# A COMPARATIVE ANALYSIS OF DATABASE ARCHITECTURES: SQL AND NOSQL DATABASES

**PATEL KIRANBEN BHARATBHAI**
**Research Scholar,**
**Department of Computer Science,**
**VNSGU, Surat, Gujarat, India.**

**DR. NIMISHA A. MODI**
**Assistant Professor,**
**Department of Computer Science**
**VNSGU, Surat, Gujarat, India.**

## Abstract:

The continuous growth of data on various domains has created major challenges to traditional data management systems. Relational Database Management Systems (RDBMS), while long-established and reliable for structured data and transactional processing, face limitations in handling large-scale, distributed, semi-structured, and unstructured data. Alternatively, NoSQL databases have gained popularity due to their flexible schemas, ability to scale horizontally, and efficiency in handling various types of data. This paper presents a comparative analysis of relational (SQL) and non-relational (NoSQL) databases, based on a systematic literature review of relevant research articles, conference papers, and technical reports. The study examines crucial dimensions including data models, query languages, scalability, consistency, performance, indexing, Big Data integration, and applicability across various application domains. The findings indicate that while RDBMSs are well-suited for scenarios demanding strong consistency and complex queries, NoSQL databases excel in real-time analytics and dynamic, distributed environments. Additionally, the paper explores the strengths and limitations of both database types and highlights the increasing interest in hybrid models that integrate features from both paradigms.

## 1. Introduction:

Data is essential in the every organization operation. With organizational growth, the variety of data has increased, resulting in generating large volumes of structured, semi-structured, and unstructured data. Traditional Relational Database Management Systems (RDBMS) such as Oracle, MySQL, MS SQL Server, PostgreSQL, and Sybase have long served as the cornerstone for storing, managing, and retrieving structured data due to their well-defined schema, ACID compliance, and support for complex querying. However, as modern applications demand real-time analytics, distributed architectures, and greater flexibility in handling heterogeneous data, the limitations of relational databases have become increasingly evident.

To overcome these limitations, NoSQL (Not Only SQL) databases have become a robust alternative. Designed with scalability, schema flexibility, and high performance, NoSQL databases utilize a distributed, schema-less architecture and support various data models, including key-value stores, column-oriented stores, document-oriented systems, and graph databases. Examples such as MongoDB, Cassandra, CouchDB, Redis, and Neo4j are widely used in cloud computing, real-time analytics, high-volume web and mobile applications, Internet of Things (IoT) environments, and content management systems.

This paper provides a comparative analysis of relational (SQL) and non-relational (NoSQL) database systems. The study evaluates and contrasts both database architectures based on key criteria such as data models, performance, scalability, consistency, indexing mechanisms, flexibility, and their suitability for various modern applications. This will provide various users with the ability to select the most suitable database technology according to their specific application and organizational requirements. The study also emphasizes the need for hybrid database systems that integrate the strengths of both relational (SQL) and non-relational (NoSQL) database technologies.

## 2. Literature Review

To further understand the relative merits of SQL and NoSQL systems, prior research has compared them across multiple dimensions. This section presents the findings from selected research papers and examines the fundamental differences, performance, and application suitability of relational (SQL) and non-relational (NoSQL) database systems.

Boicea et al. [1] compare Oracle and MongoDB databases based on various criteria, including data model, performance, scalability, system requirements, and theoretical foundations. They describe the main conceptual differences and similarities, and syntactical differences using create, drop, insert, update, and delete commands. Oracle uses the SQL language and follows ACID principles, while MongoDB relies on JavaScript-based API calls and follows BASE principles. Performance tests show that Oracle easily works with small datasets but struggles with large data volumes, while MongoDB performs better at horizontal scaling and inserting large-scale datasets, but slower with smaller datasets. For update and delete operations, MongoDB remains consistent, whereas Oracle's time increases with the data volume growth. The paper concludes that MongoDB offers speed, flexibility, and ease of use, while Oracle is preferred for complex relational data handling.

Similarly, Faraj and Bilal [2] analyze the differences between relational (Oracle) and non-relational (MongoDB) databases based on query performance and aggregate functions (Sum, Count, Avg) of retrieving a large set of data. The result of this comparison shows that for retrieving a high number of records, MongoDB performs better or faster than Oracle. However, for each aggregation function (Sum, Count, Avg), Oracle is better than MongoDB. This experiment proves that both can coexist; it is only the needs of the customer that determine which one suits better.

Gaikwad [3] compares the Oracle (SQL database) and MongoDB (document-oriented NoSQL database) based on theoretical concepts, basic terminology, different commands, operators, importing files, executing statistics, and use in both databases' main features, such

as Sharding and MapReduce. It compares the Oracle ACID properties with the MongoDB BASE model. Oracle has a fixed schema structure, while MongoDB has a schemaless structure for data storage. Oracle scales up until it has maximized the resources, while MongoDB scales horizontally across servers. Sharding is a time-consuming process for Oracle, while it is an automatic and easy process in MongoDB. Map-reduce operations are implemented in MongoDB using the mapReduce database command, while many internal and external solutions for extending Oracle Database 11g and Oracle Table functions provide a robust, scalable way to implement MapReduce. Thus, this paper suggests that if one needs a fast, flexible database, then choose a NoSQL Database, whereas if relations between the tables are required, then choose a traditional RDBMS.

Shetty [4] presents a query performance comparison between structured Oracle and unstructured MongoDB with a few identified operations like insert, update, delete, and search being tested, using 10, 100, 1000 and 10,000 records. Hypothesis testing suggests that MongoDB performs better only for insert, delete and select operations, whereas for update operations, Oracle still performs better in the given environment. Overall, non-relational databases perform better than relational databases.

Singh and Tiwari [5] present the differences between Oracle and MongoDB using syntax differences and time comparison for performance. The challenges of model transformation and data migration during the process of transferring an SQL database to MongoDB. The result shows that NoSQL databases are quicker, better, and easier to store large amounts of data than SQL databases in terms of velocity and flexibility. Using the Erwin HAWK tool, SQL databases are easily migrated to NoSQL databases.

Nayak et al. [6] describe the NoSQL data model, the types of NOSQL data stores based on their characteristics and features, query languages used in NOSQL, the advantages and disadvantages of NOSQL over RDBMS and the prospects of NOSQL. NOSQL does not have a standard query language. Most of the NoSQL databases have created their own query language, such as Cassandra uses CQL (Cassandra query language), MongoDB uses mongo query language, etc. UnQL, pronounced ‘uncle’, stands for Unstructured Query Language. It is a type of SQL that works with unstructured data. It’s a collaborative effort to introduce familiar and standardized data definitions and data manipulation languages to the NoSQL platform. Working on a hybrid data model is the prospect for RDBMS as well as NoSQL databases.

Ahmad [7], Győrödi [8], Ceresnak and Chovancova [9], Patel et al. [10], and Ticiana [11] compared MySQL and MongoDB and focused on the scalability, performance, application

requirements, characteristics, and types of NoSQL databases. They conducted experiments on different workloads. The results show that MongoDB performs better, scales horizontally, and is more suitable for Big Data and unstructured data in web applications, while RDBMSs maintain superiority in structured, relational workloads with complex queries.

Malik [12] and Elburki [13] compared MS SQL Server and MongoDB based on performance with unstructured data in Big Data applications. The result indicates MongoDB is a good choice for handling unstructured data, offering easy scalability, flexibility, low cost, and better performance, while SQL Server supports transactional integrity, JOIN operations and handling structured data.

Radoev [14], Mohamed [15], Kunda and Phiri [16] and Sahatqija et al. [17] compare traditional Relational databases (SQL) and modern NoSQL databases, evaluating their strengths and weaknesses of each other. They focus on the characteristics, data model, scalability, integrity, security, performance and explore architectural differences. They suggest that RDBMS is preferable when data consistency, data integrity by ACID compliance and complex joins are needed, while NoSQL databases prioritize BASE properties, eventual consistency, flexible, high-volume data storage and distributed architecture for unstructured data-driven applications or big data applications.

Khan [18] used practical experiments to evaluate SQL(MySQL) and NoSQL(MongoDB) performance under various workloads in handling data for scientific and enterprise applications, based on file size and evaluating the performance. It highlights the structural differences between SQL provides relational, predefined schemas, manages structured data, handles complex queries, and NoSQL provides non-relational, flexible schemas, manages unstructured data, and offers scalability. He discusses various data models, like documents, key-value pairs, and graphs. Using PHP to connect and test both databases, the experiment analyzed loading, response, and retrieval times for datasets containing 10,000, 20,000, and 30,000 records. Results indicated that SQL databases are consistently more efficient and have better performance than NoSQL in all operations. Despite NoSQL's advantages in scalability and schema flexibility, SQL proved more efficient and reliable for the tested performance metrics. Therefore, SQL (MySQL) is found to be more effective than NoSQL (MongoDB) for performance-critical data operations or this type of application.

Recent studies by Klaus [19] and Mihai [20] compare relational (SQL) and non-relational (NoSQL) databases in terms of scalability, consistency, query performance, and suitability for large-scale data operations. Unlike relational databases, NoSQL models are **non-relational, distributed, often open-source, and horizontally scalable**, making them suitable for

decentralized systems and diverse data types. They are especially useful for real-time, large-scale applications with changing requirements. However, relational databases remain better suited for structured data and applications requiring complex queries and strict consistency. In short, **SQL and NoSQL are complementary**, with each being ideal for different use cases. NoSQL supports agile development and scalability, making it a strong fit for modern web and big data applications. Thus, an organization must choose a database based on specific workload demands.

N. Modi and J. Patel [21] describe a comparative analysis of query processing architecture differences in relational and document-oriented databases, which differ in handling selection, sorting and join operations. Their study focuses on schema design, scalability, consistency, indexing strategies and query execution, which will be useful for selecting an appropriate database for specific application requirements. Relational databases execute complex queries, ensuring strict consistency and maintaining data integrity through ACID properties. Unlike a document-oriented database, it challenges schema flexibility in handling semi-structured data and horizontal scalability. The document-oriented database is preferable for modern or dynamic applications; however, it lacks advanced join capabilities, complex query optimization, and strict consistency features. The gap or future research scope described in this paper is the emergence of hybrid, multi-model systems and the adoption of polyglot persistence, which allows for optimizing system performance and selecting the most appropriate storage model for each use case, and also recommends exploring the integration of AI-driven query optimizers.

## 3. Key Findings and Discussion

Based on the literature reviewed in Section 2, this analysis summarizes studies published between 2011 and 2025. The findings highlight significant differences and similarities between relational (SQL) and non-relational (NoSQL) database systems across multiple parameters, including data modeling, scalability, performance, consistency, indexing, query languages, applicability, and suitability for Big Data environments.

### 3.1 Data Models and Flexibility

Relational databases manage structured data using a fixed schema, typically in tables with predefined relationships. They are rigid in their schema, do not allow changes in fields when required, and cannot store semi-structured or unstructured data

NoSQL databases use a schema-less data model. They provide a wide range of data models, some of them are key-value store, column-oriented, document-oriented, graph-based and object-oriented Databases. They are flexible, schema-less or dynamic

schemas, allowing rapid and dynamic changes in data structure, for handling semi-structured and unstructured data [1, 6, 14, 20].

### 3.2 Scalability and Distributed Architecture

Relational databases allow vertical scaling, handle a large amount of data using adding more resources to a single server, leading to increasing hardware costs and difficulty with distribution.

NoSQL databases are popular for allowing their horizontal scalability, easy distribution of high-volume data across multiple nodes, providing automatic sharding, replication and fault tolerance, and offering superior performance in distributed environments. For this, it becomes more popular day by day [1, 6, 11, 13, 14, 15, 17, 19, 20, 21].

### 3.3 Performance and Query Processing

The results of the Performance evaluation are based on the type of context, different workloads, various operations, Indexing or various application domains.

Relational databases provide better performance for structured data, complex joins, transaction processing and analytical queries, which require strong ACID properties. They decrease their performance when dealing with large datasets or unstructured data.

In contrast, NoSQL databases outperform in flexible, large amounts of data, unstructured data, and high concurrency. They perform better in **retrieve** and **insert** operations with a large amount of data in some applications, worse in others [4, 9, 10, 11, 12, 13, 18].

### 3.4 Consistency and Reliability

Relational databases implement ACID (Atomicity, Consistency, Isolation, Durability) properties to maintain strong consistency and transactional integrity. Relational databases ensure immediate consistency

In contrast, NoSQL databases follow BASE principles (Basically Available, Soft state, Eventual consistency). According to the CAP theorem, they prioritize availability and partition tolerance, with eventual consistency. NoSQL systems allow temporary inconsistencies to achieve high availability and partition tolerance [2, 5, 13, 14, 19, 21].

### 3.5 Indexing

Indexing is performing a major role for query performance in terms of speed and memory space in both systems [12].

Relational or SQL databases support robust or mature indexing techniques [3, 9, 13, 21].

NoSQL databases have custom or model-specific fields for indexing, support single field indexes, compound indexes, geospatial indexes, and text search indexes, etc [5, 10, 13].

### 3.6 Query Language

Relational databases have a standard query language (SQL), support advanced operations like complex joins, nested queries, and aggregations.

NOSQL does not have a standard query language. However, they use various custom query mechanisms depending on the data model, and they use their query language, such as Cassandra uses CQL (Cassandra query language), MongoDB uses mongo query language or Gremlin for graph databases, etc. Therefore, it is not easy to switch from one to another NOSQL database. Hence, there is a need for a common query language like SQL, which can be used for all NoSQL databases. The UnQL is Unstructured Query Language, being developed by the creators of Couch and SQLite, as a superset of SQL, a collective effort to bring a familiar and standardized data definition and data manipulation language to the NOSQL platform [5, 6].

For example, Oracle Database uses SQL as its query language, whereas MongoDB relies on API calls, JavaScript, and REST for data interactions [1].

### 3.7 Integration with Big Data

For big data integration, the system requires flexibility, scalability, handling a huge volume of data, semi-structured or unstructured data and is suitable for a distributed environment. For this, NoSQL-MongoDB is best suitable for big data integration rather than Relational databases. Relational databases can require significant architectural modifications or the use of additional tools for adapting Big Data [7, 13, 19].

### 3.8 Applicability

The different applications have different requirements or needs; selecting the appropriate or most suitable database system is a crucial decision. This review paper also helps with this task.

Relational databases are best suitable for OLTP systems, Banking, ERP, HR, Financial systems, enterprise applications and applications requiring complex queries, strong consistency and transactional integrity.

NoSQL databases are best suitable for E-commerce, social media, content management, web and cloud applications, sensor data, IoT and mobile applications, Big data and real-time analytics. [11, 19, 20, 21].

## 4. Research Gap

Research comparing relational and non-relational database architectures is extensive, but significant gaps remain. Most studies highlight differences in schema design, scalability, and performance, but do not provide a structured framework to guide database selection for different applications. Many evaluations are performed in experimental or small-scale environments, which do not reflect real-world workloads with heterogeneous data, high concurrency, and evolving schemas. The trade-offs between ACID and BASE properties are often discussed conceptually but lack detailed experimental validation. There is limited evidence on how these trade-offs impact reliability, latency, and throughput in mission-critical and real-time systems.

Another gap exists in the study of indexing, query processing, and query language standardization. Relational databases use mature indexing methods, while indexing in NoSQL remains model-specific and less optimized for large datasets. Few studies explore how indexing directly affects performance at scale. Interoperability challenges between SQL and NoSQL systems also remain underexplored, especially in hybrid environments. In addition, most comparative studies focus on traditional applications such as OLTP or e-commerce. Emerging domains like IoT, AI-driven analytics, and edge computing are rarely considered. These gaps indicate the need for more experimental, application-oriented, and hybrid-focused research to address the demands of modern data-intensive systems.

## 5. Conclusion

Relational databases (Oracle, MySQL, MS SQL Server) remain robust and reliable choices for applications that demand structured data, complex queries, and strong consistency through ACID (Atomicity, Consistency, Isolation, Durability) compliance. They have standardized query languages as a SQL support for joins or nested queries, traditional processing systems and transactional workloads [1, 2, 3, 6, 8, 10, 15, 19, 21].

Conversely, NoSQL databases like MongoDB offer for handle high volumes of semi-structured and unstructured data in distributed environments. They provide schema flexibility, horizontal scalability, and high-speed data ingestion capabilities, making them well-suited for Big Data, real-time analytics, and modern web-scale

applications. Performance evaluations across multiple studies have shown that NoSQL systems outperform relational systems in terms of large volume datasets, read/write speed, scalability and dynamism. [4, 9, 10, 11, 12, 13, 18].

However, NoSQL lacks strong consistency, support for complex queries or strict transactional guarantees, which can be critical in financial or enterprise environmental systems. For this, the selection between relational (SQL) and non-relational (NoSQL) database systems should be guided by application-specific requirements, including data structure, scalability needs, consistency guarantees, and query complexity. [12, 14, 16, 17, 20, 21].

In conclusion, neither database model is fulfil for all application needs; rather, each has to offer unique advantages and trade-offs depending on the use case. A thoughtful evaluation of system requirements, informed by performance benchmarks and domain-specific needs, is essential when selecting the most appropriate database architecture.

## References:


[1] A. Boicea, F. Radulescu, and L. I. Agapin, "MongoDB vs Oracle - database comparison," in *Proc. 2012 IEEE 8th Int. Conf. Electronics, Computers and Artificial Intelligence (ECAI)*, 2012.

[2] A. Faraj and Bilal, "Comparative study of relational and non-relational database performances using Oracle and MongoDB systems," *J. Eng. Dev.*, vol. 18, no. 6, pp. 90–96, 2014.

[3] R. G. Gaikwad, "SQL and NoSQL: Which is better," *Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET)*, vol. 4, no. 9, pp. 3434–3437, 2015.

[4] S. Shetty, "Performance analysis of queries in RDBMS vs NoSQL," in *Proc. IEEE ICICICT*, 2019.

[5] S. Singh and A. Tiwari, "Comparative study of MongoDB and NoSQL databases," *Int. J. Comput. Appl.*, vol. 182, no. 12, pp. 14–18, 2019.

[6] A. Nayak, A. Poriya, and D. Poojary, "Type of NoSQL databases and its comparison with Relational databases," *International Journal of Applied Information Systems*, vol. 5, no. 4, pp. 16–19, Mar. 2013.

[7] A. Ahmad, “From Relational databases to NoSQL databases: Performance evaluation,” *Int. J. Comput. Sci. Mobile Comput.*, vol. 4, no. 9, pp. 256–260, 2015.

[8] R. Győrödi, “A comparative study: MongoDB vs MySQL,” *J. Comput. Appl.*, vol. 6, no. 2, pp. 13–17, 2015.

[9] R. Ceresnak and O. Chovancova, "Comparison of query performance between MySQL and MongoDB database," in Proceedings of the 2019 17th International Conference on Emerging eLearning Technologies and Applications (ICETA), 2019, pp. 165–170.

[10] S. Patel, S. Kumar, S. Katiyar, R. Shanmugam, and R. Chaudhary, “MongoDB vs MySQL: A comparative study of MongoDB and MySQL based on their performance,” *Int. J. Comput. Appl.*, vol. 177, no. 6, pp. 10–15, 2020.

[11] Ticiana, “Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation,” *J. Comput. Sci. Softw. Dev.*, vol. 8, no. 1, pp. 22–30, 2023.

[12] M. Malik, “A comparative study of unstructured data with SQL and NoSQL DBMS,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 11, no. 3, pp. 230–236, 2020.

[13] A. Elburki, “Performance comparison between SQL and NoSQL in terms of use with Big Data,” *Int. J. Data Sci. Technol.*, vol. 6, no. 1, pp. 55–64, 2024.

[14] M. Radoev, “A comparison between characteristics of NoSQL databases and traditional databases,” *Int. J. Database Manage. Syst.*, vol. 9, no. 4, pp. 1–10, 2017.

[15] M. A. Mohamed, “Relational vs. NoSQL databases: A survey,” *J. Comput. Eng. Appl.*, vol. 8, no. 4, pp. 78–83, 2014.

[16] D. Kunda and H. Phiri, “A comparative study of NoSQL and relational database,” *Zambia ICT J.*, vol. 1, no. 1, pp. 14–23, 2017.

[17] K. Sahatqija, J. Ajdari, X. Zenuni, and B. Raufi, “Comparison between relational and NoSQL databases,” in *Proc. 9th Int. Conf. Inf. Soc. Technol. (ICIST)*, pp. 68–73, 2018.

[18] M. Z. Khzn, “Comparative case study: An evaluation of performance computation between SQL and NoSQL database,” *J. Emerg. Technol. Innov. Res.*, vol. 10, no. 2, pp. 134–140, 2023.

[19] F. Klaus, “A comparative analysis of relational and non-relational database models in high volume data environments,” *Int. J. Comput. Sci. Inf. Technol. Res.*, vol. 11, no. 1, pp. 52–58, 2023.

[20] G. (Rizescu) Mihai, “Comparison between relational and NoSQL databases,” *Int. J. Adv. Comput. Sci. Appl. (IJACSA)*, vol. 12, no. 2, pp. 456–462, 2021.

[21] N. Modi and J. Patel, "Evaluating Query Processing Architectures in Relational and Document Databases," *International Journal of Innovative Research in Technology (IJIRT)*, vol. 12, no. 2, pp. –, Jul. 2025. ISSN: 2349-6002.