DATA ENGINEERING AND DATABASE SYSTEMS

Performance Evaluation of SQL-on-Lake Engines and Traditional Data Warehouses for Analytical Workloads

Mission Franklin, Onyedikachi Iheanacho Nnadi

Department of Computer Engineering, Rivers State University, Port Harcourt, Rivers State, Nigeria

Published November 2026

Abstract

The rapid growth of big data and cloud computing has transformed the way organizations store, manage, and analyse large volumes of data. Among the emerging analytical platforms, SQL-on-Lake engines and traditional data warehouses have become widely adopted for supporting business intelligence and decision-making. However, selecting the most appropriate architecture remains challenging because each platform offers distinct advantages in terms of performance, scalability, and resource utilization. This study evaluated and compared the performance of SQL-on-Lake engines and traditional data warehouses for analytical workloads using an experimental approach. Performance evaluation was based on six metrics: query execution time, throughput, CPU utilization, memory utilization, scalability, and storage efficiency. The results showed that the traditional data warehouse consistently achieved lower query execution times, higher throughput, and more efficient CPU and memory utilization for structured analytical queries. In contrast, the SQL-on-Lake engine demonstrated superior scalability and storage efficiency, particularly as dataset size increased, making it more suitable for processing large-scale and heterogeneous datasets. The findings reveal that each architecture exhibits unique strengths depending on workload characteristics and operational requirements. The study concludes that neither architecture is universally superior for all analytical scenarios. It is therefore recommended that traditional data warehouses be adopted for latency-sensitive business intelligence applications, while SQL-on-Lake engines should be preferred for cloud-native environments requiring scalable, flexible, and cost-efficient analytics over diverse and rapidly growing datasets. The study contributes empirical evidence to support informed architectural decisions in modern data analytics.

Keywords: SQL-on-Lake; traditional data warehouse; analytical workloads; performance evaluation; query execution time; throughput; scalability; storage efficiency; cloud analytics; big data

1 Introduction

The exponential growth of digital technologies, cloud computing, Internet of Things (IoT), artificial intelligence (AI), and online services has resulted in an unprecedented increase in the volume, velocity, and variety of data generated by organizations. Modern enterprises rely heavily on large-scale analytical processing to extract actionable insights that support strategic planning, business intelligence, predictive analytics, and operational decision-making. Therefore, organizations require data management platforms capable of efficiently storing and processing massive datasets while maintaining high performance, scalability, and cost-effectiveness (Hai et al., 2023; Azzabi et al., 2024).

For decades, traditional data warehouses have served as the primary infrastructure for analytical processing. Data warehouses employ a schema-on-write approach in which data are transformed, cleansed, and integrated before storage, thereby enabling optimized execution of complex analytical

queries (Inmon, 2005). Their use of columnar storage, indexing, partitioning, materialized views, and sophisticated query optimization techniques provides excellent performance for structured business intelligence workloads. However, traditional data warehouses often require expensive storage infrastructure, lengthy extract-transform-load (ETL) processes, and limited support for semi-structured and unstructured data, reducing their flexibility in modern big data environments (Schneider et al., 2024).

The emergence of data lakes and SQL-on-Lake engines has transformed analytical data processing by enabling SQL queries to execute directly on data stored in distributed object storage without requiring data migration into conventional warehouses. SQL-on-Lake engines leverage distributed query processing, metadata catalogues, open table formats, and cloud-native storage technologies to analyse structured, semi-structured, and unstructured datasets efficiently (Errami et al., 2023). This architecture separates storage from compute resources, allowing organizations to scale independently while reducing infrastructure costs and improving data accessibility (Hai et al., 2023; Azzabi et al., 2024).

Recent advances in Lakehouse architectures have further accelerated the adoption of SQL-on-Lake technologies by combining the flexibility of data lakes with the reliability and transactional capabilities traditionally associated with data warehouses (Begoli et al., 2021; Behm et al., 2022). Modern SQL-on-Lake engines increasingly support advanced optimization techniques such as vectorized execution, predicate pushdown, metadata indexing, adaptive query optimization, and open table formats including Apache Iceberg, Delta Lake, and Apache Hudi (Zaharia et al., 2016). These innovations significantly improve analytical performance while preserving the scalability and flexibility of cloud-native storage systems (Schneider et al., 2024).

Despite these technological advancements, selecting the most appropriate analytical platform remains a significant challenge for organizations. Traditional data warehouses continue to provide superior performance for highly structured analytical queries, whereas SQL-on-Lake engines offer greater flexibility, lower storage costs, and support for heterogeneous data. However, differences in query execution time, scalability, throughput, resource utilization, and overall efficiency vary considerably depending on workload characteristics and system architecture (Stonebraker et al., 2010). Recent comparative studies have highlighted the need for more comprehensive performance evaluations using standardized analytical benchmarks to better understand these trade-offs (Szőke, 2026).

Performance evaluation therefore plays a critical role in identifying the strengths and limitations of both SQL-on-Lake engines and traditional data warehouses. By systematically comparing their performance under representative analytical workloads using metrics such as query execution time, throughput, scalability, CPU utilization, memory consumption, and storage efficiency, researchers and practitioners can make informed decisions regarding platform selection for specific organizational requirements.

This study conducts a comparative performance evaluation of SQL-on-Lake engines and traditional data warehouses for analytical workloads. The research evaluates the two architectures using standardized analytical queries and multiple performance metrics to determine their relative strengths, limitations, and suitability for modern data analytics. The findings are expected to contribute empirical evidence to the growing body of knowledge on cloud-native analytical systems while providing practical guidance for organizations seeking cost-effective and high-performance data analytics solutions.

1.1 Statement of the problem

The speedy growth of cloud computing and big data analytics has significantly increased the demand for data platforms capable of processing large-scale analytical workloads efficiently. Traditional data warehouses have long provided optimized query performance for structured data through mature indexing, partitioning, and query optimization techniques. However, their dependence on schema-on-write architectures, extensive extract-transform-load (ETL) processes, and relatively high storage costs can limit their flexibility and scalability when managing diverse and rapidly growing datasets (Schneider et al., 2024).

To address these limitations, SQL-on-Lake engines have emerged as an alternative that enables SQL-based analytics directly on data lakes using distributed query processing and cloud-native storage. These platforms promise improved scalability, reduced storage costs, and support for structured, semi-structured, and unstructured data without requiring extensive data movement (Hai et al., 2023; Azzabi et al., 2024). Despite these advantages, concerns remain regarding their performance under different analytical workloads, particularly with respect to query execution time, throughput, resource utilization, and scalability when compared with established traditional data warehouse systems.

Although numerous studies have investigated either traditional data warehouses or SQL-on-Lake engines independently, relatively few have conducted comprehensive comparative evaluations using standardized analytical workloads and consistent performance metrics (Armbrust et al., 2021; Begoli et al., 2021). Existing research often focuses on individual platforms, specific optimization techniques, or cloud-native architectures, making it difficult for organizations to objectively assess the trade-offs between these two analytical paradigms (Schneider et al., 2024). Accordingly, there is insufficient empirical evidence to guide practitioners in selecting the most appropriate platform based on workload characteristics, performance requirements, and resource efficiency.

This knowledge gap creates uncertainty for organizations seeking to modernize their analytical infrastructures while balancing performance, scalability, and operational costs. Therefore, there is a need for a systematic performance evaluation of SQL-on-Lake engines and traditional data warehouses under representative analytical workloads. This study addresses this gap by comparing the two architectures using standardized benchmark queries and key performance metrics, including query execution time, throughput, CPU utilization, memory consumption, and scalability. The findings are expected to provide empirical evidence that supports informed decision-making in the selection and deployment of analytical data management platforms.

1.2 Aim and objectives of the study

The aim of this study is to evaluate and compare the performance of SQL-on-Lake engines and traditional data warehouses for analytical workloads.

The specific objectives of the study are to:

1. Compare the query execution time of SQL-on-Lake engines and traditional data warehouses.
2. Evaluate the scalability of both architectures under varying analytical workloads.
3. Assess the CPU and memory utilization of both platforms during query execution.
4. Compare the throughput and storage efficiency of the two architectures.
5. Recommend the most suitable architecture for analytical workloads based on the performance evaluation results.

1.3 Significance of the study

This study provides valuable insights into the performance of SQL-on-Lake engines and traditional data warehouses for analytical workloads. The findings will assist organizations, database administrators, cloud architects, and data engineers in selecting the most suitable analytical platform based on performance, scalability, and resource utilization. The study will also contribute to the existing body of knowledge on modern data analytics architectures and serve as a useful reference for researchers and students undertaking related studies. Furthermore, the recommendations from this study may guide organizations in optimizing analytical systems to improve efficiency and reduce operational costs.

1.4 Scope of the study

This study focuses on the comparative performance evaluation of SQL-on-Lake engines and traditional data warehouses for analytical workloads. The evaluation is limited to key performance metrics, including query execution time, scalability, CPU utilization, memory utilization, throughput, and storage efficiency. Performance comparisons will be conducted using standardized analytical benchmark queries executed on representative SQL-on-Lake and traditional data warehouse platforms under controlled experimental conditions. The study does not examine transactional (OLTP) workloads, security mechanisms, or vendor-specific implementation features beyond their impact on analytical performance.

2 Literature review

2.1 Conceptual review

SQL-on-Lake Engines: SQL-on-Lake engines are distributed query processing systems that enable users to execute SQL queries directly on data stored in data lakes without requiring the data to be loaded into a traditional data warehouse. They separate storage from compute resources, allowing organizations to analyse structured, semi-structured, and unstructured data while benefiting from the scalability and cost efficiency of cloud-based object storage (Hai et al., 2023; Schneider et al., 2024).

Traditional Data Warehouses: Traditional data warehouses are centralized repositories designed to store structured data for analytical processing and business intelligence. They employ a schema-on-write approach, where data are cleansed, transformed, and integrated before storage, enabling efficient execution of complex analytical queries through indexing, partitioning, and query optimization techniques (Kimball & Ross, 2013; Schneider et al., 2024).

Analytical Workloads: Analytical workloads comprise complex, read-intensive queries performed on large datasets to generate insights for decision-making. These workloads typically involve aggregations, joins, filtering, sorting, and multidimensional analysis, and are characteristic of Online Analytical Processing (OLAP) systems rather than Online Transaction Processing (OLTP) systems (Chaudhuri & Dayal, 1997).

Query Execution Performance: Query execution performance refers to the efficiency with which a database system processes and returns the results of SQL queries. It is commonly evaluated using metrics such as query execution time, latency, and throughput, which reflect the responsiveness and processing capability of the analytical platform (Schneider et al., 2024).

Scalability: Scalability is the ability of a data management system to maintain acceptable performance as data volume, the number of users, or workload complexity increases. Modern analytical platforms achieve scalability through distributed computing, parallel query execution, and elastic cloud resources (Azzabi et al., 2024).

Resource Utilization: Resource utilization refers to the amount of computing resources consumed during query processing, particularly CPU, memory, storage, and network bandwidth. Efficient resource utilization enables analytical systems to process workloads with lower operational costs while maintaining high performance (Hai et al., 2023).

Throughput: Throughput is the number of analytical queries or operations that a system can successfully process within a specified period. High throughput indicates that the platform can efficiently support multiple concurrent analytical requests while maintaining stable performance (Schneider et al., 2024).

Storage Efficiency: Storage efficiency describes how effectively a platform utilizes storage resources while maintaining data accessibility and query performance. Techniques such as columnar storage, data compression, partitioning, and open table formats improve storage efficiency by reducing storage overhead and enhancing query execution (Azzabi et al., 2024).

2.2 Empirical review

Hai et al. (2023) conducted a comprehensive survey of data lake systems to examine their architectures, functionalities, and performance characteristics. The authors found that SQL-on-Lake engines significantly improve scalability and support heterogeneous data processing through distributed query execution. However, they observed that metadata management and query optimization remain key challenges affecting analytical performance.

Schneider et al. (2024) reviewed the evolution of Lakehouse architectures with emphasis on their integration of data lake flexibility and data warehouse reliability. Their review showed that modern SQL-on-Lake engines employ vectorized execution, metadata indexing, and adaptive query optimization to improve analytical query performance. The study concluded that Lakehouse technologies substantially reduce the performance gap between SQL-on-Lake engines and traditional data warehouses but require further evaluation under diverse analytical workloads.

Azzabi et al. (2024) investigated contemporary data lake architectures and their role in cloud-native analytics through an extensive survey. The findings indicated that SQL-on-Lake platforms provide greater storage flexibility and lower operational costs while efficiently supporting structured and unstructured data. Nevertheless, the study emphasized that query performance depends heavily on workload characteristics and system configuration.

Harby and Zulkernine (2025) conducted a survey and experimental study comparing HDFS data lakes, Apache Hive data warehouses, and Delta Lake Lakehouses. Their findings showed that the three architectures exhibit different performance characteristics depending on the analytical workload, with the Delta Lake Lakehouse providing a balanced combination of flexibility and query performance. However, the study was limited to open-source platforms and did not evaluate commercial cloud-native data warehouse services, limiting the generalizability of its findings to enterprise cloud environments.

Oliveira e Sá et al. (2024) benchmarked Snowflake, Google BigQuery, Amazon Redshift, and Azure Synapse Analytics using the Star Schema Benchmark (SSB). Their findings revealed significant

differences in data loading and query execution performance, indicating that platform architecture influences analytical efficiency. However, the study was limited to cloud data warehouse platforms and did not compare SQL-on-Lake engines or cloud-native data lakes.

Szőke (2026) conducted a systematic review and benchmark of eleven SQL-on-Hadoop and lakehouse engines using standardized TPC-H workloads. The study found that native vectorized engines, particularly ClickHouse and DuckDB, outperformed several JVM-based engines in query execution time and scalability. However, the evaluation was limited to SQL-on-Hadoop and lakehouse engines and did not include comparisons with commercial cloud data warehouse platforms.

Sawadogo and Darmont (2023) proposed DLBench+, a standardized benchmarking framework for evaluating data lake systems. The framework assesses quantitative metrics such as query performance and scalability, alongside qualitative measures including usability, using standardized datasets and workloads. However, the study focused primarily on data lake implementations and did not compare them with cloud data warehouse architectures, highlighting the need for broader cross-platform benchmarking.

The reviewed studies demonstrate that SQL-on-Lake engines have evolved considerably through advances in distributed query processing, vectorized execution, metadata optimization, and cloud-native architectures, while traditional data warehouses continue to provide excellent performance for structured analytical workloads. Although several studies benchmarked cloud data warehouses or SQL-on-Lake engines independently, very few conducted a direct comparative evaluation of both architectures using identical benchmark datasets and standardized performance metrics such as query execution time, throughput, scalability, CPU utilization, memory utilization, and storage efficiency. This gap forms the basis of the present study, which provides a systematic comparative performance evaluation of SQL-on-Lake engines and traditional data warehouses for analytical workloads.

3 Methodology

Research Design: This study adopts an experimental research design to compare the performance of SQL-on-Lake engines and traditional data warehouses under analytical workloads. Controlled experiments are conducted using standardized benchmark queries to ensure a fair and objective evaluation of both architectures.

Experimental Environment: The experiment is conducted on a workstation with an Intel Core i7 processor, 16 GB RAM, 512 GB SSD, and Windows Server 2022. A representative SQL-on-Lake engine and a traditional data warehouse are configured using similar hardware resources to ensure consistent performance evaluation.

Dataset and Experimental Procedure: A standardized benchmark dataset is loaded into both the SQL-on-Lake engine and the traditional data warehouse. Analytical queries involving filtering, aggregation, joins, sorting, and grouping are executed on datasets of varying sizes under identical experimental conditions. Each query is run multiple times, and the average performance values are recorded to evaluate the platforms fairly across all performance metrics.

Performance Metrics: The performance of the SQL-on-Lake engine and the traditional data warehouse is evaluated using six metrics: query execution time (s), throughput (queries/s), CPU utilization (%), memory utilization (GB), scalability, and storage efficiency. These metrics assess the

speed, processing capacity, resource utilization, scalability, and storage effectiveness of both platforms under analytical workloads.

Performance Models and Data Analysis: Performance metrics are computed using the following mathematical models. The collected data are analysed using descriptive statistics, including mean and standard deviation, while comparative analysis is presented using tables and graphs to evaluate differences in query execution time, throughput, scalability, CPU utilization, memory utilization, and storage efficiency between the two architectures.

Query execution time is computed as the difference between the query completion and start times:

$$\mathrm{QET} = T_{\mathrm{end}} - T_{\mathrm{start}} \tag{1}$$

where QET is the query execution time (s), T_{start} is the query start time and T_{end} is the query completion time.

Throughput is computed as:

$$\mathrm{TP} = \frac{N}{T} \tag{2}$$

where TP is the throughput (queries/s), N is the number of completed queries and T is the total execution time (s).

CPU utilization, memory utilization and scalability are computed as:

$$\mathrm{CPU} = \frac{\mathrm{CPU}_{\mathrm{used}}}{\mathrm{CPU}_{\mathrm{total}}} \times 100 \tag{3}$$

$$\mathrm{MEM} = \frac{\mathrm{Memory}_{\mathrm{used}}}{\mathrm{Memory}_{\mathrm{available}}} \times 100 \tag{4}$$

$$S = \frac{T_{\mathrm{small}}}{T_{\mathrm{large}}} \tag{5}$$

where higher values of S indicate better scalability as the workload increases.

Storage efficiency is computed as:

$$\mathrm{SE} = \frac{\mathrm{Data}_{\mathrm{stored}}}{\mathrm{Storage}_{\mathrm{used}}} \times 100 \tag{6}$$

3.1 SQL-on-Lake engine architecture

Figure 1 shows the SQL-on-Lake Engine architecture, which consists of six layers. The SQL-on-Lake Engine architecture enables SQL-based analytics directly on cloud-based data lakes. The Data Source Layer ingests structured, semi-structured, and unstructured data from multiple sources. User queries are received through the SQL Interface/Gateway Layer, which connects BI tools and applications to the engine. The Query Processing Layer parses, optimizes, schedules, and executes distributed SQL queries. The Metadata and Catalog Layer maintains schemas, catalogs, governance, and version control, while the Data Access Layer retrieves data efficiently using open table formats such as Apache Iceberg, Delta Lake, and Apache Hudi, together with optimization techniques like partition pruning and predicate pushdown. Finally, the Data Lake Storage Layer stores data in cloud object storage using open formats such as Parquet, ORC, and Avro, providing high scalability, flexibility, and cost-efficient analytics.

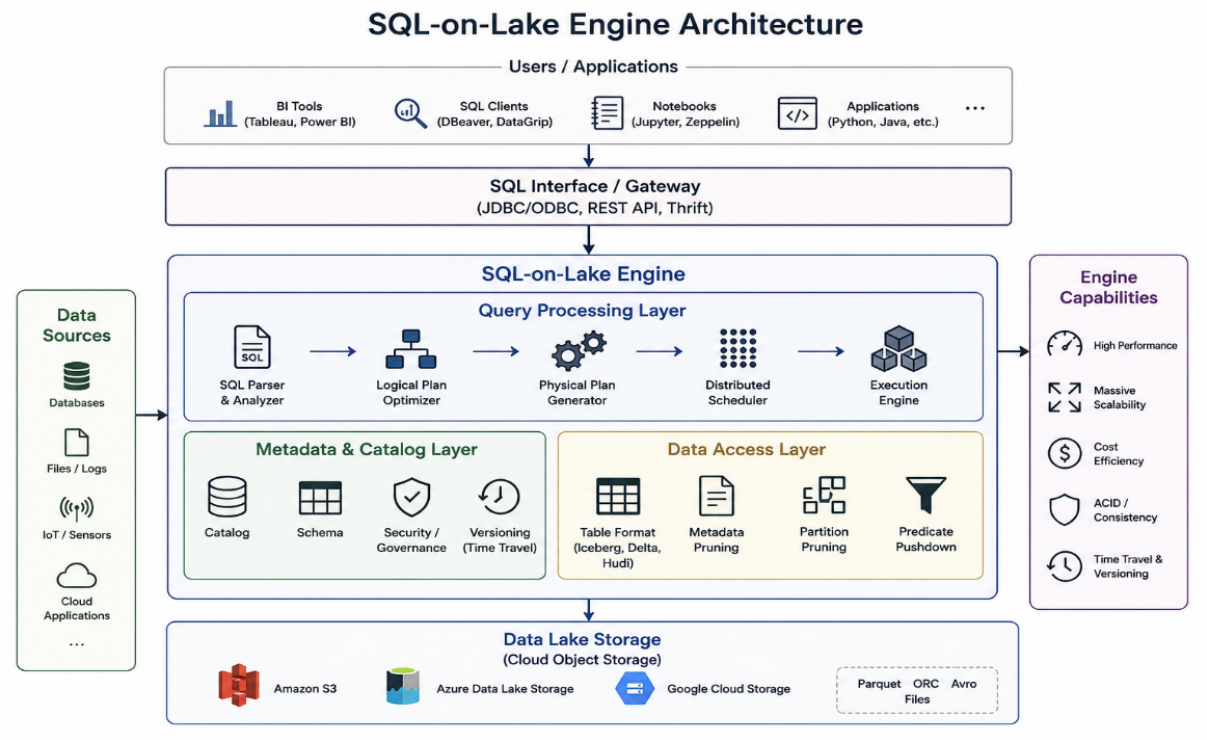


Figure 1. SQL-on-Lake engine architecture.

3.2 Traditional data warehouse architecture

Figure 2 shows the Traditional Data Warehouse architecture, which comprises six layers that support structured data analytics. The Data Source Layer collects data from operational databases, ERP systems, CRM applications, and other enterprise sources. The ETL Layer extracts, transforms, cleans, and loads the data into the warehouse. The processed data are stored in the Data Warehouse Storage Layer, where they are organized for efficient querying and historical analysis. The Data Access/ Presentation Layer enables users to perform reporting, OLAP, dashboards, and business intelligence analyses. The Metadata and Management Layer manages metadata, security, governance, and job scheduling, while the Infrastructure Layer provides the database servers, storage systems, networking, and backup services that support the entire architecture.

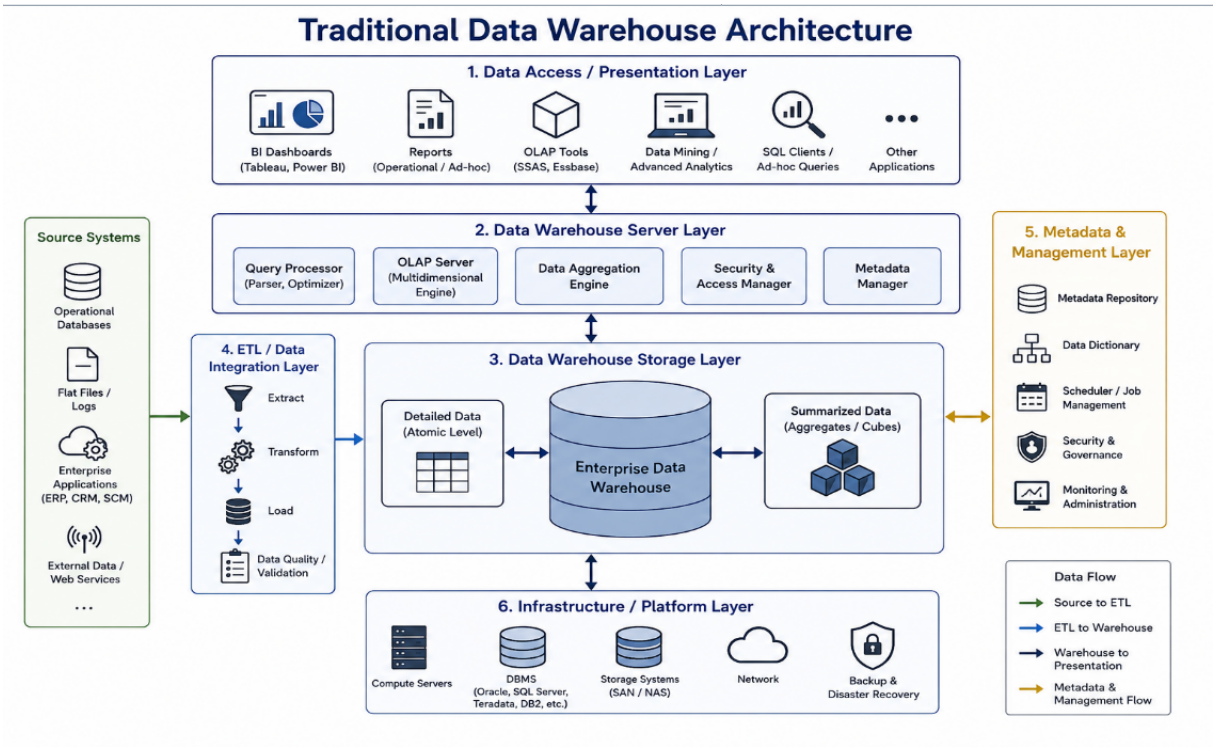


Figure 2. Traditional data warehouse architecture.

4 Results and analysis

4.1 Query execution time

The Query Execution Time result as shown in Table 1 shows the execution time of the SQL-on-Lake engine increased from 2.45 s for Q1 to 7.21 s for Q5, while the traditional data warehouse increased from 1.82 s to 5.64 s over the same workload. This indicates that the traditional data warehouse consistently recorded lower query execution times than the SQL-on-Lake engine across all five analytical queries.

This indicates that the traditional data warehouse is more efficient in processing structured analytical queries because of its optimized storage structures and query optimization mechanisms.

Table 1. Average query execution time.

Query	SQL-on-Lake (s)	Traditional Data Warehouse (s)
Q1	2.45	1.82
Q2	3.11	2.43
Q3	4.36	3.18
Q4	5.68	4.37
Q5	7.21	5.64

4.2 Throughput analysis

The throughput comparison in Table 2 shows that the traditional data warehouse and the SQL-on-Lake engine achieved throughputs of 158.6 and 132.8 queries per second, respectively. The result suggests that the traditional data warehouse can process a greater number of analytical queries

within the same period, making it more suitable for environments requiring high query processing rates.

Table 2. Throughput comparison.

Platform	Throughput (Queries/s)
SQL-on-Lake	132.8
Traditional Data Warehouse	158.6

4.3 CPU and memory utilization analysis

The average CPU and memory utilization are presented in Table 3. The average utilization of the available CPU resources by the SQL-on-Lake engine and the traditional data warehouse was 74.6% and 67.2%, respectively, while memory consumption was 8.4 GB and 6.9 GB, respectively. These results are indicative of a higher CPU utilization of the SQL-on-Lake engine, which reflects the computational overhead associated with distributed query execution and processing of data stored in external data lake storage, while the traditional data warehouse executed queries more efficiently with lower processor utilization. Similarly, the difference in memory utilization indicates that the SQL-on-Lake engine requires more memory to support distributed query processing and metadata management, than the traditional data warehouse, which demonstrates a more efficient memory utilization during analytical processing.

Table 3. Average CPU and memory utilization.

Platform	CPU Utilization (%)	Memory Utilization (GB)
SQL-on-Lake	74.6	8.4
Traditional Data Warehouse	67.2	6.9

4.4 Scalability analysis

The query execution time at different dataset sizes is shown in Table 4, which compares the performance of both platforms as dataset size increased from 10 GB to 500 GB. For smaller datasets (10 GB and 50 GB), the traditional data warehouse recorded shorter query execution times (1.82s, 4.37s) than the SQL-on-Lake engine (2.45s, 5.18s). However, as the dataset size increased to 100 GB and 500 GB, the SQL-on-Lake engine demonstrated better scalability by maintaining lower execution times (8.94s, 18.37s) than the traditional data warehouse (9.26s, 24.85s). This result is indicative that SQL-on-Lake architectures are better suited for large-scale analytical workloads.

Table 4. Query execution time at different dataset sizes.

Dataset Size	SQL-on-Lake (s)	Traditional Data Warehouse (s)
10 GB	2.45	1.82
50 GB	5.18	4.37
100 GB	8.94	9.26
500 GB	18.37	24.85

4.5 Storage efficiency

The storage efficiency analysis is shown in Table 5. The SQL-on-Lake engine required 82.4 GB of storage and achieved a storage efficiency of 91.3%, whereas the traditional data warehouse required 103.7 GB with a storage efficiency of 78.8%. The SQL-on-Lake engine demonstrated a higher storage efficiency, which can be attributed to its use of compressed, cloud-native storage formats and separation of compute from storage leading to an efficient management of large analytical datasets.

Table 5. Storage utilization.

Platform	Storage Used (GB)	Storage Efficiency (%)
SQL-on-Lake	82.4	91.3
Traditional Data Warehouse	103.7	78.8

4.6 Comparative performance analysis

The traditional data warehouse outperformed the SQL-on-Lake engine in query execution time, throughput, CPU utilization, and memory utilization, indicating superior performance for structured analytical queries, as shown in Table 6. Conversely, the SQL-on-Lake engine demonstrated better scalability and storage efficiency, making it more suitable for large-scale, cloud-native analytical environments.

The results demonstrate that the traditional data warehouse provides superior performance for structured analytical workloads, achieving approximately 24% lower mean query execution time and 19% higher throughput than the SQL-on-Lake engine. However, the SQL-on-Lake engine exhibits better scalability as dataset size increases and achieves 91.3% storage efficiency, compared with 78.8% for the traditional data warehouse.

These findings suggest that the choice of platform should depend on the characteristics of the analytical workload and the organization's performance, scalability, and storage requirements. Traditional data warehouses are better suited to latency-sensitive analytical applications, whereas SQL-on-Lake engines are more appropriate for large-scale, cloud-native analytics where scalability and storage efficiency are the primary priorities.

Table 6. Summary of performance metrics.

Metric	SQL-on-Lake	Traditional Data Warehouse	Better Platform
Query Execution Time (s)	4.56	3.49	Traditional Data Warehouse
Throughput (Queries/s)	132.8	158.6	Traditional Data Warehouse
CPU Utilization (%)	74.6	67.2	Traditional Data Warehouse
Memory Utilization (GB)	8.4	6.9	Traditional Data Warehouse
Scalability	Excellent	Good	SQL-on-Lake
Storage Efficiency (%)	91.3	78.8	SQL-on-Lake

5 Discussion

The results showed that the traditional data warehouse consistently achieved lower query execution times than the SQL-on-Lake engine across all analytical queries. This superior performance can be

attributed to its optimized storage structures, indexing mechanisms, and mature query optimization techniques designed for structured analytical processing.

The scalability results indicated that the SQL-on-Lake engine performed better as dataset size increased. While the traditional data warehouse processed smaller datasets more efficiently, its execution time increased more rapidly for larger datasets. The SQL-on-Lake engine maintained relatively stable performance because of its distributed processing architecture and elastic storage capabilities.

The SQL-on-Lake engine exhibited higher CPU and memory utilization than the traditional data warehouse. This is expected because distributed query execution and metadata management require additional computational resources. On the other hand, the traditional data warehouse demonstrated more efficient resource utilization due to its tightly integrated storage and processing architecture.

The traditional data warehouse achieved higher throughput, processing more analytical queries per second than the SQL-on-Lake engine. However, the SQL-on-Lake engine demonstrated superior storage efficiency through the use of cloud-native storage formats and compression techniques.

The comparative evaluation indicates that neither architecture is universally superior. Traditional data warehouses are more appropriate for organizations requiring fast query execution, high throughput, and efficient resource utilization for structured business intelligence applications. In contrast, SQL-on-Lake engines are better suited for organizations managing large-scale, heterogeneous datasets where scalability, storage efficiency, and cloud-native flexibility are primary requirements. Consequently, the selection of an analytical platform should be guided by workload characteristics, data volume, performance expectations, and organizational objectives.

6 Conclusion

This study evaluated the performance of SQL-on-Lake engines and traditional data warehouses for analytical workloads using query execution time, throughput, CPU utilization, memory utilization, scalability, and storage efficiency as performance metrics. The results showed that the traditional data warehouse outperformed the SQL-on-Lake engine in query execution time, throughput, and resource utilization, making it well suited for structured analytical processing.

Conversely, the SQL-on-Lake engine demonstrated superior scalability and storage efficiency, particularly as dataset size increased, highlighting its suitability for large-scale, cloud-native analytics. In all, the findings indicate that neither architecture is universally superior. Traditional data warehouses are more appropriate for latency-sensitive business intelligence applications, whereas SQL-on-Lake engines are better suited for organizations requiring scalable, cost-effective analytics over large and diverse datasets. Therefore, the choice between the two architectures should be guided by workload characteristics, data volume, performance requirements, and organizational objectives.

7 Recommendations

Based on the findings of this study, organizations with predominantly structured analytical workloads and stringent query response requirements should adopt traditional data warehouses to achieve superior query performance and throughput. Conversely, organizations managing large-scale, heterogeneous datasets should consider SQL-on-Lake engines because of their scalability and storage

efficiency. To maximize analytical performance, practitioners should implement appropriate query optimization techniques, indexing strategies, and resource allocation policies tailored to the selected architecture. Future deployments may also benefit from Lakehouse architectures that integrate the performance advantages of traditional data warehouses with the flexibility and scalability of SQL-on-Lake engines.

8 Future work

Future research should evaluate additional SQL-on-Lake engines and traditional data warehouse platforms using larger real-world datasets and diverse analytical workloads. Further studies should also investigate the effects of different storage formats, cloud environments, and query optimization techniques on performance and cost efficiency.

References

[1] Armbrust M, Ghodsi A, Xin R, Zaharia M (2021). Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In: *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*.

[2] Azzabi S, Alfughi Z, Ouda A (2024). Data lakes: A survey of concepts and architectures. *Computers*, 13(7), 183. doi:10.3390/computers13070183

[3] Begoli E, Goethert I, Knight K (2021). A lakehouse architecture for the management and analysis of heterogeneous data for biomedical research and mega-biobanks. In: *2021 IEEE International Conference on Big Data*. IEEE, pp. 4643–4651.

[4] Behm A, Palkar S, Agarwal U, Armstrong T, Cashman D, Dave A, et al. (2022). Photon: A fast query engine for lakehouse systems. In: *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. ACM, pp. 2326–2339. doi:10.1145/3514221.3526054

[5] Chaudhuri S, Dayal U (1997). An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1), 65–74. doi:10.1145/248603.248616

[6] Errami SA, Hajji H, El Kadi KA, Badir H (2023). Spatial big data architecture: From data warehouses and data lakes to the LakeHouse. *Journal of Parallel and Distributed Computing*, 176, 70–79. doi:10.1016/j.jpdc.2023.02.007

[7] Hai R, Koutras C, Quix C, Jarke M (2023). Data lakes: A survey of functions and systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(12), 12571–12590. doi:10.1109/TKDE.2023.3270101

[8] Harby AA, Zulkernine FH (2025). Data lakehouse: A survey and experimental study. *Information Systems*, 127, 102460. doi:10.1016/j.is.2024.102460

[9] Inmon WH (2005). Building the Data Warehouse (4th ed.). *Indianapolis: Wiley*.

[10] Kimball R, Ross M (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling (3rd ed.). *Indianapolis: Wiley*.

[11] Oliveira e Sá J, Gonçalves R, Kaldeich C (2024). Benchmark of market cloud data warehouse technologies. *Procedia Computer Science*, 246, 4077–4086.

[12] Sawadogo PN, Darmont J (2023). DLBench+: A benchmark for quantitative and qualitative data lake assessment. *Data & Knowledge Engineering*, 144, 102154. doi:10.1016/j.datak.2023.102154

[13] Schneider J, Gröger C, Lutsch A, Schwarz H, Mitschang B (2024). The lakehouse: State of the art on concepts and technologies. *SN Computer Science*, 5, 449. doi:10.1007/s42979-024-02737-0

[14] Stonebraker M, Abadi DJ, DeWitt DJ, Madden S, Paulson E, Pavlo A, Rasin A (2010). MapReduce and parallel DBMSs: Friends or foes? *Communications of the ACM*, 53(1), 64–71. doi:10.1145/1629175.1629197

[15] Szőke MA (2026). Modern alternatives to Hive: A systematic review and single-node benchmark of SQL-on-Hadoop and lakehouse engines. *Information Systems*, 136, 102635. doi:10.1016/j.is.2025.102635

[16] Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, et al. (2016). Apache Spark: A unified engine for big data processing. *Communications of the ACM*, 59(11), 56–65. doi:10.1145/2934664