Comparison of Query Performance Between MySQL and MongoDB Database

Roman Ceresnak, Olga Chovancova

***Abstract*—** Nowadays exists enormously variety of databases. They are essential for storing and managing information using operations in the query. In the article, MySQL and MongoDB query performance was compared. The time duration of operations for body mass index (BMI), was taken into account. Experiments were executed on generated sample data of 100, 1~000, 10~000 items size. We conducted that relational database is more suitable for a straightly defined data structure from the perspective of time. Our results showed, which type of database is faster for the selection of data for watching BMI index of individual groups.

***Keywords*—** MySQL, MongoDB, DBMS, NoSQL, BMI index.

I. Introduction

A database is a combination of information that is organized so that it can efficiently be obtained, managed, and updated. Among operations belongs these operations: create, read, update and delete (CRUD). CRUD operations are used for managing the primary function of persistent storage [1].

Database Management Systems (DMSs) are higher-level software programs that work with lower-level application programming interfaces that take care of CRUD operations.

New kinds of DBMSs, like relational and non-relational (NoSQL) databases, have been developed to help to solve different kinds of problems. Applications programs like MySQL, MongoDB, PostgreSQL were developed to implement these kinds of DBMSs. Most of them, are using for accessing databases, the most common standardized language a SQL. The SQL is a structured query language.

The most popular relational SQL database among open source community is MySQL. It is mostly used for web sites, which are running on open source systems. The most popular non-relational SQL refers to MongoDB. It allows for exploring new ways of storing information. It becomes popular mostly due to their scalability and flexibility for Cloud Computing and Big Data. This article focuses on a comparison of query performance between MySQL and MongoDB database.

Section II. contains background for this topic. Experiments and results are described in Section III. In Section IV., results are being summarized, and potential future work will be discussed. Section V. is oriented for an overall conclusion.

II. Background

There are many databases commonly, relational and non-relational (NoSQL) databases. The concept of a relational database is that, a data structure that allows linking information from different tables, or different types of data buckets. A non-relational database only stores data without explicit and structured mechanisms to link data from different buckets to one another. Relational databases usually work with structured data, and non-relational databases are work with semi-structured data.

Truica [2] examined CRUD operations for MongoDB and MySQL databases. Asynchronous replication, which is necessary for a scalable and flexible system, was examined in the paper of [3].

Roman Ceresnak, Faculty of Management Science and Informatics, University of Zilina, Slovakia, (email: roman.ceresnak@fri.uniza.sk)
Olga Chovancova, Faculty of Management Science and Informatics, University of Zilina, Slovakia, (email: olga.chovancova @fri.uniza.sk)

They used for testing, the execution time for CRUD operations for a single database instance and a distributed environment.

In the research [4], a comparative study of non-relational databases and relational databases was presented. Their primary focus was on the comparison of MongoDB to MySQL. Their results stated that MongoDB is more efficient than MySQL. They used a no-relational database for integration in a forum in the field of personal and professional development. They also presented a framework for database integration. Performance evaluation of MySQL and MongoDB was also performed in paper [5].

A comprehensive comparison of SQL and MongoDB databases for various CRUD operations and large datasets was performed in work of [6].

Continuation of Gyordodi work [7] was a comparative study between the usage abilities of MongoDB, and MySQL, as a back-end for an online platform. They presented advantages for using MongoDB compared to a MySQL. They integrated their results on an online platform for publishing articles, books, and so on, with the possibility of sharing them with other users. The primary outcome of their work is highlighting differences between MySQL and Mongo for executed operations in a parallel system.

In research [8], the main concepts of NoSQL databases were compared to four selected products databases (Riak, MongoDB, Cassandra, Neo4J) according to their capabilities concerning consistency, availability, and partition tolerance, as well as performance, were presented.

III. Databases

In following section, relational database MySQL and non-relational MongoDB will be shortly described.

A. MySQL Database

An ACID is an abbreviation regarding atomicity, consistency, isolation, and durability. Those features are all useful in a database system and connected to the understanding of a transaction. Atomic units of operation are called transactions, and they can be rolled back or committed. When a transaction saves changes to the database, either all the changes are, or they are rolled back. A consistent state is persisted in a database for all times. It is also after each commit or rollback, and during progressing of transactions. If data are updated across various tables, then queries returns old values or new values. Old and new values are not combined.

Indexes are significant features of query performance. Many applications require fast lookups in a query. It is necessary to design better tables, queries, and indexes for better performance. The typical database design applies a covering index wherever possible. The query results are determined totally of the index, without viewing the original table data. Respectively foreign key constraint additionally demands an index, to efficiently check if values exist either in parent and child tables.

A query is an operation, which reads information from a table. It could be one or more tables. Optimization by index depends on the parameters and structure of data. Join is a query if multiple tables are included. Example of MySQL query is shown in Fig. 1.

```
select name, surname
   from user
       join height using (op)
       join weight using (op)
   where current_year = 2019
       and (weight.current_weight
           /(height.current_height
           * height.current_height*)
        ) < 19,9;
```

Fig. 1. Example of query code for MySQL database.

B. MongoDB Database

MongoDB is a non-relational database. It is also open-source and document-based. High performance, automatic, and high availability provides [9].

The term "MongoDB" originates from the word "humongous." That is mainly because of databases ability to scale up with ease, and it allows containing enormous amounts of data. Documents are stored in collections within databases [10].

MongoDB performs requests to read data from the database. MongoDB uses a JSON-like query language. Its language includes a variety of query operators which begins with character *$*. In the mongo shell, can be called query using the commands for methods like *db.collection.find(), db.collection.findOne()*.

In Fig. 2., example of MongoDB query is shown. The query creates a collection of users by aggregating BMI index which is computed by dividing the current weight of user against the power of two of current height. If values are less than 19, then they are selected, which results in selecting all underweight users from 10 000 data sample.

```
db.getCollection('User3')
   .aggregate([
    { "$project": {
      "total": {
        "$divide": [
          "user.current_weight"
          { "$multiply": [
            { "$multiply":
               [
               "user.current_height",
                user.current_height"
               ]
            },
            10000
          ]},
          :{ $lte: 19.9}
      ]}}}
])
```

Fig. 2. Example of query code for MongoDB database.

IV. EXPERIMENTS

Experiments were performed on the machine with the following configuration:

- **Computer**: MacBook Pro (Retina, Early 2015),
- **CPU**: 2,9 GHz Intel Core i5,
- **RAM**: 8 GB 1867 MHz DDR3,
- **Hard Disk**: 500 GB SSD,
- **Operating system**: macOS Mojave.

Sample data were generated for conducting experiments. These data will be used for calculation of body mass index, shortly BMI. Users were divided, according to the results of BMI, into five groups, which are shown in Table 1.

Users were created based on the mentioned groups. The main idea of experiments is to compare the speed of gaining individual data about BMI index from users' weight and height. Records in the constructed table for experiments contained 100, 1 000, and 1~000 items, which represented users.

The objective of the experiments is to make a query, which shows the count of users with the same BMI index. BMI is measured by is Eq. 1:

$$BMI = {}^{m}/_{h^2} \tag{1}$$

where *m* is body mass in kilograms, and *h* is body height in meters.

In Table 1, the groups of BMI based on health risk are shown. This table is referential for

evaluating category in our experiments.

TABLE I
BMI INDEXES ACCORDING TO THEIR HEALTH AND WEIGHT CATEGORY

BMI	Health risk	Weight
0 - 19.9	middle	underweight
20 - 24.9	low	normal weight
25 - 29.9	middle	overweight
30 - 39.9	high	obesity
> 40	very high	extreme obesity

Attributes such as name, surname, height, and weight are needed for selecting the BMI group. The first step is creating the data, and the next step is indexing up the data, so it could be used for observing the changes in the gaining of values for each BMI group.

V. RESULT

In this section are described results from experiments of query performance between MySQL relational database a MongoDB non-relational database.

A. MySQL results

Results of first experiment are shown in Fig. 3. Results of the experiment are measured in milliseconds by retrieving each user (records) with related BMI index.

For finding out the value of people we will need attributes such as name, surname, height, and weight and so, in the next step we will create the index up to the data and we will also watch the changes of the times needed for the gaining of values for each BMI group.

Fig. 4 shows times measured for the gaining of all people with using the index up to the chosen data with the same BMI for the chosen number of lines in the MySQL database.

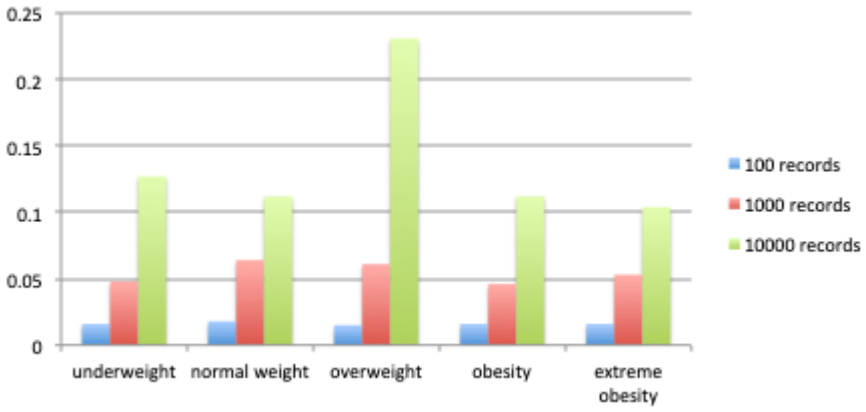

Fig. 3. Results of Experiment 1 on MySQL database

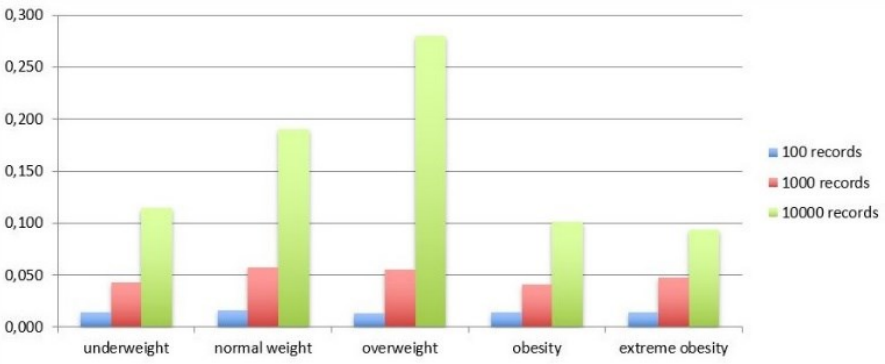

Fig. 4. Results of indexing up on MySQL database

B. MongoDB results

The experiment with the speed of gaining the data was also implied in non-relational database MongoDB, and up to the created attributes we created the index too. Differences between the speed of data are possible to compare in Fig. 5 and. Fig. 6. presents times measured for the gaining of every person with the same BMI for the chosen number of lines in the MongoDB database. Fig. 6. shows times measured for the gaining of all people with using the index up to the chosen data with the same BMI for the chosen number of lines in the MongoDB database.

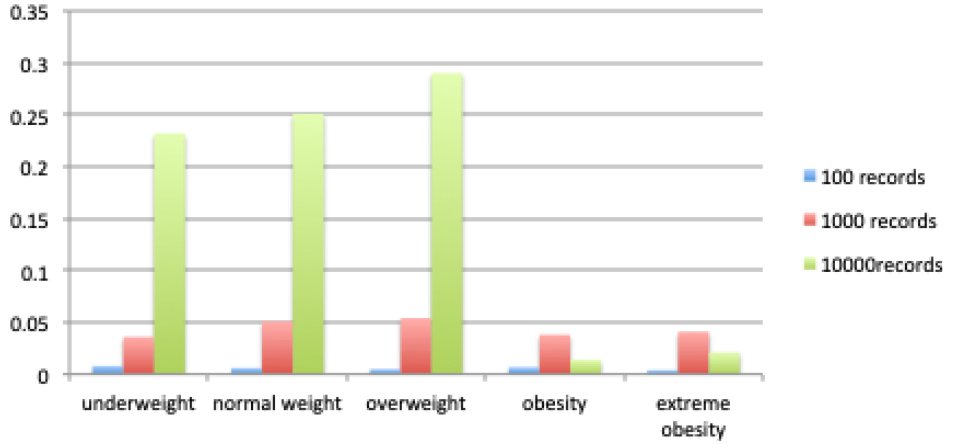

Fig. 5. Results of Experiment 1 on MongoDB database

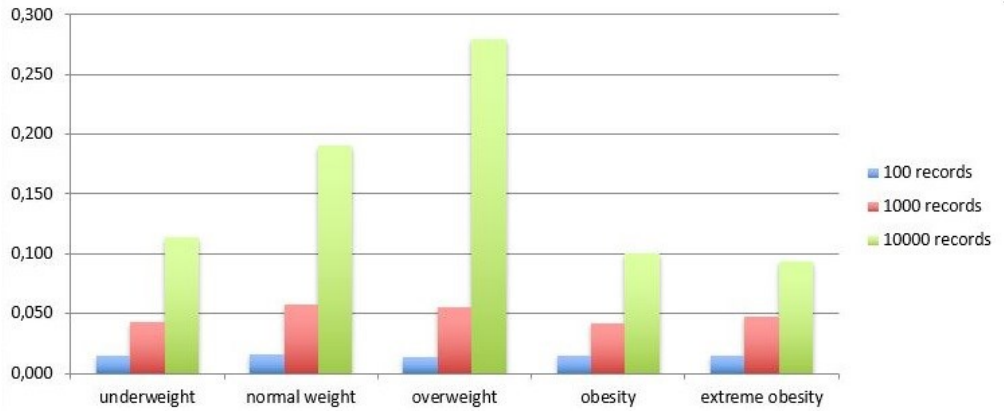

Fig. 6. Results of indexing up on MongoDB database

C. Experiment with indexes

For the comparing reasons we chose two types of data, there are visible in Fig. 7 and 8. For the first reason, we chose the table with line value 10 000 and watched the difference of times with and without using. Subsequently, we did this experiment for the non-relational database too.

Fig. 7. presents comparation of difference in using respectively no-using of the index in relational database MySQL on the sample of 10 000 data. Fig. 8. displays comparation of difference in using respectively no-using of the index in non-relational database MongoDB on the sample of 10 000 data.

As we can see in Fig. 7. and 8., in the case of optimization the speed for the gaining of data amount, it is suitable to use the index, which will make the demand up to the individual data faster. In increasing the data amount, the speed of choosing wanted data in non-relational database MongoDB is decreasing, and the effectiveness of the relational database is increasing.

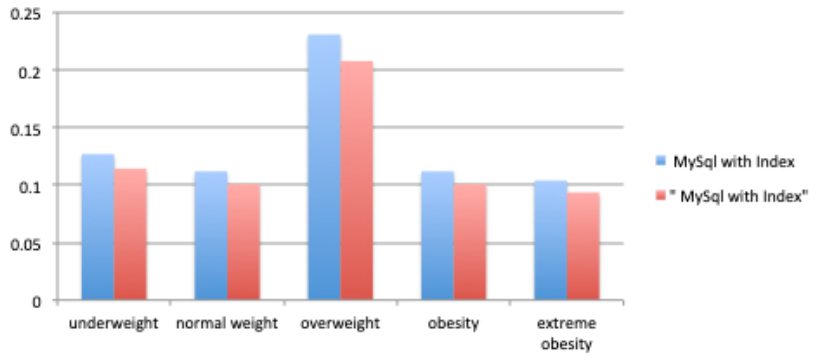

Fig. 7. Comparation of MySQL on the sample of 10 000 data

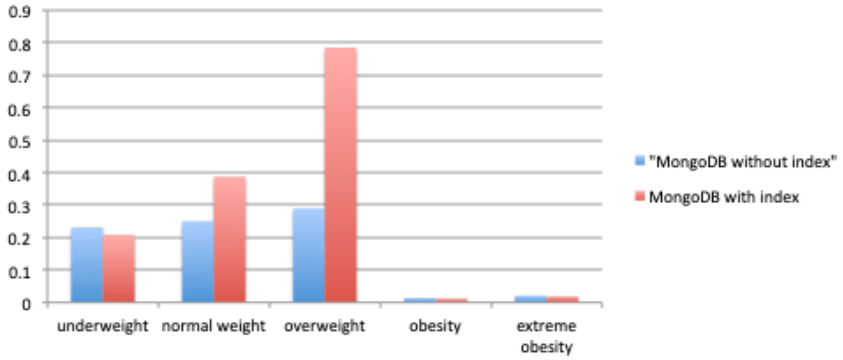

Fig. 7. Comparation of Mongo on the sample of 10 000 data

VI. Discussion

We did the experiments, which shows us that with the straightly defined data structure in the number of data 10 000 and more is from the time perspective more suitable to use a relational database. Then we examined how the choice of data is affected by the creation of the current index up to the given data. This also made the selecting of data in watching of body mass index of individual groups faster.

In the case of optimization, the speed for the gaining of data amount, it is suitable to use the index, which will make the demand up to the individual data faster. In increasing the data amount, the speed of choosing wanted data in non-relational database MongoDB is decreasing, and the effectiveness of the relational database is increasing.

VII. Conclusion

In this article, we focused on query performance between most used relational MySQL database and most used a non-relational MongoDB database.

For a demonstration of experiments, we created a simple table of users and filled up with random data. Later, was created a query which selected BMI category based on weight and height. We used a small data size because we were observing the query performance on small data.

Our finding was conducted for a most used type of query, which is select, and better suited is relational database MySQL instead of MongoDB database.

It gives a better and faster result for 10 000 data sample size from the viewpoint of a time scale. For 10,000 records, we found out that in the most common select operation, the relational database will be suited for faster results.

For future work, we plan to conduct more experiments, and with more significant and more relationships between tables by comparing more types of databases.

References

[1] M. G. Lindquist, “Managing the data-base environment,” Inf. Process. Manag., 2002.
[2] C.-O. Truica, A. Boicea, and I. Trifan, “CRUD Operations in MongoDB,” 2013.
[3] C. O. Truica, F. Radulescu, A. Boicea, and I. Bucur, “Performance evaluation for CRUD operations in asynchronously replicated document oriented database,” in Proceedings - 2015 20th International Conference on Control Systems and Computer Science, CSCS 2015, 2015.
[4] C. Gyorodi, R. Gyorodi, G. Pecherle, and A. Olah, “A comparative study: MongoDB vs. MySQL,” in 2015 13th International Conference on Engineering of Modern Electric Systems (EMES), 2015, pp. 1–6.
[5] D. Damodaran B, S. Salim, and S. M. Vargese, “Performance Evaluation of MySQL and MongoDB Databases,” Int. J. Cybern. Informatics, 2016.
[6] R. Aghi, S. Mehta, R. Chauhan, S. Chudhary, and N. Bohra, “A comprehensive comparison of SQL and MongoDB databases,” Int. J. Sci. Res. Publ., 2015.
[7] C. Gyorödi, R. Gyorödi, I. Andrada, and L. Bandici, “A Comparative Study Between the Capabilities of MySQl Vs. MongoDB as a Back-End for an Online Platform,” Int. J. Adv. Comput. Sci. Appl., 2016.
[8] T. Wiktorski, “NOSQL Databases,” in Advanced Information and Knowledge Processing, 2019, pp. 77–84.
[9] D. Hows, P. Membrey, E. Plugge, and T. Hawkins, “Introduction to MongoDB,” in The Definitive Guide to MongoDB, Berkeley, CA: Apress, 2015, pp. 1–16.
[10] MongoDB Inc., “The MongoDB 3.2 Manual,” MongoDB Manual 3.2. 2016.