www.ijecs.in
International Journal Of Engineering And Computer Science
Volume 14 Issue 09 September 2025, Page No. 27724-27730
ISSN: 2319-7242 DOI: 10.18535/ijecs/v14i09.5264

Next-Generation Hybrid Databases: Bridging SQL Consistency, NoSQL Scalability, and AI-Driven Optimizations

Brahim Ben lhoucine[1], Khalid Tatane[2]

[1]Image and Pattern Recognition – Intelligent and Communicating Systems Laboratory (IRF-SIC)
Faculty of Science, Agadir
IBN ZOHR University, Agadir, Morocco
[2]Information Technology, Data and Applied Mathematics Science Team (ESTDIMA)
National School of Applied Science
IBN ZOHR University, Agadir, Morocco

Abstract:
The rise of big data, IoT, and real-time applications has exposed the scalability limits of traditional relational databases (SQL), while NoSQL systems often sacrifice consistency for flexibility and throughput. This paper reviews the evolution of data management systems, with a focus on **hybrid SQL–NoSQL frameworks**. We analyze four key dimensions: **architectural approaches** (polystores, multi-model systems, HTAP, and cloud-native hybrids), **consistency and integration mechanisms** (ACID vs BASE, tunable consistency, CDC, and event-driven middleware), **benchmarks and case studies** (YCSB and domain-specific workloads), and the emerging role of **AI-enhanced data management** (learned indexes, adaptive optimizers). A comparative analysis highlights how hybrid solutions aim to combine SQL's reliability with NoSQL's scalability, yet challenges persist regarding **semantic mismatches, latency, consistency guarantees, and security risks**. The review identifies open research directions and calls for standardized benchmarks, adaptive consistency strategies, and AI-driven query optimization to advance the next generation of hybrid database systems.

Keywords : SQL, NoSQL, hybrid databases, polystores, consistency, benchmarks, AI in databases

Introduction

The landscape of data management has evolved dramatically over the past two decades. Relational databases (SQL) long dominated enterprise applications by providing robust **ACID guarantees** (atomicity, consistency, isolation, durability) and a well-defined query language. However, the rise of **big data, social media, IoT, and cloud computing** revealed their limitations in **horizontal scalability** and **schema flexibility**. This drove the emergence of **NoSQL systems**, such as MongoDB, Cassandra, Redis, and Neo4j, which support semi-structured and unstructured data with high throughput and distributed deployment models.

Despite their success, NoSQL systems trade off **transactional consistency and standardization** for scalability. In contrast, SQL systems remain preferred for **mission-critical applications** requiring strict consistency and structured schemas. This dichotomy led to the exploration of **hybrid solutions** that aim to combine the **strengths of both paradigms**—the reliability of SQL with the scalability and flexibility of NoSQL.

Recent innovations have expanded the hybrid ecosystem through:

1. **Polystores and polyglot persistence**: loosely coupled systems integrating multiple specialized databases.

2. **Multi-model databases (MMDBs)**: supporting different data models natively.

3. **Hybrid Transactional/Analytical Processing (HTAP)**: bridging OLTP and OLAP in real time.

4. **Cloud-native hybrids**: managed platforms such as Amazon Aurora and Azure Cosmos DB with tunable consistency and flexible integration.

At the same time, **integration mechanisms** such as Change Data Capture (CDC), event-driven pipelines (Kafka, Debezium), and consistency models (from ACID to eventual/tunable consistency) have become critical enablers of hybrid adoption. Moreover, the introduction of **AI-driven optimizations**—such as learned indexes and adaptive query optimizers—suggests a new wave of database intelligence that may enhance hybrid systems further.

This article provides a **structured literature review** of these developments. It consolidates contributions across architectures, integration methods, benchmarks, and AI-enabled techniques. The goal is to offer both practitioners and researchers a roadmap of **where hybrid systems stand today, what challenges remain, and what opportunities lie ahead**.

I. Background: SQL vs NoSQL

SQL databases. Relational databases have been the backbone of enterprise systems for decades, offering a rigorous data model, the SQL language, and **ACID** guarantees suited to mission-critical workloads [1]. While they excel at complex joins and integrity constraints, classic deployments can struggle with horizontal scale and schema agility for rapidly evolving, high-volume applications [2].

NoSQL databases. Emerging from web-scale demands, NoSQL systems favor flexible schemas and distributed designs aligned with the **BASE** philosophy (Basically Available, Soft state, Eventually consistent) [3]–[6]. Major families include **document stores** (e.g., MongoDB, CouchDB), **key-value stores** (e.g., Redis, Dynamo-style systems), **column stores** (e.g., Cassandra, HBase), and **graph databases** (e.g., Neo4j) [4], [6]–[8]. These systems often deliver high throughput and elasticity across commodity clusters, but many trade strict transactional consistency for availability and partition tolerance [3]–[6].

Why hybrid? Modern applications frequently combine **transactional** operations (orders, payments, authentication) with **semi-structured/analytic** workloads (catalog/reviews, feeds, recommendations). As a result, organizations increasingly adopt **hybrid SQL–NoSQL** approaches to leverage SQL's strong consistency where it matters and NoSQL's scalability where it pays off [2], [4], [5].

II. Evolution of Data Management Systems

The limitations of "SQL-only" and "NoSQL-only" approaches have stimulated a wave of research into **hybrid strategies**. These range from **loosely coupled polystores** to **multi-model systems**, **federated query engines**, and **cloud-native hybrids**. Each class reflects different trade-offs in performance, integration, and usability.

A. Polystore Systems and Polyglot Persistence

Polystore systems unify queries across heterogeneous stores. Examples include **BigDAWG** and **Polypheny**, which provide middleware to abstract multiple backends [9], [10]. Similarly, the concept of **polyglot persistence** advocates that applications employ multiple specialized databases—SQL for structured data, NoSQL for scalability, and graph/document stores for analytics—coordinated through an integration layer [6], [11]. While this approach maximizes flexibility, it often suffers from **semantic mismatches**, **latency overhead**, and **cross-store consistency issues**.

B. Multi-model Databases (MMDBs)

A related line of work is **multi-model databases**, which aim to natively support multiple paradigms (e.g., document, graph, key-value) within a single engine. **ArangoDB** and **OrientDB** exemplify this approach, and the **UniBench** benchmark has emerged as a standard tool to evaluate MMDBs [12]. Studies show that MMDBs simplify integration but can underperform compared to specialized systems [13].

C. Federated Query Engines and Unified Interfaces

Federated query engines (e.g., **Presto/Trino**, **Apache Calcite**) expose a unified **ANSI SQL interface** over heterogeneous data sources [14]. These systems ease adoption by preserving SQL familiarity while extending queries to non-relational stores. However, they often lack **strong transactional guarantees** and incur overhead when joining across stores [15]. This underscores the need for **adaptive query engines** that balance expressiveness with optimization.

D. Consistency Models and Cloud-native Hybrids

Consistency remains a **core challenge**. Relational systems enforce **strong ACID compliance**, whereas many NoSQL platforms rely on **eventual consistency** for scalability [16]. Cloud-native hybrids, such as **Google Spanner** and **Azure Cosmos DB**, introduce new models: Spanner achieves **global consistency** through its **TrueTime API** [17], while Cosmos DB offers **tunable consistency levels** (strong, bounded staleness, session, consistent prefix, eventual) [18]. These innovations highlight the spectrum between **strict correctness** and **high availability**, but practical deployments must still balance latency and complexity.

III. Integration Mechanisms

A. Change Data Capture (CDC)

One of the most widely used techniques for hybrid integration is **Change Data Capture (CDC)**, which tracks modifications in relational databases and propagates them to other systems in near real time. Tools such as **Debezium** and **Oracle GoldenGate** support reliable CDC pipelines, ensuring consistency without requiring distributed transactions [19].

B. The Outbox Pattern

The **outbox pattern** has emerged as a robust approach to event-driven integration. By storing events in a dedicated relational table (the “outbox”), systems can ensure that both data changes and events are committed atomically. A separate process (or CDC tool) then publishes these events to a message broker such as **Kafka**, enabling downstream NoSQL or cache systems to remain synchronized [20].

C. Middleware and Event-Driven Synchronization

Modern hybrid deployments rely heavily on **message brokers** (Kafka, RabbitMQ) and **ETL pipelines** (Apache NiFi, Spark Streaming) to propagate updates across stores. This middleware enables **eventual consistency** across heterogeneous systems while maintaining high throughput [21].

D. Limitations of Current Integration Approaches

While CDC and middleware improve interoperability, they face challenges:

- **Latency overheads** in high-throughput settings.
- **Failure handling** (e.g., dropped events, message duplication).
- **Complexity of configuration** in distributed, multi-cloud deployments.

These limitations highlight the need for **adaptive consistency managers** and **AI-assisted monitoring tools** to optimize synchronization in hybrid systems [15], [21].

IV. Benchmarks and Case Studies

A. Standard Benchmarks

Evaluating hybrid systems requires **standardized benchmarking tools** to ensure fair comparisons. The **Yahoo! Cloud Serving Benchmark (YCSB)** is the most widely used for NoSQL and hybrid systems, testing workloads such as **read-heavy**, **write-heavy**, and **mixed operations** [22]. For multi-model systems, **UniBench** extends benchmarking by covering **cross-model queries** and diverse workloads [12]. These benchmarks highlight performance trade-offs in throughput, latency, and scalability.

B. SQL vs NoSQL vs Hybrid Comparisons

Experimental studies show that SQL systems maintain superior **transactional consistency**, while NoSQL systems excel in **horizontal scalability** and **high-throughput workloads** [13], [15]. Hybrid systems, by combining specialized engines, often outperform single-model deployments in **mixed workloads** where both ACID compliance and scalability are needed [23].

C. Domain-Specific Case Studies

Hybrid deployments have been validated across multiple domains:

- **E-commerce**: Combining SQL for transactional orders with MongoDB for product catalogs and Redis for session caching improves end-to-end latency [11].

- **Social networks**: Hybrid designs integrate SQL for authentication, graph databases (Neo4j) for relationship queries, and Redis for real-time feeds [24].

- **IoT and sensor networks**: SQL is used for device metadata, Cassandra for high-velocity time-series ingestion, and Redis for dashboard caching, synchronized via Kafka [25].

D. Key Insights

Across these domains, **hybrid deployments consistently outperform SQL-only and NoSQL-only systems** in terms of **balanced performance, scalability, and reduced latency**. However, experiments also reveal challenges such as **cross-store consistency overhead**, **increased system complexity**, and **the absence of standardized hybrid benchmarks** beyond YCSB and UniBench [12], [23].

V. AI in Databases

A. Learned Indexes

Recent research has shown that **machine learning can replace traditional index structures**. The **NeuroCard** system (2020) applied deep learning for **cardinality estimation**, enabling more accurate query plans [26]. Similarly, **learned index structures** treat indexes as models trained on data distributions, reducing lookup costs [27]. Surveys confirm growing interest in applying learned indexes to multi-dimensional and hybrid workloads [28].

B. Learned Query Optimizers

Traditional optimizers rely on cost models that may not generalize well to hybrid or distributed environments. **Bao** (2021) introduced a **reinforcement learning–based query optimizer**, showing improvements over hand-tuned heuristics in adaptive environments [29]. These approaches demonstrate the potential of AI to handle **complex cross-store queries** in hybrid architectures.

C. Natural Language Interfaces (NLIs)

AI also expands **query accessibility**. Recent surveys on **Text-to-SQL systems** highlight how **large language models (LLMs)** and deep learning are enabling users to issue queries in natural language, which are then translated into SQL or NoSQL queries [30]. This trend could be extended to hybrid databases, where NLIs can automatically route workloads to the most suitable backend.

D. Implications for Hybrid Systems

For hybrid SQL–NoSQL databases, AI can contribute to:

- **Adaptive routing** of queries based on workload patterns.
- **Dynamic consistency tuning** (e.g., adjusting Cosmos DB levels automatically).
- **Predictive caching** using learned access patterns.

These directions suggest that AI will not only optimize individual systems but also serve as a **coordination layer across hybrid frameworks**.

VI. Critics and Research Gaps

While hybrid SQL–NoSQL systems present a promising direction, several **open challenges** remain. Existing literature highlights gaps in architecture, integration, benchmarks, and security. Addressing these issues is critical for hybrid systems to reach widespread, production-ready adoption.

A. Architectural Complexity

Hybrid systems often combine multiple specialized engines under middleware or federation layers. While this enables flexibility, it introduces **operational complexity**: administrators must manage different data models, replication policies, and query optimizers simultaneously. Research is needed into **simplified orchestration frameworks** that reduce the overhead of managing heterogeneous backends [9], [10], [23].

B. Consistency Trade-offs

Although tunable and hybrid consistency models (e.g., Cosmos DB) provide flexibility, developers struggle to select the “right” trade-off for each workload. Studies show that changes in consistency levels can significantly impact latency and throughput [16], [18]. Future work should focus on **adaptive consistency managers** that dynamically adjust guarantees based on workload characteristics and SLA requirements.

C. Integration and Latency Overheads

CDC, outbox patterns, and event-driven middleware (Kafka, Debezium) improve interoperability but often introduce **latency overheads** and **failure handling complexities**. Hybrid systems need **self-healing integration mechanisms** that automatically detect dropped events, avoid duplication, and optimize cross-store synchronization [19], [20], [21].

D. Benchmarking Limitations

Most experimental studies use **YCSB** for NoSQL or **UniBench** for multi-model databases. However, no widely accepted benchmark exists that **captures hybrid workloads across SQL and NoSQL simultaneously**. Without standardized benchmarks, it is difficult to compare hybrid approaches rigorously. Future research should propose **hybrid workload benchmarks** spanning transactions, analytics, graph queries, and caching [12], [22].

E. Security and Privacy

SQL systems benefit from decades of research on **role-based access control** and **transactional security**. Many NoSQL systems, however, still lack equally robust mechanisms [15], [23]. Hybrid systems must address **cross-store vulnerabilities**, ensuring that sensitive data (e.g., financial records in SQL, user activity in NoSQL) are protected with **uniform security policies** [18].

F. AI-Enhanced Coordination

Although AI-driven optimizers (e.g., NeuroCard, Bao) show promise, their integration into hybrid systems is still in its infancy. Research is needed into **AI-assisted workload routing, consistency tuning, and predictive caching**. These methods could reduce latency, improve query planning, and balance workloads dynamically across SQL and NoSQL engines [26]–[30].

Tale 1: Research Gaps and Future Directions in Hybrid SQL–NoSQL Systems

Research Gap	**Description**	**Future Direction**
Architectural Complexity	Hybrid deployments require managing multiple engines, schemas, and optimizers.	Develop simplified orchestration frameworks to reduce operational overhead.
Consistency Trade-offs	Hard to choose between strong ACID vs. eventual consistency for each workload.	Build adaptive consistency managers that auto-tune based on workload and SLAs.
Integration Latency	CDC/outbox/middleware add latency and risk of event duplication/loss.	Design self-healing integration mechanisms with automated failure detection.
Benchmarking Limitations	Lack of standardized benchmarks for SQL–NoSQL hybrid workloads.	Propose hybrid workload benchmarks covering transactions, analytics, graphs, caching.
Security and Privacy	SQL has robust controls; NoSQL often lacks equivalent policies.	Enforce cross-store, unified security and access control frameworks.

AI-Enhanced Coordination	AI tools exist (learned indexes, optimizers) but not integrated into hybrids.	Integrate AI for workload routing, consistency tuning, and predictive caching.

VII. Conclusion

The evolution of data management has moved from **relational databases** with strong ACID guarantees to **NoSQL systems** optimized for scalability and flexibility, and now toward **hybrid SQL–NoSQL solutions** that combine the strengths of both paradigms. This review has synthesized contributions across **architectures** (polystores, multi-model systems, HTAP, cloud-native hybrids), **integration mechanisms** (CDC, outbox, middleware), **consistency models** (ACID, BASE, tunable), **benchmarks and case studies** (YCSB, UniBench, domain-specific applications), and **AI-driven innovations** (learned indexes, learned optimizers, natural language interfaces).

Findings confirm that hybrid systems offer **balanced performance**—achieving scalability and flexibility without fully sacrificing transactional integrity. Yet, critical challenges remain: **architectural complexity, integration overhead, limited benchmarking standards, and fragmented security policies**. These gaps represent opportunities for future work.

Looking ahead, three promising directions stand out:

1. **Adaptive consistency managers** that tune guarantees dynamically based on workload and SLA requirements.

2. **Standardized hybrid benchmarks** covering diverse workloads such as transactions, analytics, graph queries, and caching.

3. **AI-enhanced coordination** layers capable of workload routing, predictive caching, and query optimization across heterogeneous backends.

By consolidating fragmented research and highlighting these open challenges, this article provides a roadmap for both **academia and industry** to advance the **next generation of hybrid database systems**.

References:

1. M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*, 4th ed. Springer, 2019.
2. M. Stonebraker and U. Çetintemel, "One Size Fits All: An Idea Whose Time Has Come and Gone," *IEEE Data Eng. Bull.*, vol. 27, no. 1, pp. 11–17, 2005.
3. D. Pritchett, "BASE: An ACID Alternative," *Queue*, vol. 6, no. 3, pp. 48–55, 2008.
4. J. Han, E. Haihong, G. Le, and J. Du, "Survey on NoSQL Database," in *Proc. 2011 Int'l Conf. Pervasive Computing and Applications*, pp. 363–366.
5. R. Hecht and S. Jablonski, "NoSQL Evaluation: A Use Case Oriented Survey," in *Proc. Int'l Conf. Cloud and Service Computing*, 2011, pp. 336–341.
6. M. Fowler and P. J. Sadalage, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley, 2012.
7. G. DeCandia *et al.*, "Dynamo: Amazon's Highly Available Key-value Store," in *Proc. SOSP*, 2007, pp. 205–220.
8. I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*, 2nd ed. O'Reilly, 2015.
9. J. Duggan *et al.*, "The BigDAWG Polystore System," *SIGMOD Record*, vol. 44, no. 2, pp. 11–16, 2015.
10. P. Bressel, A. Grulich, and K. Böhm, "Polypheny: Fast Prototyping for Polystores," in *Proc. EDBT*, 2019.
11. I. Bjeladinovic *et al.*, "Polyglot Persistence in Practice: A Case Study in E-Commerce and Healthcare," *J. Syst. Softw.*, vol. 183, p. 111129, 2022.
12. A. Gubichev *et al.*, "UniBench: Benchmarking Multi-Model Database Systems," in *Proc. BTW*, 2021.

13. R. Gupta *et al.*, “Hybrid Database Systems: A Systematic Literature Review,” *J. Big Data*, vol. 7, no. 112, pp. 1–19, 2020.
14. M. Armbrust *et al.*, “SQL Interface for Data Lakes and Heterogeneous Data Sources: Apache Calcite and Trino,” *Proc. VLDB Endow.*, vol. 14, no. 12, pp. 3135–3147, 2021.
15. A. Kumar and B. C. Ooi, “SQL vs NoSQL: Revisiting the Trade-offs,” *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–23, 2019.
16. K. Kaur and R. Rani, “Modeling and Querying Data in NoSQL Databases,” in *Proc. IEEE Big Data*, 2013, pp. 1–7.
17. J. C. Corbett *et al.*, “Spanner: Google’s Globally-Distributed Database,” in *Proc. OSDI*, 2012, pp. 261–264.
18. Microsoft Azure Documentation, “Consistency Levels in Azure Cosmos DB,” 2022. Available: https://learn.microsoft.com
19. Debezium Documentation, “Change Data Capture with Debezium,” 2023. Available: https://debezium.io/documentation/
    Microsoft Documentation, “Implementing the Outbox Pattern,” 2023. Available: https://learn.microsoft.com/

20. F. Chen, J. Rao, and K. Lakshmanan, “High Performance and Fault Tolerant Middleware for Cloud Databases,” *Future Generation Computer Systems*, vol. 117, pp. 64–76, 2021.
21. B. F. Cooper *et al.*, “Benchmarking Cloud Serving Systems with YCSB,” in *Proc. 1st ACM Symposium on Cloud Computing (SoCC)*, 2010, pp. 143–154.
22. S. Ghafari, A. Al-Dubai, and E. Ekonomou, “SQL–NoSQL Hybrid Solutions: A Performance and Scalability Evaluation,” *Future Generation Computer Systems*, vol. 118, pp. 145–160, 2021.
23. Y. Shen and L. Shao, “Hybrid Databases: Integrating SQL and NoSQL for Modern Applications,” *Int. J. Computer Applications*, vol. 975, pp. 17–25, 2020.
24. C. Li and S. Manoharan, “A Performance Comparison of SQL and NoSQL Databases,” in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 2013, pp. 15–19.
25. J. Yang *et al.*, “NeuroCard: One Cardinality Estimator for All Tables,” in *Proc. SIGMOD*, 2020, pp. 1099–1114.
26. T. Kraska *et al.*, “The Case for Learned Index Structures,” in *Proc. SIGMOD*, 2018, pp. 489–504.
27. A.-A. Mamun, H. Wu, Q. He, J. Wang, and W. G. Aref, “A Survey of Learned Indexes for the Multi-Dimensional Space,” *arXiv preprint arXiv:2403.06456*, 2024.
28. R. Marcus and O. Papaemmanouil, “Bao: Making Learned Query Optimization Practical,” in *Proc. SIGMOD*, 2021, pp. 1278–1293.
29. Z. Hong *et al.*, “Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL,” *Proc. VLDB Endow.*, vol. 17, no. 5, pp. 960–973, 2024.