Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation

Conference Paper · February 2023

DOI: 10.1109/ICDABI56818.2022.10041513

Comparison of SQL and NoSQL databases with different workloads: MongoDB *vs* MySQL evaluation

Ticiana Capris
Computer Science Department
Polytechnic Institute of Viseu
Viseu, Portugal
estgv17486@alunos.estgv.ipv.pt

Pedro Melo
Computer Science Department
Polytechnic Institute of Viseu
Viseu, Portugal
estgv18336@alunos.estgv.ipv.pt

Nuno M. Garcia
Instituto de Telecomunicações
Universidade da Beira Interior
Covilhã, Portugal
ngarcia@di.ubi.pt

Ivan Miguel Pires
Instituto de Telecomunicações
Universidade da Beira Interior
Covilhã, Portugal
impires@it.ubi.pt

Eftim Zdravevski
Faculty of Computer Science and Engineering
University Ss Cyril and Methodius
Skopje, North Macedonia
eftim.zdravevski@finki.ukim.mk

***Abstract*—One of the most important considerations when selecting a database is how relational (SQL) and non-relational (NoSQL) data structures will interact. While all options are viable, consumers should take certain distinctions into account before choosing. Since SQL databases are vertically scalable, you can typically scale server components like CPU, RAM, or SSD. NoSQL databases, on the other hand, support horizontal scaling. As a result, you can increase the capacity of your NoSQL database by fragmenting (data partitioning), or by adding extra servers. Why, then, is it still challenging to choose the instance that is most appropriate for a given application and requires the least amount of runtime? Because data that will be conveyed via the internet uses a cloud in computer networks as a metaphor. To determine which model to utilize, it is required to conduct a comparison study of SQL-oriented database engines. SQL has a form created for another side of non-productive data and is offered in the form of ordered data, but NoSQL databases are horizontally expandable. Workload management solutions are therefore also in charge of automating organizational procedures, i.e., they carry out activities without requiring manual employee attendance. For businesses trying to implement continuous delivery methods and enhance the effectiveness of customer service delivery, they are unavoidable.**

***Index Terms*—Non-relational database, Database engines, SQL vs. NoSQL, big data,**

I. Introduction

In general, because both relational (SQL) and non-relational (NoSQL) data structures are feasible possibilities, it is desirable to combine them when choosing a database to store information about a system. To pick the best choice, you must consider various distinctions, quote Kaur 2013 Modeling. This article compares and contrasts relational and non-relational database engines, including MySQL and MongoDB, that are both SQL and NoSQL oriented.

The business world of today is getting more and more competitive, requiring organizations to adapt quickly to changes and stay competitive. Agile decision-making is necessary for the organization at the strategic and tactical or operational levels. Organizations must be equipped to facilitate the gathering, processing, and analysis of massive amounts of data in order to lay the groundwork for new information to be discovered. As a result, it is becoming more understood that creating systems that support decision analysis is essential to raising the amount and caliber of information available for decision-making within an organization.

With the vast data growth, big data architectures become indispensable for efficient, robust and timely processing of it [10]. In turn, it entails the use of efficient algorithms for cluster-size and cost optimisation [6], [11], as well as for scalable feature selection and dimensionality reduction [13].

The methodology was qualitative, utilizing bibliographical research on books, papers, and monographs that experimentally illustrate the topic at hand as well as well-known websites on various software application types, uses, and scalability. Therefore, the proposed study intends to accelerate performance benefits by optimizing the execution of database applications in virtualized systems, highlighting resource sharing and implementation of dynamic tuning of disk resources.

Comparative comparison of relational and non-relational databases will be done as part of this study in order to gauge how well each system performs. These outcomes allow one to gauge how effectively the database is optimized.

In order to help readers choose a database system for various applications, this paper's main contribution is a realistic database comparison of SQL and NoSQL. Consequently, utilizing a tool to display the results of workloads and investigate the database performance space

Six sections make up this research. The structure of this essay is as follows. Section 2 provides a recap of the background work. The distinction between relational and non-relational databases is compared in Section 3. The applied experimental setup is described in Section 4. The performance findings are shown in Section 5, and the study is concluded and directions for further research are presented in Section 6.

II. Background

A systematic collection of data can be referred to as a database. The database (DBMS) manages a system that deals with data, transactions, issues, or other elements. Traditional database systems are comparable DBMSs that employ the Structured Query Language (SQL) [1].

NoSQL systems are solutions where the maintenance of the database schema is transferred to the application code and users do not declare a database schema [5]. Consequently, these two datasets are comparable prospective rivals. The goal of this study is to find appropriate databases for comparison analysis between databases that are NoSQL and SQL-oriented and a non-relational database engine. Several articles currently emphasize the connection between relational and NoSQL databases by illustrating their differences and similarities through a broad notion. NoSQL primarily aims to lessen the influence of SQL, highlight the distinction between structured and unstructured databases, and enhance system performance without modifying the hardware or acquiring more powerful servers to increase the network. In the relational database review article, this leads to an improvement in network scalability when employing low-cost commodity hardware. There are pros and cons to NoSQL and its properties. Since the majority of developers are inexperienced with the technology, the drawbacks and issues with NoSQL databases will be discussed in terms of their complexity, consistency, economics, and limitations, highlighting the fact that relational databases will still be required in the future. The only application lines they will support are those that assist company operations. Nevertheless, NoSQL databases will support expansive, widely used, and content-focused applications.

III. Difference between relational and non-relational databases

SQL or Structured Query Language is the most widely used expression in the world for executing commands in relational databases, based on tables [3]. Through SQL, one creates databases, tables, columns, and indexes where it guarantees and revokes user distinctions by querying and storing data.

Therefore, SQL is a robust language divided into the command sets of Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL), Transactional Control Language (TCL), and Data Query Language (DQL). SQL is one of the most widely employed choices, making it a safe and beneficial option for complex services. However, it can be limited.

All your data must follow the same structure. It can require a lot of initial preparation, and one error can bring the entire system to a halt. Today, an SQL database is designed for unproductive data and provided as organized data. Its flexibility allows each document to have its structure by allowing various database syntaxes for other databases. As dataset sizes grow, big data architectures become necessary for efficient and timely processing of it [10]. In turn, it entails the use of efficient algorithms for cluster-size and cost optimisation [6], [11], as well as for scalable feature selection and dimensionality reduction [13].

On the other hand, NoSQL databases are scalable [9]. This base object supports more traffic by fragmenting (partitioning data), i.e., adding more servers to its database. This absolution makes NoSQL more powerful and the most chosen for projecting larger sets by allowing constant changes.

The dynamic schema of NoSQL databases easily supports agile development, which requires extensive and fast iterations. However, the term NoSQL has two definitions of different origins, where one refers to relational databases that do not use the SQL language for querying, created by Carlo Strozzi [7]. The other is related to the NoSQL movement, which defines a tool structure that does not use a relation body.

The NoSQL database consists of files organized in a hierarchical database format and uses the UNIX Shell as an interaction tool. The NoSQL database models have a completely different structure, each using an incompatible query form. NoSQL databases define four types of structured data, such as documents, graphs, key-value pairs, and vast column stores. One of the most general types we have in the NoSQL world, with schemas defined as flexible, allows records and documents to have different types and numbers of fields without needing a fixed schema.

Such documents are defined in JSON format through concepts contiguous to the object-oriented model, where it is equivalent to documents. In addition, new fields can be added, allowing for rapid application development, as in the case of MongoDB and CouchDB.

A. *Types of software and their metrics that the debriefing tools allow you to perform*

Quality Assurance is a significant concern in the software development industry because most companies today use this application to manage their business, products, and customer relationships, which requires more excellent reliability and quality. Therefore, quality assurance measures are critical to the success of any software application. One of the simplest and cheapest is software measurement. Software measurement aids in decision making by providing quantitative data to inform which aspects of a product do or do not meet specified quality standards and by allowing you to assess the benefits of new engineering methods and tools [4].

For example, understanding and improving production processes, evaluating return on investment, and managing the project based on facts rather than "guesswork.

Various metrics are used to measure software, like the types of measurement used in a software system, documentation, or related process. With these metrics, you can determine the effort or time required to complete a task or the size of a product. In addition, software metrics are easily calculated, understood, tested, and independent of the observer applying them. They are also a good source for statistical research on the software life cycle.

Scope, developers, and environment are all factors that affect the development process. Therefore, the comparison must be carefully analyzed.

Metrics can and should be applied during the software development phase, ensuring their positive impact on the final product. However, according to some experts, measurements must be defined according to specific objectives to measure software artifacts through meaningful metrics. Furthermore, according to some experts, measurements must be determined according to particular purposes to measure software artifacts through meaningful metrics. In this regard, GQM (Goal/Question/Metric) is an approach to applying metrics to improve the software development process (and, consequently, the software products generated) while maintaining the organization's business goals and technical goals. Writing and reading from big data requires careful analysis of the usage patterns in order to find suitable row key designs so that data is partitioned and the load is evenly distributed across nodes [12].

It is a top-down approach to establishing systematic measurements of goals related to the development process. The team establishes organizational goals, sets measurement targets, enters questions to address specific goals, and identifies metrics that provide answers.

B. Runtime and scalability

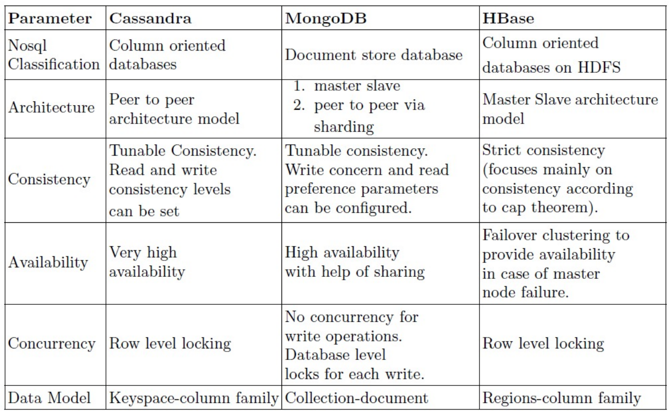
A study focused on the application in a virtualized environment makes it possible to define resource allocation adjustments statically, prioritizing an increase in performance among the various machines that make up this application. Therefore, as organizations increasingly use virtualization to reduce the operational costs of physical servers in the data center and consolidate applications, concerns arise about the concurrency of resources in a consolidated virtualized environment. As a result, critical applications are more vulnerable to performance bottlenecks, such as database transactions.

C. Study and comparative evaluation of databases type column

The use of mobile applications and web is increasing, and it is generating massive unstructured data has led to the invention of various NoSQL data stores. As a result, web-scale demands are increasing daily, and NoSQL databases are evolving to meet industry requirements.

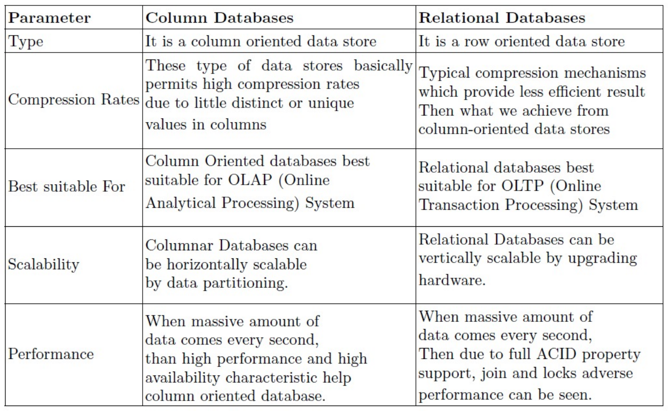
Column-oriented databases are also known as extensible record stores and vast columnar stores. All stores are inspired by Google's Bigtable, a distributed storage system for managing structured data designed to scale to a huge size [8].

Differently to the relational databases, column stores in NoSQL are hybrid row/column stores. Although column stores use massively distributed architectures to store data instead of tables, they share the idea of column-by-column storage



Parameter	Cassandra	MongoDB	HBase
Nosql Classification	Column oriented databases	Document store database	Column oriented databases on HDFS
Architecture	Peer to peer architecture model	1. master slave 2. peer to peer via sharding	Master Slave architecture model
Consistency	Tunable Consistency. Read and write consistency levels can be set	Tunable consistency. Write concern and read preference parameters can be configured.	Strict consistency (focuses mainly on consistency according to cap theorem).
Availability	Very high availability	High availability with help of sharing	Failover clustering to provide availability in case of master node failure.
Concurrency	Row level locking	No concurrency for write operations. Database level locks for each write.	Row level locking
Data Model	Keyspace-column family	Collection-document	Regions-column family

Fig. 1. Comparison of Cassandra, MongoDB and HBase



Parameter	Column Databases	Relational Databases
Type	It is a column oriented data store	It is a row oriented data store
Compression Rates	These type of data stores basically permits high compression rates due to little distinct or unique values in columns	Typical compression mechanisms which provide less efficient result Then what we achieve from column-oriented data stores
Best suitable For	Column Oriented databases best suitable for OLAP (Online Analytical Processing) System	Relational databases best suitable for OLTP (Online Transaction Processing) System
Scalability	Columnar Databases can be horizontally scalable by data partitioning.	Relational Databases can be vertically scalable by upgrading hardware.
Performance	When massive amount of data comes every second, than high performance and high availability characteristic help column oriented database.	When massive amount of data comes every second, Then due to full ACID property support, join and locks adverse performance can be seen.

Fig. 2. Database parameters

with columnar databases and columnar extensions to row-based databases. Each key in column storage is connected to one or more attributes. A column store saves its data so it can be aggregated rapidly with less I/O activity. It offers high scalability in data storage.

The data stored in the database is based on the sort order of the column family. Columns can be categorized into column families, which is crucial for splitting and organizing data. At runtime, columns and rows can be added flexibly.Still, column families have to be predefined often, which leads to less flexibility than key-value or document stores. The column family data stores include Hbase, Hypertable, and Cassandra.

IV. Experimental Setup

The experiments require a standard, flexible and customizable benchmarking platform. Yahoo! Cloud Serving Benchmark [2] (YCSB) is the de facto choice for benchmarking NoSQL and SQL databases. Yahoo! The Cloud Serving Benchmark Framework aims to facilitate performance comparison of next-generation cloud data service systems. It defines a core set of benchmarks and reports the results. In this regard, a vital feature of the YCSB framework/tool is that it is extensible

- it allows for easy definition of new workloads and facilitates benchmarking of new systems.

The benchmarking tools help generate synthetic data, a string of random characters indexed by a primary identifier. In addition, it has a workload executor to handle multiple client threads and performs defined CRUD operations.

For both models, different workloads were considered. These are:

- Workload A, 50% read, 50% write
- Workload B, 100% write
- Workload C, 100% read
- Time and throughput

Other benchmarks, such as TPC-H or SSB, can be used to test performance. However, these benchmarks are more appropriate for decision support. YCSB was chosen for its simplicity and for testing the two primary operations performed in NoSQL and SQL databases, get and put. The tests were performed on a single node with a MacBook Pro macOS, Intel i9 9880H, 2.3 GHz processor, 32 GB of memory, and 500 GB of SDD. The section shows the performance evaluation using workloads A, B, and C.

V. Analysis of the Results

In this chapter, the performance results for both MySQL and MongoDB will be presented. Figures V and V show the throughput and runtime for all three workloads.

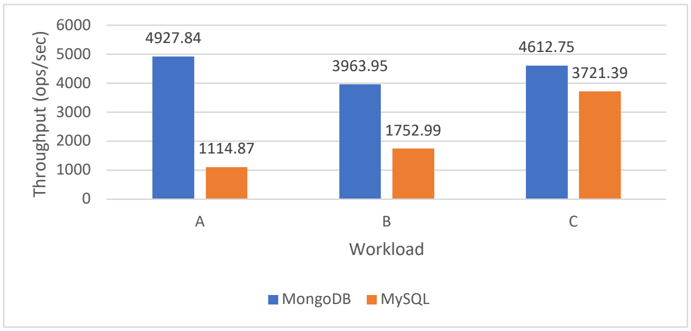


Fig. 3. Throughput (ops/second) for the three workloads.

A. Workload A

Workload A consists of 50% reads and 50% writes, plus 10,000,000 records.

The MySQL and MongoDB databases are tested, and MongoDB performs 55.77% better. However, MySQL exhibits slower performance with 95 minutes and 8 seconds, while MongoDB performs better with 42 minutes and 5 seconds.

B. Workload B

Workload B consists of 100% writes plus 10,000,000 records.

The MySQL and MongoDB databases are tested, and MongoDB performs 77.37% better. However, MySQL exhibits slower performance with 149 minutes and 49 seconds, while MongoDB performs better with 33 minutes and 82 seconds.

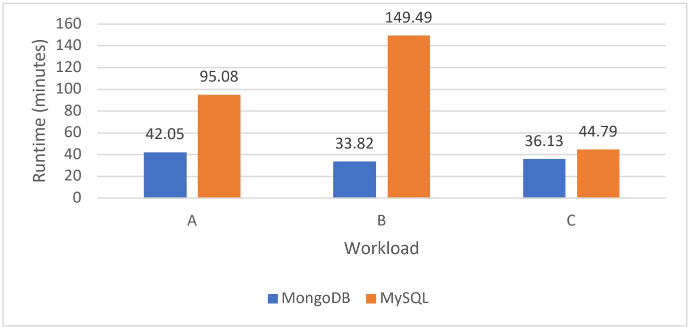


Fig. 4. Runtime for the three workloads

Metrics	MySQL	MongoDB
Workload A, 50% reads, 50% writes	When analyzing the results of execution of workload A, a good performance is not achieved by the SQL database	When analyzing the results of execution of workload A, a good performance is achieved by the Document Store NoSQL database.
Workload B, 100% writes	When analyzing the results of execution of workload B, a good performance is not achieved by the SQL database.	When analyzing the results of execution of workload B, Compare to MySQL it takes less time for write operation.
Workload C, 100% reads	When analyzing the results of execution of workload C, MySQL is slower due to C or C++ language written engines.	When analyzing the results of execution of workload C, a good performance is achieved due to JavaScript.
Time and Throughput	In this case while operation is perform, MySQL is taking more time than MongoDB.	In this case while operation is perform, MongoDB is taking less time than MySQL.

Fig. 5. MySQL vs MongoDB metric comparison

C. Workload C

Workload C consists of 100% reads plus 10,000,000 records.

The MySQL and MongoDB database are tested, and MongoDB performs 19.33% better. However, MySQL exhibits slower performance with 44 minutes and 79 seconds, while MongoDB performs better with 36 minutes and 13 seconds.

VI. Discussion and Conclusions

Although there are IT professionals who specialize in database operations, such as DBA's, a basic knowledge of performance is essential for any programmer. Since databases are the backdrop of most applications used on the Web today, including websites and applications, there is a sudden need for a professional, or even a novice, to act with SQL. Server consolidation reduces the number of physical servers by reducing the physical space in the data center. Through consolidation, organizations can reduce the number of physical servers and operating costs by reducing the power consumption required to keep them running, directly impacting the amount of cooling needed to keep those servers running at optimal temperatures.

The YCSB benchmark is used for various workloads to test the performance of the database. The results show that the MongoDB database loads, inserts, and scans data, even on an extensive database. It is because it does not check the schema and perform vital foreign checks, but when reading data by attributes and searching, it is not always faster, especially if it does not have index keys.

Exponential data growth overwhelmed the relational data model and drove the development of non-relational databases, including NoSQL. Since the database is the heart of every application, choosing the ideal database system is highly critical. NoSQL databases generally guarantee good performance for simple operations on potentially large datasets. However, we can confirm that the selection criteria for a suitable solution depend on the application requirements, the nature of the operations performed on the manipulated data, and the context in which they are used.

Using both databases in the default configuration, without optimization tuning, based on the suggested test environment, we can conclude that MongoDB achieves better results. NoSQL databases were born to meet performance needs, leaving other details like atomicity in the background. NoSQL and relational databases use different paradigms and, in turn, have different goals but the same goal: persistent data. According to performance tests, the MongoDB database is a good choice for applications with high-load database queries, such as web services. However, the MySQL database is better if the application does not require a more robust security layer with database access control.

For future work, a wide range of possibilities related to this topic, such as: tackling other relational and NoSQL databases like PostgreSQL, FireBird, and CouchDB, increasing queries using more criteria and tables, and performing backups, replication, and concurrency tests.

ACKNOWLEDGMENTS

This work is funded by FCT/MEC through national funds and co-funded by FEDER—PT2020 partnership agreement under the project **UIDB/50008/2020** (*Este trabalho é financiado pela FCT/MEC através de fundos nacionais e cofinanciado pelo FEDER, no âmbito do Acordo de Parceria PT2020 no âmbito do projeto UIDB/50008/2020*).

This article is based upon work from COST Action CA19130 (FinAI) – Fintech and Artificial Intelligence in Finance – Towards a transparent financial industry (fin-ai.eu), supported by COST (European Cooperation in Science and Technology). COST is a funding agency for research and innovation networks. Our Actions help connect research initiatives across Europe and enable scientists to grow their ideas by sharing them with their peers. It boosts their research, career, and innovation. More information on www.cost.eu.

The author affiliated with the Faculty of Computer Science and Engineering at the Ss.Cyril and Methodius University, Skopje, Macedonia acknowledges its support for the work on this article.

Furthermore, the authors also would like to thank the University of Bahrain for their support.

REFERENCES

[1] Antonio Celesti, Maria Fazio, and Massimo Villari. A study on join operations in mongodb preserving collections data models for future internet applications. *Future Internet*, 11(4):83, 2019.

[2] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154, 2010.

[3] Sandra L Emerson, Marcy Darnovsky, and Judith Bowman. *The practical SQL handbook: using structured query language*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[4] Norman E Fenton and Martin Neil. Software metrics: roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 357–370, 2000.

[5] Paola Gómez, Rubby Casallas, and Claudia Roncancio. Data schema does matter, even in nosql systems! In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–6. IEEE, 2016.

[6] Marek Grzegorowski, Eftim Zdravevski, Andrzej Janusz, Petre Lameski, Cas Apanowicz, and Dominik Slezak. Cost optimization for big data workloads based on dynamic scheduling and cluster-size tuning. *Big Data Research*, 25:100203, 2021.

[7] Adam Lith and Jakob Mattsson. Investigating storage solutions for large data - a comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data. Master's thesis, Chalmers University of Technology, 2010.

[8] V Manoj. Comparative study of nosql document, column store databases and evaluation of cassandra. *International Journal of Database Management Systems*, 6(4):11, 2014.

[9] Jaroslav Pokorny. Nosql databases: a step to database scalability in web environment. *International Journal of Web Information Systems*, 2013.

[10] Eftim Zdravevski, Petre Lameski, Cas Apanowicz, and Dominik Slezak. From big data to business analytics: The case study of churn prediction. *Applied Soft Computing*, 90:106164, 2020.

[11] Eftim Zdravevski, Petre Lameski, Ace Dimitrievski, Marek Grzegorowski, and Cas Apanowicz. Cluster-size optimization within a cloud-based etl framework for big data. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3754–3763, 2019.

[12] Eftim Zdravevski, Petre Lameski, and Andrea Kulakov. Row key designs of nosql database tables and their impact on write performance. In *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 10–17, 2016.

[13] Eftim Zdravevski, Petre Lameski, Andrea Kulakov, Sonja Filiposka, Dimitar Trajanov, and Boro Jakimovski. Parallel computation of information gain using hadoop and mapreduce. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 181–192. IEEE, 2015.