MINAR International Journal of Applied Sciences and Technology
ISSN: 2717-8234
Article type: Research Article

Received: 04/06/2022 **Accepted:** 19/06/2022 **Published:** 01/09/2022

A COMPARISON BETWEEN NOSQL AND RDBMS: STORAGE AND RETRIEVAL

Sameera Abbas FADHEL [1]
University of Mosul, Iraq
Enas Ali JAMEEL [2]
University of Mosul, Iraq

Abstract

Relational database management systems (RDBMS) emerged as the solution for data storage in the past decades. All data storage systems and applications utilize a RDBMS in the heart of the system to store and retrieve the data. In the past few years a new data storage model, named Not Only Structured Query Language (NOSQL), has emerged to produce less complex data storage systems and to tackle the data's massive volume performance degradation of used systems. In this work, a comparison study is conducted between MySQL as an example of RDBMS and Monogodb as an example of NOSQL systems using threading and machine resources to show the differences for developers to select one of these models for their applications. The results show that the performance of NoSQL is less than MySQL for small datasets and few database operations, such as few thousands of records and hundreds of operations per day. However, with the introduction of threads and volumes of data, the performance of Mongodb overcomes My Structured Query Language (MySQL). In addition, the results have shown that Mongodb requires more memory usage and CPU resources than MySQL to complete their tasks. Finally, the images were saved as byte data inside both platforms. The storing process for this data inside Mongodb was faster than the MySQL platform.

Keywords: Mongodb, NoSQL, Relational Database Management System (RDBMS), MySQL, CPU utilization, Threading.

http://dx.doi.org/10.47832/2717-8234.12.18

[1] sameeraabbasfadhel@uomosul.edu.iq, https://orcid.org/0000-0002-5811-4911
[2] EnasAliJameel@uomosul.edu.iq, https://orcid.org/0000-0003-3973-4282

Introduction

Data is the new oil [1], a sentence that has been used widely in the past few years. In 2020, it has been reported that 2.5 quintillion bytes of data was created in daily bases. Moreover, it has been estimated in 2025 that the world will have 175 zettabytes of data. With the introduction of the Internet of things (IoT) and their applications, billions of devices will generate a massive amount of data daily, such as smartphones, smart-watches, computers, tablets, smart glasses and even home appliances [3]. All of this data has to be transferred from one location to another location for analytics and analysis. However, before processing this massive amount of data, this data has to be stored. Databases have dominated in the storage area. Nevertheless, with the introduction of this massive amount of data, the classical database structure, named the RDBMS, revealed many performance issues [4]. To tackle these issues, different database systems and paradigms have been proposed in the past few years. One of these paradigms that became popular is NoSQL.

NoSQL has been proposed to handle data volumes in more efficient methods. Its roles and structure does not follow the relational model of classical RDBMS. In RDBMS, relations "tables" are utilized to record the data. Each table has a special unduplicated key to identify each row "data record". These keys are leveraged to create connections and links between the relations. This means that before storing any data records in RDBMS systems, a database schematic has to be created. In addition, data normalization has to be performed on the data to create a database schematic without duplicated data. These roles are not followed in NoSQL systems. Moreover, NoSQL does not satisfy atomicity, consistency, isolation, and durability (ACID) [5,6]. NoSQL adopts different data models, such as, column, key-values, documents and graphs. Documents data model has dominated in NoSQL approach. In this way, the database is a collection of volumes of independent documents. Each document stores its schematic and its own data. As mentioned no database schematic is used. This means that the sored documents may have different schemas and structures. The documents have to be encoded to be stored in the NoSQL model. Different encoding methods can be leveraged, such as, extensible markup language (XML), JavaScript object notation (JSON), binary JSON (BSON), human-readable data-serialization language (YAML) and binary format [7]. This document based data storing model allows the NoSQL system to be easily divided over numbers of servers in cloud and data center. The question that emerges is, which database paradigm to select for your application? Does the application require NoSQL or RDBMS?

In this work, a comparison study between NoSQL and RDBMS paradigm is conducted. Mongodb has been selected as the example of the NoSQL model. It has been selected since it is the most popular NoSQL system in the world [8]. MySQL has been selected as the example of a DBMS system. These two systems have been compared according to many different performance metrics. First, the impact of threading and parallel requirements on these systems will be studied. Different numbers of threads will be created to perform simple and complex data retrieval operations over these systems. Second, Python connector for these two systems will be investigated. Python will be leveraged to generate the threads and attempts to connect with the database. Python has been leveraged since it has dominated in the data science and machine learning area in the past few years. Finally, the CPU utilization and memory usages will be measured to show the load of these systems on the computer resources. In this experiment study, the following questions should be answered:

- Which platform is faster under the usage of threading?
- Which platform has fewer loads on the system resources, such as memory and CPU?
- Which platform can handle massive amounts of data faster?
- Is the Python connector a good selection for these systems ?
- How do both systems deal with different data structures, such as images?

The rest of this paper is organized as follows; in the next section, some of the works and experiments that have been conducted in the area of database system comparisons are overviewed. Section 3 introduces the method that has been used and the leveraged tools. Section 4, overviews the obtained results. Finally, the conclusion of this paper in section 5.

2. Related Works:

All The migration process from RDBMS to NoSQL platforms proliferated in the past few years [Bansal et al. 2021]. Different technical details and developments are proposed to convert the structural relational datasets into document oriented models [Abdelhedi et al. 2022, Namdeo et al. 2012]. Other researchers attempted to design an advisor for schema design to migrate from RDBMS to NoSQL [Dabowsa et al. 2021]. However, why migrate from RDBMS to NoSQL systems?

Comparing relational databases and NoSQL databases have attracted researchers over the past years. These

comparisons attempted to measure the time required to perform data manipulating and controlling for both platforms. As an example of RDBMS systems, MySQL and Mongodb have dominated for the comparisons in research papers. In addition, different datasets have been leveraged. For example, in [Jose et al. 2021], the authors attempted to compare Mongo and MySQL for data enquiries over two different datasets. The authors have shown that the time required for simple select, select with condition, updating and inserting for Mongodb is less than MySQL for all different scenarios. In [Reichardt et al. 2021], the authors compared four different database systems, three NoSQL platforms “Mongodb, Redis, Cassandra” and one RDBMS platform “MySQL”. The author leveraged the Python driver library to connect with the database platforms. The author reported that NoSQL platforms are faster than MySQL for updating, inserting, reading and writing for small datasets. However, MySQL performed better for large values “300K values”. In [Pereira et al. 2018], the authors compared three different NoSQL platforms’; Mongodb, Rethnkdb and Couchbase. The author reported that the performance of Monogodb and Couchbase overcomes Rethinkdb. Moreover, the authors have shown that Monogodb overcomes Couchbase for multi-thread operations. However, Couchbase scored betere results for select with condition “GET with ID”. In [Győrödi et al. 2020], the authors compared the performance of MySQL and the Couchbase NoSQL platform. CRUD operations have been investigated. The results reported in the work have shown that MySQL performs better for small operations and structured data. However, for complex and massive operations, NoSQL performed better. Another comparison between MySQL and Mongodb has been reported in [Matallah et al. 2021], the author utilized Yahoo Cloud Serving Benchmark [Cooper et al. 2010]. The author reported that Mongodb operation execution time is less than MySQL for all types of operations .

In [Sánchez-de-Madariaga et al. 2018], medical data records have been used to compare RDBMS and NoSQL. The authors reported that Mongodb execution time is lower than MySQL for all types of operations. In [Das et al. 2019], loading and data scanning metrics have been used to compare MySQL and NoSQL HBase platforms. The author reported that HBase performs better in these metrics than MySQL. In [Yassine et al. 2018], the migration process from MySQL to Monogdb has been investigated. The performance of complex operations has been compared between the two platforms, Mongodb outperformed Mysql for complex operations. In [Chakraborty et al 2021], a dataset of COVID-19 genome has been used for the comparison purpose between MySQL and Mongodb. Two main operations have been used for the comparison purpose, data loading and complex inquiries. Mongodb has outperformed MySQL for both operations .

With the introduction of cloud computing and their services, database as a service has been introduced by different cloud vendors. RDBMS and NoSQL have been designed to work with different cloud platforms. In [Shareef et al. 2022], a survey and a comparison of different cloud based database systems have been reviewed. The scalability of these systems have been overviewed. In [Zaman et al. 2021], a comparison between Azure SQL and Atlas Mongodb has been conducted. The results have shown that data loading, retrieving and uploading time for Azure is much less than Atlas. These results show that the cloud platform selecting process is an important process .

This work differs from other works in three main folds. First, storing and retrieving special data types, such as images. Second, Python database connector will be utilized to load the data and to perform the inquiries. In this way, the performance of the databases with their interface was measured. Finally, inquiries' resources, such as, the memory usage and CPU utilizations using different threads to access the databases in parallel will be measured.

3. Experiment Details

This section consists of two main parts. The first part describes the process used to generate the dataset utilized in the comparison. In the second part, the operations and inquiries used will be overviewed.

3.1 The Dataset

To compare the two databases' platforms, a dataset has to be used. The dataset from the students' records and their classes were created. Moreover, the information of 500 students with 20 lecturers has been added to create the database schematic.

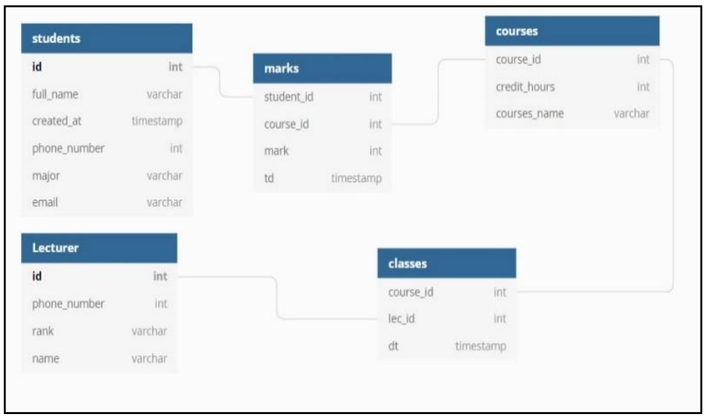

Figure 1 .RDBMS Database Schema Design

Figure 1 shows the designed schematic. The first table consists of the information of the 500 students. The second table consists of the information of the 20 lecturers. The third table has the grade information of these students. This table consists of 20,000 records. The fourth table consists of the classes' information. The fifth table consists of the curriculums' information. This table is created from the relation between the curriculum, students and the lecturers. To fill out these tables, a home page has been created and each student has been asked to fill the grades off all the classes that have been attended. Subsequently, the data has been extracted from the databases' tables into csv files. Moreover, this data has been exported from MySQL to be loaded into new databases. In addition, these files have been analyzed to generate data documents for each student, grads, classes and lecturers. These documents have been converted into JSON files to be loaded into Mongodb.

Mongodb version 5.0.6 has been used for NoSQL databases and MySQL server version 8.0.28 has been used for the RDBMS. The PC used in this experiment is Intel i5 10 generation with 8G RAM and 1 Tbyte storage. Windows 10 has been used as the operating system for the PC. Python 3.6 has been utilized to connect to the databases. To export the dataset from the first created database, the export tool of MySQL has been used. To create the schema for MySQL, the MySQL client tool has been used. Finally, to connect the databases with Python, two connector libraries have been downloaded. The first connector is Pymongo that connects to the Mongodb database. On the other hand, Mysql-connector-python-8.0.28 connector has been used to connect Python to MySQL.

Table 1 .Load Time of Both Platforms

Platform	Time
MySQL	11 mins
Mongodb	5 mins

The exported MySQL version of the files and the JSON file to load them into newly created databases was utilized. Table 1 shows the time required to load both files into the new databases. The observation from the table that the time required to load the JSON files into Mongodb is less than the time required to load the same data into MySQL since the imported file from MySQL has SQL statements that should be executed one by one to create the same database as the original one. However, in Mongodb, the JSON file has been used to generate documents for each entry in the file only. In addition, JSON files for all the data "students, lecturer, classes and grades" have been inserted into the same collection without creating a new schema. This is not the case with MySQL that requires the creation of a new schema for each table and inserting its data in a separate process.

3.2 Operations and Inquires

To compare the two platforms' performances utilizing the generated dataset, four different operations have been leveraged. The following subsections introduces these operations.

A- Select or Find

The first operation is the find or "select" inquiry. Three versions have been used from this inquiry. The first is a common simple find command on the mark table that consists of 20,000 records. This select operation retrieved the record limited to a different number of records. In the second form of this inquiry, a condition has been added to select only the marks of the students in the same major only. The last version is to utilize 10 threads to retrieve the marks for ten different majors from the same database.

Table 2 .The Select Inquiries Performed

Operation	Platform	Inquiry
Simple Select	MySQL	Select * from marks LIMIT
Complex Select	MySQL	Select * from marks where "student_id" like {%num%}
Simple Select	Mongodb	db.marks.find.limit(). explain()
Complex Select	Mongodb	db.marks.findOne({"student_id": {$regex:"num"}})

20 different threads have been created using the threading library in Python. Each

thread creates a different connection using the connector library for each platform. The time is recorded when all threads retrieve the data from the database. Moreover, CPU utilization has been recorded for each system when utilizing these threads. Table 2 shows the inquiries that have been used in the first operation.

B- Update Inquiries

The second type of used inquiry is the update command. Three versions of update have been used for the comparison process.

Table 3 .The Update Inquiries Performed

Operatio n	**Platfor m**	**Inquiry**
Simple Update	MySQL	update marks set marks =50 where “student_id”=xx
Complex Update	MySQL	update marks set marks =50 where “ student_id” like {%num%}
Simple Update	Mongod b	db.marks.explain().update({“studnet_id”:xx},{$set:{”mark”:50}})
Complex Update	Mongod b	db.marks.explain().update({“studnet_id”:{$regex:num}},{$set:{”mark ”:50}, {$multi:true}})

The first update inquiry is to update a select field in the only one record in the mark table. The second version is to update all the marks for all the students in the same major. Finally, the last version is to update only one record for 20 students utilizing threads. The threads have been created and configured as in the select inquiry. Table 3 shows the update inquiries performed.

C- Delete Operation

The third operation is the data deleting process. As in the select and update, this operation consists of three versions.

Table 4 .The Delete Inquires Executed

Operation	**Platform**	**Inquiry**
Single Delete	MySQL	Delete from marks where “student_id”=xx
Multiple Delete	MySQL	Delete from marks where “ student_id” like {%num%}
Single Delect	Mongodb	db.marks.explain().deleteOne({“studnet_id”:xx},{$set:{”mark”: 50}})
Multiple Delete	Mongodb	db.marks.explain().deleteMany({“studnet_id”:{$regex:num}},{$ set:{”mark”:50}, {$multi:true}}){$multi:true}})

In the first type, one record will be deleted. In the second version multiple recorders are selected for the deletion process. Finally, 20 threads are used to delete one 20 records. Table 4 shows the inquires executed against the two systems.

4. Results

This section consists of four main subsections. Each one of the subsections discusses the results obtained by the inquiries in each case.

4.1 Select Case

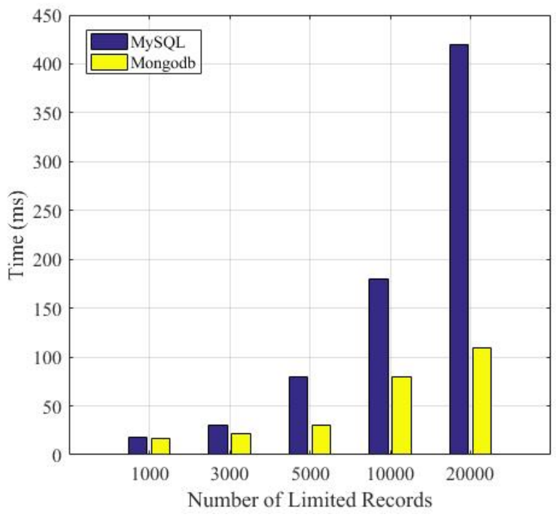

Figure 2 .Simple Select Operation

Figure 2 shows the time of executing the simple select operation utilizing one thread with a different number of retrieved records. The observation that the time is almost the same for both systems with a small number of records. However, with higher numbers Monogdb recorded less execution time.

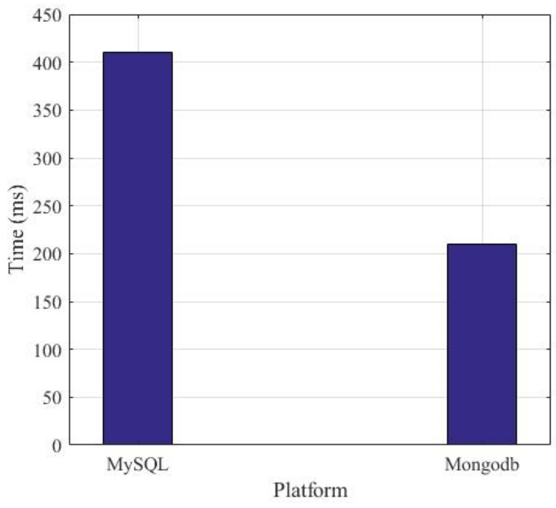

Figure 3 .Threading Comparison for Simple Select

Figure 3 shows the recorded execution time for the compex inquiry against the two platforms. The observation observe that Mongodb recorded less retrieval time than MySQL.

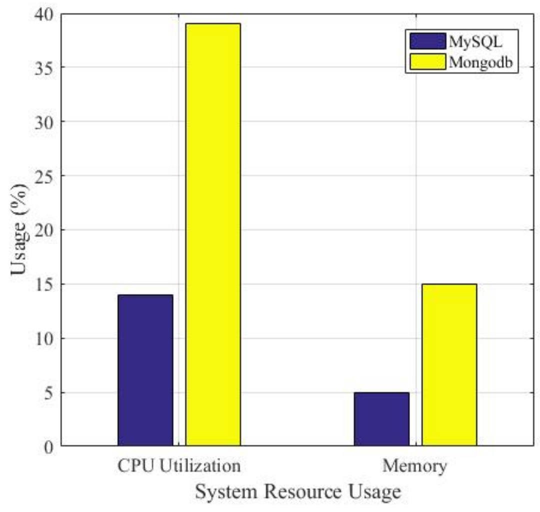

Figure 4 .System Resource for Threading in Simple Select

Finally, figure 4 shows how both systems operate with different numbers of threads. The observation of the Mongodb has less execution time however. In figure 4, the observation of the Mongodb utilized another 25% more CPU utilization among all the created threads. In addition, the memory usage of Mongodb compared with MySQL server for these threads is shown in figure 4. Memory usage of Mongodb is higher with more than 10% when utilizing these threads. This means that there is a trade-off between the system resource and execution time for these two platforms. It worth mentioning that is not compare the complex select since it is similar to the multiple updates shown in the next section.

4.2 Update Case

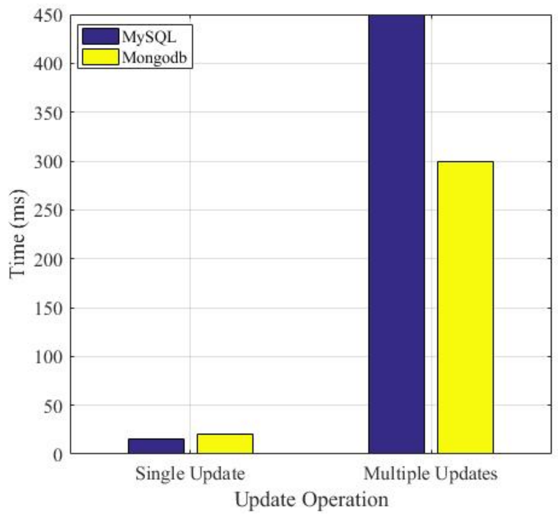

Figure 5 .Update Comparison

Figure 5 shows the execution time of record updating using both platforms. The observation for a single update MySQL recorded a better time. However, when the number of records increases, the time for Mongodb decreases compared with MySQL.

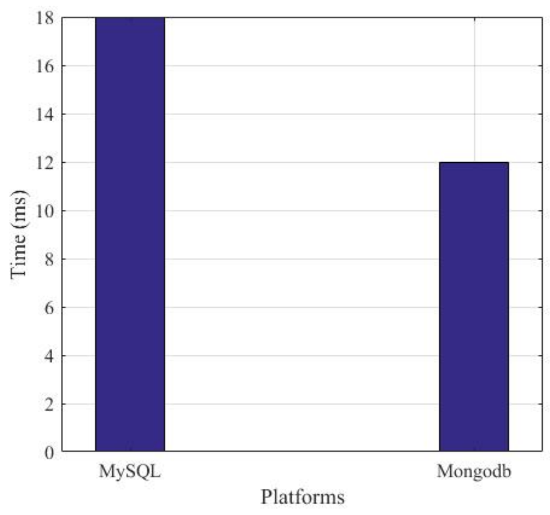

Figure 6 .Threading comparison for Update Process

Figure 6 shows the updating record time utilizing threads. Thread connections are less time consuming in Mongodb compared to MySQL.

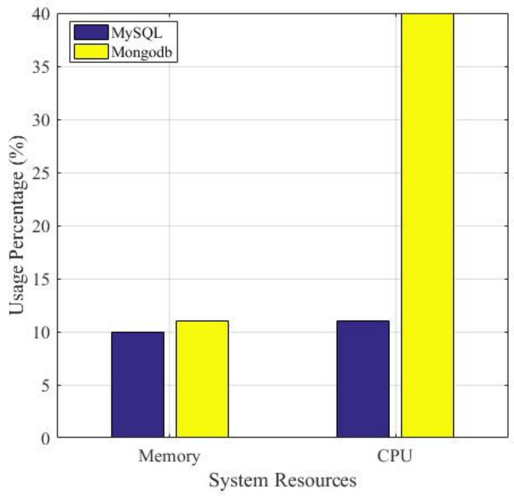

Figure 7 .System Resource Usage in Update

However, figure 7 shows the memory usage and CPU utilization of both Mongodb and MySQL. The observation for updating, the memory usage of both systems is approximately the same. However, for CPU utilization, Mongodb increased the CPU utilization up to 40% compared to MySQL that increased it with 11% only.

4.3. Data Deletion Case

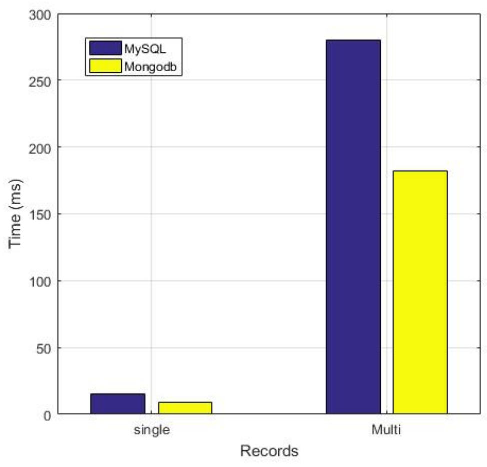

Figure 8 .Delete Comparison between the Two Platforms

Figure 8 shows the time required for data deletion in both systems. The observation for deleting a document is easier than deleting a record in a table.

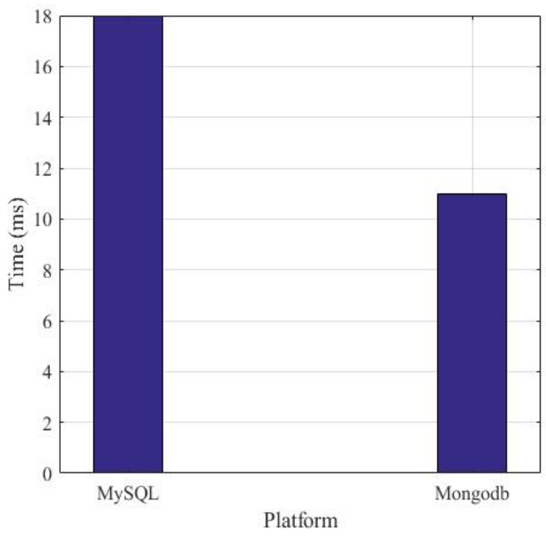

Figure 9 .Threading Comparison in Deletion

The execution time for Mongodb is better than MySQL even for one record. Moreover, for the number of records, Mongodb has recorded a lower execution time. For threading, Mongodb has also recorded better time as shown in Figure 9. However, for system resources, the same results as shown for data updating inquiries were recorded.

4.4 Storing Images

The constructed system by adding images of the students and the lecturers in the databases was enhanced. To save images in databases, systems store these images directly in operating system file system folders and save a link as a "varchar" in one column in the tables in classical RDBMS databases. The same can be followed in NoSQL. However, images can be stored as byte data inside Mongodb and can be stored as a "blob". There is a data type called longblob that can be leveraged in MySQL. The images were saved inside Mongodb and MySQL.

Table 5 .Image insertion and Retrieving Comparison

Operation	Platform	Time
Insertion	MySQL	110 ms
Retrieval	MySQL	87 ms
Insertion	Mongodb	90 ms
Retrieval	Mongodb	77 ms

Table 5 shows the time required to save one image and to retrieve it for both systems. The observation of the time is high for both platforms. However, it is better in Mongodb. This method is not recommended for both systems.

5. Conclusion:

In this work, a comparison study has been conducted between RDBMS and NoSQL platforms for different data operations. A new dataset has been harvested from the university students. Threading and system resources have been leveraged as a performance metric for the comparison process. Moreover, images have been also utilized as a complex data type for insertion and retrieving. The data shows that Mongodb overcomes MySQL for complex data types and large data volumes. However, Mongodb consumes more system resources than the MySQL platform for memory and CPU. This makes an issue for developers that leverage and lease resources from cloud systems. The observation for developing NoSQL Mongodb is easier than MySQL since the developer does not require to learn SQL language. Moreover, no schema, data normalizations are required. In addition, importing the data and exporting is easier with NoSQL since it deals with documents.

6. References:

[1] Humby, C. (2006). Data is the new oil. Proc. ANA Sr. Marketer's Summit. Evanston, IL, USA.

[2] Data Statistics. (october, 2021). How much data is creted everyday. https://seedscientific.com/how-much-data-is-created-every-day

[3] Masoud, M, Jaradat, Y, Manasrah, A, Jannoud, I. (2019). Sensors of smart devices in the internet of everything (IoE) era: big opportunities and massive doubts. *Journal of Sensors.*

[4] Kumawat, D and Pavate, A. (2018). Correlation of NOSQL & SQL Database. *IOSR J. Comput. Eng.(IOSR-JCE),* 18, 70-74.

[5] Truica, C, Radulescu, F, Boicea and A, Bucur, I. (2015). Performance evaluation for CRUD operations in asynchronously replicated document oriented database. *In 2015 20th International Conference on Control Systems and Computer Science,* 191-196.

[6] Gupta, S and Rani, R. (2016). *Data Transformation and Query Analysis of Elasticsearch and CouchDB Document Oriented Databases*. PhD diss.

[7] Meier, A and Kaufmann. (2019). M. SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management; Springer Vieweg: Wiesbaden, Germany.

[8] DB engines. (2022). Database system ranking. https://db-engines.com/en/ranking_trend

[9] Bansal, N, Soni, K and Sachdeva, S. (2021). Journey of Database Migration from RDBMS to NoSQL Data Stores. In International Conference on Big Data Analytics, 159-177 .

[10] Abdelhedi, F, Jemmali, R and Zurfluh, G. (2022). Relational Databases Ingestion into a NoSQL Data Warehouse. *arXiv preprint arXiv:2203.06949.*

[11] Namdeo, B and Suman U. (2012). Schema design advisor model for RDBMS to NoSQL database migration. *International Journal of Information Technology*, 13 (1), 277-286.

[12] Dabowsa, N, Maatuk, A, Elakeili, S and Ali, A. (2021). Converting Relational Database to Document-Oriented NoSQL Cloud Database. *In 2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA*, 381-386.

[13] Jose, B and Abraham S. (2020). Performance analysis of NoSQL and relational databases with MongoDB and MySQL. *Materials today: PROCEEDINGS 24* , 2036-2043.

[14] Reichardt, M, Gundall, M and Schotten, H. (2021). Benchmarking the Operation Times of NoSQL and MySQL Databases for Python Clients. *In IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society*, 1-8.

[15] Pereira, D, Morais, W and Freitas E. (2018). NoSQL real-time database performance comparison. *International Journal of Parallel, Emergent and Distributed Systems* , 2, 144-156.

[16] Győrödi, C, Dumşe-Burescu, D, Zmaranda, D, Győrödi, R, Gabor, G, and Pecherle, G. (2020). Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Application's Data Storage. *Applied Sciences*, 23 (8524).

[17] Matallah, H, Belalem, G and Bouamrane, K. (2021). Comparative study between the MySQL relational database and the MongoDB NoSQL database. *International Journal of Software Science and Computational Intelligence (IJSSCI)*, 3, 38-63.

[18] Cooper, B, Silberstein, A, Tam, E, Ramakrishnan, R and Sears, R. (2010). Benchmarking cloud serving systems with YCSB. *In Proceedings of the 1st ACM symposium on Cloud computing* ,143-154.

[19] Sánchez-de-Madariaga, R, Muñoz, A, Castro, A, Moreno, O and Pascual, M. (2018). Executing complexity-increasing queries in relational (MySQL) and NoSQL (MongoDB and EXist) size-growing ISO/EN 13606 standardized EHR databases. *Journal of Visualized Experiments*, e57439.

[20] Das, N, Paul, S, Sarkar, B and Chakrabarti, S. (2019). NoSQL overview and performance testing of HBase over multiple nodes with MYSQL. *In Emerging technologies in data mining and information security*, 269-279.

[21] Yassine, F and Awad, M. (2018). Migrating from SQL to NOSQL Database: Practices and Analysis. *In 2018 International Conference on Innovations in Information Technology (IIT)* , 58-62.
[22] Chakraborty, S, Paul, S and Hasan, KM. (2021). 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)*, 324-328.
[23] Shareef. T, Sharif. K, and Rashid. B. (2022). A Survey of Comparison Different Cloud Database Performance: SQL and NoSQL. *Passer Journal of Basic and Applied Sciences*, 4(1), 45-57.
[24] Zaman. F, Khuhro. M, Kumar, K, Mirbahar, N, Khan, M and Kalhoro A. (2021). Comparative Case Study Difference Between Azure Cloud SQL and Atlas MongoDB NoSQL Database. *International Journal*, 9(7).