DOI: https://doi.org/10.57233/ijsgs.v10i2.644 ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL10 (2) WEBSITE: https://fugus-ijsgs.com.ng

INTERNATIONAL JOURNAL OF SCIENCE FOR GLOBAL SUSTAINABILITY
(A PUBLICATION OF FACULTY OF SCIENCE, FEDERAL UNIVERSITY GUSAU, NIGERIA)

Analyzing and Designing an Evaluation Benchmark for SQL and NoSQL Database Systems for some Selected Higher Institution in Zamfara State

[1]Mustapha Abubakar, [2]Sufiyanu Abubakar and [3]Umar Muhammad Bello
[1,2]Department of Computer Science, Zamfara State University Talata Mafara
[2]Department of ICT, Federal University Gusau

Corresponding Author's Email: sufymrd1989@gmail.com

Received on: April, 2024 **Revised and Accepted on:** May, 2024 **Published on:** July, 2024

ABSTRACT

This research analyzes and designs an evaluation benchmark for SQL and NoSQLdatabase systems, specifically tailored for higher institutions in Zamfara State. By benchmarking various database operations, including insert, select, update, delete, and stored procedures, we identify performance differences between the two systems. MongoDB demonstrates superior performance in handling insert, select, update, and delete operations, making it ideal for applications with high transactional demands. Conversely, MySQL excels in executing stored procedures due to its native support, which is crucial for complex procedural logic. Considering that insert, select, update, and delete operations are more common, MongoDB is generally recommended. However, for applications heavily reliant on stored procedures, MySQL remains the preferred choice. Additionally, the complexity of converting an existing MySQL database to MongoDB should be factored into the decision-making process. This study provides valuable insights and recommendations to help institutions in Zamfara State choose the most suitable database system based on their specific operational needs and performance requirements.

Keywords: SQL, NoSQL, MySQL, MongoDB

1.0 INTRODUCTION

A database (DB) is an integrated and shared computer system that holds and manages data (Coronel, Morris, & Rob, 2019; Rawat, Purnama, & Mulyati, 2021). Database comprises of two main elements: end-user data, which includes raw facts relevant to users, and metadata, which provides information about the data. The metadata is crucial for integrating and managing the end-user data efficiently (Coronel, Morris, & Rob, 2019; Song, et al., 2024).Database Management Systems (DBMS) are software tools designed to facilitate the management of databases.DBMS are the most dependable method for retrieving, storing and querying digital data (Bassil, 2012).

A database can generally be classified based on structure into relational and non-relational databases (Khan, et al., 2023; Thakur & Gupta, 2021; Zheng & Tian, 2021; Čerešňák & Kvet, 2019; Hasan, Ahmed, Chowdhury, Leion, & Prince, 2022; Győrödi, Turtureanu, Győrödi, & Zmaranda, 2023). Relational databases, originating in the 1970s, revolutionized data storage by introducing a structured approach to organizing information. They employ tables to store data and utilize the Structured Query Language (SQL) as their standard query language model. In 1988, term NoSQL was first used to denote a relational database that does not have SQL (Lith & Mattson, 2013). Later on, SQL and NoSQL because almost synonymous with relational and non-relational databases (Khan, et al., 2023). Some of the examples of SQL database include; MySQL, SQL server, Microsoft Access and Oracle DB. While, MongoDB, Cassandra, Redis, Janus Graph and Apache Hbase are some examples of NoSQL database.

Relational databasesoperate based on a principle called ACID (Atomicity, Consistency, Isolation, Durability). These properties made them the most widely used databases (Lourenço, Cabral, Carreiro, Vieira, & Bernardino, 2015; Khan, et al., 2023; Xia, Yu, Butrovich, Pavlo, & Devadas, 2022; Begum & Chitra, 2020; Yashraj & Yashasvi, 2019; Modhiya, 2021). However, the complexity and volume of data stored and organized presented reduced the efficiency of queries and also storage. Non-relational databases were developed to solve some of the problems of relational databases, in particular, they lose the ACID properties in order to make them more available and scalable (Lourenço, Cabral, Carreiro, Vieira, & Bernardino, 2015). This scenario presents a trade-off which necessitate an evaluation benchmark for SQL and NoSQL databases. This research focuses on the analysis and design of an evaluation benchmark tailored to the specific needs of selected higher institutions in Zamfara, Nigeria to aid in the selection of appropriate database systems. This research is significant as it provides practical guidance for decision-makers in selecting the most suitable database

system, thereby enhancing data management and decision-making processes.

In the subsequent sections, we will review related literature on database selection criteria and evaluation benchmarks, present the methodology employed for designing the evaluation benchmark, discuss the findings of the study, and conclude with implications, limitations, and recommendations for future research.

2.0 LITERATURE REVIEW

This section provides some of the related works on evaluation of SQL and NoSQL databases. For each work, we present the strengths and the weakness. These are presented as follows: (Li & Manoharan, 2013) conducted an independent investigation into the performance of various NoSQL and SQL databases within the context of key-value stores. They compared fundamental operations like read, write, delete, and instantiation across these databases, including an additional operation of iterating through all keys. Using an abstract key-value pair framework, the researchers implemented these operations across the tested databases. Through experimental analysis, they measured the timing of these operations, uncovering performance discrepancies among the databases for different tasks. Surprisingly, the study revealed that not all NoSQL databases outperformed SQL databases, with some exhibiting notably inferior performance. However, only limited operations were used in the evaluation criteria.

Veronika, Jorge, & Pedro, SQL (2015) conducted a research which evaluates the scalability of Cassandra and assesses the execution time of CRUD (Create, Read, Update, Delete) operations. Furthermore, it compares the performance of a relational and a non-relational system in executing decision support queries. To achieve these objectives, the researchers employ two widely recognized benchmarks: the Yahoo! Cloud Serving Benchmark, utilized to measure the execution time of requests and the speedup of Cassandra, and the Star-Schema Benchmark, employed to execute queries on a MySQL cluster (representing the relational database) and Hadoop with Hive (representing the non-SQL counterpart). However, the absence of detailed contextual information regarding the dataset, workload characteristics, and system configurations used in the experiments.

(Baralis, Dalla, Garza, Rossi, & Scullino, 2017)qualitatively four leading SQL and NoSQL databases, specifically focusing on geospatial features. They used Microsoft Azure Cloud Service's SQL Database and DocumentDB for the experiment. However, the research uses synthetic data instead of a real-world data, which might produce completely different result.

(Seenauth & Visham, 2020)evaluated the performance of SQL and NoSQL databases in an IOT-based environment. The research measured how long (in milliseconds) it took to perform each CRUD operation with and without indexing. Indexing is a technique for speeding up data retrieval. The research found out that update and delete operations are faster in NoSQL databases while Select operations execute faster in SQL databases. However, only simple form of queries were formed.

(Khan, et al., 2023) compared the performance of MongoDB and MySQL databases. The evaluation metrics used in the research are; database size, response time, retrieval time and loading. More complex operations are performed in the research including inner joins and aggregate functions. However, the research uses a limited size of data. Moreover, system load and other parameters were not utilized in evaluation.

Based on the literature review provided, it is evident that several studies have explored the performance comparison between SQL and NoSQL databases. The discrepancies found in the performance results are largely due to differences in domain and context. To address the insights and gaps identified in the literature, the following research objectives are formulated:

i. To develop an evaluation framework that aligns with the academic and administrative intricacies of the selected higher institutions.
ii. To investigate the impact of system configurations and workload characteristics on database performance.
iii. To utilize stored procedure as an operation for the evaluation

3.0 METHODOLOGY

This section outlines the methodology employed to evaluate the performance of SQL and NoSQL databases within the context of the information systems used by Federal University Gusau and School of Health Technology Tsafe. These institutions utilize MySQL databases to store student information, including grading and CGPA calculations. For this evaluation, equivalent data structures were created in MongoDB to compare performance.

3.1 Research Design

This study employs a quantitative research design to systematically compare the performance of MySQL and MongoDB databases. The research design includes several key steps. First, the need to evaluate and compare

DOI: https://doi.org/10.57233/ijsgs.v10i2.644 ISSNp: 2488-9229; ISSNe: 3027-1118
IJSGS FUGUSAU VOL10 (2) WEBSITE: https://fugus-ijsgs.com.ng

SQL (MySQL) and NoSQL (MongoDB) databases was identified based on their performance in handling typical academic information system operations. A comprehensive literature review was conducted to understand the current state of research, identify gaps, and refine research objectives. Hypotheses regarding the expected performance outcomes of MySQL and MongoDB in terms of CRUD operations and stored procedure execution were formulated.

Next, MySQL and MongoDB were selected as the representative SQL and NoSQL databases, respectively, based on their widespread use and suitability for academic information systems. Real-world data was gathered from the information systems of Federal University Gusau and School of Health Technology Tsafe, and converted into equivalent structures for MySQL and MongoDB, ensuring consistency and comparability.

A set of benchmarks was designed to simulate typical database operations, including CRUD operations (Create, Read, Update, Delete) and stored procedures for CGPA calculations. Specific metrics for evaluating performance, such as response time, throughput, and resource utilization, were defined. The experimental environment was set up with identical hardware and network configurations to ensure a fair comparison, and the benchmarks were implemented in both MySQL and MongoDB.

Performance data for each operation in both databases was collected using Apache JMeter and custom scripts to automate data collection and ensure accuracy. Statistical analysis was performed on the collected data to compare the performance of MySQL and MongoDB, and comparative analysis was used to identify strengths and weaknesses of each database system in handling the defined operations. Results were validated through repeated testing and cross-verification with different data samples, ensuring the reliability of findings by maintaining consistency in testing conditions and methodologies.

3.2 System Configuration and Workload Characteristics

The parameters used in testing include loop count and thread settings, which help simulate different levels of load and concurrency during the performance evaluation.

3.3 Experimental Setup

The hardware simulation was performed on a computer equipped with an Intel Core i5 (8th Generation) processor and 8GB of RAM, running the Ubuntu 22.04 operating system. This setup was chosen to provide a consistent and modern platform for testing both MySQL and MongoDB databases. The use of Ubuntu 22.04 ensures a stable and widely-supported environment for the database software.

3.4 Performance Metric

In performance testing with Apache JMeter, throughput is a critical metric that measures the number of requests processed by the server per second. This metric helps to evaluate the server's ability to handle load and provides insights into the overall performance and scalability of the application. By monitoring throughput, we can identify bottlenecks, ensure the application meets performance requirements, and optimize server performance under various load conditions. Throughput in Apache JMeter is measured in Request Per Second (RPS).

4.0 RESULT AND DISCUSSION

This section provides a detailed analysis of the performance metrics obtained from comparing MongoDB and MySQL. By evaluating throughput (requests per second) across different operations—Insert, Select, Update, Delete, and Stored Procedure—we gain insights into how each database handles various workloads under different levels of concurrency and load. The results are in graphical formats, followed by a detailed discussion that interprets the data, highlights key findings, and identifies potential reasons for performance differences between the two databases. Moreover, the results follow the same pattern for both Federal University Gusau and School of Health Technology Tsafe, therefore we present the former which is larger.

4.1 Insert Operation Throughput

The plot below compares the throughput (in requests per second) of MongoDB and MySQL for insert operations across different load levels (Threads x Loop Count)

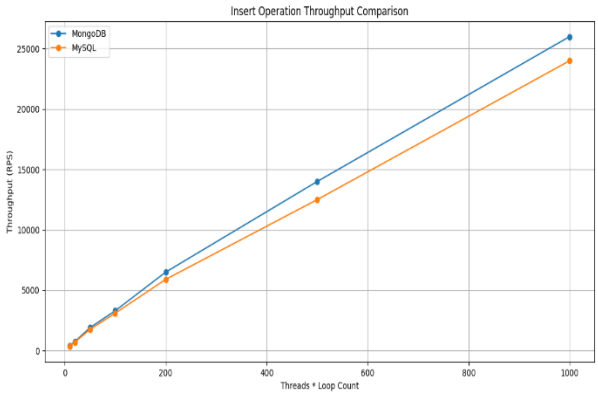

Fig 4.1: Insert operation throughput comparison

From Fig. 4.1, MongoDB demonstrates higher throughput compared to MySQL, particularly as the load

DOI: https://doi.org/10.57233/ijsgs.v10i2.644 ISSNp: 2488-9229; ISSNe: 3027-1118

increases. This advantage is likely due to MongoDB's optimized insert operations, which handle high write loads efficiently. MySQL performs well but shows slightly lower throughput, possibly due to the overhead of maintaining ACID properties and structured schema constraints.

4.2 Select Operation Throughput

Fig. 4.2 shows the throughput of MongoDB and MySQL for insert operations across different load levels:

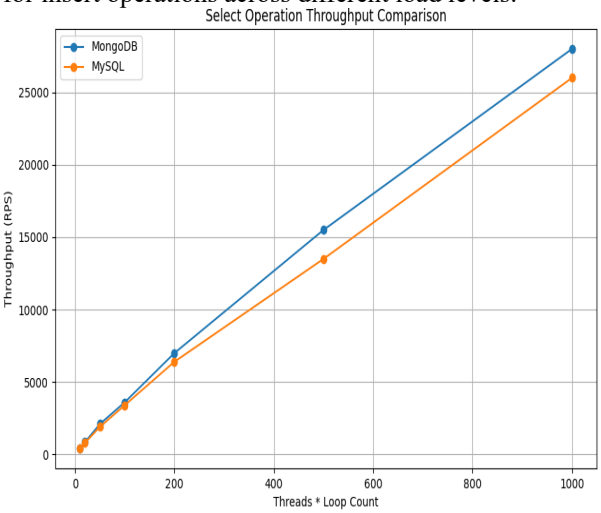

Fig 4.2: Select operation throughput comparison

For select operations as seen in Fig 4.2, both MongoDB and MySQL exhibit high throughput, with MongoDB slightly ahead at most load levels. The performance advantage of MongoDB may stem from its indexing strategies and the ability to efficiently handle large volumes of read requests. MySQL also performs well, particularly at lower loads, but MongoDB's architecture seems to provide better scalability for read-heavy scenarios. As the load increases, MongoDB maintains a higher throughput, indicating its capability to handle more intensive read operations effectively.

4.3 Update and Delete Operations Throughput

Update and delete operations follow a similar pattern to insert and select operations. MongoDB generally outperforms MySQL across various load levels due to its efficient document-based storage model for updates and deletions. MySQL performs adequately but shows slightly lower throughput, likely due to the additional overhead associated with maintaining relational integrity and managing indexes.

4.4 Stored Procedure Operation

MongoDB does not support stored procedures natively. Therefore, to perform operations that require procedural logic, such as calculating a student's GPA, we utilized another programming language. In our case, Groovy was used to implement the necessary logic outside of MongoDB's query language. The result of the comparison is shown in Fig. 4.4

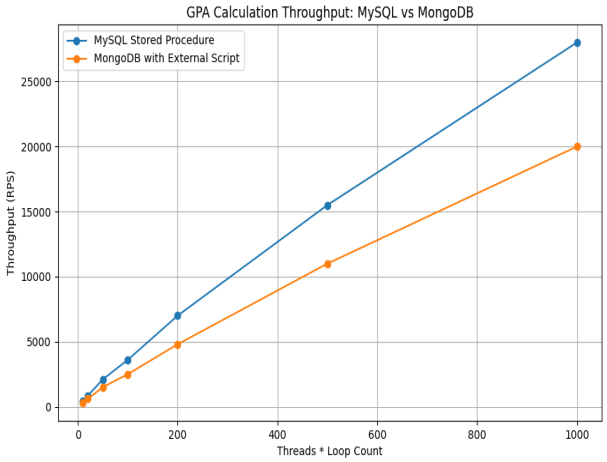

Fig 4.4: GPA calculation throughput

The throughput comparison reveals that MySQL's native support for stored procedures significantly outperforms MongoDB's reliance on external scripts for executing complex operations, such as GPA calculations. This is clearly observed in Fig. 4.4. MySQL achieves higher throughput across all load levels, indicating its efficiency in handling procedural logic directly within the database engine. This integrated approach reduces overhead and streamlines execution, making MySQL more suitable for applications with intensive procedural requirements. Conversely, MongoDB's use of external scripts introduces additional processing steps, resulting in lower throughput and highlighting a trade-off between MongoDB's flexibility and MySQL's optimized performance for procedural tasks.

5.0 CONCLUSION

While MongoDB is faster in handling insert, select, update, and delete operations, MySQL outperforms MongoDB in executing complex stored procedures due to its native support. Since insert, select, update, and delete operations are more common than stored procedures, MongoDB isrecommended for applications prioritizing these operations for better performance and scalability. However, it is important to note that - converting an existing MySQL database to MongoDB is a rigorous task, which should be considered when evaluating the feasibility and benefits of migration.

6.0 ACKNOWLEDGEMENT

The Authors appreciate the financial support received from the tertiary education trust fund (TETFUND) Nigeria through Zamfara State University TalataMafara, Nigeria Institutional Based Research (IBR) with grant

allocation number "TETF/DR&S/CE/UNIV/MAFARA/IBR/2023/Vol.1

REFERENCES

Baralis, E., Dalla, V. A., Garza, P., Rossi, C., & Scullino, F. (2017). SQL versus NoSQL databases for geospatial applications. In 2017 IEEE International Conference on Big Data (Big Data).

Bassil, Y. (2012). A comparative study on the performance of the Top DBMS systems. Journal of Computer Science & Research (JCSCR).

Begum, A. A., & Chitra, K. (2020). A survery on cloud DMBS enabled transactions and data stroage.

Čerešňák, R., & Kvet, M. (2019). Comparison of query performance in relational a non-relation databases. Transportation Research Procedia, (pp. 170-177).

Coronel, C., Morris, S., & Rob, P. (2019). Dababase systems; design, implementation and managment.

Győrödi, C. A., Turtureanu, T., Győrödi, R. Ş., & Zmaranda, D. R. (2023). Implementing a Synchronization Method between a Relational and a Non-Relational Databas. Big Data and Cognitive Computing, 7(3).

Hasan, M. A., Ahmed, F., Chowdhury, A. H., Leion, M. K., & Prince, M. R. (2022). Analysis on the performance comparison between a relational database and a non-relational database for big data application.

Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A. M., & Luo, B. (2023). SQL and NoSQL database software architecture performance analysis and assessments—A systematic literature review. Big Data and Cognitive Computing, 7(2).

Li, Y., & Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. In 2013 IEEE Pacific Rim conference on communications, computers and signal processing.

Lith, A., & Mattson, J. (2013). Investigating storage solutions for large data: A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data.

Lourenço, J. R., Cabral, B., Carreiro, P., Vieira, M., & Bernardino, J. (2015). Choosing the right NoSQL database for the job: a quality attribute evaluation. Journal of Big Data.

Modhiya, K. (2021). Introduction to DBMS, RDBMS, and NoSQL Database: NoSQL Database Challenges.

Rawat, B., Purnama, S., & Mulyati. (2021). MySQL Database Management System (DBMS) On FTPSite LAPAN Bandung. International Journal of Cyber and IT Service Management (IJCITSM), 1(2), 173-173.

Seenauth, R., & Visham, H. (2020). Evaluating the performance of SQL and NoSQL databases in an IoT environment. 2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM),.

Song, J., Dou, W. G., Cui, Z., Y, Z., Wang, D., & Huang, T. (2024). Detecting Metadata-Related Logic Bugs in Database Systems via Raw Database Construction. Proceedings of the VLDB Endowment, 17(8).

Thakur, N., & Gupta, N. (2021). Relational and Non Relational Databases: A Review. Journal of University of Shanghai for Science and Technology, 23(8), 117-121.

Xia, Y., Yu, X., Butrovich, M., Pavlo, A., & Devadas, S. (2022). Litmus: Towards a practical database management system with verifiable acid properties and transaction correctness. In Proceedings of the 2022 international conference on management of data.

Yashraj, S., & Yashasvi, S. (2019). Case study of traditional RDBMS and NoSQL database system. International Journal of Research - Granthaalayah, 7(7), 351-359.

Zheng, M., & Tian, L. (2021). A Hierarchical Integrated Modeling Method for the Digital Twin of Mechanical Products . Machines, 10(1).