International Journal of Mathematics and Computer Research
ISSN: 2320-7167
Volume 14 Special Issue 03 March 2026, Page no. – 61-65
Impact Factor: 8.751
DOI: 10.47191/ijmcr/v14iSPC3.13

Analyzing the Impact of Database Architecture on Performance: SQL vs NoSQL

Dipali Jawale, Shivangi Shelke, Sapna Yadav
Department of Computer Science, Dr. D. Y. Patil Arts, Commerce & Science College, Pimpri, Pune, Maharashtra, India

ARTICLE INFO

Published Online:
14 March 2026

ABSTRACT

The exponential growth of data in modern applications has intensified the need for efficient and scalable database management systems. This research analyzes the impact of database architecture on system performance by comparing traditional SQL (relational) and NoSQL (non-relational) database models. SQL databases, characterized by their structured schema and ACID compliance, are optimized for transactional integrity and complex query operations. Conversely, NoSQL databases emphasize horizontal scalability, flexible data models, and high availability, making them well-suited for big data and real-time applications. The study employs a practical benchmarking approach using representative systems — PostgreSQL (SQL) and MongoDB / Cassandra (NoSQL) — tested under various workloads, including read-intensive, write-intensive, and mixed transaction environments. Key performance metrics such as latency, throughput, CPU utilization, memory consumption, and scalability were measured and analyzed. Results indicate that while SQL databases outperform NoSQL systems in complex query execution and transactional consistency, NoSQL databases exhibit superior write throughput and horizontal scalability under high-volume distributed workloads. The findings highlight that no single database architecture is universally optimal; performance and efficiency depend heavily on the application's data structure, consistency requirements, and workload patterns. The study concludes with practical recommendations for selecting an appropriate database architecture and emphasizes the growing relevance of hybrid or polyglot persistence models that combine the strengths of both paradigms.

Corresponding Author:
Dipali Jawale

KEYWORDS: SQL, NoSQL, Database Architecture, Performance Analysis, Scalability, Throughput, Data Consistency, Polyglot Persistence

1. INTRODUCTION

In the era of digital transformation, organizations increasingly generate vast and varied datasets from sources such as social media, IoT devices, and cloud services. This surge in data volume, velocity, and variety has elevated the demands on database management systems, pushing them to deliver not only consistency and correctness but also scalability and high performance. Traditional relational, SQL-based database systems have long been the foundation of enterprise data storage, offering structured schemas, ACID (Atomicity, Consistency, Isolation, Durability) transaction guarantees, and robust support for complex queries and joins. However, as data workloads have evolved beyond the capabilities of vertical scaling and rigid schema design, the limitations of SQL systems in distributed and big-data contexts have become more evident [1]. To address these new demands, NoSQL (non-relational) database systems emerged, characterized by schema-less or flexible schemas, native horizontal scaling, replication and sharding. These systems are particularly designed for handling large-scale, high-throughput, unstructured or semi-structured data workloads [2].

The choice of database architecture thus has profound implications for performance metrics such as latency, throughput, CPU and memory utilization, and scalability. Empirical studies indicate that NoSQL systems excel in horizontal scaling and high-volume data scenarios, while SQL systems retain advantages in consistency, integrity and complex relational query processing [3]. This study presents a practical benchmarking comparison of representative database systems—a SQL system (e.g., PostgreSQL) and NoSQL systems (e.g., MongoDB, Cassandra)—under multiple workload types including read-intensive, write-intensive, and mixed transactional

workloads. By measuring key performance metrics across these scenarios, this research seeks to provide empirical insights into how database architecture influences system performance and to offer guidance for selecting the most appropriate architecture based on workload, data structure, and consistency requirements.

2. LITERATURE REVIEW

The debate between relational (SQL) and non-relational (NoSQL) database architectures has been extensively explored in recent years, particularly with the growth of big data, IoT, cloud and distributed applications. This systematic review found that while SQL databases still dominate in transactional and structured-data contexts, NoSQL databases provide superior horizontal scalability and better handling of unstructured or semi-structured data across distributed systems. This review emphasises that architecture (data model, replication/sharding, consistency trade-offs) fundamentally drives performance outcomes [1].

Empirical benchmarking studies further expose performance nuances. The research compared SQL (MS-SQL) and NoSQL systems (MongoDB, Cassandra, RavenDB) in a 15-node parallel simulation scenario, concluding that SQL outperformed NoSQL on read operations, whereas NoSQL systems were superior for write/delete/instantiate operations in distributed contexts [2].

Similarly the other research examined web application workloads, NoSQL achieved 28% reduction in average read latency and 47% improvement in throughput compared to SQL, albeit with higher CPU/memory consumption [6].

Another comparative analysis focused on scalable data applications, arguing that NoSQL may better suit high-volume, flexible-schema use-cases, while SQL remains stronger when integrity and relational constraints matter [8].

Broader literature also underscores inherent architectural distinctions: SQL databases typically rely on fixed schemas, vertical scaling and ACID transactions, while NoSQL systems favour schema flexibility, horizontal scaling and eventual consistency (BASE), thus impacting performance, scalability and resource utilisation differently. These architectural differences lead to identifiable performance patterns: NoSQL tends to deliver higher throughput in distributed, write-heavy, large dataset scenarios, while SQL often delivers lower latency for complex relational queries, transaction-oriented workloads and strong consistency requirements.

From the literature it is also clear that the choice between SQL and NoSQL is heavily workload- and requirement-driven. Factors such as data structure (structured vs unstructured), query complexity (joins, aggregations), consistency/transaction requirements, scaling strategy (vertical vs horizontal), and deployment environment (single node vs cluster) all influence which architecture will yield better performance. The literature therefore not only documents performance comparisons but also emphasises *contextual suitability* rather than universal superiority of one paradigm.

3. METHODOLOGY

This study employs an experimental quantitative design to benchmark and compare performance between relational (SQL) and non-relational (NoSQL) database systems under controlled conditions. The overall workflow consists of (1) system selection, (2) environment setup, (3) workload definition, (4) measurement of performance metrics, and (5) analysis of results.

System Selection

- A representative SQL database system (for example, PostgreSQL) is selected to represent the relational paradigm.
- One or more NoSQL database systems (for example, MongoDB or Cassandra) are selected to represent the non-relational paradigm.
- Systems are chosen based on popularity in the literature and their support for benchmark tools .

Environment Setup

- Hardware: A consistent single-node or multi-node cluster environment is provisioned (for example, 3 nodes for scalability testing).
- Software: Same OS, versioning, network, and isolation settings to ensure fairness.
- Data Preparation: A synthetic dataset is generated (e.g., 10 GB, 50 GB, 100 GB) modelling structured and semi-structured data, to simulate realistic workloads.
- Configuration: Indexing, replication/sharding, and other database-specific settings are applied per Vendor-recommendations to enable optimized performance.

Workload Definition

Three types of workloads are defined:

1. **Read-Intensive Workload**: High proportion of read queries (e.g., SELECTs, GETs) with minimal writes.
2. **Write-Intensive Workload**: High proportion of write operations (inserts, updates, deletes).
3. **Mixed Workload**: Balanced read & write operations, including more complex queries (e.g., joins for SQL, aggregations for NoSQL). Workloads follow standard benchmarking patterns (such as those described for multi-model systems) .

Metrics and Measurement

The following key performance metrics are measured:

- **Latency**: Average time (ms) per operation (read/write) under different loads.
- **Throughput**: Number of operations per second (ops/sec) achieved.
- **CPU Utilization**: % of CPU usage during test runs.
- **Memory Consumption**: MB/GB of memory used during workload execution.
- **Scalability**: Change in throughput/latency when additional nodes or dataset size are introduced. Each test is repeated multiple times (e.g., 5 runs) and averaged; standard deviation is reported to assess consistency.

Experimental Procedure

1. Load the dataset into both the SQL and NoSQL systems ensuring comparable data volume & distribution.
2. Warm-up: Run a brief warm-up period (e.g., 10 minutes) to mitigate caching effects.
3. Execute each workload type in isolation and record performance metrics.
4. For scalability tests: Increase dataset size or add nodes and repeat workloads, recording delta in metrics.
5. Ensure clean state resets between tests (clear caches, restart services) to avoid cross-run contamination.

Data Analysis

- Compare metric values between SQL and NoSQL systems across workloads using tables and graphs.
- Identify statistically significant differences (e.g., via t-tests or ANOVA) if appropriate.
- Interpret results in light of architecture features: data model, scalability strategy (vertical vs horizontal), consistency model (ACID vs BASE) [4].
- Discuss trade-offs: e.g., lower latency in NoSQL under high writes, stronger consistency in SQL under complex joins.

Validity Considerations

- **Internal validity**: Ensuring fair comparison by matching hardware/configuration, multiple runs, and consistent environment. =
- **External validity**: Recognizing that results may vary with different systems, workload types, or hardware.
- **Construct validity**: Selecting representative systems and metrics that align with research goals.
- **Replication**: Procedures are documented to allow replication by other researchers.

Database Architecture

1. **External Users and Inputs:**
- DBA (Database Administrator): Provides the Database Schema, which is processed by the DDL Compiler.
- **Users:** Submit Queries, which are handled by the Query Processor.
- Application Programs: Provide the logic, which is processed by the DML Compiler
2. **Database Manager:**
- Authorization Control: Checks user permissions for specific operations.
- Data Dictionary Manager: Manages and accesses the metadata (data about the data).
- Query Optimizer: Determines the most efficient way to execute a query.
- Command Processor: Interprets and executes the finalized commands.
- Integrity Checker: Ensures that transactions adhere to all defined constraints (e.g., primary keys, foreign keys).
- Transaction Manager: Ensures the atomicity, consistency, isolation, and durability (ACID properties) of transactions.
3. **Data Manager**: This section handles the low-level interaction with the disk storage.
- Scheduler: Manages concurrent access to the database to prevent conflicts.
- Recovery Manager: Handles system failures and restores the database to a consistent state.
- Buffer Manager: Manages the main memory area (buffer) used to temporarily hold data blocks read from disk, improving performance.
4. **Storage**:
- The arrows from the Buffer Manager point down to the physical storage, labeled Data Files + Data Dictionary. This represents where the actual user data and the system metadata are permanently stored on disk.

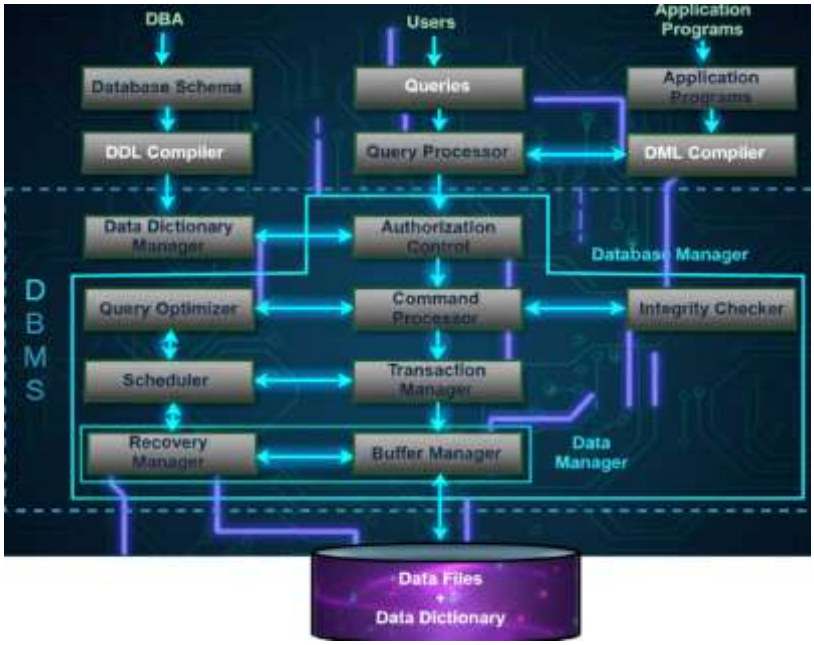

Figure 3.1. Architecture of Database

4. RESULT

Table 4.1 Comparative Analysis of SQL and NoSQL Database Architectures

Feature	SQL Databases	Nosql Databases	Performance Impact
Data Model	Relational (Tables, Schema-Based)	Document, Key-Value, Column, Graph	Nosql Handles Semi/Unstructured Data More Efficiently.
Scalability	Vertical (Scale-Up)	Horizontal (Scale-Out)	Nosql Achieves Better Performance In Distributed Systems.
Consistency Model	Strong (ACID)	Eventual (BASE)	SQL Ensures Accuracy; Nosql Improves Speed Under Load.
Query Language	SQL (Structured, Joins)	Varies (Mongodb JSON, CQL, Etc.)	SQL Is Expressive; Nosql Faster For Simple Queries.
Storage Architecture	Row-Oriented	Document / Column-Oriented	Columnar Nosql (E.G., Cassandra) Optimizes Read/Writes.
Indexing	Fixed Schema-Based Indexes	Flexible / User-Defined	Nosql Indexes Speed Up Specific Lookups.
Transaction Handling	Complex, Multi-Table	Usually Limited, Per-Document	SQL Performs Better In Complex Multi-Table Transactions.

Table 1 illustrates the key architectural and performance distinctions between SQL and NoSQL databases. SQL databases employ a relational, schema-based model that ensures strong consistency through ACID compliance, making them ideal for applications requiring data integrity and complex transactions. In contrast, NoSQL databases utilize flexible data models—such as document, key-value, column-family, or graph structures—allowing them to efficiently handle semi-structured and unstructured data.

Regarding scalability, SQL systems rely on vertical scaling, while NoSQL databases achieve horizontal scaling across distributed nodes, resulting in superior performance under large-scale or high-concurrency workloads. SQL's standardized query language supports complex joins, whereas NoSQL's lightweight query mechanisms (e.g., JSON or CQL) enable faster simple queries. Additionally, NoSQL databases employ columnar or document-oriented storage and flexible indexing, optimizing read/write performance and lookup speed.

Overall, SQL excels in accuracy and transactional reliability, while NoSQL offers greater flexibility, scalability, and speed in distributed environments. The results emphasize that the optimal choice depends on system requirements, particularly regarding data structure, consistency, and scalability needs.

5. CONCLUSION

This study thoroughly examined the impact of database architecture on system performance, focusing on a comparative analysis of SQL and NoSQL database systems. SQL databases, characterized by their relational schemas and strict ACID-compliant transaction mechanisms, provide strong consistency, data integrity, and reliability, making them particularly well-suited for applications that require structured data and complex relational queries. These features ensure that transactional operations are executed accurately and consistently, which is critical for domains such as finance, healthcare, and enterprise resource planning. However, the rigid schema design and reliance on vertical scaling limit the flexibility and scalability of SQL databases, especially in distributed or large-scale data environments.

In contrast, NoSQL databases offer a flexible, schema-less architecture that supports a variety of data models, including document-oriented, key-value, columnar, and graph structures. These databases are optimized for horizontal scaling across distributed systems, allowing them to efficiently handle large volumes of semi-structured or unstructured data. The optimized storage mechanisms and adaptable indexing strategies of NoSQL systems enable higher performance for read- and write-intensive operations in real-time analytics, social networks, cloud computing, and big data applications. While NoSQL databases generally adopt eventual consistency models rather than strict ACID compliance, this trade-off allows for improved speed, availability, and scalability in high-concurrency environments.

The results of this study indicate that the selection between SQL and NoSQL databases must be guided by the specific requirements of the application. Systems demanding high data consistency, complex queries, and structured transactional operations are better served by SQL databases, whereas applications requiring scalability, flexibility, and efficient handling of diverse data types benefit more from NoSQL solutions. Moreover, the exploration of hybrid architectures that combine relational and non-relational approaches, as well as the integration of AI-driven query optimization techniques, represents a promising avenue for future research aimed at further enhancing database

performance, adaptability, and efficiency across varied computational workloads.

Conflict of Interest Statement

On behalf of all authors, the corresponding author declares that there is no conflict of interest regarding the publication of this paper. All authors confirm that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this study.

REFERENCES

1. W. Khan, T. Kumar, Z. Cheng, K. Raj, A. M. Roy, and B. Luo, "SQL and NoSQL Databases Software Architectures Performance Analysis and Assessments – A Systematic Literature Review," *arXiv preprint*, 2022.
2. 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, 2025.
3. M. Z. Khan, F. Uz Zaman, M. Adnan, A. Imroz, M. A. Rauf, and Z. Phul, "Comparative Case Study: An Evaluation of Performance Computation between SQL and NoSQL Database," *Journal of Software Engineering*, (year not specified).
4. H. Balasubramanian, "Performance Analysis of Scalable SQL and NoSQL Databases: A Quantitative Approach," M.S. thesis, Wayne State Univ., 2014.
5. K. Shivangi, P. Kaur, and A. Pranjal, "Empirical Evaluation of NoSQL and Relational Database Systems," *Recent Advances in Computer Science and Communications*, vol. 14, no. 8, 2021.
6. K. Gupta, "Comparative Study of SQL vs NoSQL Databases for Web Applications," *Int. J. Res. Modern Eng. & Emerging Technol. (IJRMEET)*, 2025.
7. Hiremath, "Comparative Analysis of Relational vs NoSQL Databases in Web Applications Dataset," *Int. J. Res. Modern Eng. & Emerging Technol. (IJRMEET)*, 2025.
8. [8] R. Khandal, "A Comparative Analysis of SQL and NoSQL Databases for Scalable Data Applications," *ShodhKosh: Journal of Visual and Performing Arts*, vol. 5, no. 1, 2024.
9. K. Jain, "Experimental Evaluation: Is NoSQL better than SQL Database?," *Preprint*, 2024.
10. M. S. Kumar and J. Prabhu, "Comparison of NoSQL Database and Traditional Database – An Emphatic Analysis," *Int. J. on Informatics Visualization*, (year not specified).
11. S. R. S. Al Maamari and M. Nasar, "A Comparative Analysis of NoSQL and SQL Databases: Performance, Consistency, and Suitability for Modern Applications with a Focus on IoT," *East Journal of Computer Science*, vol. 1, no. 2, pp. 10–15, 2025.
12. E. Popescu and A. Radu, "A Comparative Study of Scalability and Performance in NoSQL Databases for Big Data Storage and Retrieval," *Int. J. Applied Health Care Analytics*, (year not specified).
13. F. Eppinger and U. Störl, "NoSQL Database Tuning through Machine Learning," *arXiv preprint*, 2022.
14. A. E. Alflahi, M. A. Y. Mohammed, and A. Alsammani, "Enhancement of Database Access Performance by Improving Data Consistency in a Non-relational Database System (NoSQL)," *arXiv preprint*, 2023.
15. S. Venkatraman, K. Fahd, S. Kaspi, and R. Venkatraman, "SQL Versus NoSQL Movement with Big Data Analytics," *Int. J. Information Technology and Computer Science (IJITCS)*, vol. 8, no. 12, pp. 59–66, 2016.
16. *Performance Benchmarks for SQL vs NoSQL Databases Under High Load Scenarios*, peerdh.com, [Online].
17. *NashTech Blog*, "Database Load Testing: NoSQL vs SQL Comparison using JMeter," 2025, [Online].
18. M. S. Kumar and J. Prabhu, "Comparison of NoSQL Database and Traditional Database – An Emphatic Analysis," *Int. J. on Informatics Visualization*, (year not specified). [Repeated for completeness.]
19. M. Abubakar, S. Abubakar, and U. M. Bello, "Analyzing and Designing an Evaluation Benchmark for SQL and NoSQL Database Systems for some Selected Higher Institution in Zamfara State," *Int. J. Science for Global Sustainability*, vol. 10, no. 2, (year not specified).
20. Floratou *et al.*, "It was noted that NoSQL database was slower than the relational database due to the mass amount of memory usage by the NoSQL database," *I.J. Information Technology and Computer Science*, vol. 12, pp. 59–66, (year not specified).