Performance Study of MySQL and MongoDB for IoT Data Processing and Storage

Tsvetelina Mladenova
Department of Computer Systems and Technologies
University of Ruse
Ruse, Bulgaria
tsmladenova@uni-ruse.bg

Irena Valova
Department of Computer Systems and Technologies
University of Ruse ation
Ruse, Bulgaria
ivalova@ecs.uni-ruse.bg

2022 International Conference Automatics and Informatics (ICAI) | 978-1-6654-7625-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICAI55857.2022.9960134

***Abstract*—One of the most critical moments in the development of software development is the emergence and use of databases. With the emergence of the Internet and web applications, as well as with the entry of IoT devices into literally all spheres of our lives, the amount of data that is generated and processed increases many times. The popular SQL databases cannot meet the requirements of the new volumes of data, and this necessitates the search for other solutions and the emergence of NoSQL databases. It is challenging to compare and evaluate the two types when designing and developing software applications.**

This paper presents a study and performance comparison of the two types (SQL and NoSQL) databases for data from IoT devices and sensors. Two basic operations - Insert and Select - have been tested, in different technical variants. The results show that MySQL is a suitable choice when there is a need for multiple select queries and MongoDB is suitable when there are numerous insert queries that have to be processed by the DBMS. The experiments show that a combination of both databases is an optimal solution.

***Keywords*—databases, MySQL, MongoDB, database comparison, IoT data**

I. Introduction

One of the most notable software developments is the development of the database. The quantity of the data collected via different software products increases drastically and there is a need for constant development of new methods and technologies for data storage optimization and data processing. The development of database management systems is considered to be a very important moment in the history of software development. The development of databases itself is considered to be in three revolutionary phases – the first is the emergence and development of the computer (from around 1951 to 1972), the second – the development of the relational database (up until 2005), and the third – the emergence and dynamical development of the nonrelational databases and their modifications [1].

In relational databases, a common term is a *relation*. The data is organized in tables that have a defined, clear, and intuitive representation format. Every row of the table is a record that describes an object and has a unique identificator or key. The integrity of the data is guaranteed by defining a primary key, and with the definition of foreign keys, the relationships between the tables are formed. The relational databases are subject to the ACID properties (atomicity, consistency, isolation, durability) that guarantee the data integrity while the transaction is performed. All these characteristics have contributed to the popularity of these databases.

They are already on the market long enough and on the one hand, they have secure and easy support, and on the other hand, there are good enough environments to work with them, there are many developed applications with them and respectively many customers who use them. They are undeniably very popular. Certainly, the biggest advantage of RDB is the use of the SQL language and its standards created over the years. There are slight differences in the data description language in different DBMS, but they are minor, and the possibility to use a unified approach or query language from all DBMS manufacturers is a big advantage.

Despite all this, the continuous generation of data on the Internet and the daily use of multiple IoT devices leads to the generation of huge amounts of data that cannot be organized and processed in a relational environment. The emergence of the so-called big data also leads to the search for suitable models for them and, as an alternative, NoSQL solutions appear. They do not use tables and records and do not have a clear and simple data organization structure. NoSQL is interpreted as not only SQL, which means that they use this language as an auxiliary, and not as the main language of querying. In general, this type of database uses different languages, even very different ones. These different languages are required because of the support of unstructured and semi-structured data and reduce the degree of data abstraction.

Undoubtedly, this leads to an increase in the workload and work of programmers. NoSQL databases do not use a fixed data model, have no join operations, and are subject to the CAP (consistency, availability, and partitioning) rule, not ACID.

The most popular representative of relational databases is the MySQL DBMS (Fig. 1). It is used both for training purposes and for the implementation of sufficiently serious software projects. MongoDB is the most popular of the NoSQL DBMSs (Fig. 1).

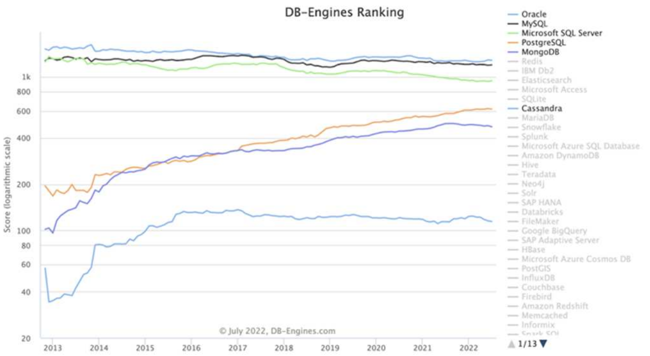


Fig. 1 Databases Ranking [2]

II. Related Work

Since relational or SQL databases are so popular and used in practice and are used in a huge number of information systems, it is logical to look for comparisons with NoSQL databases. NoSQL databases became a possibility after 2005, and to this point, there is still uncertainty as to whether and when to use them. Therefore, there have been and continue to be studies and comparisons between different NoSQL systems and between NoSQL and SQL.

According to [3] databases and their management systems are divided into three types - SQL, newSQL and NoSQL. A special place is devoted to the evaluation of the performance of NoSQL databases, which are divided into groups according to the data organization model they use - key-value databases, column-oriented databases, document-oriented, and graph databases. The article is a useful overview with concrete data on the performance of NoSQL DBMS.

In [4] MySQL is used as a traditional SQL database and MongoDB as a NoSQL database to compare the performance of CREATE, SELECT, INSERT, DELETE and IMPORT commands. The paper shows the difference in execution times of various queries in relational and non-relational databases and concludes that in all cases non-relational databases have better performance than relational databases and as data volume increases their performance increases.

The authors in [5] and [6] make detailed comparisons of the two types of DB using different evaluation criteria: scalability, performance, flexibility, queries, and security, and conclude about the conditions and when to use which type of DB.

There are similar studies in the field of business applications [7], and the goals of applications in the context of Industry 4.0 [8], for collecting and storing data from IoT devices [9].

The motivation for this study is to check how the two DBMSs would behave for collecting, storing, and processing data from a measuring station for monitoring soil and air indicators. Although NoSQL databases are recommended for such use cases, our experience, familiarity, and existing relational database applications already in operation make us hesitant in our choice.

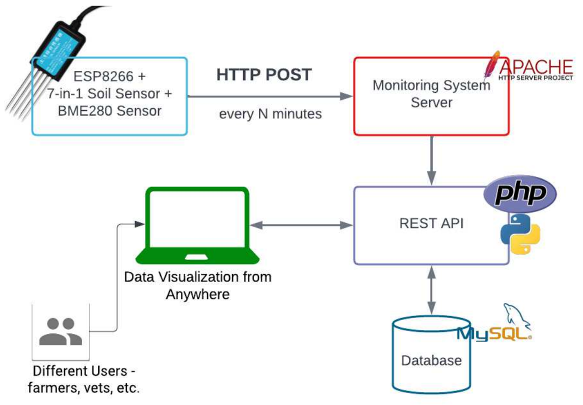
III. Experiment

A. Setup

The experiment setup consists of several IoT devices, such as sensors for measuring soil parameters (ph, temperature, moisture, salinity, nitrogen, phosphorus, potassium), sensors for measuring air parameters (temperature, humidity, pressure), and video cameras. The data is considered to be IoT data because of its origin and purposes and its usage is described in [10] and [11].

The measurements from the sensors are collected with Arduino devices and sent to a server on which a REST Api is responsible for the data processing and storing in the database. Fig. 2 shows the architecture of the setup.

It should be noted that the current system uses a MySQL database and hence its presence in the figure. This article aims to examine the possibility of replacing MySQL with MongoDB or combining the two databases if it's proven that the combination of the two databases is optimal for the purposes of the system.



ESP8266 + 7-in-1 Soil Sensor + BME280 Sensor
HTTP POST
every N minutes
Monitoring System Server
APACHE
HTTP SERVER PROJECT
REST API
php
Data Visualization from Anywhere
Different Users - farmers, vets, etc.
Database
MySQL

Fig. 2 Architecture of the System

The experiments are conducted with the following setup:

- **Computer**: HP Pavillion 15
  - **Processor**: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz
  - **RAM**: 16GB
- PHP 7.4
- MySQL 5.7
  - PHP extension: mysqli 7.4.6
- MongoDB 4.2.2
  - PHP driver: 1.13

B. Dataset

Table I. shows the structure of the data used in the tests. Visible from the sample is that the data consist primarily of doubles with the exception of the date. All of the numerical fields are stored as doubles, both in MySQL and MongoDB, and the date field is stored as a date, again both in MySQL and MongoDB.

TABLE I. TABLE 1 DATASET DESCRIPTION

Column	Data type	E.g.
Soil moisture	Double	16.5
Soil temperature	Double	25.1
Conductivity	Double	228
Ph	Double	5.3
Nitrogen	Double	16
Phosphorus	Double	22
Potassium	Double	55
Salinity	Double	125
TDS	Double	114
Air temperature	Double	36.29
Pressure	Double	1011.42
Altitude	Double	15.24
Air humidity	Double	24.57
Date	Date	2022-07-13 11:56:42

C. *Test Cases*

Since the primary objective of the study is to compare both databases through the prism of IoT, the following test cases are taken into consideration:

- **INSERT** – Simulation of continuous stream of data to the server;

The test of the *insert* case is done for 1 000, 10 000, 20 000, 30 000, 40 000, 50 000, and 60 000 data points, simulating different workloads of the sensors and the server. In this experiment, the insert queries are done individually, meaning that for every new measurement, an *insert* query is performed rather than storing several measurements and sending them to the databases in a bulk. While the bulk insert is a beneficial and more optimized solution in such cases, sometimes there are storage limitations, network bandwidth limitations, etc. that make this type of insert inapplicable.

Therefore, this experiment tests the capability of both databases when a relatively big number of queries is performed for a short time.

- **SELECT** – selecting parts of the data with different criteria.

Having a system that monitors different parameters via sensors means that there would be a need for selecting data with different criteria. The select test cases include selecting all of the data for one date, for a period of three days, and for a period of seven days. The selected ranges mimic the behavior of most monitoring systems and their functionality to visually represent a past state, usually a short past period.

IV. Results

Fig. 3 shows the results obtained in one of the insert test scenarios – insert of multiple rows in MySQL. As expected, as the number of records grows, the execution time grows as well. What is interesting is that when inserting the records as-it-is and without any primary key, the execution time does not follow any trend apart from the growing one. While inserting the records with an autoincrement value as a primary key, the execution time growth is more predictable and smooth. The same experiment, insert with and without index, is not performed in MongoDB, as MongoDB does not allow insert without index, or ObjectId as it is called. Hence, the experiment is done only in MySQL.

Fig. 4 shows the execution time of the operation *insert* in MySQL (orange line) and MongoDB (blue line). The diagram shows the drastic difference between the two DBMS regarding the operation. It should be noted that the insert in both tables is done with some form of an index – in MySQL, it is a primary key with autoincrement value and in MongoDB, it is the automatically created ObjectId. In both DBMS, the execution time grows with the growth of the inserted records, as expected. The obvious difference is that the operation *insert* is extremely fast and optimized in MongoDB in which 60 000 records are processed and stored for less than the time that MySQL needs for processing and storing 10 000 records. That makes MongoDB the obvious choice if the system expects multiple *insert* operations.

Fig. 5 shows the execution times of the operation *select* with MySQL and MongoDB. In this test case, the DBMS that performs better is undoubtedly MYSQL. Again, the experiment is done on tables and collections that have index and its presence is visible in the results. The primary key in MySQL contributes to the faster querying of the data. Every select query is done with the same criteria – filtering by dates for both MySQL and MongoDB. And in the case of a system, that needs to display data, i.e. to select data from the database, MySQL is the better choice.

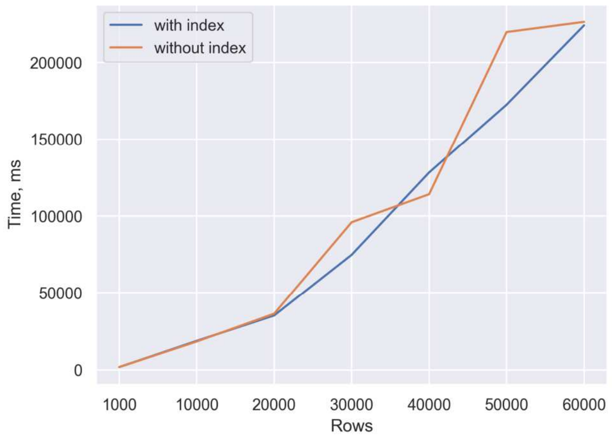


Fig. 3 MySQL insert with and without indexes

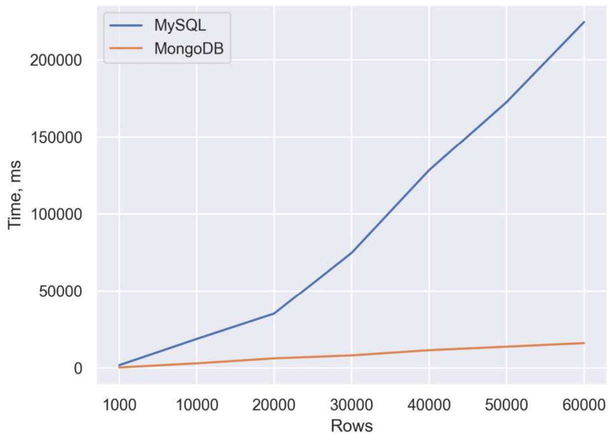


Fig. 4 Insert with MySQL and MongoDB

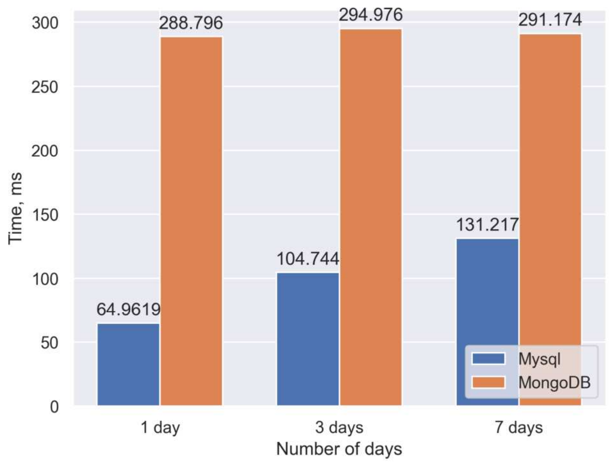


Fig. 5 Select with MySQL and MongoDB

V. Conclusions

It is very difficult to choose which of the two approaches to use - SQL or NoSQL, when the developers have extensive experience with relational databases and know the SQL language well, especially if an already existing information system is being upgraded with additional functionalities. Experiments show that when adding records to the database, MySQL does not perform as well as MongoDB, but in terms of data extraction, this DBMS gives undoubtedly better results. In reality, the data is such that there is no need for a primary key in the tables, but an autoincrement attribute was added just for the experiment, which is defined as a primary key. The SELECT operations performed do not use the key attribute, but usually date and time.

For the purposes of the soil and air monitoring system, we can still use the relational database because the insertion of records is not very time-intensive - a single record is added to each table every 15 minutes, and data retrieval is more often, since it is necessary to graphically present information to the user for a time interval chosen by them.

Acknowledgment

This paper is supported by project 2022–EEA–01 “Analysis of big data processing algorithms and their application in multiple subject domains”, funded by the Research Fund of the “Angel Kanchev” University of Ruse.

The research leading to these results has received funding from the Ministry of education and science under the National science program INTELLIGENT ANIMAL HUSBANDRY, grant agreement № Д01-62/18.03.2021.

References

[1] Guy, H. (2015). Next generation databases: NoSQL, newSQL, and big data.

[2] https://db-engines.com/en/ranking_trend

[3] Khasawneh, T. N., AL-Sahlee, M. H., & Safia, A. A. (2020, April). Sql, newsql, and nosql databases: A comparative survey. In 2020 11th International Conference on Information and Communication Systems (ICICS) (pp. 013-021). IEEE.

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

[5] A. Oussous, F. Z. Benjelloun, A. A. Lahcen, S. Belfkih, “Comparison and Classification of NoSQL Databases for Big Data”, International Journal of Database Theory and Application, vol. 6 (4), 2013

[6] T. Partel and T. Eltaieb, “Relational Database vs NoSQL,” in Journal of Multidisciplinary Engineering Science and Technology (JMEST), vol. 2, pp. 691–695, April 2015

[7] Chang, M. L. E., & Chua, H. N. (2018, April). Sql and nosql database comparison. In Future of Information and Communication Conference (pp. 294-310). Springer, Cham.

[8] de Oliveira, V. F., Pessoa, M. A. D. O., Junqueira, F., & Miyagi, P. E. (2021). SQL and NoSQL Databases in the Context of Industry 4.0. Machines, 10(1), 20.

[9] S. Reetishwaree and V. Hurbungs, "Evaluating the performance of SQL and NoSQL databases in an IoT environment," 2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM), 2020, pp. 229-234, doi: 10.1109/ELECOM49001.2020.9297028.

[10] N. Valov et al., "Design of a Sensor Measuring Station for Pasture Parameters Remote Monitoring," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2022, pp. 1-4, doi: 10.1109/HORA55278.2022.9800039.

[11] T. Mladenova, I. Valova and N. Valov, "Design of a smart system for monitoring and management of pastures and meadows: The Relational Database Approach," 2022 8th International Conference on Energy Efficiency and Agricultural Engineering (EE&AE), 2022, pp. 1-5, doi: 10.1109/EEAE53789.2022.9831393.