Electronic Journal of University of Aden for Basic and Applied Sciences
EJUA-BA Vol. 7 No. 1 (2026)
https://doi.org/10.47372/ejua-ba.2026.1.498
ISSN: 2708-0684

RESEARCH ARTICLE

EXPERIMENTAL PERFORMANCE EVALUATION OF MONGODB AND SQL SERVER UNDER LARGE-SCALE WORKLOADS

Maria Othman Saleh Makcha[1,*], and Khaled Ahmed Abood Omer[1]

[1] *Dept. of Computer Science and Engineering, Faculty of Engineering, University of Aden, Yemen*

***Corresponding author: Maria Othman Saleh Makcha; E-mail: omothman20921@gmail.com**

Received: 19 February 2026 / Accepted: 12 March 2026 / Published online: 31 March 2026

Abstract

The rapid growth of large-scale and heterogeneous data generated by web applications, cloud platforms, and Internet of Things (IoT) systems has increased the need for efficient and scalable data management solutions. Traditional relational database management systems (RDBMS), such as Microsoft SQL Server, ensure strong consistency and data integrity, while NoSQL systems, like MongoDB, provide schema flexibility and horizontal scalability. Selecting an appropriate database architecture remains a critical design decision for modern applications. This study presents a controlled experimental performance evaluation of Microsoft SQL Server and MongoDB under identical deployment conditions. Both systems were containerized using Docker and tested with standardized datasets ranging from 10K to 5M records. Performance was assessed in terms of insertion time, query latency, update and delete execution time, CPU utilization, memory consumption, and scalability behavior. Monitoring was conducted using Prometheus and Grafana to capture system metrics. Experimental results indicate that MongoDB shows better performance in insert operations, queries, and resource efficiency, while SQL Server shows advantages in structured and type-based queries. The findings highlight that database selection should be driven by workload characteristics and application requirements rather than general performance assumptions.

Keywords: Performance evaluation; MongoDB; SQL Server; Database; SQL; NoSQL.

1. Introduction:

The exponential growth of digital data in modern computing environments has fundamentally transformed database system requirements. Applications operating in cloud computing, distributed systems, e-commerce platforms, and IoT ecosystems demand scalable, high-performance, and flexible data storage solutions. Traditional relational database systems were originally designed for structured data and transactional integrity, whereas modern applications increasingly process semi-structured and unstructured data at large scale. The need to manage these new types of massive data has led to the emergence of NoSQL databases, which are more responsive and scalable. As a result, researchers began analyzing and comparing these two systems to determine the best uses of each and to guide organizations in choosing the most suitable solution according to their needs [1].

MongoDB is a NoSQL database system, meaning it's a non-relational database that stores data in documents rather than traditional tables. This data is often stored in a format similar to JSON or BSON, which gives MongoDB significant flexibility in storing non-static data without requiring a fixed schema before data entry [2]. Its features include a dynamic schema that allows for high flexibility in data structure, enabling easy addition or deletion of fields and modification of document structures at any time. It doesn't support relationships by default, relying on a document model, but allows for embedding or linking documents as needed. No predefined schema is required; data can be created and modified without pre-designing a database structure.

MongoDB is fast, flexible, and suited for applications that require rapid response, handling large datasets, and horizontal expansion. Its non-static model allows for structure changes without prior schema modification, making it perfect for rapid application development and constantly changing data requirements [3].

Microsoft SQL Server is a relational database management system (RDBMS) developed by Microsoft.

Its defining characteristic is its table-based relational architecture, where data is stored in structured tables composed of rows and columns. It requires a data schema definition before data entry, specifying data types and their allowed values. It supports relationships and transactions, allowing data to be linked via keys, and provides support for transaction-based operations to ensure data integrity. It uses SQL as a standard query language for accessing and managing data. SQL Server is a reliable, high-performance database used in business applications and applications requiring data integrity and accuracy [3].

SQL is the ideal choice for systems that require data integrity and accuracy, such as transactional systems and sensitive data, while NoSQL offers greater flexibility in handling unstructured and massive data, with high scalability and performance in big and distributed data environments. Therefore, the choice between them remains dependent on the nature of the application and the business needs. It is wise for organizations to adapt to the characteristics of each type to achieve the best performance and efficiency in managing their data[1].

Despite numerous comparative studies between SQL and NoSQL systems, performance outcomes often vary depending on workload characteristics, data modeling strategies, indexing mechanisms, and hardware configurations. Therefore, empirical evaluation under controlled and reproducible conditions remains essential.

This research aims to provide a systematic performance comparison between Microsoft SQL Server and MongoDB under identical experimental conditions. The primary objectives are to evaluate insertion performance across increasing data volumes, analyze query latency for different query types, measure update and delete efficiency, assess CPU and memory consumption during workload execution, and examine scalability behavior under large-scale datasets. By conducting controlled containerized experiments, this study demonstrates empirical evidence to guide database selection decisions for large-scale data environments.

The research is organized as follows: Section 2 reviews previous studies and compares SQL Server with MongoDB. Section 3 explains the methodology used in this research. Section 4 discusses the experimental results of both systems. Section 5 concludes with the findings and future work.

2. Related work:

Numerous studies have compared the structure, characteristics, and architecture of SQL and NoSQL databases, highlighting the differences between them in aspects, including the type of data stored and the applications of each type. These studies have shown that the choice of database type depends largely on the application requirements and the organization. SQL relational databases are ideal for processing structured data that requires data integrity, transaction support, and high stability. In contrast, NoSQL is more flexible and scalable, making it suitable for handling large and diverse amounts of data, especially in parallel computing environments and modern web applications. However, it lacks rigorous processing features like ACID, which may be necessary in some cases. Therefore, there is no single database that works for everyone; each type of database has its advantages and disadvantages, and the choice should be based on the performance, scalability, stability, and security requirements [1,4]

Qumins et al. compared SQL Server, a relational database, with MongoDB as a non-relational (NoSQL) database. They highlighted the differences between them, explaining that SQL Server relies on predefined tables that use a fixed schema, while MongoDB uses dynamic schemas without constraints on data structure. It also outlined some key differences between the systems, identifying the most suitable applications for each type, processing, structure, and performance. The study further showed that SQL Server is more popular than MongoDB, according to db-engines.com, and has greater support resources. MongoDB is used in situations that require fast response and high flexibility, in web and mobile applications [5].

I. Qaddara et al. compared the efficiency of read, write, delete, and create operationsin the context of key and value stores implemented using both NoSQL databases (MongoDB, RavenDB, and Cassandra) and SQL databases (MS-SQL) over an average of three runs. The first experiment measured the time taken to create a database instance. The results showed that RavenDB had the fastest instance creation time, followed by MongoDB and Cassandra, while MS SQL showed the slowest performance in this process. The second experiment measured the time taken to read. The results showed that MS SQL gave the best average read time, while RavenDB gave the worst. The third experiment measured the time taken for deletion. MS SQL gave the worst time, while RavenDB gave the best time, followed by MongoDB [6]

A. Malik et al. conducted a comparative analysis between Microsoft SQL Server relational database and MongoDB a non-relational database within the context of unstructured JSON data representation. A series of experiments was performed. In the first experiment, 100,000 entries were entered into both the RDBMS (SQL) and the NoSQL (Mongo) databases. It was concluded that MongoDB was faster than the other two databases used in the experiment, generally fifteen seconds faster than SQL Server. In the second experiment, a random string was searched ten times within both databases, and the results of each iteration were compared. The results of these experiments were very intensive, revealing a significant difference between

the two types of databases. In fact, searching for a specific string occurrence was extremely efficient and easy to implement within MongoDB. In the third experiment, the search was performed using a randomly generated identifier, as follows: the primary key is the SQL Server ID, while the default index in MongoDB is in the _id field. SQL Server proved to be a very fast and efficient tool when searching using the ID field, which contains the primary key [7].

A. Rudniy designed and implemented a data warehouse (DW) for an online learning platform using three technologies: Microsoft SQL Server, MongoDB, and Apache Hive. The three systems were evaluated in terms of text structure and descriptive analytics. Apache Hive was found to achieve the best processing time, followed by SQL Server and MongoDB. In terms of analytical queries, SQL Server performed best, followed by MongoDB and Hive [8].

C. Ming Wu et al. also conducted comparisons between MongoDB and SQL Server. The study showed that MongoDB in NoSQL is ten times more efficient in reading and writing than MS-SQL databases. This confirms that NoSQL database technology is a highly viable option for future use [9].

3. Research Methodology:

This section describes the experimental methodology used to compare and evaluate the performance of MongoDB and SQL Server in managing intellectual property data. The setup includes environment configuration, data generation, implementation, and the performance metrics used for evaluation.

3.1. Hardware and Software Environment:

These experiments are conducted on a Windows 11 machine with an Intel Core i7 processor, 8 GB of RAM, and Docker Desktop (v4.36.1) [10]. The two databases, viz. MongoDB 3.6 and SQL Server 2019 were used in isolated Docker containers. Python 3.9 is used to handle all operations with Prometheus and Grafana monitoring tools in this research work. The same hardware configuration was used for all tests to maintain consistency and comparability.

3.2. Experimental setup:

All experiments were conducted using a containerized simulation environment using Docker Compose to emulate a network on a single machine [11]. Each database system had its own independent container, with CPU and memory resources to ensure consistent resource availability, fair comparison, and minimize the impact of external factors. Identical workloads and dataset sizes were applied to both systems. This enabled realistic execution, resource isolation, and reproducible performance measurement. System metrics were collected using Prometheus and visualized with Grafana. Each experiment was repeated multiple times, and average values were recorded to minimize random fluctuations and improve measurement reliability.We ran each system separately to ensure that metric measurements were isolated for fair comparison. Further, the database was reinitialized before each experiment.

3.2.1. Workflow Diagram:

Figure 1 illustrates the architecture of the prototype that shows the interactions among MongoDB, SQL Server, and the utility services used. It illustrates that all containers operate on a single network within Docker. Data is entered into both SQL Server and MongoDB using the Python script. and also shows that all of this is monitored using the monitoring tools Grafana and Prometheus, which are also available on the same Docker network as all the containers we are working on.

Figure 2 illustrates the workflow for the experimental setup, including the data generation, insertion, query, monitoring, and analysis phases. This diagram shows the parallel operations applied to both MongoDB and SQL Server to ensure fairness and equality.

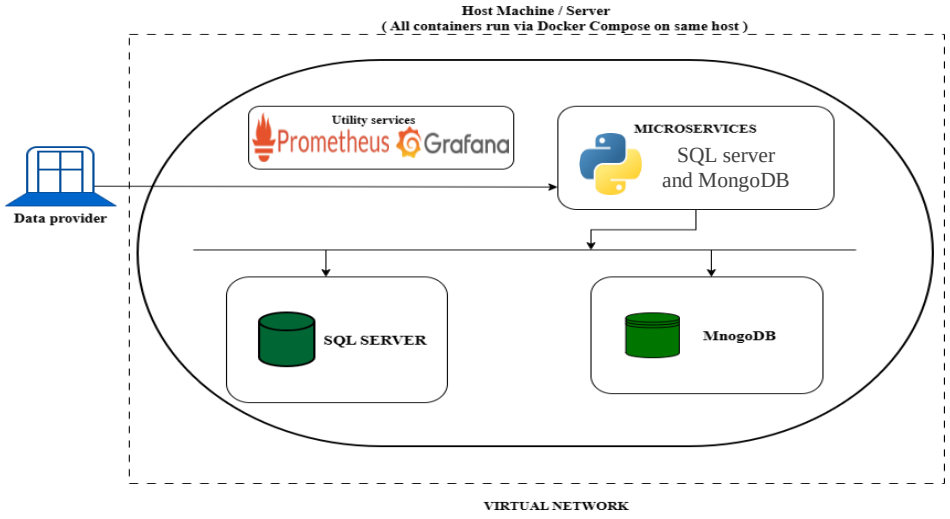

Fig. 1: High-level architecture of the prototype interacting with MongoDB and with SQL server

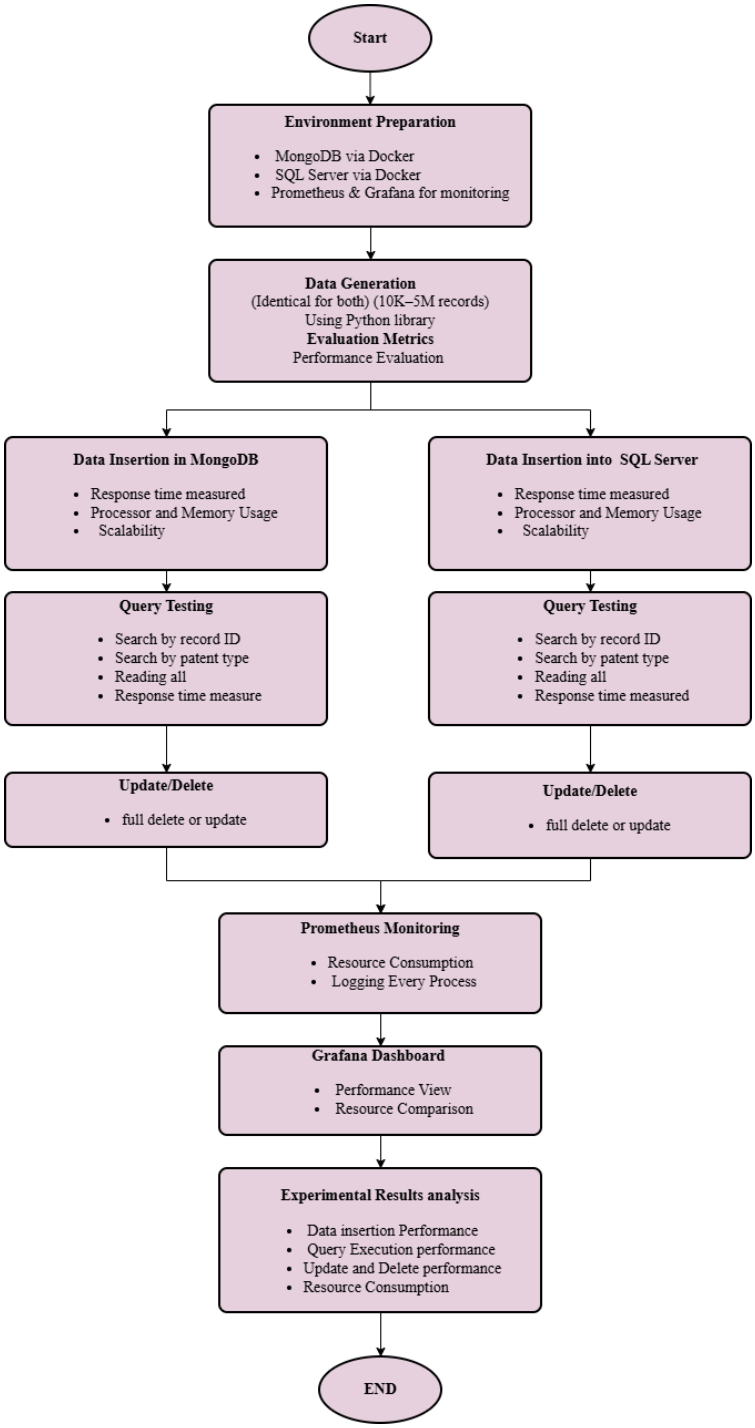

Fig. 2: Flow diagram of experimental setup

3.3. Dataset Preparation:

Due to the absence of publicly available large-scale intellectual property datasets, synthetic data were generated using the Faker Python library [12]. The dataset simulated intellectual property records with the following attributes: ‘record ID’, 'name', 'description', 'owner', 'type', 'status', 'date', 'document_url'. The datasets were prepared with 10k, 50k, 100k, 500k, 1M, 2M, and 5M records, respectively, to evaluate performance at different scales. This incremental scaling enabled evaluation of system behavior under increasing data volumes.

3.4. System Implementation:

We performed the following operations: data insertion, querying, deletion, update, and time measurement using Python scripts. To insert data into SQL Server, we used the pyodbc library to execute INSERT commands in batches. Similarly, in MongoDB, we used pymongo to send CREATE transactions, where each record represented a new patent.

Identical queries were performed on both databases: Record_ID, patent_type, and Query All. Also, update/deletion operations were performed, so that the UPDATE and DELETE commands were executed directly in both SQL Server and MongoDB without any intervention.

3.5. Monitoring

Prometheus continuously collects system-level metrics, while Grafana provides real-time visualization. Prometheus was configured within the isolated environment to collect container and system performance metrics using Cadvisor, including CPU and memory consumption for each system. A default data collection period of 15 seconds was used in all experiments. The same settings were applied to both systems under investigation to ensure measurement fairness and reproducibility. Monitoring was performed concurrently with workload execution to capture realistic resource behavior under load conditions.

3.6. Performance Evaluation Metrics

1. **Data Entry Time**: The time it takes to enter data into the database. Measured in seconds using a Python script.
2. **Query, Delete, and Update Execution Time**: The time it takes the system to retrieve a record or set of records based on a specific query, delete and update.
3. **Processor and Memory Usage**: This measures CPU and RAM usage during execution. It was monitored using Grafana and Prometheus.
4. **Scalability**: This measures the system's ability to maintain performance under increased data or process load.

3.7. Fairness Considerations and Limitations

Both systems were evaluated under identical hardware and workload conditions to provide a controlled environment that ensured a fair and consistent comparison between the evaluated systems. The evaluation framework focused on a single-node operating environment using a containerized simulation environment using Docker Compose, and artificially generated datasets of increasing size to systematically observe performance behavior under controlled workloads. This design promotes the systematic consistency of the experiment,

Furthermore, the experiments were limited by the hardware resources available in the test environment, which restricted the maximum dataset size and the possibility of scaling the experiment to a larger number of nodes. Future work could expand the evaluation to include multi-node distributed environments and more complex real-world data.

4. Experimental Results and Discussion

This section presents the experimental results obtained for evaluating MongoDB and Microsoft SQL Server under identical deployment conditions. The analysis focuses on insertion performance, query latency, Update and delete, resource utilization, and scalability.

4.1. Data insertion Performance

The experimental results demonstrate a consistent performance advantage for MongoDB in bulk insertion operations. Across all dataset sizes from 10K to 5M records, MongoDB required significantly less execution time than SQL Server. At 10K records, MongoDB completed insertion substantially faster than SQL Server, while at 5M records, MongoDB maintained lower execution time and more stable resource utilization.

MongoDB exhibited gradual increases in CPU and memory consumption as the dataset size grew, indicating stable scalability behavior. In contrast, SQL Server demonstrated higher and more variable CPU usage, often exceeding 50% utilization during large insert operations. Memory consumption for SQL Server remained consistently higher than that of MongoDB across all scales.

These results suggest that MongoDB’s document-oriented architecture and batch insertion mechanism provide advantages in high-throughput write-intensive workloads.

Figure 3 shows that SQL Server execution times were large compared to those of MongoDB and increased with the increase in the number of inserted records.

Figure 4 shows that MongoDB demonstrated stable resource consumption, operating at relatively consistent values compared to SQL Server.

Figure 5 shows that MongoDB demonstrated stable resource consumption, operating at relatively consistent values compared to SQL Server.

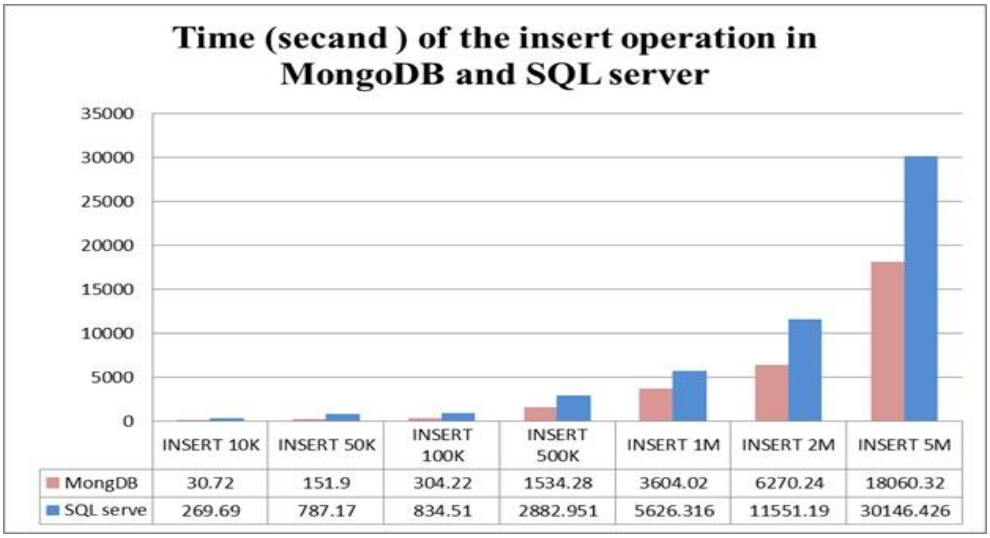

Fig. 3: Execution time(second) for insert operation

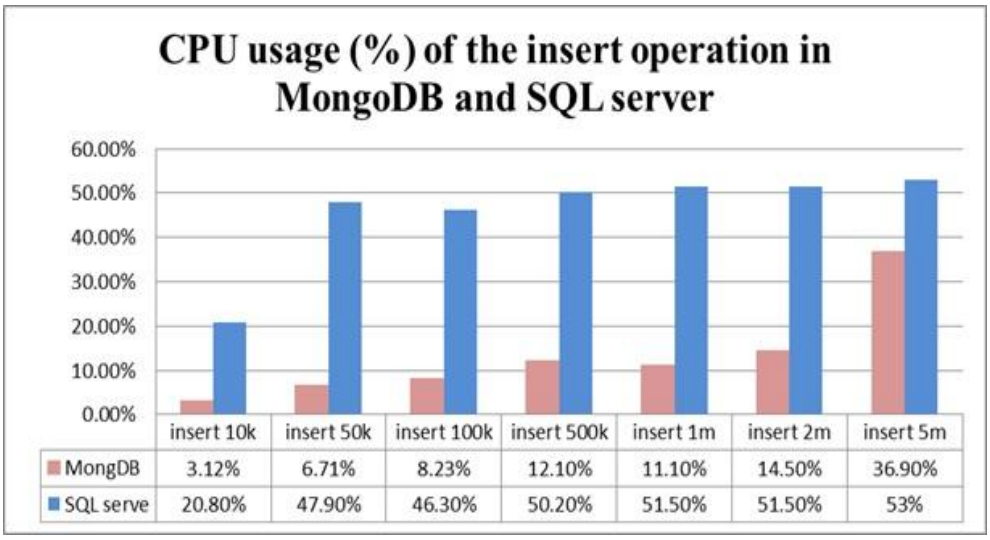

Fig. 4: CPU usage (%) for insert operation

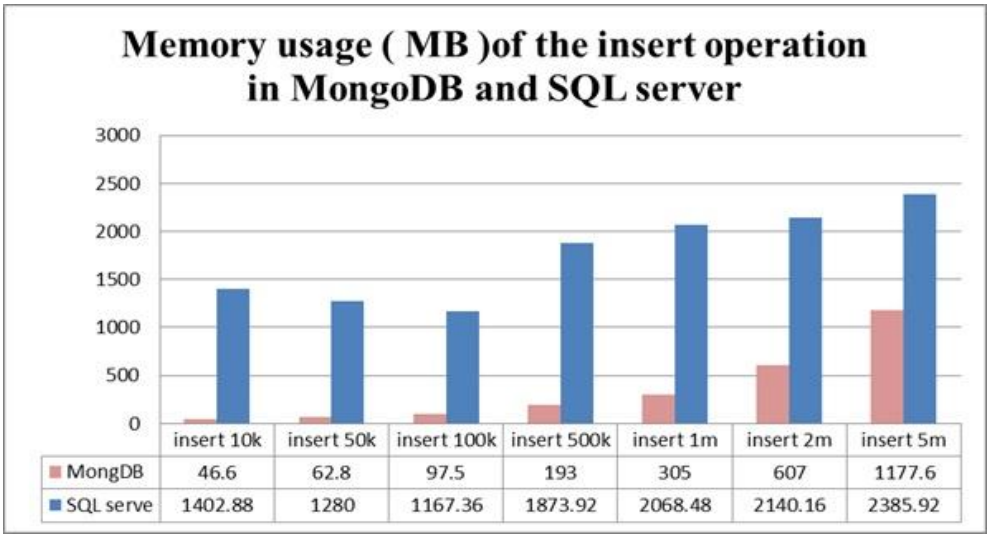

Fig. 5: Memory usage (MB) for insert operation

4.2. Query Execution Performance:

To evaluate the data retrieval efficiency of both systems, a set of queries was executed under identical conditions. All queries were executed using Python scripts, with both query response time and resource usage measured. Query performance varied depending on query type as follows:

4.2.1 Record ID Query: MongoDB achieved faster retrieval times and significantly lower resource consumption when querying by record ID. This is attributable to its default indexing on the _id field.

4.2.2 Query All Records: For full dataset retrieval, MongoDB again showed lower execution time and more moderate CPU and memory usage, particularly at larger scales.

4.2.3 Query by Patent Type: SQL Server outperformed MongoDB in structured type-based queries. This result reflects the efficiency of relational indexing and optimized query planning for structured data filtering.

Overall, MongoDB demonstrated superior performance in most query scenarios, particularly those involving large result sets. SQL Server performed better in structured attribute filtering operations, as shown in Figures 6 7, and 8 below.

Figure 6 shows that MongoDB's execution time was minimum compared to that of SQL Server, except for patent type search queries, where SQL Server had the minimum time.

The comparison in Figures 7 and 8 revealed resource consumption. MongoDB outperformed SQL Server in almost all operations. But in the patent type search, SQL Server outperformed MongoDB.

4.3. Update and Delete Operations

Update and delete experiments revealed that MongoDB consistently required less execution time and lower resource consumption compared to SQL Server.

For update operations: MongoDB completed modifications faster with minimal CPU impact, while SQL Server exhibited higher memory usage during update execution.

For delete operations, MongoDB showed reduced latency and stable resource behavior, whereas SQL Server required more memory and processing time.

These results indicate that MongoDB handles modification operations efficiently under the tested workload conditions.

Figure 9 shows that MongoDB performs better than SQL Server regarding execution time.

Figures 10 and 11 revealed the resource consumption. MongoDB outperformed SQL Server regarding resource consumption in all operations.

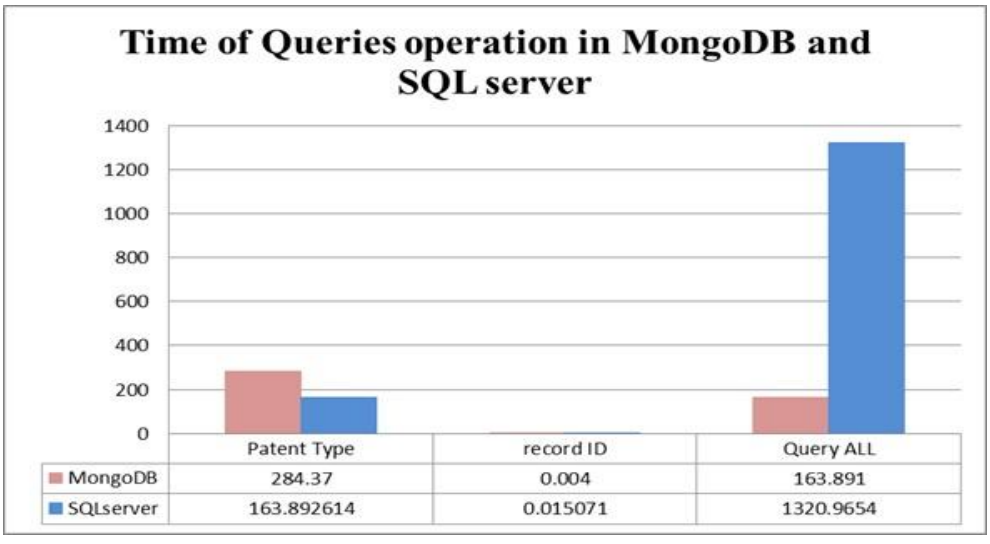

Fig. 6: Execution time(second) for Query operation

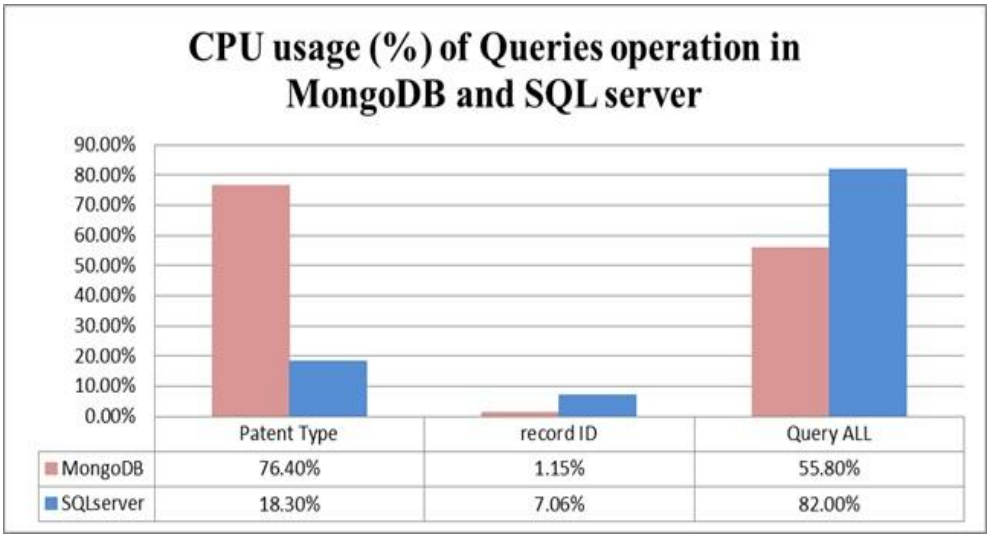

Fig. 7: CPU usage mean (%) for Queries operation

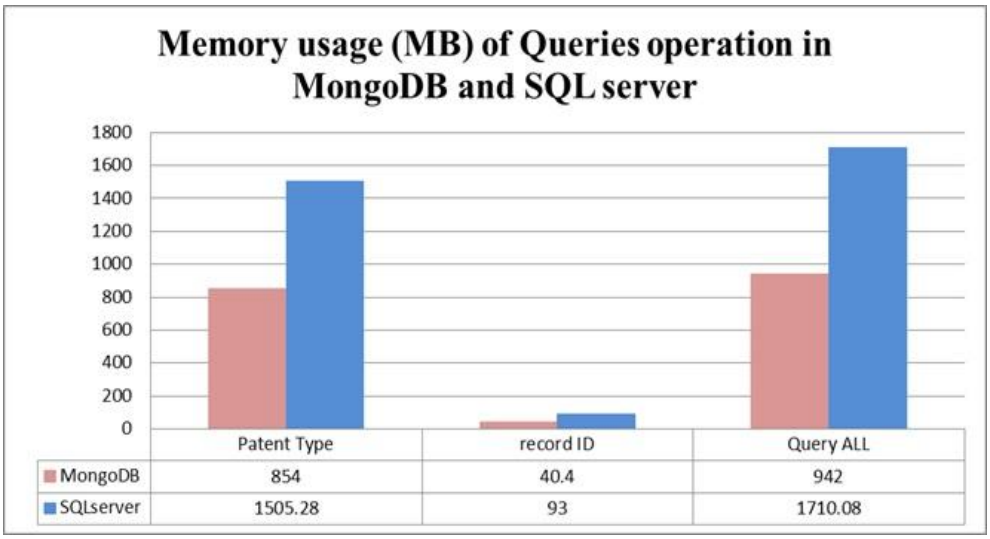

Fig. 8: Memory usage mean (MB) for Queries operation

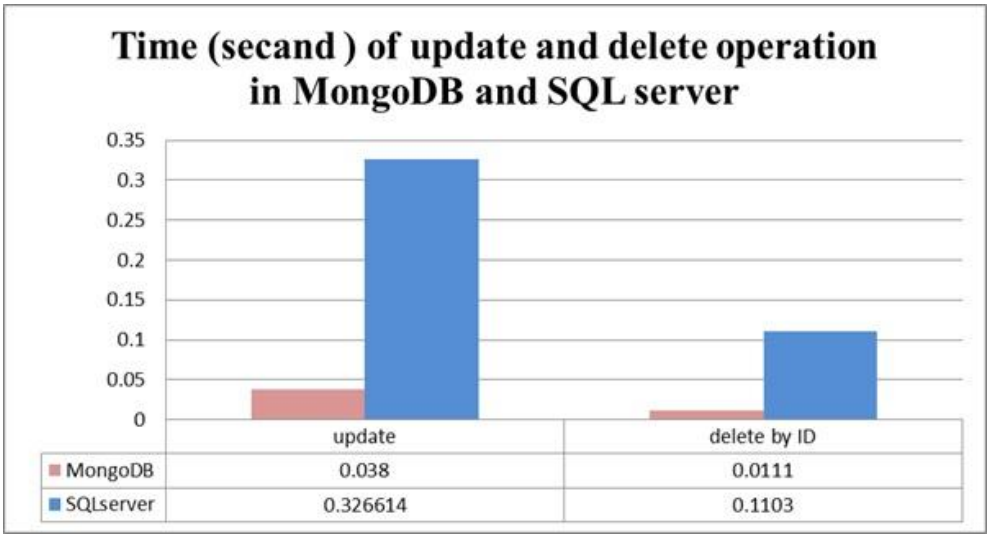

Fig. 9: Execution time(second) for update and delete operation

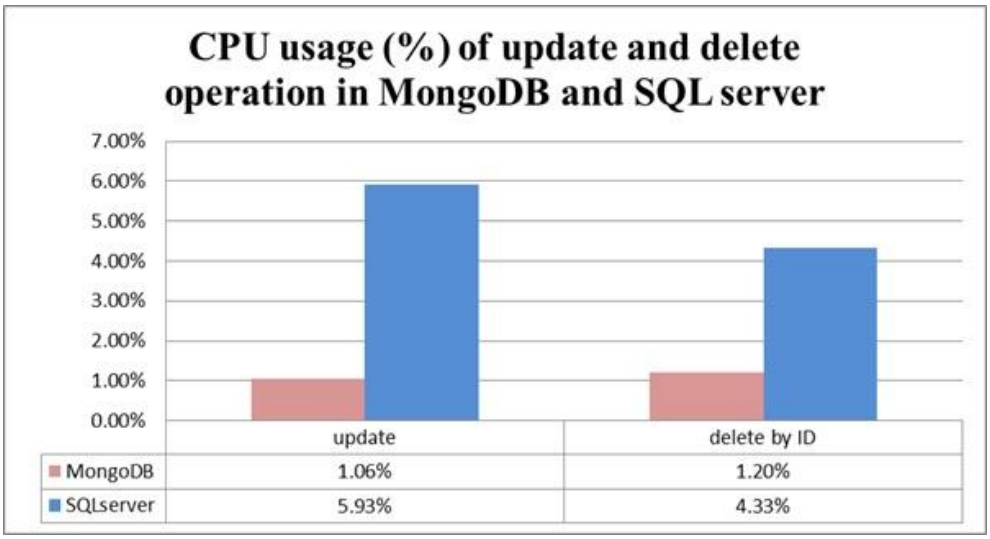

Fig. 10: CPU usage mean (%) for update and delete operation

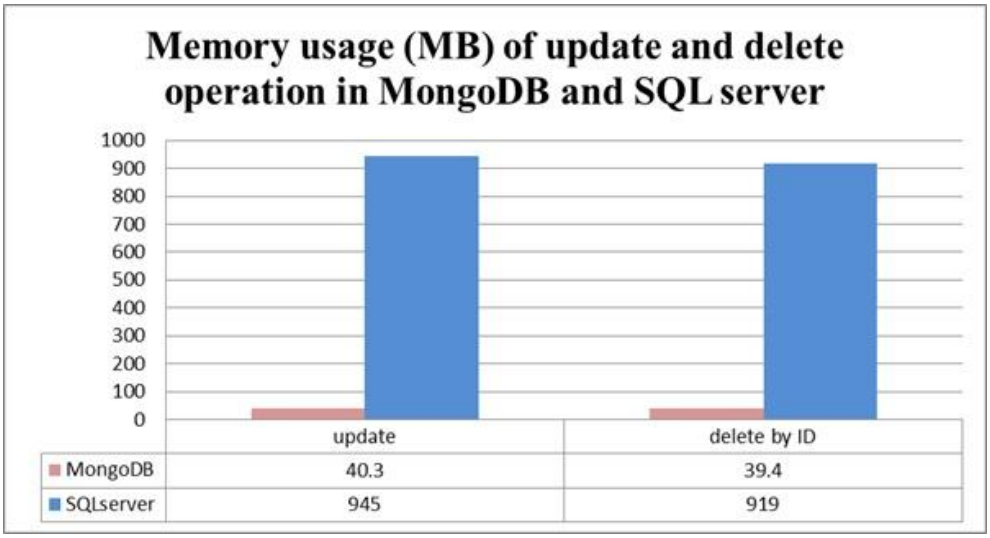

Fig. 11: Memory usage mean (MB) for update and delete operation

4.4 Discussions:

Previous experimental results evaluating the performance of the two databases across the studied core operations revealed distinct performance characteristics. The results showed that MongoDB recorded lower execution times and resource consumption for all insertion operations compared to SQL Server, even as the data size increased from 10K to 5M records. This can be attributed to the documentary nature of MongoDB databases, which allows for flexible data representation and efficient insertion operations when handling increasing data volumes.

Regarding query operations, MongoDB also exhibited lower execution times and resource consumption for most queries, except in queries based on structured data types. In these cases, SQL Server demonstrated lower execution time and resource consumption, reflecting the efficiency of its relational indexing and optimized query planning for filtering structured data.

For the remaining operations, such as modification and deletion, MongoDB consistently outperformed SQL Server in terms of execution time and resource consumption.

The results showed that SQL Server consumes more CPU and memory in most workloads. This is expected because relational systems optimize performance using efficient indexing, caching, and query processing algorithms. In contrast, MongoDB exhibits lower CPU and memory usage despite faster execution times. This is due to its non-relational documentation model, which provides greater flexibility in storing unstructured data

and faster queries with large workloads or horizontal expansion.

The experimental results demonstrate a clear trade-off between SQL and NOSQL database architectures. MongoDB delivered high performance in almost all operations, thanks to its NOSQL documentation model.

In contrast, SQL Server exhibits lower performance in most operations but demonstrates its strength in complex queries that rely on structured data and complex relations, making it suitable for complex operations and structured data.

5. Conclusions and Future Work

5.1. Conclusions

This study conducted a controlled experimental comparison between Microsoft SQL Server and MongoDB under identical containerized conditions. The evaluation covered insertion, querying, updating, deleting, scalability, CPU utilization, and memory consumption across datasets ranging from 10K to 5M records.

The findings demonstrate that MongoDB provides superior performance in: insertion operations, and semi-structured queries, update and delete execution, resource efficiency, and scalability. In contrast, SQL Server exhibits stronger performance in structured filtering queries and scenarios requiring relational optimization. MongoDB is better suited for high-throughput, schema-flexible, and large-scale distributed applications. SQL Server remains advantageous in structured enterprise environments where relational integrity and optimized joins are critical.

Despite these findings, the choice between them ultimately depends on the nature of the application and business needs. Organizations must adapt to the characteristics of each type to achieve optimal performance and efficiency in data management. SQL Server is the ideal choice for systems requiring data integrity and accuracy, while MongoDB offers greater flexibility in handling unstructured and large datasets, along with high scalability and performance in large and distributed data environments.

Although the methodology used in this study was carefully designed to provide a performance comparison between SQL Server and MongoDB. There are some limitations to this methodology. The experiments were conducted in a resource-limited environment and with hardware configurations that may not fully reflect large-scale production environments. Furthermore, the data volume processed was still below the scales we hoped to use and systems might encounter in real-world applications. These limitations do not diminish the value of the results, but they clearly define the framework within which they should be interpreted.

5.2. Future Work

Although this study provides a comprehensive performance comparison, several research extensions are possible.Future experiments should evaluate both systems in multi-node distributed cluster configurations rather than single-node Docker deployments to better reflect large-scale deployment scenarios in real-world environments.

The current study focuses on executing workloads on a single, controlled machine. Future research analyzing concurrent transaction processing, lock disputes, and throughput in concurrent multi-user environments can provide a deeper understanding of system behavior under real-world operating conditions.Future studies could also involve using real-world datasets or more complex data structures to further validate the findings of this study.

Future research could investigate the impact of advanced indexing techniques, including composite indexes, text indexes, and more complex query patterns, to understand their effect on database performance across different data models.

Using synthetic datasets ensures controlled testing, but it may not fully represent real-world anomalies. Future studies should include large, publicly available datasets to validate practical performance behavior. Additional statistical analysis techniques can also be used to enhance the reliability and strength of performance comparisons.

References:

[1] A. Malik, A. Burney, and F. Ahmed, “A Comparative Study of Unstructured Data with SQL and NO-SQL Database Management Systems”, J. Comput. Commun., vol. 08, no. 04, pp. 59–71, 2020

[2] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, “SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review”, Big Data Cogn. Comput., vol. 7, no. 2, p. 97, May 2023.

[3] A. Shamel Abdullah and A. R. Suleiman, “Comparison of SQL Server and Mongo DB, International Journal of Engineering and Innovative Technology (IJEIT) Volume 6, Issue 4, October 2016”, vol. 6, no. 4, 2016.

[4] G. Mihai, “Comparison between Relational and NoSQL Databases”, *Ann. Dunarea Jos Univ. Galati Fascicle Econ. Appl. Inform.*, vol. 26, no. 3, pp. 38–42, Dec. 2020.

[5] Computer Department, College of Science and Technology, Quminsˇet *al.*, “Performance Comparison between SQL and NoSQL in Terms of Use with Big Data”, *Int. Sci. Technol. J.*, vol. 34, no. 1, pp. 1–19, Apr. 2024.

[6] I. Qaddara, Y. Alraba'nah, and M. O. Hiari, "Evaluation of SQL and NoSQL Databases on Parallel Processing", *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 4, pp. 24298–24304, Aug. 2025.

[7] A. Malik, A. Burney, and F. Ahmed, "A Comparative Study of Unstructured Data with SQL and NO-SQL Database Management Systems", *J. Comput. Commun.*, vol. 08, no. 04, pp. 59–71, 2020

[8] A. Rudniy, "Data Warehouse Design for Big Data in Academia", Computers, Materials & Continua, vol. 71, no. 1, pp. 979–992, 2022.

[9] C. Ming Wu, Y. Fu Huang, and J. Lee, "Comparisons Between MongoDB and MS-SQL Databases on the TWC Website", AJSEA, vol. 4, no. 2, p. 35, 2015.

[10] Docker . https://www.docker.com/

[11] N. A. Sultan and R. PutrosQasha, "Container-Based Virtualization For Blockchain Technology: A Survey" Jordanian J. Comput. Inf. Technol. JJCIT, vol. 9, no. 3, Sept. 2023.

[12] Faker Python package. https://pypi.org/project/Faker/

[13] MongoDB. https://www.mongodb.com/

مقالة بحثية

تقييم الأداء التجريبي لـ MongoDB وSQL Server تحت أحمال عمل واسعة النطاق

ماريا عثمان صالح مقشع[1،*]، و خالد أحمد عبود عمر[1]

[1] قسم علوم وهندسة الحاسوب، كلية الهندسة، جامعة عدن، اليمن

* الباحث الممثل: ماريا عثمان صالح مقشع؛ البريد الالكتروني: omothman20921@gmail.com

استلم في: 19 فبراير 2026 / قبل في: 12 مارس 2026 / نشر في 31 مارس 2026

المُلخّص

أدى النمو السريع للبيانات واسعة النطاق وغير المتجانسة الناتجة عن تطبيقات الويب، ومنصات الحوسبة السحابية، وأنظمة إنترنت الأشياء (IoT) إلى زيادة الحاجة إلى حلول فعّالة وقابلة للتوسع لإدارة البيانات. تضمن أنظمة إدارة قواعد البيانات العلائقية التقليدية (RDBMS)، مثل Microsoft SQL Server، اتساقًا قويًا وسلامة للبيانات، في حين توفر أنظمة NoSQL، مثل MongoDB، مرونة في بنية البيانات وقابلية للتوسع الأفقي. لا يزال اختيار بنية قاعدة البيانات المناسبة قرارًا تصميميًا حاسمًا للتطبيقات الحديثة. تقدم هذه الدراسة تقييمًا تجريبيًا مُتحكمًا لأداء كل من Microsoft SQL Server وMongoDB تحت ظروف نشر متطابقة. تم تغليف كلا النظامين باستخدام Docker، وتم اختباره باستخدام مجموعات بيانات معيارية تتراوح من 10 آلاف إلى 5 ملايين سجل. تم تقييم الأداء من حيث زمن الإدراج، وزمن استجابة الاستعلام، وزمن تنفيذ عمليات التحديث والحذف، واستهلاك وحدة المعالجة المركزية، واستهلاك الذاكرة، وسلوك قابلية التوسع. كما تم إجراء المراقبة باستخدام Prometheus وGrafana لالتقاط مؤشرات أداء النظام. تشير النتائج التجريبية إلى أن MongoDB يُظهر أداءً أفضل في عمليات الإدراج والاستعلام وكفاءة استخدام الموارد، بينما يُظهر SQL Server تفوقًا في الاستعلامات المنظمة والمعتمدة على الأنواع. وتبرز النتائج أن اختيار قاعدة البيانات يجب أن يستند إلى خصائص عبء العمل ومتطلبات التطبيق بدلاً من الافتراضات العامة حول الأداء.

الكلمات المفتاحية: تقييم الأداء؛ MongoDB؛ SQL Server؛ قاعدة البيانات؛ SQL؛ NoSQL.

How to cite this article:

M. O. S. Makcha, and K. A. A.Omer, "EXPERIMENTAL PERFORMANCE EVALUATION OF MONGODB AND SQL SERVER UNDER LARGE-SCALE WORKLOADS", *Electron. J. Univ. Aden Basic Appl. Sci.*, vol. 7, no. 1, pp. 46-56, Mar. 2026. DOI: https://doi.org/10.47372/ejua-ba.2026.1.498

Copyright © 2026 by the Author(s). Licensee EJUA, Aden, Yemen. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-NC 4.0) license.