An Experimental Evaluation of a Hybrid SQL–NoSQL Architecture for Scalable E-Commerce Analytics

Brahim Ben Lhoucine
Information Technology, Data and Applied Mathematics Science Team (ESTIDMA), National School of Applied Science, Ibn Zohr University, Agadir, Morocco
Brahim.benlhoucine@edu.uiz.ac.ma (corresponding author)

Khalid Tatane
Information Technology, Data and Applied Mathematics Science Team (ESTIDMA), National School of Applied Science, Ibn Zohr University, Agadir, Morocco
k.tatane@uiz.ac.ma

Laila Dahr
Information Technology, Data and Applied Mathematics Science Team (ESTIDMA), National School of Applied Science, Ibn Zohr University, Agadir, Morocco
l.dahr@uiz.ac.ma

Received: 17 April 2026 | Revised: 20 April 2026 and 11 May 2026 | Accepted: 15 May 2026

 DOI: https://doi.org/10.48084/etasr.18578

ABSTRACT

The rapid growth of data volume, structural heterogeneity, and real-time processing demands in modern e-commerce platforms challenges the scalability and flexibility of single-model database systems. Hybrid database architectures, which integrate multiple storage paradigms within a coordinated framework, offer a promising alternative; however, controlled empirical evaluation across heterogeneous models remains limited. This study presents a structured experimental assessment of a hybrid SQL–NoSQL architecture integrating relational, document-oriented, in-memory, column-family, graph, and streaming components within a unified deployment environment. The architecture incorporates PostgreSQL, MongoDB, Redis, Cassandra, and Neo4j, coordinated through Kafka-based streaming with Change Data Capture (CDC) synchronization using Debezium. Performance evaluation focused on reproducible benchmarking of representative subsystems under standardized hardware constraints. Key performance indicators—including throughput, response latency, and scalability under increasing concurrency—were measured using established workload simulation tools. Results demonstrate clear workload-dependent specialization: relational processing maintained stable transactional behavior, document-oriented storage supported higher ingestion throughput, and in-memory caching achieved minimal access latency. The findings indicate that distributing workload categories across specialized storage engines improves operational balance compared with monolithic deployments. Beyond quantitative benchmarking, the study provides a reproducible architectural framework illustrating how coordinated multi-model systems can support scalable and adaptable data infrastructures for real-time e-commerce analytics.

Keywords-hybrid database architecture; SQL–NoSQL integration; polyglot persistence; distributed data systems; database benchmarking; Change Data Capture (CDC); e-commerce analytics; real-time processing

I. INTRODUCTION

Modern digital platforms generate massive volumes of heterogeneous data that must be processed efficiently for transactional and analytical purposes. Traditional Relational Database Management Systems (RDBMSs) have historically served as the backbone of enterprise data infrastructures due to their strong consistency guarantees and structured schema design. However, the rapid growth of large-scale web applications and e-commerce platforms has exposed limitations in relational systems regarding horizontal scalability and flexible data representation [1, 2]. To address these limitations, NoSQL databases were introduced, offering distributed architectures, flexible schemas, and high scalability across multiple nodes. Systems such as Cassandra and MongoDB demonstrate significant advantages in handling large-scale

datasets and high-throughput workloads typical of modern web environments [2, 3].

Distributed systems must also manage trade-offs between consistency, availability, and partition tolerance, commonly described through the CAP theorem framework [4, 5]. Because different applications require different trade-offs, relying on a single database technology may not always provide optimal performance. Hybrid database architectures have therefore emerged as a promising solution. These architectures combine multiple database paradigms—including relational, document, key–value, and graph databases—within a unified infrastructure, allowing each system to handle the workloads for which it is best suited [6-8]. Additionally, modern data architectures increasingly rely on event-driven data pipelines and Change Data Capture (CDC) mechanisms to synchronize data between heterogeneous systems. Technologies such as streaming platforms enable near real-time propagation of database changes, facilitating scalable and flexible data integration strategies [9, 10].

This study investigates the performance and practical feasibility of hybrid SQL–NoSQL architectures in the context of scalable e-commerce data systems. Using standardized Yahoo! Cloud Serving Benchmark (YCSB) benchmarking workloads and real-time data synchronization pipelines, this study evaluates the performance characteristics of multiple database models and demonstrates how hybrid architectures can effectively support heterogeneous workloads in modern digital platforms.

A. Contributions

The primary contributions of this study are summarized as follows:

1. A unified experimental framework for evaluating hybrid SQL–NoSQL architectures under controlled and reproducible conditions.
2. A workload-driven comparative benchmarking analysis of relational, document-oriented, and in-memory database components within a coordinated multi-model deployment.
3. An architectural integration model demonstrating practical coordination of heterogeneous storage engines using streaming synchronization and CDC mechanisms.
4. An application-oriented evaluation contextualizing hybrid database performance within scalable e-commerce analytics environments.

This study contributes to the literature by providing an experimental evaluation of a hybrid SQL–NoSQL architecture deployed in a unified containerized environment. The study focuses on workload-driven performance analysis using YCSB benchmarking and real-time CDC pipelines.

II. RELATED WORK

Database technologies have evolved significantly in response to the rapid growth of data generated by modern digital platforms. Traditional RDBMSs have long been the dominant technology for structured data storage due to their strong transactional guarantees and well-established query mechanisms [1, 2]. However, the increasing scale and diversity of data produced by web applications have exposed limitations in relational systems, particularly regarding horizontal scalability and schema flexibility. To overcome these limitations, NoSQL databases emerged as an alternative paradigm designed for distributed environments. Systems such as Cassandra introduced decentralized architectures capable of scaling across multiple nodes while maintaining high availability [3]. Similarly, document-oriented databases allow flexible schema definitions and are widely used in applications that require rapid data evolution.

Several studies have evaluated the performance differences between relational and NoSQL systems under large-scale workloads. Authors in [2] demonstrated that NoSQL databases often outperform relational systems in write-intensive scenarios when deployed in distributed environments. Recent work by authors in [11] has also evaluated SQL and NoSQL database performance in parallel processing environments, highlighting workload-dependent behavior across database models. Other comparative studies confirm that NoSQL architectures provide advantages in handling high-throughput and unstructured data workloads [9, 10].

More recently, researchers have explored hybrid and multi-model database architectures that combine different storage paradigms within a unified data infrastructure. Authors in [12] argued that modern data systems should adopt specialized storage models optimized for specific workloads rather than relying on a single universal database system. Similarly, authors in [8] proposed hybrid database frameworks capable of integrating relational and NoSQL systems to support heterogeneous workloads.

Polyglot persistence has therefore become an increasingly popular architectural approach. This strategy allows organizations to deploy multiple database technologies simultaneously, selecting the most appropriate system for each data processing requirement [13]. Such architectures enable improved scalability, flexibility, and performance compared with traditional monolithic database solutions. In addition to storage technologies, modern data architectures increasingly incorporate streaming platforms and event-driven data pipelines. These mechanisms enable asynchronous data synchronization between heterogeneous databases and support near-real-time analytics capabilities [9]. Despite these advancements, there remains limited empirical evaluation of hybrid database architectures within realistic application scenarios such as e-commerce platforms. Consequently, this study aims to experimentally evaluate the performance of hybrid SQL–NoSQL architectures and assess their suitability for scalable transactional and analytical workloads.

Recent research has increasingly focused on the evolution of hybrid and multi-model database systems. Comparative studies have evaluated the performance of multi-model databases against polyglot persistence approaches, highlighting trade-offs between integration complexity and query efficiency [14]. More recent work has explored hybrid querying techniques that integrate relational databases with advanced

processing systems, demonstrating the growing importance of hybrid architectures in modern data environments [15]. Furthermore, recent experimental studies have compared hybrid multi-model architectures with native multi-model databases, showing that while hybrid systems provide stronger transactional consistency, native multi-model solutions may offer improved performance for cross-model queries [16].

These recent developments indicate that hybrid database systems remain an active research area. However, many existing studies focus either on theoretical frameworks or specific system optimizations, with limited emphasis on unified experimental evaluation using real-time data synchronization pipelines. This gap motivates the present study.

III. METHODOLOGY

This study evaluates the performance of hybrid database architectures through a controlled experimental framework combining multiple database models and standardized benchmarking workloads. The objective is to analyze how different database paradigms perform individually and when integrated within a hybrid architecture designed for scalable e-commerce systems.

A. Experimental Environment

All experiments were conducted within a containerized environment using Docker to ensure reproducibility and isolation between system components. The experimental infrastructure included several database technologies representing different data models:

- PostgreSQL representing relational database systems (SQL model).
- MongoDB representing document-oriented NoSQL databases.
- Cassandra representing distributed column-family databases.
- Redis representing in-memory key–value data stores.
- Neo4j representing graph database systems.

These systems were deployed simultaneously within the experimental environment to simulate a heterogeneous data infrastructure typical of modern digital platforms.

B. Hybrid Architecture Design

The hybrid architecture implemented in this study integrates multiple database systems through an event-driven data pipeline. PostgreSQL served as the primary transactional database responsible for processing structured order transactions. To enable real-time data synchronization across systems, a CDC mechanism was implemented using Debezium. Debezium monitors the PostgreSQL Write-Ahead Log (WAL) and captures database changes as structured events. These events are then transmitted to Apache Kafka, which acts as a distributed event-streaming platform responsible for propagating data updates across the system. Kafka topics therefore function as a real-time data bus connecting the different storage technologies. Downstream databases such as MongoDB can subscribe to Kafka topics and ingest the change events, enabling asynchronous data replication and cross-database synchronization.

C. Benchmarking Workloads

Performance evaluation was conducted using the YCSB, which is widely used for evaluating the performance of distributed database systems [17]. The benchmark simulated typical application workloads consisting of:

- Read operations.
- Update operations.
- Mixed read–write workloads.

These workloads were executed against each database system individually in order to measure their performance characteristics under comparable experimental conditions.

D. Performance Metrics

The experimental evaluation focused on several key performance indicators commonly used in database benchmarking studies:

- Throughput (operations per second).
- Average latency.
- 95th percentile latency.
- System stability under sustained workloads.

These metrics allow direct comparison of database performance under identical benchmarking scenarios.

E. Experimental Procedure

The experimental procedure consisted of two main phases. First, baseline performance tests were executed for each database system independently in order to establish reference performance metrics. These experiments measured the ability of each system to handle transactional workloads under standardized YCSB benchmarks. Second, the hybrid architecture was evaluated by enabling real-time data propagation through the CDC pipeline connecting PostgreSQL, Kafka, and downstream systems. This phase allowed observation of data synchronization behavior and evaluation of the system's capability to support distributed data processing across heterogeneous storage engines. This methodology enables a comprehensive evaluation of hybrid database architectures and provides insights into their practical applicability for scalable data-intensive applications.

Each benchmark scenario was executed five times under identical experimental conditions. Reported values represent averaged measurements obtained from repeated runs in order to reduce transient system variability. Observed measurement variance remained low across repeated executions, indicating stable benchmarking behavior and acceptable experimental consistency.

Benchmark workloads were generated using standardized YCSB configurations adapted for relational, document-oriented, and in-memory database systems. The experimental environment was deployed using Docker-based

containerization to facilitate reproducibility of the evaluation process.

IV. SYSTEM ARCHITECTURE AND EXPERIMENTAL SETUP

A. Hybrid SQL–NoSQL Architecture Design

The proposed experimental framework implements a hybrid multi-model database architecture designed to distribute workload categories across heterogeneous storage engines. The architecture integrates relational, document, in-memory, column-family, graph, and streaming components within a coordinated deployment environment. PostgreSQL is used to manage structured transactional data requiring ACID compliance and complex relational querying. MongoDB supports semi-structured data ingestion and flexible schema storage. Cassandra provides distributed column-family storage optimized for write-intensive workloads and horizontal partitioning [3]. Redis functions as an in-memory key–value store to reduce latency for frequently accessed session and lookup data. Neo4j is deployed to evaluate relationship-intensive queries and graph traversal operations. Inter-system synchronization is achieved using Apache Kafka as the event-streaming backbone, with CDC mechanisms used to propagate transactional updates across subsystems. Event-driven streaming architectures are widely used to enable scalable data synchronization between distributed storage systems [9]. This configuration enables near real-time data consistency while maintaining loose coupling between storage engines.

The overall system architecture is illustrated in Figure 1.

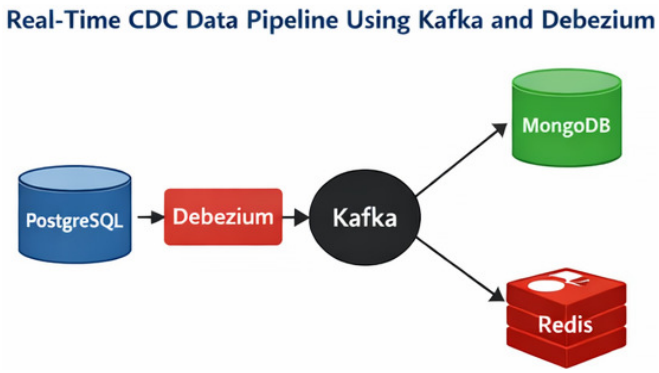

Fig. 1. Hybrid SQL–NoSQL architecture with CDC-based data synchronization.

B. Data Model and Schema Distribution

Data distribution within the hybrid architecture follows workload specialization principles. Structured transactional records are maintained within relational schemas. Semi-structured product and behavioral datasets are mapped to document collections. High-frequency lookup data is allocated to in-memory storage to minimize response latency. Event-driven analytical records are directed to column-family storage to support distributed write scalability. Graph-based datasets encode relationships among customers, products, and interactions to support recommendation-style and network-oriented queries. This workload-aligned segmentation enables evaluation of performance characteristics across heterogeneous data models within a unified system. The database components and their primary functional roles are presented in Table I.

TABLE I. DATABASE COMPONENTS AND FUNCTIONAL ROLES

Component	Database type	Primary role
PostgreSQL	Relational	Transaction processing
MongoDB	Document	Semi-structured storage
Redis	Key–value	Caching
Cassandra	Column-family	Event storage
Neo4j	Graph	Relationship analytics
Kafka/Debezium	Streaming	Data synchronization

C. Experimental Environment

Experiments were conducted in a standardized containerized environment using Docker to ensure uniform resource allocation across all database instances. CPU cores, memory limits, and storage parameters were maintained consistently to prevent configuration bias.

System deployment followed documented configuration guidelines for each technology platform to ensure operational stability. Streaming pipelines responsible for event propagation remained stable throughout the evaluation process in order to avoid measurement interference. The experimental environment configuration is presented in Table II.

TABLE II. EXPERIMENTAL ENVIRONMENT CONFIGURATION

Parameter	Description
Dataset size	1 million records
Transaction types	Read / write / mixed
Concurrency levels	100 / 500 / 1,000 users
Benchmark tool	YCSB [17]
Measurement metrics	Throughput, latency
CPU	Apple M2 (8 cores)
RAM	16 GB
Storage	SSD
Deployment	Docker containerized environment

All experiments were conducted under identical hardware and container configurations to ensure consistency and reproducibility of results.

D. Benchmarking Procedures

Workloads representative of e-commerce analytics were executed across database subsystems. These workloads included transactional inserts, catalog retrieval queries, session lookups, event ingestion, and graph traversal operations. Performance evaluation followed standardized benchmarking conventions consistent with cloud-serving evaluation methodologies such as YCSB [17]. Throughput and latency were measured under progressively increasing concurrency levels in order to assess scalability behavior. Each benchmark scenario was executed multiple times to ensure measurement consistency. Average values were calculated to reduce the influence of transient system variability. Comparative analysis was then conducted across storage engines to identify workload-dependent performance characteristics.

V. RESULTS AND ANALYSIS

Workloads were categorized into three types:

- Read-intensive workloads.
- Write-intensive workloads.
- Mixed workloads.

This classification enables a clearer analysis of system performance under different operational scenarios.

The throughput performance of the evaluated systems is presented in Figure 2.

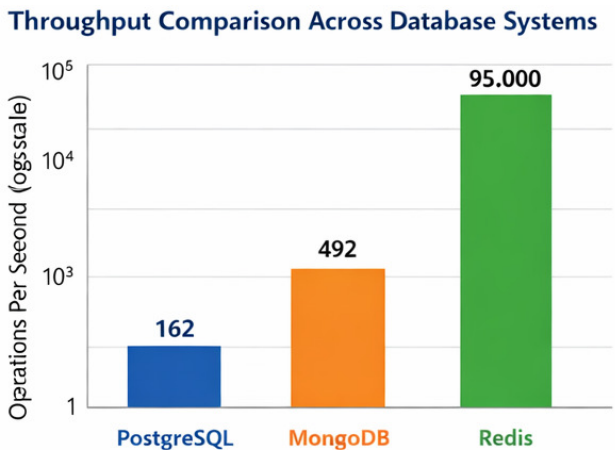

Fig. 2. Throughput comparison across database systems under YCSB workloads.

A. Performance Evaluation

The hybrid database architecture was evaluated under read-intensive, write-intensive, and mixed workloads representative of e-commerce operations. Empirical benchmarking focused on components for which consistent performance measurements were obtained under controlled execution conditions, specifically PostgreSQL, MongoDB, and Redis. MongoDB demonstrated higher ingestion throughput for semi-structured document operations compared to relational transaction processing. PostgreSQL maintained stable performance for structured transactional workloads under moderate concurrency. Redis achieved significantly higher operation rates for cache retrieval tasks due to memory-resident execution. The throughput comparison of the benchmarked components is presented in Table III.

TABLE III. THROUGHPUT COMPARISON OF BENCHMARKED COMPONENTS

Database engine	Workload type	Throughput (ops/sec)
PostgreSQL	Transaction processing	162
MongoDB	Document ingestion	491
Redis	Cache retrieval	100,000+

B. Latency Analysis

Latency measurements show clear differentiation among evaluated components. Redis exhibited the lowest response times, reflecting in-memory execution advantages. MongoDB demonstrated moderate latency variability under concurrent document workloads. PostgreSQL maintained predictable latency patterns for structured transactional queries.

Latency behavior supports workload-aligned allocation within the hybrid architecture, where caching, document ingestion, and transactional processing are distributed across specialized engines.

Latency performance is a critical metric for evaluating real-time system responsiveness. The latency comparison across database systems is presented in Figure 3.

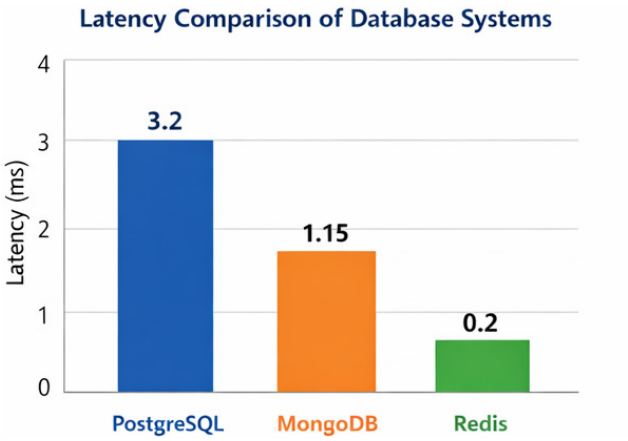

Fig. 3. Latency comparison of database systems.

Figure 3 shows that Redis achieves the lowest latency due to its in-memory architecture, followed by MongoDB, whereas PostgreSQL exhibits higher latency due to transactional overhead.

To further analyze Redis performance under different workloads, latency measurements for various operations are presented in Figure 4.

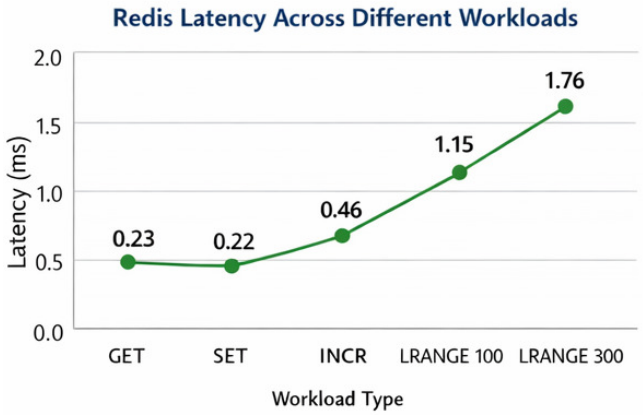

Fig. 4. Redis latency across different workloads.

C. Scalability Analysis

Scalability testing was conducted by increasing concurrency levels from 100 to 1,000 simulated users.

MongoDB sustained improved ingestion scalability under increasing load. PostgreSQL demonstrated stable behavior under moderate concurrency with gradual performance decline at higher contention levels. Redis maintained stable performance for high-frequency retrieval tasks.

These results indicate that distributing workload responsibilities across specialized components reduces bottlenecks typically observed in monolithic database deployments.

VI. DISCUSSION

A. Interpretation of Findings

The experimental evaluation indicates that performance within a hybrid database environment is strongly workload-dependent. Rather than demonstrating universal superiority of a particular storage model, the results show that different engines exhibit strengths aligned with specific operational patterns. PostgreSQL maintained stable transactional behavior under structured workloads, while MongoDB achieved higher ingestion throughput for semi-structured operations. Redis consistently delivered minimal latency for high-frequency retrieval tasks. These findings support the principle that architectural specialization improves operational efficiency. These findings are consistent with recent studies on hybrid and multi-model database systems, which emphasize workload-dependent performance and the benefits of integrating heterogeneous storage paradigms. Instead of relying on a single database paradigm to satisfy heterogeneous workload demands, distributing responsibilities across storage models reduces performance bottlenecks and improves responsiveness. The results therefore reinforce the practical viability of workload-aligned data segmentation within hybrid deployments.

B. Addressing Scalability Considerations

Scalability testing revealed that document-oriented and in-memory components sustained performance more effectively under increasing concurrency levels than purely relational processing in write-intensive scenarios. PostgreSQL exhibited predictable performance degradation under higher contention, reflecting typical transactional coordination constraints. The hybrid configuration mitigated these limitations by redistributing workload categories. Caching reduced read pressure on transactional storage, while document-based ingestion absorbed flexible data input streams. This redistribution illustrates how coordinated multi-model systems can manage concurrency growth without requiring uniform scaling of a single storage engine.

Streaming-based synchronization further supported architectural cohesion by enabling timely data propagation between subsystems. While the evaluation focused on controlled benchmarking rather than production-scale deployment, the integration approach demonstrates practical feasibility for distributed environments.

C. Strengths and Limitations

A primary strength of this study lies in its controlled cross-model benchmarking under standardized conditions. Evaluating multiple storage paradigms within a unified framework improves comparability and reduces environmental variability that often complicates independent system assessments. However, certain limitations must be acknowledged. Synthetic workload generation, while necessary for repeatability, does not fully replicate real-world behavioral complexity. Additionally, experiments were conducted within resource-constrained environments, which may influence absolute throughput values. The findings therefore emphasize comparative trends rather than universal performance guarantees.

Furthermore, quantitative benchmarking was limited to subsystems with fully validated measurement cycles. While other components were integrated architecturally, their performance characteristics warrant more extensive dedicated evaluation in future studies.

D. Positioning Within Existing Research

The findings contribute to ongoing discussions regarding multi-model database strategies by providing controlled empirical evidence of workload-dependent performance differentiation. Rather than framing hybrid architectures as replacements for existing paradigms, the results suggest that coordinated specialization offers measurable operational benefits under heterogeneous workload conditions.

By linking empirical benchmarking outcomes to practical architectural decisions, this study complements prior theoretical analyses and system-level investigations into distributed database trade-offs. The emphasis on controlled evaluation and reproducibility strengthens its relevance within contemporary database research discourse.

Although Cassandra and Neo4j were included in the architectural design, their quantitative evaluation was limited in this study. Cassandra benchmarking in a single-node setup resulted in client overload and zero throughput, highlighting the need for distributed cluster configurations for accurate performance evaluation. Neo4j evaluation requires graph-specific workloads, which were beyond the scope of the current study and are considered for future work.

Future work will extend this study by evaluating Cassandra in a multi-node cluster and Neo4j using graph-specific query workloads.

VII. E-COMMERCE APPLICATION IMPLICATIONS

A. Practical Relevance

Modern e-commerce platforms must support transactional integrity, behavioral logging, personalization workflows, and real-time data synchronization. The evaluated hybrid architecture illustrates how separating these responsibilities across specialized storage engines can improve operational balance.

Transactional consistency remains anchored within relational storage, while document-based ingestion supports flexible catalog and interaction data. In-memory caching reduces latency for session and lookup operations. These functional separations demonstrate how workload segmentation can improve system responsiveness without overburdening a single database model.

B. Operational Considerations

From an operational perspective, hybrid architectures provide flexibility in scaling individual subsystems according to workload pressure. Rather than scaling the entire data infrastructure uniformly, organizations can allocate resources selectively to engines experiencing demand growth. This modularity reduces structural rigidity and supports incremental system evolution.

Streaming synchronization mechanisms enable coordinated data updates across subsystems while preserving loose coupling. Such integration supports analytics pipelines that require timely state propagation without centralizing all storage operations.

C. Future Research Directions

Future investigations should extend benchmarking to fully distributed multi-node configurations and incorporate real-world workload traces to validate performance under production-like conditions. Expanded evaluation of graph and column-oriented subsystems under controlled experimental designs would further clarify performance trade-offs within integrated hybrid deployments.

Additionally, emerging workloads involving machine learning inference, real-time personalization, and cross-channel integration present opportunities for evaluating hybrid architectures under increasingly complex analytical scenarios.

VIII. CONCLUSION AND FUTURE WORK

This study examined the performance behavior of a hybrid SQL–NoSQL database architecture under workloads representative of scalable e-commerce analytics environments. By integrating relational, document-oriented, in-memory, column-family, graph, and streaming components within a unified experimental framework, the research evaluated how workload-aligned specialization influences system performance. Empirical benchmarking demonstrated that performance outcomes vary according to workload characteristics. PostgreSQL maintained stable transactional behavior under structured operations. MongoDB exhibited higher ingestion throughput for semi-structured workloads. Redis consistently achieved minimal latency for high-frequency retrieval tasks. These findings indicate that distributing workload categories across specialized storage engines improves responsiveness compared with reliance on a single database model.

While additional components such as column-family and graph subsystems were integrated within the architecture, quantitative performance conclusions were limited to subsystems with fully reproducible benchmarking measurements. Their inclusion illustrates architectural feasibility and coordination within a multi-model environment but warrants further dedicated evaluation for comprehensive performance characterization.

Overall, the results support the practical viability of hybrid deployment strategies in data-intensive environments. Rather than promoting a universal storage solution, the findings reinforce the importance of workload-driven architectural segmentation in modern database ecosystem design. Within the context of e-commerce analytics, such segmentation enables simultaneous support for transactional integrity, flexible ingestion, and low-latency retrieval.

A. Future Work

Future research should extend the experimental scope to multi-node distributed deployments and larger-scale datasets to evaluate behavior under production-level variability. Incorporating real-world workload traces would further enhance ecological validity and enable more precise scalability modeling.

Additional investigation into automated workload routing and adaptive query optimization within hybrid environments may improve dynamic performance balancing. Expanding benchmarking coverage to fully quantify graph and column-oriented subsystems under controlled distributed configurations would also strengthen comparative analysis.

Finally, applying the hybrid framework to alternative application domains beyond e-commerce analytics may improve generalizability and clarify domain-specific performance trade-offs.

DECLARATION OF COMPETING INTERESTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

ACKNOWLEDGMENT

The authors acknowledge the technical support and collaborative input provided during the development and evaluation of the experimental framework.

DATA AVAILABILITY

The experimental data generated and analyzed during this study are available from the corresponding author upon reasonable request. Benchmark workloads were generated using standardized YCSB configurations within a Docker-based experimental environment.

AI USE AND DECLARATION OF GENERATIVE AI USE

The authors used generative Artificial Intelligence (AI) tools solely for language refinement, editing assistance, and manuscript preparation support. All scientific content, experimental design, implementation, data collection, analysis, interpretation of results, and final manuscript validation were performed and verified by the authors. The authors take full responsibility for the content of this publication.

REFERENCES

[1] M. Stonebraker, "SQL databases v. NoSQL databases," *Communications of the ACM*, vol. 53, no. 4, pp. 10–11, Apr. 2010, https://doi.org/10.1145/1721654.1721659.

[2] T. N. Khasawneh, M. H. AL-Sahlee, and A. A. Safia, "SQL, NewSQL, and NOSQL Databases: A Comparative Survey," in *2020 11th International Conference on Information and Communication Systems*, Irbid, Jordan, 2020, pp. 013–021, https://doi.org/10.1109/ICICS49469.2020.239513.

[3] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, Apr. 2010, https://doi.org/10.1145/1773912.1773922.

[4] D. Abadi, "Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story," *Computer*, vol. 45, no. 2, pp. 37–42, Feb. 2012, https://doi.org/10.1109/MC.2012.33.

[5] M. Kleppmann, *Designing Data-intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. Sebastopol, CA, USA: O'Reilly Media, 2017.

[6] J. Lu, I. Holubová, and B. Cautis, "Multi-model Databases and Tightly Integrated Polystores: Current Practices, Comparisons, and Open Challenges," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, Torino, Italy, 2018, pp. 2301–2302, https://doi.org/10.1145/3269206.3274269.

[7] S. Bjeladinovic, Z. Marjanovic, and S. Babarogic, "A proposal of architecture for integration and uniform use of hybrid SQL/NoSQL database components," *Journal of Systems and Software*, vol. 168, Oct. 2020, Art. no. 110633, https://doi.org/10.1016/j.jss.2020.110633.

[8] J. Han, H. E, G. Le, and J. Du, "Survey on NoSQL database," in *2011 6th International Conference on Pervasive Computing and Applications*, Port Elizabeth, South Africa, 2011, pp. 363–366, https://doi.org/10.1109/ICPCA.2011.6106531.

[9] A. E. Topcu *et al.*, "Evaluating the Performance of NoSQL Databases for Big Data in Cloud Computing Environments," *HighTech and Innovation Journal*, vol. 6, no. 3, pp. 808–830, Sept. 2025, https://doi.org/10.28991/HIJ-2025-06-03-05.

[10] G. C. Deka, "Chapter Nine - NoSQL Polyglot Persistence," in *A Deep Dive into NoSQL Databases: The Use Cases and Applications*, P. Raj and G. C. Deka, Eds. Amsterdam, Netherlands: Elsevier, 2018, pp. 357–390, https://doi.org/10.1016/bs.adcom.2017.08.003.

[11] I. Qaddara, Y. Alraba'nah, and M. O. Hiari, "Evaluation of SQL and NoSQL Databases on Parallel Processing," *Engineering, Technology & Applied Science Research*, vol. 15, no. 4, pp. 24298–24304, Aug. 2025, https://doi.org/10.48084/etasr.10620.

[12] S. Idreos and T. Kraska, "From Auto-tuning One Size Fits All to Self-designed and Learned Data-intensive Systems," in *Proceedings of the 2019 International Conference on Management of Data*, Amsterdam, Netherlands, 2019, pp. 2054–2059, https://doi.org/10.1145/3299869.3314034.

[13] E. H. Tama and D. Mark, "The Future of Database Management in the Era of Big Data and Cloud Computing," *International Journal of Information Technology & Computer Engineering*, vol. 4, no. 5, pp. 48–60, Sept. 2024, https://doi.org/10.55529/ijitc.45.48.60.

[14] D. Van Landuyt, J. Benaouda, V. Reniers, A. Rafique, and W. Joosen, "A Comparative Performance Evaluation of Multi-Model NoSQL Databases and Polyglot Persistence," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, Tallinn, Estonia, 2023, pp. 286–293, https://doi.org/10.1145/3555776.3577645.

[15] F. Zhao, D. Agrawal, and A. E. Abbadi, "Hybrid Querying Over Relational Databases and Large Language Models." arXiv, Nov. 15, 2024, https://doi.org/10.48550/arXiv.2408.00884.

[16] M. M. Chaudhari and S. B. Shirude, "Next-Generation Library Information Systems: Evaluating Native Multi-Model Database Technology," *International Journal on Advanced Computer Theory and Engineering*, vol. 15, no. 1S, pp. 170–181, Jan. 2026, https://doi.org/10.65521/ijacte.v15i1S.1315.

[17] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proceedings of the 1st ACM symposium on Cloud computing*, Indianapolis, IN, USA, 2010, pp. 143–154, https://doi.org/10.1145/1807128.1807152.