

Revolutionising Database Management Using MongoDB and Its Comparative Analysis

Tushar Singh Shekhawat^{1*}, Subhasmita Pradhan², Prafull Meena³ and Rishabh Pareek⁴, Akshay Arya⁵

1,2,3,4 B. Tech Student, Department of AIDS, Arya College of Engineering, Jaipur 5 Asst. Professor, Department of AIDS, Arya College of Engineering, Jaipur

Abstract—The exponential growth of data in modern applications has necessitated a shift from traditional Relational Database Management Systems (RDBMS) to more flexible and scalable solutions. MongoDB, a leading NoSQL document-oriented database, has emerged as a revolutionary tool in database management due to its dynamic schema, high scalability, and efficient handling of unstructured data. This paper reviews MongoDB's transformative impact on database management, focusing on its key features such as sharding and load balancing. A comparative analysis with MySQL, a prominent RDBMS, and other NoSQL databases is conducted, evaluating performance, scalability, and flexibility. Experimental results from prior studies, including insertion and retrieval operations, demonstrate MongoDB's superior performance for large-scale, dynamic datasets. The paper concludes by discussing MongoDB's advantages for modern applications and future research directions in database technologies.

Keywords—RDBMS, NoSQL, MySQL, MongoDB

I. Introduction

The rapid increase in data volume, velocity, and variety has challenged traditional database management systems. According to IBM, 90% of the world's data was generated in the two years prior to 2013, a trend that has only accelerated [1]. Relational Database Management Systems (RDBMS), such as MySQL, excel in handling structured data but struggle with the scalability and flexibility required for unstructured and rapidly evolving datasets. This has led to the rise of NoSQL databases, with MongoDB being a prominent document-oriented solution. MongoDB's ability to manage large-scale, dynamic data through features like sharding and a flexible schema has revolutionized database management. This review paper explores MongoDB's transformative impact, compares its performance with MySQL and other NoSQL databases, and highlights its advantages in modern applications.

II. Background

A. Relational Databases

RDBMS, such as MySQL, store data in structured tables with predefined schemas. They support SQL for querying and ensure atomicity, consistency, isolation, and durability (ACID) properties, making them ideal for transactional applications. However, their rigid schema and limited horizontal scalability pose challenges for handling large, unstructured datasets [2].

B. NoSQL Databases

NoSQL databases, meaning "Not Only SQL," address the limitations of RDBMS by offering flexible data models and high scalability. They are categorized into four types:

- **Key-Value:** Simple hash table-based databases (e.g., Amazon SimpleDB).
- **Column-Oriented:** Designed for large-scale data across multiple servers (e.g., Cassandra, HBase).
- **Graph-Stored:** Represent data as nodes and edges (e.g., Neo4j).
- **Document-Oriented:** Store data as JSON-like documents (e.g., MongoDB, CouchDB).

MongoDB, a document-oriented NoSQL database, provides dynamic schemas and robust querying capabilities, making it suitable for agile development and big data applications [2].

C. MongoDB Features

MongoDB's key features include:

- **Dynamic Schema:** Allows documents to have varying fields, supporting polymorphic data.
- **Sharding:** Distributes data across multiple servers to balance load and enhance scalability.
- **Replication:** Ensures high availability through data redundancy.
- **Robust Querying:** Supports complex queries on document fields.

III.

Literature Review

Several studies have compared MongoDB with MySQL and other NoSQL databases. Kumar et al. [3] analyzed MongoDB and MySQL, highlighting MongoDB's superior handling of unstructured data due to its flexible schema and auto-sharding. Their comparison included query syntax, execution speed, and integration methods, concluding that MongoDB outperforms MySQL for large datasets.

Györödi et al. [4] conducted a performance comparison using a dynamic application forum. They evaluated insertion, selection, deletion, and update operations, finding that MongoDB was faster at higher data loads due to its document-oriented model and customization capabilities.

Aghi et al. [5] compared SQL and MongoDB across various datasets, noting that MySQL is efficient for simple

queries on small datasets, while MongoDB excels in complex queries on large datasets. Their analysis emphasized MongoDB's scalability for big data applications.

Abramova et al. [6] evaluated five NoSQL databases (Cassandra, HBase, MongoDB, OrientDB, Redis) using the Yahoo! Cloud Serving Benchmark. They found that MongoDB and Redis are optimized for read operations, while Cassandra and HBase excel in updates. MongoDB's performance was notable for its balance of read and write efficiency.

Patil et al. [2] conducted an experiment comparing MongoDB and MySQL in a web application with one million records. Their results showed that MongoDB's insertion and retrieval times were significantly lower than MySQL's, attributed to sharding and document-level atomicity.

IV. Comparative Analysis

A. Performance

Patil et al. [2] measured the performance of MongoDB and MySQL in a web application with registration and login pages. For a single record insertion, MongoDB took 0.283 seconds compared to MySQL's higher time (exact value not specified). For login validation, MongoDB required 1.051 seconds, again outperforming MySQL. When inserting multiple records (10 to 50), MongoDB's times ranged from 0.005 to 0.01 seconds, while MySQL's ranged from 0.0511 to 0.0698 seconds (Table I).

TABLE I
Time Taken for Insertion (MySQL vs. MongoDB) [2]

Test #	No. of Records	MySQL (s)	MongoDB (s)
1	10	0.0511	0.005
2	20	0.0520	0.007
3	30	0.0565	0.01
4	40	0.0598	0.01
5	50	0.0698	0.01

B. Scalability

MongoDB's sharding distributes data across multiple servers, reducing the load on individual servers and enabling horizontal scalability. MySQL, reliant on vertical scaling (upgrading server resources), is less efficient for large datasets. Sharding allows MongoDB to handle increasing data loads by partitioning data into smaller, manageable chunks [2].

C. Flexibility

MongoDB's dynamic schema supports evolving data structures without requiring schema migrations, unlike MySQL's rigid schema. This flexibility is critical for applications with rapidly changing requirements, such as agile development environments [4].

D. Atomicity

MySQL ensures ACID compliance at the transaction level, suitable for mission-critical applications. MongoDB provides atomicity at the document level, allowing faster updates but lacking full transaction atomicity across multiple documents. This trade-off enhances MongoDB's performance for non-transactional applications [2].

V.

Discussion

MongoDB's features, particularly sharding and dynamic schemas, have revolutionized database management by addressing the limitations of RDBMS. Its ability to scale horizontally and handle unstructured data makes it ideal for big data and real-time applications. The comparative analysis shows that MongoDB outperforms MySQL in insertion and retrieval operations, especially for large datasets, due to its document-oriented model and load balancing capabilities. However, MySQL remains preferable for applications requiring strict transactional integrity.

Compared to other NoSQL databases, MongoDB balances read and write performance, unlike Redis (optimized for reads) or Cassandra (optimized for updates) [6]. Its ease of use and robust querying capabilities further enhance its adoption in diverse applications, from e-commerce to IoT.

Limitations of MongoDB include its lack of full ACID compliance and higher memory usage compared to some NoSQL databases. Future research could explore hybrid models combining MongoDB's scalability with RDBMS transactional features.

VI.

Conclusion

MongoDB has transformed database management by offering a scalable, flexible, and high-performance solution for modern applications. Its sharding, dynamic schema, and document-oriented model enable efficient handling of large, unstructured datasets, outperforming MySQL in insertion and retrieval operations. While MySQL excels in transactional applications, MongoDB's advantages make it a preferred choice for big data and agile development. Future research should focus on enhancing MongoDB's transactional capabilities and optimizing its performance for emerging technologies like IoT and cloud computing.

References

- [1] A short history of databases: From RDBMS to NoSQL and beyond, <https://www.3pillarglobal.com/insights/short-history-databases-rdbms-nosql-beyond,2013>.
- [2] M. M. Patil, A. Hannifi, C. Tejeshwar, and P. Patil, "A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retrieval operations using a web/android application to explore load balancing - sharding in MongoDB and its advantages," in Proc. Int. Conf. I-SMAC, 2017, pp. 325–328.
- [3] L. Kumar, D. S. Rajawat, and K. Joshi, "Comparative analysis of NoSQL (MongoDB) with MySQL database," IJATER, 2016.
- [4] C. Györödi, R. Györödi, G. Pecherle, and A. Olah, "A comparative study: MongoDB vs MySQL," ResearchGate Conf. Paper, Jun. 2015, doi: 10.13140/RG.2.1.1226.7685.

- [5] R. Aghi, S. Mehta, R. Chauhan, S. Chaudhary, and N. Bohra, "A comprehensive comparison of SQL and MongoDB databases," *Int. J. Sci. Res.*, vol. 5, no. 2, Feb. 2015.
- [6] V. Abramova, J. Bernardino, and P. Furtado, "Which NoSQL database? A performance overview," *Open J. Databases*, vol. 1, no.2,2014.

