JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 35 (2025) 184–190
Received: 09 February 2025
Accepted: 25 March 2025

The Impact of Relational and Non-Relational Databases on Application Performance

Wpływ relacyjnych i nierelacyjnych baz danych na wydajność aplikacji

Jakub Olszak*, Maria Skublewska-Paszkowska

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The purpose of this study was to assess the performance of two database systems, PostgreSQL (relational) and MongoDB (non-relational), in handling data and their impact on application performance. The evaluation was conducted through a series of performance tests including read, write, delete, update, and aggregation operations on small and large datasets. Key metrics such as task completion time, response times were used for comparisons. Based on the results obtained during the experiments, it is difficult to definitively conclude which database system performs better, as both systems demonstrated advantages in different scenarios depending on the specific use case.
Keywords: SQL; NoSQL; database; performance; application

Streszczenie

Celem tego badania była ocena wydajności dwóch systemów baz danych, PostgreSQL (relacyjnego) i MongoDB (nierelacyjnego), w obsłudze danych i ich wpływu na wydajność aplikacji. Ocenę przeprowadzono za pomocą serii testów wydajności, w tym operacji odczytu, zapisu, usuwania, aktualizacji i agregacji na małych i dużych zbiorach danych. Do porównań wykorzystano kluczowe wskaźniki, takie jak czas wykonania zadania, czasy odpowiedzi. Na podstawie wyników uzyskanych podczas eksperymentów trudno jest ostatecznie stwierdzić, który system baz danych działa lepiej, ponieważ oba systemy wykazały zalety w różnych scenariuszach w zależności od konkretnego przypadku użycia.
Słowa kluczowe: SQL; NoSQL; baza danych; wydajność; aplikacja

*Corresponding author

Email address: jakub.olszak@pollub.edu.pl (J. Olszak)

1. Introduction

The performance of database systems is a critical aspect in the development and operation of applications, particularly as data volume and complexity increase. The choice between relational databases and non-relational databases depends on the specific needs and constraints of an application. Relational databases, such as PostgreSQL, have traditionally been favored for their robust ACID (Atomicity, Consistency, Isolation, Durability) properties and their ability to handle complex queries and transactions. These databases use a structured schema, which ensures data integrity and supports complex relationships through foreign keys and joins. However, the rise of big data and the need for scalable, high-performance systems have led to the popularity of NoSQL databases like MongoDB. NoSQL databases are designed to handle large volumes of unstructured data with high efficiency and flexibility They often abandon some of the strict consistency guarantees of the RDBMS (Relational Database Management System) in favor of partition availability and tolerance, as described in the CAP theorem. This makes them particularly suited for applications requiring high write throughput, flexible schema design, and horizontal scalability. This study aims to compare the performance of these two types
of databases—relational and non-relational—in managing data. The specific objectives of this research are to evaluate read performance, measure the time taken to retrieve data based on specific queries in both PostgreSQL and MongoDB and examine scalability by analyzing how the performance of PostgreSQL and MongoDB scales with increasing dataset sizes. The guiding hypotheses for this study are as follows:

H1: NoSQL databases, such as MongoDB demonstrate superior performance in handling large volumes of unstructured data, particularly in write-heavy scenarios, due to their schema-less architecture and optimized handling of diverse data types.

H2: Relational databases, like PostgreSQL offer advantages in terms of data integrity and complex queries, making them more reliable for applications requiring consistency and sophisticated data manipulation.

Understanding the performance characteristics of relational and non-relational databases is crucial for making informed decisions in database selection, especially as applications scale and data complexity increases. This study provides insights into the trade-offs between the two types of databases, helping developers and system architects choose the most appropriate database technology for their specific needs. It highlights the scenarios where each type of database excels, aiding in the optimization of application performance and resource utilization. In conclusion, this study aims to provide a comprehensive comparison of PostgreSQL (representing relational databases) and MongoDB (representing non-relational databases) in terms of read and

write performance, and scalability. This analysis aims to contribute valuable knowledge to the field of database management and support the development of more efficient and scalable applications.

2. Related Works

The comparison of relational and non-relational databases has been a subject of extensive research, with several studies highlighting the distinct strengths and weaknesses of each approach.

Studies such as [1][2][3] emphasize the superior performance of NoSQL databases, particularly MongoDB, in high write-throughput scenarios and large-scale data processing. MongoDB's schema-less design enables faster data ingestion and better scalability for unstructured or semi-structured datasets. These studies found that while MongoDB excels in bulk operations like INSERT and DELETE, relational databases such as Oracle Database and SQL Server perform better in structured queries and tasks requiring strong consistency or aggregation.

Research focusing on specific use cases, including GeoTIFF data [4] and genomic datasets [5], highlights the flexibility of NoSQL databases like MongoDB and Cassandra in managing complex or dynamic data structures. These databases outperformed relational alternatives in ingestion speed and adaptability to unstructured data, although relational databases like PostgreSQL showed advantages in handling structured queries with spatial extensions or complex joins.

Scalability and workload performance were key topics in studies such as [6][7][8], which compared various NoSQL solutions, including MongoDB, Couchbase, and Cassandra. While MongoDB generally excelled under increased load, Couchbase demonstrated strengths in real-time data processing, and Cassandra proved efficient for large-scale, unstructured datasets. These findings underline the importance of selecting a database based on specific workload requirements.

Other studies, such as [9][10], compared SQL and NoSQL databases in business and scientific applications. While SQL databases like MySQL and Oracle excelled in transactional consistency and complex queries, NoSQL databases offered better performance for read and write operations on smaller or semi-structured datasets. However, MongoDB's efficiency diminished with larger data volumes in certain scenarios, whereas MySQL benefited from index optimization.

Collectively, these studies illustrate the trade-offs between relational and non-relational databases. NoSQL databases, such as MongoDB, offer flexibility, scalability, and superior performance for unstructured data and high-volume operations. Conversely, relational databases like MySQL and PostgreSQL remain indispensable for applications requiring strong consistency, complex queries, and transactional integrity.

Building on these insights, this research aims to synthesize previous findings and provide a detailed comparative analysis of PostgreSQL and MongoDB. By conducting rigorous performance evaluations, this study seeks to guide developers and system architects in selecting and optimizing database solutions to meet specific application requirements and performance goals.

3. Scientific method

The study focused on comparing the performance of two popular database systems, each representing a different data management paradigm, allowing for an in-depth analysis of the advantages and disadvantages of both solutions.

The databases studied were:

- **PostgreSQL (version 13.15)** – a relational database with strong support for transactions, indexing, and advanced SQL queries,
- **MongoDB (version 8.0.1)** – a non-relational database known for its flexibility and scalability, making it well-suited for unstructured data.

Additionally, the application used for testing was built on Java Spring 3.0 with Java 17, providing a modern, robust framework for database interactions and ensuring compatibility with the latest features and performance optimizations

3.1. Dataset

The dataset used in the experiment comes from an self-created online bicycle shop. It includes information on various products, customers, and orders, reflecting typical e-commerce data. Whole database and dataset was created for the purpose of the study. The dataset included the following records:

- 500,000 orders**,**
- **a**pproximately 1.5 million order details**,**
- 100,000 users**,**
- 3 user roles**,**
- 3 records each in all of the following tables: delivery_method, payment_method, status, brand, category, and size.

Full database schema is shown in Figure 1.

3.2. Research stand

The experiments were conducted in controlled conditions, meaning that all background programs, including Windows updates and other non-essential services, were disabled to minimize any potential disruptions to the tests. This ensured that the database performance comparisons were as accurate and reliable as possible, with minimal interference from external processes.

The research was performed on a PC rig featuring the following specifications:

- Processor: AMD Ryzen 7 5800X3D
- Memory: 32 GB RAM
- Operating System: Windows 11 Pro, version 23H2
- Disk NVMe M.2 SSD 1TB 960 EVO

These specifications were chosen to efficiently handle the high data volume and ensure consistent, reliable performance results during the database tests.

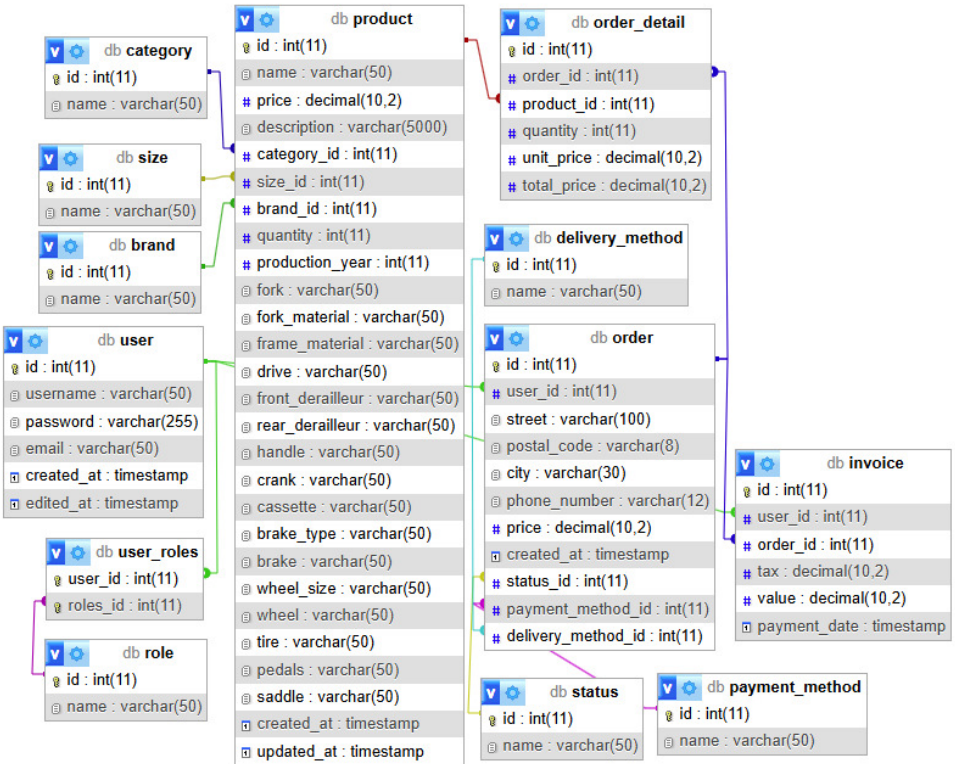

Figure 1: Database schema.

3.3. Research metrics

To objectively assess the performance of both systems, the focus was on key metrics that are crucial for applications running on databases. These metrics allow for evaluating both the performance of operations and the efficiency of data storage. All tests were conducted time-based, meaning the time elapsed from sending the query to its execution was measured to accurately determine the speed of operation execution.

The metrics included:

- read time performance – measured by the time taken to retrieve data based on queries,
- write time performance – the time required to insert new data into the database,
- delete time performance – the time taken to remove data from the database,
- efficiency time of links between tables – the time taken to retrieve data including join of the other table based on id of the record,
- additionally, the average operation times, standard deviation, median, shortest, and longest query execution times through the backend API were examined.

The results of these measurements were recorded in a file and then analyzed, allowing for a precise comparison of the performance between the two database systems.

3.4. Description of the research experiment

The experiment was conducted in several steps, which allowed for a comparison of system performance on both small and large datasets. Small dataset refers to 100 records and large dataset refers to 100,000 records, however for operations including links between tables small dataset refers to 500 records and large dataset refers to 50,000 records. The tests were repeated 100 times for each dataset. Caching was enabled during the tests, however, the first query results were always rejected, as they included caching effects or index creation, which could impact the results for the SQL database. The key stages of the experiment were as follows:

- data preparation: Organizing the datasets into appropriate tables for PostgreSQL and collections for MongoDB,
- small dataset: Testing CRUD operations on a dataset of 100 records to evaluate performance at a small scale,
- large dataset: Repeating the tests on a dataset of 100,000 records to assess scalability and performance under higher load,
- API calls: Testing CRUD operations via the API and accurately measuring response times,
- result analysis: Comparing the performance and efficiency of both systems under different conditions and scenarios.

3.5. Executed Queries and code

In this experiment, the majority of queries were executed using built-in Spring mechanisms, specifically leveraging JpaRepository for PostgreSQL and MongoRepository for MongoDB. These repositories allow for easy access and interaction with the databases, providing high-level query management while abstracting

much of the complexity associated with raw SQL. For join operations in PostgreSQL, native SQL queries were used instead of JPQL. Native queries allow direct interaction with the relational database, providing more control over SQL execution and potentially improving performance in complex operations. In contrast, for MongoDB, the Aggregation framework was used to handle more complex queries efficiently.

To perform this, there is a need to create Order object. Listing 1 shows the code to perform write operation on an single object on both relational and non-relational databases:

Listing 1: Write query for PostgreSQL and MongoDB

```
@Override
public Order createOrder(Order order) {
    return orderRepository.save(order);
}
```

To perform delete operation, the build-in repository methods were used. In Java Spring to delete record, first there is a need to find that record in database. The deletion operations were performed as in Listing 2 for both PostgreSQL and MongoDB.

Listing 2: Delete 100 records query for PostgreSQL and MongoDB

```
@Override
public void deleteOrder100() {
    List<Order> orders = orderReposito-
ry.findTop100ByOrderByIdDesc();
    for (Order order : orders) {
        orderRepository.delete(order);
    }
}
```

To perform operations involving table associations, custom queries were used. In PostgreSQL, JPQL was utilized to handle relationships between entities, such as joining tables based on foreign key relationships. In MongoDB, the Aggregation Framework was used, specifically the $lookup stage, to join collections and retrieve associated data. The table association operations were performed as shown in Listing 3 for PostgreSQL and Listing 4 for MongoDB:

Listing 3: Join query for PostgreSQL

```
@Query(value = "SELECT o.* FROM `order` o JOIN
order_detail od ON o.id = od.order_id " +
        "WHERE od.product_id = :productId ORDER
BY o.id ASC LIMIT 100", nativeQuery = true)
List<Order> findOrdersByProductIdWithLim-
it100(@Param("productId") Long productId);
```

Listing 4: Lookup query for MongoDB

```
@Aggregation(pipeline = {
        "{ $lookup: { from: 'order_detail',
localField: '_id', foreignField: 'order_id',
as: 'order_details' } }",
        "{ $match: { 'or-
der_details.product_id': ?0 } }",
        "{ $sort: { _id: 1 } }",
        "{ $limit: 100 }",
        "{ $project: { order_details: 0 } }"
})
List<Order> findOrdersByProductIdWithLim-
it100(String productId);
```

4. Results

After the research experiments were conducted, the results were collected in separate files for all research scenarios. The results were then analyzed and described in detail for each activity conducted.

4.1. Read operation results

First, tests were performed on the application server's response time to database read queries. The reading was performed on two data sets: small dataset and large dataset, then the steps were repeated a hundred times to get reliable results.

The results showing reading with a small set of data from each database are shown in Table 1.

Table 1: Read response times for 100 records

	MongoDB (ms)	PostgreSQL (ms)
Median	3.06	1.51
Average	3.38	1.11
Longest time	5.60	3.09
Shortest time	1.51	0.0>
Standard deviation	0.82	0.68

Performance analysis for MongoDB and PostgreSQL when processing 100 records shows a clear advantage for PostgreSQL in terms of speed and stability of operations. The results indicate that PostgreSQL offers shorter and more predictable response times in compared to MongoDB, which is evident in both the mean and median processing times processing. In addition, the maximum values for PostgreSQL remain lower than for MongoDB. Additionally, the standard deviation for the relational database is lower, which indicates more stable and consistent operation times.

Results showing reading with a large data set from each database are shown in Table 2.

Table 2: Read response times for 100,000 records

	MongoDB (s)	PostgreSQL (s)
Median	1.97	0.59
Average	2.01	0.60
Longest time	2.49	0.73
Shortest time	1.87	0.56
Standard deviation	0.11	0.03

When processing a large data set, PostgreSQL significantly outperforms MongoDB in terms of time performance. The mean and median times for PostgreSQL are significantly shorter, indicating that the relational database provides faster access to large data sets. PostgreSQL also shows a smaller standard deviation, suggesting greater stability of operations and less fluctuation in response times. For both a small dataset and a large dataset, the relational database had shorter response times for queries executed.

4.2. Write operation results

Next, the application server's response time to queries for writing data to the database was tested. Each test set was repeated a hundred times to obtain reliable and authoritative results. The Results showing records for the smaller data set in each database are presented in Table 3.

Table 3: Write response times for 100 records

	MongoDB (ms)	PostgreSQL (ms)
Median	28.08	30.36
Average	28.80	31.47
Longest time	53.37	66.72
Shortest time	26.30	12.53
Standard deviation	2.94	5.07

The write results for 100 records in MongoDB and PostgreSQL are similar, suggesting comparable performance for smaller datasets. However, the most notable difference is the significantly higher standard deviation and longer write times for the relational database, which may be caused by integrity checks and transactions management.

The results for larger datasets in each database are presented in Table 4.

Table 4: Write response times for 100,000 records

	MongoDB (s)	PostgreSQL (s)
Median	29.00	50.49
Average	30.09	50.47
Longest time	35.05	71.39
Shortest time	27.06	43.33
Standard deviation	2.51	4.01

Comparing the write time of 100,000 records in MongoDB and PostgreSQL, the results clearly show that MongoDB provides faster write time with large data sets. The average time and median for the non-relational database are significantly lower than for the relational one, showing its advantage when writing large volumes of data. The standard deviation for MongoDB is lower, indicating less fluctuation in write times and more consistent operations compared to PostgreSQL, where write times sometimes increase significantly.

4.3. Delete operation results

In the next step, server response times to data deletion queries were compared. The results showing the results for the smaller dataset in each database are presented in Table 5.

Table 5: Delete response times for 100 records

	MongoDB (ms)	PostgreSQL (ms)
Median	30.19	348.51
Average	31.01	694.54
Longest time	50.22	3671.08
Shortest time	27.01	95.80
Standard deviation	3.61	776.69

The results of deleting 100 records in MongoDB and PostgreSQL, you can see a significant advantage for MongoDB in terms of performance. The median time and average deletion time for MongoDB are significantly shorter than for PostgreSQL, indicating faster deletion operations in a non-relational database. That could be caused by integrity checks and transactions management in relational database.

Results showing writes for the larger data set in each database are presented in Table 6.

Table 6: Delete response times for 100,000 records

	MongoDB (s)	PostgreSQL (s)
Median	29.36	151.54
Average	29,90	152.53
Longest time	36.33	183.95
Shortest time	27.99	129.18
Standard deviation	16.67	10.85

The results of deleting 100,000 records in MongoDB and PostgreSQL show significant differences in the performance of the two databases. MongoDB achieves

significantly better times both median and average deletion times are as low as a few seconds, indicating the efficiency of this database for large data sets. Ostensibly for both a small data set and a large data set, the non-relational database achieves significantly lower server response times for deletion attempts.

4.4. Association operation results

The final stage of the study examined the server's response times to queries for table and document linkage operations.

The results showing the linkage for a smaller data set in each database are presented in Table 7.

Table 7: Association response times for 500 records

	MongoDB (s)	PostgreSQL (s)
Median	1.46	0.79
Average	1.48	0.78
Longest time	1.76	0.97
Shortest time	1.44	0.65
Standard deviation	0.04	0.06

Analyzing the results of join operations in MongoDB and PostgreSQL on a dataset consisting of 500 records, we can observe clear differences in the performance of both databases. The median time for PostgreSQL is significantly shorter than for MongoDB, indicating a faster server response for join queries in PostgreSQL. The standard deviation for both databases shows that PostgreSQL has less variation in response times, which may indicate greater stability in join operations.

The results for join operations on smaller datasets in each database are presented in Table 8.

Table 8: Association response times for 50,000 records

	MongoDB (s)	PostgreSQL (s)
Median	1.62	1.18
Average	1.63	1.18
Longest time	1.85	1.18
Shortest time	1.85	1.17
Standard deviation	0.39	0.01

Analyzing the results of linkage operations in MongoDB and PostgreSQL on a data set of 50,000 records, there are differences in the performance of the two databases. The median time for PostgreSQL is shorter than for MongoDB, indicating a faster server response for linkage queries. The median response time for PostgreSQL is also lower compared to MongoDB, which may suggest higher efficiency of operations in this database. The longest time in MongoDB significantly exceeds the longest time for PostgreSQL, which may indicate occasional delays in the operation of MongoDB. The standard deviation for PostgreSQL is much smaller, indicating its stability in the context of linkage operations.

5. Discussion

The findings from this study offer valuable insights into the performance characteristics of relational and non-relational databases—specifically PostgreSQL and MongoDB—across different types of database operations. In particular, the study explores the ability of these systems to handle large volumes of unstructured data, perform complex queries, and maintain high write throughput. The performance results have been evaluated in light of the initial hypotheses (H1–H2), and comparisons are made with the findings of related studies.

5.1. Read operations

The results of read operations (Table 1-2) show a distinct advantage for PostgreSQL in terms of speed and stability, particularly when querying small datasets. The median and average response times for PostgreSQL are significantly lower than those for MongoDB, aligning with the findings from related work [2], where PostgreSQL demonstrated superior performance in handling complex queries, especially those involving spatial data. The longer response times observed for MongoDB, particularly the maximum times, suggest that it may not be as consistent in query response times, reinforcing H2: relational databases, like PostgreSQL, offer advantages in terms of data integrity and consistency. For larger datasets (100,000 records), MongoDB's response time becomes more competitive, though PostgreSQL still maintains an advantage in terms of stability and lower variability in response times. These findings support H2's proposition that PostgreSQL outperforms MongoDB in scenarios involving complex queries and large relational datasets.

5.2. Write operations

Write operations (Table 3-4) present a more nuanced comparison. MongoDB outperforms PostgreSQL in both small and large datasets when measuring median and average write times. The results for MongoDB's write performance align with H1, which suggested that NoSQL databases, like MongoDB, perform better in write-heavy scenarios due to their schema-less design and optimized handling of unstructured data. This is consistent with related literature [1], where MongoDB demonstrated superior write throughput compared to Oracle NoSQL. For smaller datasets (100 records), MongoDB's median write time of 28.08 ms is slightly better than PostgreSQL's 30.36 ms, indicating MongoDB's advantage in raw write performance. However, for larger datasets (100,000 records), PostgreSQL is almost two times slower than MongoDB. These findings

validate H1, showing that MongoDB's design is well-suited for high write throughput, particularly for unstructured data.

5.3. Delete Operations

Delete operations highlight significant differences between the two databases (Table 5-6). MongoDB achieves significantly lower response times for both small and large datasets. For small datasets (100 records), MongoDB's median time of 30.19 ms is much better than PostgreSQL's 348.51 ms, reflecting MongoDB's efficiency in deletion tasks, especially for large-scale deletions. The same trend continues with larger datasets, where MongoDB's deletion time remains faster and more stable compared to PostgreSQL, which shows considerable fluctuation in response times. These results align with the findings of [4], where MongoDB outperformed SQL Server in deletion operations as well. The faster deletion times for MongoDB further support the hypothesis H1 that NoSQL databases will be more efficient in handling operations involving large datasets with less complex relationships.

5.4. Association Operations

Finally, association operations (joining data from multiple records) show that PostgreSQL significantly outperforms MongoDB in managing relational data dependencies (Table 7-8). For both small and large datasets, PostgreSQL's response times for these operations are lower and more consistent, reaffirming H2's hypothesis that relational databases are more efficient when managing complex relationships and joins. These findings align with [9], where relational databases consistently outperformed non-relational databases in join operations. MongoDB's performance, while competitive, lags behind PostgreSQL, particularly when handling relational data structures.

6. Conclusions

This study compares the performance of relational (PostgreSQL) and non-relational (MongoDB) databases in managing unstructured data. The study focuses on key performance aspects: read, write, delete, across small and large datasets. Results show that PostgreSQL outperforms MongoDB in read operations, particularly for smaller datasets, due to its structured schema and optimized query processing. However, MongoDB excels in write, delete, and update operations, particularly for large datasets, highlighting its scalability and efficiency in handling unstructured data. The study confirms that while relational databases like PostgreSQL are superior for complex queries and data integrity (H2), NoSQL databases like MongoDB (H1) are better suited for high-throughput write-heavy scenarios. These findings provide valuable insights for database selection, helping developers choose the right system based on application needs. Future work could focus on further optimizing database performance in Java applications, particularly by investigating how object mapping strategies and database interactions influence overall performance. Exploring object-relational mapping techniques and their impact on database operations could lead to more efficient application designs tailored to either relational or non-relational environments.

References

[1] S. Padhy, G. M. M. Kumaran, A quantitative performance analysis between Mongodb and Oracle NoSQL, In 2019 6th international conference on computing for sustainable global development, IEEE (2019) 387-391.

[2] S. H. Aboutorabi, M. Rezapour, M. Moradi, N. Ghadir, Performance evaluation of SQL and MongoDB databases for big e-commerce data, In 2015 International Symposium on Computer Science and Software Engineering CSSE (2015) 1-7, https://doi.org/10.1109/CSICSSE.2015.7369245.

[3] A. Boicea, F. Radulescu, L. I. Agapin, MongoDB vs Oracle database comparison, In 2012 Third International Conference on Emerging Intelligent Data and Web Technologies, IEEE (2012) 330-335, https://doi.org/10.1109/EIDWT.2012.32.

[4] Y. Cheng, K. Zhou, J. Wang, Performance analysis of PostgreSQL and MongoDB databases for unstructured data, In 2019 International Conference on Mathematics, Big Data Analysis and Simulation and Modelling, MBDASM (2019) 60-62, https://doi.org/10.2991/mbdasm-19.2019.14.

[5] S. Chakraborty, S. Paul, K. A. Hasan, Performance comparison for data retrieval from nosql and sql databases: a case study for covid-19 genome sequence dataset, In 2021 2nd International Conference on Robotics, electrical and signal processing techniques (ICREST) IEEE (2021) 324-328, https://doi.org/10.1109/ICREST51555.2021.9331044.

[6] I. Carvalho, F. Sá, J. Bernardino, Performance evaluation of NoSQL document databases: couchbase, CouchDB, and MongoDB, Algorithms 16(2) (2023) 78, https://doi.org/10.3390/a16020078.

[7] Y. Li, S. Manoharan, A performance comparison of SQL and NoSQL databases. In 2013 IEEE Pacific Rim conference on communications, computers and signal processing (PACRIM) IEEE (2013) 15-19, https://doi.org/10.1109/PACRIM.2013.6625441.

[8] G. Dhasmana, P. Gujjar, G. Prasad, P. K. HR, SQL and NOSQL databases in the application of business analytics. In 2023 International Conference on Computer Science and Emerging Technologies (CSET) IEEE (2023) 1-5, https://doi.org/10.1109/CSET58993.2023.10346657.

[9] M. A. Hassan, Relational and nosql databases: The appropriate database model choice. In 2021 22nd International Arab Conference on Information Technology (ACIT) IEEE (2021) 1-6, https://doi.org/10.1109/ACIT53391.2021.9677042.

[10] R. Ceresnak, O. Chovancova, Comparison of Query Performance Between MySQL and MongoDB Database. In CERes Journal 5(1) (2019), 45-51.