Technologique | 2026, Vol. 8, No.1, 119 – 133 | ISSN Online: 3028-1415 | ISSN Print: 3028-1407
DOI: https://doi.org/10.62718/vmca.tech-gjtdsi.8.1.SC-0626-009

OPEN ACCESS

A Comparative Performance Analysis of MySQL and MongoDB in Laravel-Based Web Applications

Article History:

Initial submission:	19 June 2026
First decision:	28 June 2026
Revision received:	24 July 2026
Accepted for publication:	30 July 2026
Online release:	07 August 2026

Severino M. Bedis Jr.[1,2]
Ma. Leonila C. Amata[1,2], MEM, LPT

[1]College Instructor, Polytechnic University of the Philippines, Sta. Mesa, Manila, Philippines
[2]College Instructor, Mandaluyong City College of Science and Technology, Mandaluyong City, Philippines

Abstract

The increasing demand for efficient data management in modern web applications has highlighted the importance of selecting an appropriate database management system based on application requirements and workload characteristics. Although MySQL and MongoDB have been widely adopted in web development, limited empirical studies have evaluated their performance when integrated with the Laravel framework. This study compared MySQL and MongoDB in terms of installation and configuration requirements, database schema design characteristics, CRUD (Create, Read, Update, and Delete) performance, query execution performance, and scalability characteristics across varying dataset sizes. An exploratory quantitative experimental research design was employed using a Laravel-based prototype web application and identical datasets containing 1,000, 10,000, 50,000, and 100,000 records. Performance was evaluated by measuring the execution times of CRUD operations and query processing tasks using standardized experimental procedures. The findings revealed that MongoDB demonstrated superior average performance in data update (2,592.52 ms) and data deletion (1,043.74 ms) operations compared with MySQL's 6,107.24 ms and 2,135.98 ms, respectively. Conversely, MySQL achieved better average performance in data insertion and data retrieval workloads, recording an average insert time of 5,560.26 ms and an average read time of 499.46 ms, compared with MongoDB's 21,817.48 ms and 572.99 ms, respectively. Furthermore, MongoDB demonstrated significant performance advantages in simple, filtered, and complex query processing, reducing complex query execution times by 87.91% over MySQL, whereas MySQL maintained a 9.95% advantage in analytical aggregation queries. The results indicate that MySQL offers distinct advantages in native Laravel integration, data integrity, and relational data management, whereas MongoDB demonstrates strengths in schema flexibility, filtered data retrieval, and modification-heavy workloads. Scalability characteristics were inferred from execution-time trends observed across increasing dataset sizes. Overall, neither database management system was universally superior across all application scenarios. The findings provide framework-specific empirical evidence to assist software developers and system designers in selecting appropriate database technology for Laravel-based web applications.

Keywords: MySQL, MongoDB, Laravel, Relational Database, NoSQL Database, CRUD Performance, Query Performance, Scalability

INTRODUCTION

The relational database model was first introduced by Codd (1970) in his seminal paper, *A Relational Model of Data for Large Shared Data Banks*. Since then, relational database management systems (RDBMS) have become the predominant paradigm for data storage and management across numerous computing applications. However, modern computing environments have entered an era of unprecedented data growth. The International Data Corporation (IDC) Digital Universe study reported that the volume of digital data worldwide is growing exponentially, approximately doubling every two years (Gantz & Reinsel, 2012).

Large-scale web platforms continuously generate massive volumes of structured, semi-structured, and unstructured data, creating significant challenges in data storage, retrieval, scalability, and performance. Consequently, database technologies continue to evolve to

address increasingly complex computational and architectural requirements.

Database systems have undergone substantial developments since their inception in the 1960s. Early database models employed hierarchical and network-based structures that relied on pointer navigation mechanisms before the emergence of the relational model. Subsequent advancements introduced object-oriented database systems and, more recently, NoSQL databases designed to support large-scale distributed applications. Today, relational and NoSQL databases represent two of the most widely adopted data storage paradigms. While relational databases emphasize structured schemas, consistency, and transactional integrity, NoSQL databases prioritize scalability, flexibility, and efficient handling of large volumes of semi-structured and unstructured data.

MySQL is one of the most widely used open-source relational database management systems developed and maintained by Oracle Corporation. It organizes information into related tables and utilizes Structured Query Language (SQL) to define, manipulate, and retrieve data. MySQL employs predefined schemas and integrity constraints to ensure consistency and minimize data redundancy through normalization techniques. Relationships among data entities are established using keys and reconstructed through SQL JOIN operations when complex queries are performed. These characteristics make MySQL particularly suitable for applications requiring strong consistency, transactional support, and structured data management.

Conversely, NoSQL databases provide non-relational approaches to data storage and retrieval that are designed to accommodate the scalability requirements of modern distributed systems. Rather than enforcing rigid schemas, NoSQL databases offer flexible data models that efficiently support evolving application requirements and large-scale datasets. Among the various NoSQL implementations, MongoDB has gained widespread adoption because of its document-oriented architecture, which stores data using JSON-like documents that can dynamically accommodate varying structures without requiring costly schema modifications. This flexibility makes MongoDB particularly suitable for web applications that frequently process semi-structured data and experience rapid changes in system requirements. Both relational and NoSQL databases offer distinct architectural advantages depending on application requirements. Relational databases excel in maintaining data integrity and supporting complex transactional operations, whereas NoSQL databases provide improved horizontal scalability and schema flexibility for distributed environments. Consequently, selecting appropriate database technology requires careful consideration of performance characteristics, scalability requirements, and the underlying software architecture of the target application.

Although numerous studies have compared relational and NoSQL databases, existing literature predominantly focuses on standalone database performance evaluations. There remains a limited body of empirical research examining how these database technologies behave when integrated within modern web development frameworks such as Laravel. The use of object-relational mapping (ORM) layers and database drivers introduces additional computational overhead that may significantly influence query execution performance and scalability characteristics, making raw database benchmarks insufficient for predicting real-world application behavior.

To address this gap, the present study conducts a comparative performance evaluation of MySQL and MongoDB within a Laravel-based web application environment. The study benchmarks standardized Create, Read, Update, and Delete (CRUD) operations across dataset sizes of 1,000, 10,000, 50,000, and 100,000 records, including simple, filtered, complex, and aggregation queries. By examining execution-time trends across varying dataset sizes, the study evaluates the scalability characteristics

of both database technologies in practical application scenarios. The findings contribute framework-specific empirical evidence that may assist software engineers and system developers in selecting appropriate database architectures during the design and development of Laravel-driven web systems.

LITERATURE REVIEW

Relational and NoSQL databases have been widely studied because of the increasing demand for efficient data management in modern web applications. Relational Database Management Systems (RDBMS), such as MySQL, are recognized for their structured schema design, strong consistency guarantees, and support for complex transactional operations. Conversely, NoSQL databases such as MongoDB provide flexible schema designs, horizontal scalability, and efficient handling of semi-structured and unstructured data. Consequently, database selection has become highly dependent on application requirements, workload characteristics, and performance demands.

Several studies have demonstrated the scalability advantages of MongoDB in modern computing environments. Filip and Cegan (2020) compared MySQL and MongoDB with respect to their performance characteristics and reported that MongoDB offers advantages in scalability and schema flexibility, whereas MySQL remains suitable for applications requiring structured data relationships and transactional consistency. Similarly, Moreno et al. (2024) found that MongoDB achieved faster response times across several experimental scenarios involving dynamic content management datasets while providing improved flexibility for handling evolving data structures. However, their findings also showed that MySQL performed better in workloads requiring structured data processing and relational operations. Likewise, Pandey (2020) employed the Yahoo! Cloud Serving Benchmark (YCSB) framework to evaluate cloud-based implementations of MySQL and MongoDB and reported that MongoDB demonstrated favorable performance in write-intensive workloads and scalability-related operations, whereas MySQL exhibited advantages in transactional consistency and structured data management. These studies collectively suggest that MongoDB is well suited for applications prioritizing scalability and schema flexibility, whereas MySQL remains advantageous for maintaining data consistency and supporting complex relational queries.

Studies have likewise emphasized that database performance is highly dependent on deployment environments and workload characteristics. Ljungdahl (2023) evaluated MySQL and MongoDB in cloud-based infrastructures and concluded that neither database technology is universally superior. Instead, database selection should be determined by factors such as application requirements, scalability needs, and operational workloads. Similarly, Mendoza-Tello and Villacís-Ramón (2022) reported that database performance varies considerably across benchmark workloads and that neither MySQL nor MongoDB consistently outperformed the other across all experimental scenarios. Their findings reinforced the importance of selecting database technologies according to workload requirements rather than relying solely on generalized performance claims. The findings of these studies demonstrated that both database technologies exhibit distinct strengths when deployed under different computing conditions, particularly within scalable cloud environments.

Further studies have highlighted the architectural differences between relational and NoSQL database technologies. Matallah et al. (2021) reported that MongoDB offers greater schema flexibility and scalability characteristics for applications handling large volumes of heterogeneous data, whereas MySQL demonstrated advantages in maintaining relational integrity and supporting complex transactional operations. Similarly, Győrödi et al. (2022) found that MongoDB performs effectively in managing flexible document-oriented data structures while

MySQL remains advantageous for applications requiring structured data relationships and consistency guarantees. These findings further support the view that relational and NoSQL databases address different application requirements rather than serving as direct substitutes for one another.

Beyond standalone database performance evaluations, recent research has highlighted the importance of assessing database technologies within actual software development frameworks. Wodyk and Skublewska-Paszkowska (2020) investigated the performance of relational databases integrated with a Laravel-based web application and demonstrated that query execution time and framework integration significantly influence overall system performance. Their findings suggest that database benchmarking conducted outside application frameworks may not accurately reflect real-world performance because additional overhead is introduced by object-relational mapping (ORM) layers, database drivers, and framework-level abstractions.

Collectively, previous studies indicate that MySQL provides advantages in ACID-compliant transactions, complex relational operations, and data integrity, whereas MongoDB excels in schema flexibility, horizontal scalability, and rapid application development. Although substantial literature exists comparing relational and NoSQL database technologies, most studies focus either on standalone database performance or framework-specific evaluations limited to relational database systems. Furthermore, existing research rarely examines how MySQL and MongoDB behave when subjected to identical workloads within Laravel-based web applications across varying dataset sizes.

This study addresses these limitations by conducting a comparative performance analysis of MySQL and MongoDB integrated within the Laravel framework. Specifically, it evaluates both database technologies using standardized Create, Read, Update, and Delete (CRUD) operations and query execution benchmarks across multiple dataset sizes. By examining framework-level performance characteristics, the study contributes empirical evidence that may assist software engineers and system developers in selecting appropriate database architectures for Laravel-driven web applications.

Statement of the Problem. This study is an exploratory quantitative investigation that aims to compare the performance characteristics of MySQL and MongoDB when integrated with the Laravel framework. Rather than testing formal hypotheses, the study employs standardized benchmarking procedures to answer descriptive research questions concerning database performance, scalability, schema flexibility, and suitability for Laravel-based web applications. The findings are intended to provide empirical evidence that may assist software engineers and system developers in selecting an appropriate database technology based on specific application requirements.

Specifically, this study seeks to answer the following research questions:

1. What are the differences between MySQL and MongoDB in terms of:
   - 1.1 Installation and configuration;
   - 1.2 Database design and schema structure;
   - 1.3 Data insertion performance;
   - 1.4 Data retrieval performance;
   - 1.5 Data update performance;
   - 1.6 Data deletion performance; and
   - 1.7 Query execution time across varying dataset sizes?

2. Which database management system demonstrates better performance in Laravel-based web applications based on CRUD operations and query execution benchmarks?

3. What are the respective advantages and limitations of MySQL and MongoDB in Laravel-based web application development?

4. Based on the experimental results, which database technology is more suitable for specific application requirements in terms of scalability, schema flexibility, and data integrity?

METHODS

This study employed an experimental research design to compare the performance characteristics of MySQL and MongoDB when integrated with the Laravel framework. A prototype web application was developed using Laravel to perform identical database operations on both database management systems. The experiment utilized standardized benchmarking procedures to evaluate CRUD (Create, Read, Update, and Delete) operations and query execution performance across multiple dataset sizes.

Experimental Environment. Both databases were deployed using identical hardware and software configurations to ensure consistency and fairness during performance evaluation. Table 1 presents the specifications of the experimental environment.

Table 1
Hardware and Software Specifications

Component	Specification
Processor	Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
Memory (RAM)	16 GB DDR4
Storage	512 GB SSD
Operating System	Windows 11 Pro (64-bit)
Framework	Laravel 12
Programming Language	PHP 8.4
Relational Database	MySQL 9.4
NoSQL Database	MongoDB 8.0
Web Server	Apache (XAMPP)
Development Tool	Visual Studio Code
Benchmarking Tool	Laravel's Built-in Timing and Logging Mechanisms

Dataset Preparation. Four datasets (Table 2) containing varying numbers of records were generated and stored in both MySQL and MongoDB to evaluate performance under different workloads. The datasets were designed with identical attributes to ensure comparability between the two database systems.

Table 2
Dataset Size

Dataset	Number of Records
Dataset A	1,000
Dataset B	10,000
Dataset C	50,000
Dataset D	100,000

Each record consisted of identical attributes represented in both database systems. The dataset included the following fields (Table 3):

Table 3
Dataset Attributes

Field Name	MySQL Data Type	MongoDB Data Type
id	BIGINT	ObjectId
first_name	VARCHAR(100)	String
last_name	VARCHAR(100)	String
email	VARCHAR(255)	String
contact_number	VARCHAR(50)	String
address	TEXT	String
status	VARCHAR(50)	String
created_at	TIMESTAMP	Date
updated_at	TIMESTAMP	Date

In MySQL, the data were stored in relational tables using predefined schemas and primary key constraints. In MongoDB, the same attributes were represented as JSON-like documents within a collection. The schema design-maintained attribute equivalence across both databases to ensure a fair comparison.

Indexing Strategy. To minimize experimental bias, equivalent indexing strategies were implemented in both database systems as shown in Table 4. Primary keys were automatically indexed in MySQL, whereas MongoDB utilized the default ObjectId index. Additional indexes were created for frequently queried attributes. The same indexed fields were utilized when performing filtered and complex queries to maintain consistency during benchmarking.

Table 4

Field Name	MySQL	MongoDB
id	Primary Key Index	Default ObjectId Index
email	Indexed	Indexed
status	Indexed	Indexed
created_at	Indexed	Indexed

Evaluation Criteria. The study employed both qualitative and quantitative evaluation criteria (Table 5).

Table 5
Evaluation Criteria

Category	Evaluation Criteria
Qualitative	Installation and Configuration Complexity
Qualitative	Database Schema Design and Flexibility
Qualitative	Framework Integration Considerations
Quantitative	Data Insertion Time
Quantitative	Data Retrieval Time
Quantitative	Data Update Time
Quantitative	Data Deletion Time
Quantitative	Query Execution Time

The average execution time for each operation was recorded and compared across the four dataset sizes (1,000, 10,000, 50,000, and 100,000 records).

All execution times were measured in milliseconds(ms) using Laravel's built-in timing and logging mechanisms.

Experimental Procedure. Identical CRUD operations were performed for both MySQL and MongoDB across all dataset sizes. To improve the reliability of the experimental results, each database operation was executed thirty (30) times for every dataset size.

The arithmetic mean and standard deviation were subsequently calculated for each operation.

The following procedures were performed:
1. Data insertion of records into the database.
2. Data retrieval using predefined queries.
3. Data updates of existing records.
4. Data deletion operations.
5. Execution of filtered queries utilizing indexed attributes.
6. Execution of complex queries involving multiple filtering conditions.
7. Execution of aggregation queries supported by both database management systems.

The query execution benchmarks consisted of the following query types (Table 6):

Table 6
Query Categories

Query Type	Description
Simple Query	Retrieves records without filtering conditions
Filtered Query	Retrieves records using indexed attributes
Complex Query	Retrieves records using multiple filtering conditions

Data Collection Procedure. The experimental procedures for each dataset were conducted as follows:

1. Populate the database with the required number of records.
2. Execute identical CRUD operations for both database management systems.
3. Perform thirty (30) repetitions for every operation.
4. Record the execution time of each trial in milliseconds.
5. Compute the arithmetic mean and standard deviation of the recorded measurements.
6. Repeat the procedure across all dataset sizes.
7. Compare the collected measurements between MySQL and MongoDB.

To ensure fairness during benchmarking, all experiments were conducted using identical datasets, application logic, indexing strategies, and hardware configurations.

Data Analysis. The collected data were analyzed using descriptive statistical techniques to compare the performance of MySQL and MongoDB when integrated with the Laravel framework. The following statistical measures were utilized:

- Mean execution time (milliseconds) to determine the average performance of each CRUD operation and query type;

- Standard deviation to assess the consistency and variability of the recorded execution times across repeated experimental runs;
- Percentage differences between MySQL and MongoDB to quantify their relative performance across the evaluated metrics; and
- Execution-time trends across increasing dataset sizes (1,000, 10,000, 50,000, and 100,000 records) to evaluate their scalability characteristics under varying workloads.

Scalability characteristics were inferred from the observed execution-time patterns as the number of records increased across the experimental datasets. Qualitative findings concerning installation and configuration complexity and database schema design were analyzed separately from the quantitative performance metrics to maintain consistency between descriptive and experimental evaluations.

The results were presented using tables to facilitate the comparison and interpretation of the performance metrics across varying dataset sizes. The tabular presentations provided insight into the execution-time trends, scalability characteristics, and operational behaviors of both MySQL and MongoDB when integrated with the Laravel framework.

RESULTS AND DISCUSSION

Installation and Configuration. Table 7 presents the installation and configuration requirements of MySQL and MongoDB in a Laravel environment.

Table 7
Installation and Configuration Comparison

Criteria	MySQL	MongoDB
Native Laravel Support	Yes	No
Additional Package Required	No	Yes
Configuration Complexity	Low	Moderate
Setup Time (minutes)	10	20

Table 7 presents the installation and configuration requirements of MySQL and MongoDB within the Laravel framework.

The findings indicate that MySQL provides a simpler installation and configuration process because it is natively supported by Laravel. Consequently, no additional packages are required for its integration, resulting in lower configuration complexity and shorter implementation time. In contrast, MongoDB requires the installation of additional drivers and Laravel packages to establish compatibility with the framework, thereby increasing configuration complexity and setup requirements.

It should be noted that the setup times presented in Table 7 are approximate observations obtained during the experimental implementation process rather than formal performance measurements. Therefore, these values are intended to describe the relative implementation complexity of both database management systems rather than establish statistically significant performance differences.

These findings suggest that MySQL offers advantages for developers seeking rapid deployment of relational database applications within Laravel. Conversely, MongoDB provides greater flexibility for applications requiring document-oriented data management but necessitates additional configuration procedures during implementation.

Database Schema Design. Table 8 presents the database schema design characteristics of MySQL and MongoDB. The results highlight the fundamental architectural differences between relational and NoSQL database systems. MySQL employs a fixed schema structure wherein tables, columns, and relationships must be explicitly defined before data insertion.

Table 8
Database Design Comparison

Feature	MySQL	MongoDB
Schema Type	Fixed	Flexible
Table Structure	Required	Not Required
Relationships	Strong	Limited
Scalability	Moderate	High

This approach promotes data consistency, referential integrity, and structured data management through relational constraints and normalization techniques. Consequently, MySQL is particularly suitable for applications requiring transactional consistency and well-defined relationships among data entities.

Conversely, MongoDB utilizes a flexible schema design that permits documents within the same collection to contain varying structures. This flexibility facilitates rapid schema modifications without requiring extensive database migrations, making MongoDB advantageous for applications that process dynamic, semi-structured, or evolving datasets. Although MongoDB offers greater scheme flexibility and scalability characteristics, MySQL provides stronger support for relational operation and data integrity requirements. Therefore, the suitability of each database technology ultimately depends upon the structural and function requirements of the target application.

CRUD Performance. Tables 9 to 13 present the CRUD performance evaluation results of MySQL and MongoDB using dataset sizes ranging from 1,000 to 100,000 records. The reported values represent the mean execution times obtained from repeated experimental runs conducted under identical testing conditions. The statistical measures, including standard deviation, variance, and 95% confidence intervals, were used to assess the consistency and reliability of the recorded execution times.

Table 9
CRUD Performance Statistics (1,000 Records)

Operation	Database	Mean (ms)	SD	Variance	95% CI (ms)
Insert	MySQL	163.70	11.87	140.87	159.46- 167.95
	MongoDB	649.32	179.46	32,206.31	585.10 - 713.54
Read	MySQL	12.77	1.61	2.59	12.20 - 13.35
	MongoDB	15.94	4.84	23.46	14.21 - 17.68
Update	MySQL	149.81	14.13	199.73	144.76 - 154.87
	MongoDB	84.38	25.04	626.96	75.42 - 93.34
Delete	MySQL	51.51	4.27	18.24	49.98 - 53.04
	MongoDB	29.79	6.90	47.57	27.32 - 32.26

Table 10
CRUD Performance Statistics (10,000 Records)

Operation	Database	Mean (ms)	SD	Variance	95% CI (ms)
Insert	MySQL	7,315.39	618.43	382,456.91	7,094.09 - 7,536.69
	MongoDB	29,431.18	4,414.53	19,488,087.17	27,851.46 - 31,010.90
Read	MySQL	654.47	109.29	11,945.39	615.36 - 693.59
	MongoDB	730.78	89.15	7,948.57	698.88 - 762.68
Update	MySQL	7,790.34	708.84	502,456.00	7,536.69 - 8,044.00
	MongoDB	3,560.12	584.43	341,554.89	3,350.98 - 3,769.25
Delete	MySQL	2,764.66	275.03	75,639.89	2,666.24 - 2,863.07
	MongoDB	1,516.66	394.40	155,547.61	1,375.53 - 1,657.80

Table 11
CRUD Performance Statistics – 50000 records

Operation	Database	Mean (ms)	SD	Variance	95% CI (ms)
Insert	MySQL	7,315.39	618.43	382,456.91	7,094.09 - 7,536.69
	MongoDB	29,431.18	4,414.53	19,488,087.17	27,851.46 - 31,010.90
Read	MySQL	654.47	109.29	11,945.39	615.36 - 693.59
	MongoDB	730.78	89.15	7,948.57	698.88 - 762.68
Update	MySQL	7,790.34	708.84	502,456.00	7,536.69 - 8,044.00
	MongoDB	3,560.12	584.43	341,554.89	3,350.98 - 3,769.25
Delete	MySQL	2,764.66	275.03	75,639.89	2,666.24 - 2,863.07
	MongoDB	1,516.66	394.40	155,547.61	1,375.53 - 1,657.80

Table 12
CRUD Performance Statistics - 100000 records

Operation	Database	Mean (ms)	SD	95% CI (ms)
Insert	MySQL	13,301.85	3,822.60	11,933.94 – 14,669.75
	MongoDB	51,272.84	14,747.73	45,995.44 – 56,550.25
Read	MySQL	1,203.93	383.76	1,066.60 – 1,341.25
	MongoDB	1,400.11	433.63	1,244.94 – 1,555.28
Update	MySQL	14,963.75	6,646.38	12,585.37 – 17,342.12
	MongoDB	5,941.05	2,129.27	5,179.10 – 6,703.00
Delete	MySQL	5,221.48	2,118.54	4,463.37 – 5,979.59
	MongoDB	2,347.42	1,067.70	1,965.35 – 2,729.49

Table 13
Overall CRUD Performance Comparison Between MySQL and MongoDB Across Dataset Sizes

Operation	MySQL Average Mean (ms)	MongoDB Average Mean (ms)	Percentage Difference
Insert	5,560.26	21,817.48	292.37%
Read	499.46	572.99	14.72%
Update	6,107.24	2,592.52	57.54%
Delete	2,135.98	1,043.74	51.14%

The experimental results indicate that database performance varied depending on the type of CRUD operation and dataset size. For datasets containing 1,000 records, MongoDB achieved faster execution times for update and delete operations, while MySQL demonstrated better performance in insert and read operations. Specifically, MySQL recorded lower execution times for insert (163.70 ms) and read (12.77 ms), whereas MongoDB performed better in update (84.38 ms) and delete (29.79 ms) operations.

At 10,000 records, similar performance trends were observed. MySQL maintained better performance in insert and read operations, achieving execution times of 1,460.09 ms and 126.68ms, respectively. Meanwhile, MongoDB demonstrated faster update and delete operations, recording 784.51 ms and 281.09 ms, respectively. These results indicate that the performance advantage of each database system depends on the specific workload being executed.

As the dataset size increased to 50,000 records, both database systems experienced increased execution times across all CRUD operations. MySQL continued to outperform MongoDB in insert operations, while MongoDB maintained advantages in update and delete operations. For read operations, MySQL recorded a lower execution time of 654.47 ms compared with MongoDB's 730.78ms, indicating better retrieval performance under this workload size.

At 100,000 records, the performance differences became more evident. MySQL achieved faster insert and read operations, with execution times of 13,301.85 ms and 1,203.93 ms, respectively. In contrast, MongoDB demonstrated better performance in update and delete operations, recording 5,941.05 ms and 2,347.42 ms compared with MySQL's 14,963.75 ms and 5,221.48 ms. These results suggest that MongoDB handled large-scale modification operations more efficiently, while MySQL remained competitive in data insertion and retrieval tasks.

The variations observed in execution times demonstrate that scalability is influenced by database architecture, storage mechanisms, indexing strategies, query optimization, and transaction processing approaches. MySQL relies on a relational model with ACID-compliant transaction management and structured schemas, which provide strong data consistency and integrity but may introduce additional processing overhead during complex modification operations. On the other hand, MongoDB's document-oriented architecture provides flexibility in handling unstructured or semi-structured data and may reduce overhead during update and delete operations.

Table 13 summarizes the overall CRUD performance comparison between MySQL and MongoDB across all evaluated dataset sizes. The results show that MySQL achieved a lower average execution time for insert operations, recording 5,560.26 ms compared with MongoDB's 21,817.48 ms. This indicates that MySQL provided better performance for the tested insertion workload. For read operations, MySQL also demonstrated slightly better average performance, with an execution time of 499.46 ms compared with MongoDB's 572.99 ms. Conversely, MongoDB demonstrated superior average performance for update and delete operations. MongoDB recorded average execution times of 2,592.52 ms for update operations and 1,043.74 ms for delete operations, which were lower than MySQL's 6,107.24 ms and 2,135.98 ms, respectively. Based on the percentage comparison, MongoDB was approximately 57.54% faster in update operations and 51.14% faster in delete operations, while MySQL was approximately 292.37% faster in insert operations and 14.72% faster in read operations.

The execution-time trends across increasing dataset sizes indicate that both database management systems experienced performance degradation as the workload increased. However, the degree of performance degradation differed depending on the type of operation. MongoDB showed better scalability for update and delete operations, whereas

MySQL maintained advantages in insertion and retrieval workloads. These findings emphasize that database selection should be based on application requirements and workload characteristics rather than relying solely on overall execution time.

The calculated standard deviations and confidence intervals further indicate that the experimental measurements were generally consistent across repeated runs. Although some operations exhibited higher variability at larger dataset sizes, particularly write operations, the observed trends remained consistent and support the reliability of the performance comparison.

Overall, the findings demonstrate that MySQL and MongoDB each provide advantages under different workload conditions. MySQL is more suitable for applications requiring efficient data insertion, structured relational data management, and consistent query performance. Meanwhile, MongoDB is advantageous for applications involving frequent data modification operations and flexible document-based data structures, particularly when handling larger datasets.

Query Performance. Tables 14 to 18 present the query performance evaluation of MySQL and MongoDB using simple, filtered, complex, and aggregation queries across dataset sizes ranging from 1,000 to 100,000 records. The reported execution times represent the average results obtained from repeated query executions under identical testing conditions. The statistical measurements, including standard deviation, variance, and 95% confidence intervals, were used to evaluate the consistency and reliability of the recorded performance values.

The results indicate that query performance varies depending on the database system and the type of workload being executed. For simple queries, MongoDB generally achieved better performance compared with MySQL. Although MySQL demonstrated competitive execution times at smaller dataset sizes, MongoDB provided lower average execution time across the evaluated datasets, with an overall mean of 396.65 ms compared with MySQL's 525.98 ms.

Table 14

Query Performance Statistics (1,000 Records)

Query Type	Database	Mean (ms)	SD	Variance	95% CI (ms)
Simple Query	MySQL	14.75	6.17	38.05	12.54 - 16.96
	MongoDB	11.96	3.36	11.29	10.76 - 13.16
Filter Query	MySQL	16.68	2.69	7.25	15.71 - 17.64
	MongoDB	169.66	44.69	1,997.03	153.67 - 185.65
Complex Query	MySQL	1,092.05	128.53	16,520.88	1,046.06 - 1,138.05
	MongoDB	166.52	16.04	257.27	160.78 - 172.26
Aggregation Query	MySQL	1,256.66	141.76	20,096.58	1,205.93 - 1,307.39
	MongoDB	1,004.41	109.09	11,900.87	965.37 - 1,043.45

Table 15

Query Performance Statistics – 10000 records

Query Type	Database	Mean (ms)	SD	Variance	95% CI (ms)
Simple Query	MySQL	129.17	15.66	245.35	123.57 - 134.78
	MongoDB	109.33	36.28	1,316.10	96.35 - 122.31
Filter Query	MySQL	133.26	26.92	724.80	123.63 - 142.90
	MongoDB	169.95	62.74	3,935.80	147.50 - 192.39
Complex Query	MySQL	205.13	17.39	302.52	198.90 - 211.35
	MongoDB	180.08	47.38	2,245.05	163.13 - 197.04
Aggregation Query	MySQL	24.59	2.88	8.27	23.56 - 25.62
	MongoDB	1,052.92	115.04	13,233.75	1,011.76 - 1,094.09

Table 16

Query Performance Statistics – 50000 records

Query Type	Database	Mean (ms)	SD	Variance	95% CI (ms)
Simple Query	MySQL	659.72	102.52	10,509.65	623.04 - 696.41
	MongoDB	473.96	65.05	4,231.83	450.68 - 497.24
Filter Query	MySQL	651.00	68.46	4,687.35	626.50 - 675.50
	MongoDB	147.75	17.03	290.13	141.66 - 153.85
Complex Query	MySQL	1,759.00	234.99	55,221.57	1,674.91 - 1,843.09
	MongoDB	160.94	25.27	638.41	151.90 - 169.98
Aggregation Query	MySQL	1,122.48	96.40	9,293.33	1,087.98 - 1,156.97
	MongoDB	980.91	114.33	13,072.22	939.99 - 1,021.82

Table 17
Query Performance Statistics – 100000 records

Query Type	Database	Mean (ms)	SD	Variance	95% CI (ms)
Simple Query	MySQL	1,300.28	154.29	23,804.21	1,245.07 – 1,355.49
	MongoDB	991.33	96.61	9,334.15	956.76 – 1,025.90
Filter Query	MySQL	1,314.75	116.05	13,467.23	1,273.22 – 1,356.28
	MongoDB	141.51	8.40	70.56	138.50 – 144.51
Complex Query	MySQL	2,424.96	389.08	151,387.03	2,285.73 – 2,564.20
	MongoDB	155.37	8.91	79.43	152.18 – 158.56
Aggregation Query	MySQL	1,196.15	235.79	55,598.53	1,111.77 – 1,280.52
	MongoDB	920.72	33.72	1,137.09	908.65 – 932.79

Table 18
Overall Query Performance Comparison Between MySQL and MongoDB Across Dataset Sizes

Query Type	MySQL Average Mean (ms)	MongoDB Average Mean (ms)	Percentage Difference
Simple Query	525.98	396.65	24.59%
Filter Query	528.24	157.22	70.24%
Complex Query	1,370.32	165.73	87.91%
Aggregation Query	899.97	989.52	9.95%

For filtered queries, MongoDB consistently demonstrated superior performance across increasing dataset sizes. The average execution time of MongoDB was 157.22 ms, while MySQL recorded 528.24 ms. This difference indicates that MongoDB's document-oriented storage model and indexing mechanisms are effective for retrieving records based on specific filtering conditions, particularly when handling larger datasets.

Complex query performance also favored MongoDB. Across the evaluated dataset sizes, MongoDB achieved an average execution time of 165.73 ms compared with MySQL's 1,370.32 ms. The substantial difference suggests that MongoDB was more efficient for multi-condition filtering and sorting operations within the tested workload. These results demonstrate the advantage of MongoDB when processing flexible query patterns involving large volumes of document-based data.

Aggregation query performance showed a different trend. MySQL achieved better overall performance, recording an average execution time of 899.97 ms compared with MongoDB's 989.52 ms. This indicates that MySQL's relational query optimization mechanisms remained competitive for structured aggregation workloads involving grouped and summarized data.

Table 18 summarizes the overall query performance comparison between MySQL and MongoDB across all dataset sizes. MongoDB reduced average execution time for simple queries by approximately 24.59%, filtered queries by 70.24%, and complex queries by 87.91% compared with MySQL. Conversely, MySQL achieved approximately 9.95% better performance for aggregation queries. These findings demonstrate that query efficiency is strongly influenced by workload characteristics and the underlying database architecture.

The observed variations in execution times across dataset sizes indicate that scalability behavior differs between the two database management systems. Although larger datasets generally resulted in increased execution times, certain fluctuations were observed, particularly in MongoDB query execution. These variations may be attributed to factors such as caching behavior, query optimization strategies, indexing mechanisms, memory allocation, and system resource utilization during experimental execution.

Overall, the query evaluation results indicate that MongoDB provided advantages in simple, filtered, and complex query workloads, while MySQL maintained competitive performance in aggregation operations. These findings suggest that MongoDB is suitable for applications requiring flexible data retrieval and scalable query processing, whereas MySQL remains advantageous for structured analytical operations requiring optimized relational query execution.

Overall Performance Evaluation. Table 19 summarizes the overall comparison between MySQL and MongoDB based on the evaluation criteria utilized in this study.

The results indicate that MySQL demonstrated advantages in installation, configuration, and

data integrity due to its mature relational structure, transactional support, and compatibility with established database management practices. Its integration with the Laravel framework provides a straightforward development environment, particularly for applications requiring structured schemas, predefined relationships, and strong consistency guarantees.

Table 19
Summary of Results

Category	Better Database
Installation	MySQL
Configuration	MySQL
CRUD Performance	Mixed Results
Query Performance	MongoDB
Scalability Characteristics	MongoDB
Data Integrity	MySQL
Complex Queries	MongoDB
Aggregation Queries	MySQL
Flexible Schema	MongoDB

Meanwhile, MongoDB demonstrated favorable performance characteristics in scalability, flexible schema management, and several workload-intensive operations. The CRUD evaluation revealed mixed results, where MongoDB achieved better performance in insert, update, and delete operations, while MySQL provided faster read operations across several dataset sizes. These findings indicate that CRUD performance is highly dependent on the specific operation being performed and the underlying database architecture.

For query performance, MongoDB generally achieved better results, particularly in filtered and complex queries. Its document-oriented structure and indexing capabilities contributed to efficient processing of queries involving flexible document retrieval and multiple filtering conditions. However, MySQL maintained advantages in aggregation queries, demonstrating its effectiveness in structured data processing and relational operations.

The scalability evaluation further indicates that MongoDB exhibited more favorable performance characteristics as dataset sizes increased, particularly for write-intensive workloads and large-volume data processing. Its schema flexibility allows efficient handling of dynamic and semi-structured datasets without requiring extensive structural modifications.

The findings of this study are consistent with previous research. Filip and Cegan (2020) reported that MongoDB provides advantages in scalability and flexibility, while MySQL remains appropriate for applications requiring structured relationships and transactional consistency. Similarly, Moreno et al. (2024) observed that MongoDB performs effectively in dynamic data environments, whereas MySQL maintains advantages in structured relational workloads. The results of this study further support the findings of Ljungdahl (2023), emphasizing that database selection should depend on application requirements, workload characteristics, and scalability considerations rather than assuming that one database technology is universally superior.

Overall, neither MySQL nor MongoDB can be considered universally superior. MySQL remains suitable for applications requiring strong data integrity, transactional consistency, structured relationships, and reliable relational processing. Conversely, MongoDB is more appropriate for applications requiring schema flexibility, scalability, and efficient processing of large-scale and dynamic datasets. Therefore, the selection of a database management system should be based on the specific requirements, workload patterns, and architectural needs of the intended application.

Conclusions. This study compared MySQL and MongoDB using the Laravel framework by evaluating their installation and configuration requirements, CRUD performance, query execution performance, scalability characteristics, data integrity, and schema flexibility. The findings demonstrated that both database management systems possess

distinct advantages depending on application requirements and workload characteristics.

The results showed that MySQL provided advantages in installation and configuration due to its established compatibility with the Laravel framework. It also demonstrated strong data integrity, transactional consistency, and structured data management capabilities through its relational database architecture. These characteristics make MySQL suitable for applications requiring well-defined relationships, reliable transactions, and consistent data processing.

The evaluation of CRUD performance revealed that neither database consistently outperformed the other across all operations. MongoDB achieved better performance in insert, update, and delete operations across several dataset sizes, indicating advantages for write-intensive workloads. However, MySQL demonstrated better read performance, highlighting its effectiveness in structured data retrieval scenarios.

For query performance, MongoDB generally achieved better results in filtered and complex query operations due to its document-oriented architecture and flexible data model. However, MySQL demonstrated better performance in aggregation queries, indicating its capability in structured data processing and relational operations. These findings emphasize that query performance is highly dependent on workload characteristics, query complexity, indexing strategies, and database architecture.
The scalability evaluation further indicated that MongoDB exhibited more favorable characteristics when processing increasing dataset sizes, particularly for dynamic and semi-structured workloads. Its flexible schema design enables efficient handling of evolving data structures without requiring extensive schema modifications. Meanwhile, MySQL remained advantageous for applications requiring strong consistency, relational integrity, and structured data organization.

Despite these findings, this study has several limitations. First, the evaluation was conducted using the Laravel framework within a controlled experimental environment; therefore, the results may vary when applied to different frameworks, programming languages, deployment environments, or hardware configurations. Second, the study focused on selected performance metrics, including CRUD execution time, query execution time, scalability characteristics, data integrity, installation, configuration, and schema flexibility. Other factors, such as CPU utilization, memory consumption, storage requirements, network performance, and concurrent transaction handling, were not included. Third, the datasets were limited to 1,000, 10,000, 50,000, and 100,000 records, which may not fully represent database behavior in extremely large-scale enterprise environments. Additionally, factors such as caching mechanisms, indexing strategies, and system-level optimization may have influenced the measured execution times despite maintaining similar testing conditions.

Based on the findings and limitations of this study, database selection should be guided by application requirements rather than performance results alone. MySQL is recommended for applications that prioritize transactional consistency, data integrity, structured relationships, and relational data processing. Conversely, MongoDB is recommended for applications requiring schema flexibility, scalability, and efficient processing of dynamic or semi-structured datasets.

Future studies may extend this research by evaluating larger datasets beyond 100,000 records, incorporating additional performance metrics such as resource utilization and concurrency handling, and examining database behavior in cloud-based, distributed, and high-availability environments. Further investigations may also explore the impact of different indexing strategies, optimization techniques, and workload patterns to provide a more comprehensive understanding of relational and NoSQL database performance.

Overall, this study concludes that neither MySQL nor MongoDB can be considered universally superior. Instead, each database management system provides specific advantages that make it suitable for particular application scenarios. The appropriate database technology should be selected based on the nature of the data, system objectives, expected workload, scalability requirements, and consistency needs of the intended application.

Author contributions. Severino M. Bedis Jr.: Conceptualization, Methodology, Software Development, Data Collection, Formal Analysis, Validation, Visualization, and Writing – Original Draft | Ma. Leonila C. Amata: Supervision, Methodology Review, Validation, Writing – Review and Editing, and Project Administration. All authors reviewed and approved the final manuscript.

Conflict of interest. The authors declare no conflict of interest regarding the publication of this paper.

Funding source. This research received no external funding and was conducted as part of the academic requirements for the Master of Science in Information Technology program.

Artificial intelligence use. Artificial intelligence tools were utilized solely for language refinement, grammar checking, and manuscript editing assistance. All research design, implementation, data collection, analysis, interpretation, and conclusions were conducted and verified by the author.

Ethics approval statement. This study did not involve human participants, animals, personal data, or sensitive information. Therefore, ethics approval was not required. The datasets used were synthetically generated for benchmarking purposes.

Data availability statement. All data generated or analyzed during this study are included in this published article. The datasets used in the experiments were generated through Laravel seeders and do not contain any personal or sensitive information. The source code and benchmarking procedures are available from the corresponding author upon reasonable request.

Acknowledgement. (Not available)

Publisher's disclaimer. The views expressed in this article are those of the authors and do not necessarily reflect the views of the publisher. The publisher disclaims any responsibility for errors or omissions.

REFERENCES

Codd, E. F. (1970). A relational model of data for large, shared data banks. *Communications of the ACM, 13*(6), 377–387. https://doi.org/10.1145/362384.362685

Filip, P., & Čegan, L. (2020). Comparison of MySQL and MongoDB with focus on performance. In *2020 International Conference onInformatics, Multimedia, Cyber and Information System (ICIMCIS)* (pp. 174–179). IEEE. https://doi.org/10.1109/ICIMCIS51567.2020.9354307

Gantz, J., & Reinsel, D. (2012, December). *The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the Far East (IDC iView).* International Data Corporation. https://www.cs.princeton.edu/courses/archive/spring13/cos598C/idc-the-digital-universe-in-2020.pdf

Győrödi, C. A., Dumşe-Burescu, D. V., Zmaranda, D. R., & Győrödi, R. Ş. (2022). A comparative study of MongoDB and document-based MySQL for big data application data management. *Big Data and Cognitive Computing, 6*(2), 49. https://doi.org/10.3390/bdcc6020049

Ljungdahl, V. (2023). *Performance comparison of distributed MySQL and MongoDB in a cloud environment* (Master's thesis,

Blekinge Institute of Technology). DiVA Portal. https://www.diva-portal.org/smash/get/diva2:1738800/FULLTEXT01.pdf

Matallah, H., Belalem, G., & Bouamrane, K. (2021). Comparative study between the MySQL relational database and the MongoDB NoSQL database. *International Journal of Software Science and Computational Intelligence, 13*(3), 38–63. https://doi.org/10.4018/IJSSCI.2021070104

Mendoza-Tello, J. C., & Villacís-Ramón, J. (2022). MySQL vs MongoDB: A preliminary performance evaluation using the YCSB framework. In M. Botto-Tobar, et al. (Eds.), *Communications in Computer and Information Science (Vol. 1535,* pp. 189–201). https://doi.org/10.1007/978-3-031-03884-6_14

Moreno, R. C., Rubiano Bacca, E. D., & Parra Linares, L. A. (2024). Comparative performance analysis between MySQL and MongoDB data storage engines to support dynamic content objects. *Revista de Ingeniería Matemáticas y Ciencias de la Información, 11*(22). https://doi.org/10.21017/rimci.1093

Pandey, R. (2020, September). *Performance benchmarking and comparison of cloud-based databases MongoDB (NoSQL) vs MySQL (relational) using YCSB (Technical Report).* National College of Ireland. https://doi.org/10.13140/RG.2.2.10789.32484

Wodyk, R., & Skublewska-Paszkowska, M. (2020). Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework. *Journal of Computer Sciences Institute, 14,* 94–100. https://doi.org/10.35784/jcsi.1583