, indexing, and the use of caching mechanisms have been developed. The evolution of query engines (e.g., Apache Presto and Apache Drill) has also contributed to narrowing the performance gap, enabling faster access and analysis of large datasets stored in Data Lakes [3, 17].

5.3 Data Lakehouses

The data lakehouse architecture emerges as a compelling evolution, with the objective of synthesizing the strengths of data lakes and data warehouses while mitigating their respective weaknesses [19]. Building on the scalable and cost-effective storage foundation of data lakes, data lakehouses introduce enhanced governance and performance capabilities. By adopting open file formats and transactional table formats like Apache Iceberg or Delta Lake, these architectures enable efficient scaling of both storage and compute, similar to data lakes, while also ensuring ACID properties crucial

for reliable analytics. Furthermore, the integrated governance layers within data lakehouses address the data quality and consistency challenges often associated with traditional data lakes. This unification allows for optimized query performance through features like schema enforcement, partitioning, and indexing, narrowing the performance gap with data warehouses and providing a versatile platform for diverse analytical workloads.

6 RESULTS

As far as scalability, our observations indicate that traditional Data Warehouses face constraints when faced with massive data growth and growing processing needs. Their often proprietary structures and tightly coupled storage and compute resources can render scaling both costly and architecturally complex. On the other hand, Data Lakes, built on distributed file systems and cloud object storage, inherently offer horizontal scalability and cost-effective storage solutions for handling massive structured and processed data. Data Lakehouses inherit this advantageous scalability and low-cost storage model from Data Lakes, which are primarily open file format based. Furthermore, the decoupled nature of storage and compute within Data Lakehouse architectures allows independent resource scaling tailored to specific workload requirements, mirroring the flexibility observed in Data Lakes.

In the domain of governance, the study underscores a key issue with Data Lakes: the danger of uncontrolled, ungoverned data repositories, better known as “Data Swamps”, due to their “schema-on-read” approach and a absence of enforced structure during data ingestion. Data Warehouses, in contrast, are characterized by well-established governance practices that ensure data quality, consistency, and security through the ETL process and a “schema-on-write” methodology. Data Lakehouses are designed to bridge the gap by integrating robust governance features within the flexible storage framework of Data Lakes. By employing transactional table structures such as Apache Iceberg, Delta Lake, and Apache Hudi, Data Lakehouses enforce schema, support ACID properties, and provide key features like data versioning and lineage, thereby significantly improving the reliability and trust of data.

In terms of performance, the research finds that Data Warehouses are traditionally optimized for structured data and analytical queries, achieving efficient data retrieval through mechanisms like indexing and query optimization. Data Lakes, while exhibiting high scalability, can experience performance bottlenecks due to their need to handle raw, often unstructured data during the querying process (“schema-on-read”). Data Lakehouses aim to achieve balance by applying structure and optimization techniques to the data that reside within the data lake. The use of columnar file formats, partitioning strategies, and indexing, facilitated by transactional table formats, help improve query performance on Data Lakehouses compared to traditional Data Lakes. While Data Warehouses may still offer superior performance for specific, highly structured analytical workloads, Data Lakehouses are proposed to offer more of a generalized platform that could serve a variety of analytical types with acceptable performance.

Feature	Data Warehouse	Data Lake	Data Lakehouse
Data	Structured, processed	Structured, semi-structured, unstructured, raw	Structured, semi-structured, unstructured, both processed and raw
Processing	Schema-on-write	Schema-on-read	Schema-on-write and Schema-on-read
Storage	Expensive for large data volumes	Designed for low-cost storage	Designed for low-cost storage
Agility	Less agile, fixed configuration	Highly agile, adjustable configuration	Highly agile, adjustable configuration
Security	Mature	Maturing	Both mature and maturing
Users	Business professionals	Data Scientists et al.	Whole business environment

Table 1: Comparison of DW, DL and Data Lakehouse (source : [19])

Our analysis suggests that the Data Lakehouse architecture represents a promising evolution in data management (table 1). It effectively combines the scalable and cost-efficient storage capabilities of Data Lakes with the robust governance and performance optimization features traditionally associated with Data Warehouses. By addressing the inherent limitations of both preceding architectures, Data Lakehouses offer a unified platform that is better equipped to meet the evolving demands of modern analytics and Business Intelligence, supporting diverse data types and analytical workloads within a single, well-governed, and performant environment.

7 CONCLUSION

In this paper, we have presented a systematic comparison of Data Warehouses, Data Lakes, and Data Lakehouses across three critical dimensions: scalability, governance, and performance. Our review of traditional Data Warehouses highlighted their strength in enforcing schema-on-write, guaranteeing data quality, and delivering optimized query performance for structured analytical workloads. However, we also noted their inherent limitations in elasticity and cost-effective scaling when faced with ever-increasing volumes and varieties of data.

By contrast, Data Lakes offer virtually unlimited, low-cost storage and a flexible schema-on-read paradigm that accommodates raw, semi-structured, and unstructured data without prior transformation. Yet, this flexibility comes at the expense of governance and query efficiency, and poorly governed lakes can quickly degrade into unusable “data swamps.” We have described how metadata catalogs, automated data quality tools, and query optimization techniques can partially mitigate these challenges, though often at considerable operational complexity.

Data Lakehouses emerge as a unifying architecture that synthesizes the advantages of both predecessors. Leveraging open, transactional table formats such as Delta Lake, Apache Iceberg, and Apache Hudi, Lakehouses bring ACID guarantees, schema enforcement, and time-travel capabilities to the low-cost, scalable storage of Data Lakes. At the same time, they preserve the efficient, OLAP-style querying traditionally associated with Data Warehouses. Our analysis demonstrates that Lakehouses can independently scale storage and compute resources, enforce robust governance policies, and deliver competitive performance for a broad spectrum of analytical

workloads, thereby addressing the key shortcomings identified in both Data Warehouses and Data Lakes.

Looking forward, the continued evolution of Lakehouse technologies, particularly in areas such as cost-predictive auto-scaling, unified governance frameworks, and multi-engine query federation, will be critical to fully realize their promise. Future work should focus on large-scale benchmarks across diverse real-world workloads, seamless integration with machine learning and streaming paradigms, and comprehensive assessments of total cost of ownership in multi-cloud and on-premises environments. We believe that the Data Lakehouse paradigm, by harmonizing flexibility, reliability, and performance, will play an indispensable role in next-generation data ecosystems and the advancement of data-driven decision making.

REFERENCES

- [1] Sina Ahmadi. 2023. Elastic Data Warehousing: Adapting To Fluctuating Workloads With Cloud-Native Technologies. *Journal of Knowledge Learning and Science Technology* ISSN: 2959-6386 (online) 2, 3 (Oct. 2023), 282–301. <https://doi.org/10.60087/jkfst.vol2.n3.p301>
- [2] Soukaina Ait Errami, Hicham Hajji, Kenza Ait El Kadi, and Hassan Badir. 2023. Spatial Big Data Architecture: From Data Warehouses and Data Lakes to the LakeHouse. *J. Parallel and Distrib. Comput.* 176 (June 2023), 70–79. <https://doi.org/10.1016/j.jpdc.2023.02.007>
- [3] Michael Armbrust, Tathagata Das, Liwen Sun, Burak Yavuz, Shixiong Zhu, Mukul Murthy, Joseph Torres, Herman Van Hovell, Adrian Ionescu, Alicja Łuszczak, Michał Świtakowski, Michał Szafranski, Xiao Li, Takuya Ueshin, Mostafa Mokhtar, Peter Boncz, Ali Ghodsi, Sameer Paranjpye, Pieter Senster, Reynold Xin, and Matei Zaharia. 2020. Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores. *Proceedings of the VLDB Endowment* 13, 12 (Aug. 2020), 3411–3424. <https://doi.org/10.14778/3415478.3415560>
- [4] Michael Armbrust, Ali Ghodsi, Reynold Xin, and Matei Zaharia. 2021. Lakehouse: A New Generation of Open Platforms That Unify Data Warehousing and Advanced Analytics. (2021).
- [5] Pravin Chandra and Manoj K. Gupta. 2018. Comprehensive survey on data warehousing research. *International Journal of Information Technology* 10, 2 (June 2018), 217–224. <https://doi.org/10.1007/s41870-017-0067-y>
- [6] Sachin Chaudhary, Devendra Prasad Murala, and V K Srivastav. [n. d.]. A Critical Review of Data Warehouse. ([n. d.]).
- [7] Jill Dyché. 2000. *e-Data: Turning Data into Information with Data Warehousing*. Addison-Wesley. 323 pages.
- [8] Sasun Hambardzumyan, Abhinav Tuli, Levon Ghukasyan, Fariz Rahman, Hrnt Topchyan, David Isayan, Mark McQuade, Mikayel Harutyunyan, Tatevik Hakobyan, Ivo Stranic, and Davit Buniatyan. 2022. Deep Lake: A Lakehouse for Deep Learning. <https://doi.org/10.48550/arXiv.2209.10785> arXiv:2209.10785 [cs]
- [9] Ahmed A. Harby and Farhana Zulkernine. 2022. From Data Warehouse to Lakehouse: A Comparative Review. In *2022 IEEE International Conference on Big Data (Big Data)*. 389–395. <https://doi.org/10.1109/BigData55660.2022.10020719>
- [10] Ahmed A. Harby and Farhana Zulkernine. 2025. Data Lakehouse: A Survey and Experimental Study. *Information Systems* 127 (Jan. 2025), 102460. <https://doi.org/10.1016/j.is.2024.102460>
- [11] Ch Anwar Ul Hassan, Muhammad Hammad, Mueen Uddin, Jawaid Iqbal, Jawad Sahi, Saddam Hussain, and Syed Sajid Ullah. 2022. Optimizing the Performance of Data Warehouse by Query Cache Mechanism. *IEEE Access* 10 (2022), 13472–13480. <https://doi.org/10.1109/ACCESS.2022.3148131>
- [12] Tomislav Hlupić, Dražen Orešćanin, Domagoj Ružak, and Mirta Baranović. 2022. An Overview of Current Data Lake Architecture Models. In *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*. 1082–1087. <https://doi.org/10.23919/MIPRO55190.2022.9803717>
- [13] W H Inmon. [n. d.]. Building the Data Warehouse. ([n. d.]).
- [14] W H Inmon. [n. d.]. Wiley Computer Publishing. ([n. d.]).
- [15] Paras Jain, Peter Kraft, Conor Power, Tathagata Das, Ion Stoica, and Matei Zaharia. 2023. Analyzing and Comparing Lakehouse Storage Systems. (2023).
- [16] Dipankar Mazumdar, Jason Hughes, and J. B. Onofre. 2023. The Data Lakehouse: Data Warehousing and More. <https://doi.org/10.48550/arXiv.2310.08697> arXiv:2310.08697 [cs]
- [17] Athira Nambiar and Divyansh Mundra. 2022. An Overview of Data Warehouse and Data Lake in Modern Enterprise Data Management. *Big Data and Cognitive Computing* 6, 4 (Dec. 2022), 132. <https://doi.org/10.3390/bdcc6040132>

- [18] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. 2019. Data Lake Management: Challenges and Opportunities. *Proceedings of the VLDB Endowment* 12, 12 (Aug. 2019), 1986–1989. <https://doi.org/10.14778/3352063.3352116>
- [19] Dražen Oreščanin and Tomislav Hlupić. 2021. Data Lakehouse - a Novel Step in Analytics Architecture. In *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*. 1242–1246. <https://doi.org/10.23919/MIPRO52101.2021.9597091>
- [20] Thammasak Rujiranyong and Jonathan J. Shi. 2006. A Project-Oriented Data Warehouse for Construction. *Automation in Construction* 15, 6 (Nov. 2006), 800–807. <https://doi.org/10.1016/j.autcon.2005.11.001>
- [21] Sai Wu, Feng Li, Sharad Mehrotra, and Beng Chin Ooi. 2011. Query optimization for massively parallel data processing. In *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '11)*. Association for Computing Machinery, New York, NY, USA, 1–13. <https://doi.org/10.1145/2038916.2038928>