Proceedings of the International Conference on
Information Technologies (InfoTech-2018)
20-21 September 2018, Bulgaria

ANALYSIS AND COMPARISON OF DOCUMENT-BASED DATABASES WITH SQL RELATIONAL DATABASES: MONGODB VS MYSQL

Raif Deari, Xhemal Zenuni, Jaumin Ajdari, Florije Ismaili, Bujar Raufi

Faculty of Contemporary Science and Technologies, SEE University
e-mail(s): {rd20968,xh.zenuni,j.ajdari,f.ismaili,b.raufi}@seeu.edu.mk
Macedonia

Abstract: In this paper, we aim to make a comprehensive analysis and comparison between document-based and relational databases. We review and evaluate data storage and data management principles of each type of concerned databases. In addition, we evaluate the performance of CRUD operations using different scenarios on MongoDB and MySQL as two representatives of respected data models. The results give insights to advantages and disadvantages of each database model.

Key words: NoSQL, SQL, MongoDB, MySQL, evaluation, analysis, comparison.

1. INTRODUCTION

For a long time, relational databases have been a dominating and sometimes the only choice as a solution to data storage and data management problem [1]. However, in the last few years we witness an explosion of data. The Web, and especially social media and instrument sensors are majorly responsible for such severe data proliferation. This data has been rapidly increasing not only in volume, but in complexity as well. The idea of using relational model as ‘One size fits all’ solution has been questioned now if it can address these new challenges, and therefore new data models generally referred as NoSQL are explored.

Document-based databases [2] are one type of NoSQL databases that are well aligned with challenges of handling new and rapidly changing data types (structured, unstructured, semi-structured, polymorphic) coming in massive volumes. They offer great flexibility when working with schema less data, with horizontal scaling capabilities and where the data can be easily distributed and/or replicated in different nodes of a cluster. In addition, they provide efficient query mechanisms, ability to query on any field and a natural mapping of the document data model to objects in modern programming languages.

There is much discussion in the database world about when to use NoSQL and whether they will replace the relational databases. Therefore, in this paper we try to make a comprehensive analysis and comparison of document-based NoSQL with SQL relational databases. The aim is to compare and contrast main features of each database type and try to highlight situations when document-based NoSQL is a viable solution to implement.

2. RELATED WORK

There are many blogs and white papers that try to compare and analyze document-based versus traditional SQL databases, yet not to many research papers in this field. In [3], the authors analyze and compare the MongoDB and MySQL in the implementation of a forum for personal and professional development. In the same paper there is also an attempt to evaluate the performance of the selected databases for basic operations in such implementation in order to justify the selection of MongoDB over MySQL. However, the paper gives no insight on transaction support, query capabilities or security issues of such systems. In addition, the performance is not tested under concurrent access to data.

A similar attempt is identified in work by Soni et.al [4]. In this paper some of query support capabilities in MongoDB has been highlighted, yet the work lacks a serious analysis and comparison on query language support, transaction and security aspects. In [5], the authors focus on evaluating the effectiveness of MongoDB application to operate a website messages. A comparative performance analysis between MongoDB and MS-SQL is conducted, concluding that MongoDB outperforms the later database management system.

A similar work is observed in [6]. The authors consider to implement a specific database schema with three tables requiring join operations in MongoDB. Similarly, they evaluate and compare the performance of insert, update and with particular interest they focus on different types of select queries. The authors conclude with specific situations when the NoSQL is better choice compared to traditional SQL databases.

From the review of the literature, it can be concluded that in recent years there is an increased interest by scholars to compare and evaluate document databases versus traditional SQL databases. However, to our best knowledge, none of the existing work makes a comprehensive and overall comparison of such database approaches.

3. COMPARISON OF DOCUMENT STORE AND SQL DATABASES

The comparison of document-store and SQL databases has been conducted from four perspectives. First, the underlying principles of data organization of each concerned type of databases have been analyzed. Next, the support for transactions and mechanisms for data integrity has been compared. Third, the query language capabilities of each database type have been reviewed. And finally, security aspects and mechanisms in regard to fulfillment of NISTs security objectives have been explored.

3.1. Data Organization

In relational databases the data is organized based on relational data model which provides a declarative method for data and query specification. The data is stored in two-dimensional tables which are known as relations. Each relation has a set of named attributes, which can be considered as the name of table columns. Each column is associated with a specific domain, such as integer or string. The relations can contain a set of rows called tuples (record). Tuples contain N components, which corresponds to N attributes of belonging relation. The name of the relation together with its set of attributes represent the schema.

On the other side, in document databases the data is stored in documents which is a set of key-value pairs usually in JSON or Binary JSON format, which does not require predefined schema. Keys are represented as strings, and values can be of basic types (such as integer or string) or as structures (such as arrays or objects). Moreover, documents provide support for lists, pointers, embedded arrays or nested documents. This simplifies data access and, in many cases eliminates the need for expensive joins.

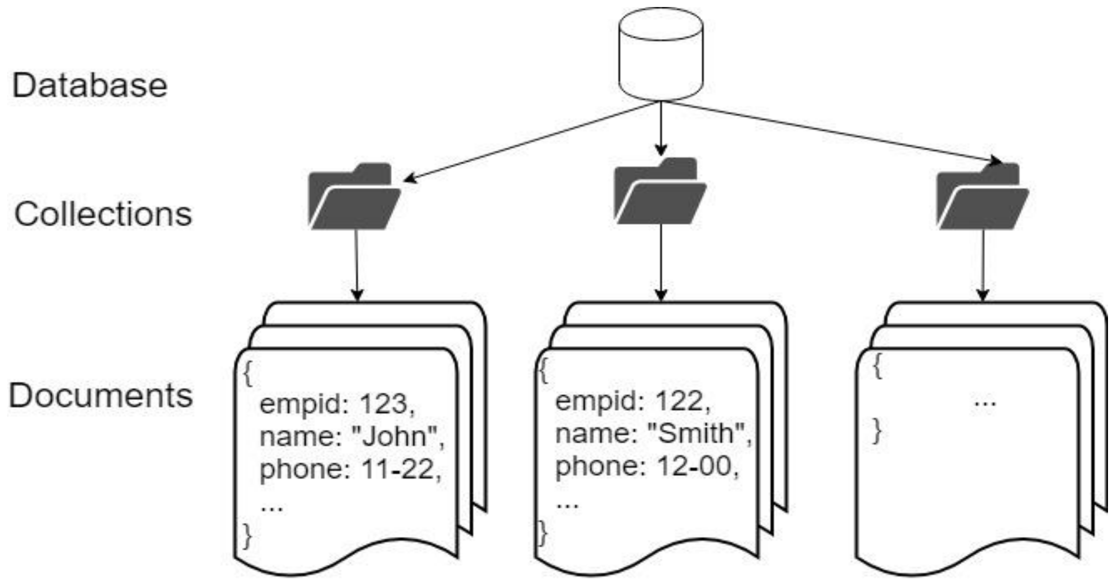


Fig. 1. The organization of data in document store databases

3.2. Transaction Support

Most of SQL databases are designed to support four basic properties in data management known as ACID (atomicity, consistency, isolation, durability) properties. These qualities contribute to the idea that the data integrity is preserved, even in cases of concurrent access to data and/or eventual failures. The ACID properties ensure that when transaction(s) will be executed on consistent database, will either complete and produce correct result, or terminate with no effect, but in both cases preserving the consistent state of data. On the other hand, NoSQL databases, including document-bases type, sacrifice ACID properties in favor of BASE (basically available, soft state, with eventual consistency) approach. This trade-off have been a critical discussion point when two types of databases are compared and discussed. However, in some cases applications can tolerate inconsistent data, as there will not be any loss when it is not provided. In some cases temporary inconsistences is an accepted trade-off when resulting in faster responses in a more scalable manner.

3.3. Query Language

SQL is a powerful query language that allows to easily manipulate the data structures and the data itself in a relational database model, in a uniform and standardized manner. On the other hand, document based databases do not use SQL, mostly for the underlying principles for data storage used in the later database model. However modern document databases, such as MongoDB [7] have also very rich query language.

Table 1 summarizes and compares the supporting features of the query languages supported in MySQL [8] and MongoDB. It can be noticed that both database systems support in an elegant way the major operations with the data, such as their creation/modification, data searching and filtering, data joining operations, or even aggregate functions.

Table 1. Major query language support constructs in MySQL and MongoDB

MySQL	***MongoDB***
CREATE, INSERT, UPDATE, DELETE, DROP	CREATE, INSERT, UPDATE, DELETE, DROP
SELECT…WHERE	SELECT…WHERE
PRIMARY KEY	PRIMARY KEY
INDEX	INDEX
JOINS	$lookup,$graphLookup EMBEDDED DOCUMENTS
GROUP_BY	Aggregation pipeline

3.4. Security Aspects

Security issues are of crucial importance and involve protecting the database and the data from unauthorized access, modification or destruction. The aspects analyzed were derived from NIST's [9] objectives and include availability, integrity, confidentiality, accountability, assurance and access control.

Table 2. Security aspects compared between MongoDB and MySQL

	MongoDB	***MySQL***
Authentication	*Username and password* *SCRAM* *x.509 Certificate Authentication* *LDAP proxy authentication* *Kerberos authetntication*	*Username and password* *SCRAM (pluggable)* *x.509 Certificate Authentication(pluggable)* *LDAP proxy authentication(pluggable)* *Kerberos authentication (pluggable)*
Authorization	*Grant/Revoke Roles and Privileges (read, write) on different levels (database, collection, ...)*	*Grant/Revoke Roles and Privileges (read, write) on different levels (database, tables,records, fields, ...)*
Transport Encryption	*TLS/SSL*	*TLS/SSL*

Table 2 summarizes the main security aspects compared between MongoDB and MySQL, but in general the following can be concluded:

- Both MongoDB and MySQL provide high availability. Replication and sharding are another important factor for MongoDB in maintaining its availability.
- SQL databases such as MySQL have been considered as huge advantage in maintaining the integrity of data through ACID properties compared to NoSQL MongoDB, which were based on BASE (basically available, soft state, with eventual consistency) principle. However, in latest version of MongoDB there is a serious shift toward the support of ACID properties as well.
- Both systems support secured communication with their clients through TLS/SSL encrypted connections.
- Both systems support logs, which can be used for auditing processes and to ensure a specific level of accountability.
- There is a lack of appropriate assurance mechanisms in both systems. User's looking for more in this regard, should look for third-party tools or systems for adequate assurance mechanisms, such as performance monitoring.
- Both systems provide fine and coarse grained access control. Access and role control is supported with GRANT and REVOKE methods, and other supplemental mechanisms such as VIEWs are implemented as well.

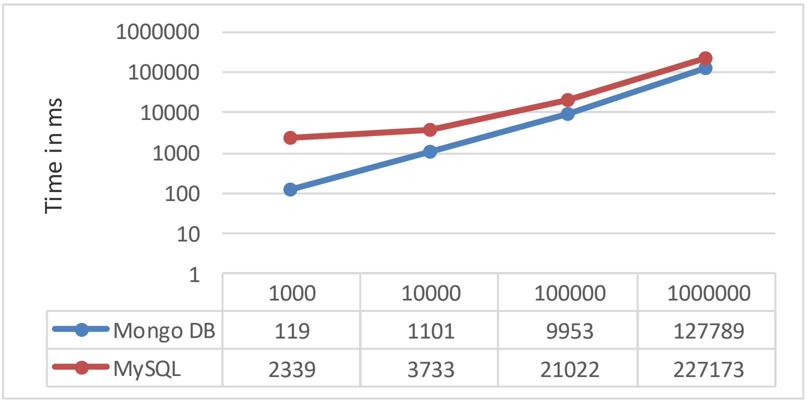
4. PERFOMANCE EVALUATION

One of the important criteria when choosing a specific database technology is its performance on CRUD (create, read, update, delete) operations. A series of performance tests or benchmarks were performed on Mongo DB 3.6 and MySQL 5.7.21 as representative systems of document – based and relational data model. These benchmarks were performed locally in a machine running Intel® Core™ i7-75000U CPU clocked @ 2.90 GHz, running Ubuntu 16.04.3 LTS 64-bit operating system, with 8GB of physical memory and 256 GB SSD disk space. Two scripts written in NodeJS were created to perform CRUD operations, one to perform on single create, read, update and delete, and the other script to perform the same operations but in concurrent mode.

Data used for these tests is a simple table holding information about a company: company id, company name and company address. All the values that are saved in different attribute fields are of the type string or integers, and these types were deliberately chosen because they are two of the most common data types used for storage in most of databases.

4.1. Insert Operation

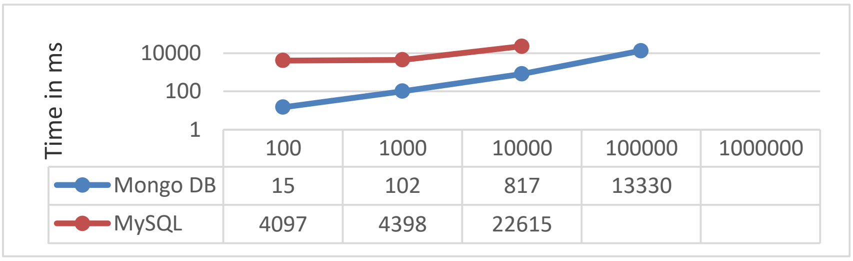
For testing data creation operation (insert operation), two tests were performed using two different NodeJS scripts. The first test is creation of multiple records but with single actions, that is creation of each record one-by-one, while the second one creates all the records concurrently.



	1000	10000	100000	1000000
Mongo DB	119	1101	9953	127789
MySQL	2339	3733	21022	227173

Fig. 2 Single insert (create) operations in empty databases

MongoDB and MySQL show similar trend in performance with creation operation on empty database, although MongoDB completes the operations in less time than MySQL, and this is true especially when working with small amounts of data.



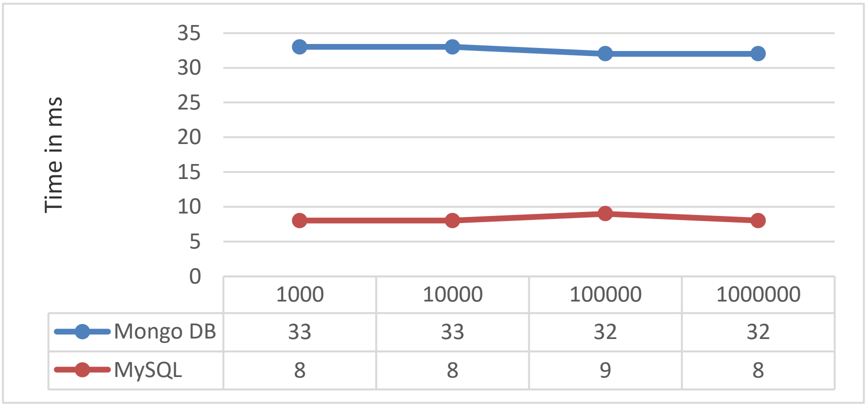
	100	1000	10000	100000	1000000
Mongo DB	15	102	817	13330	
MySQL	4097	4398	22615		

Fig. 3 Concurrent insert (create) in empty database

In concurrent record insertion, the performance evaluation shows that MongoDB is faster and can insert up to 1 million records concurrently. On the other side, MySQL crashed when attempting to insert 100000 records. MySQL was able to handle exactly 57920 concurrent insertion operations and then the insertion process failed.

4.2. Read Operation

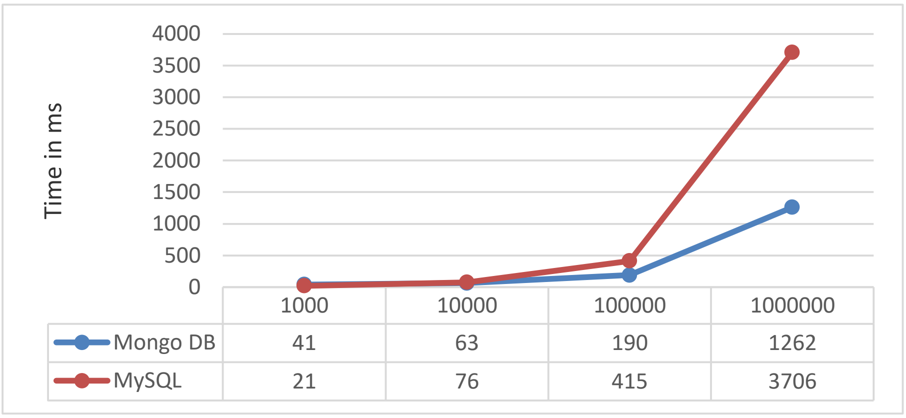
In read (select) process, the aim was to retrieve a single record both in single and concurrent mode and with different number of records in database. Figure 4 and 5 depict the performance evaluation of both systems, but in general can be seen that read operation is performed faster than creation operation.



	1000	10000	100000	1000000
Mongo DB	33	33	32	32
MySQL	8	8	9	8

Fig. 4 Reading one record with different database size(# of records)

In single mode read, both systems show similar performance trend, but this is not the case with concurrent mode and when dealing with large amount of concurrent accesses and the number of records is large. In this case, MongoDB outperformed MySQL as depicted in Figure 5.

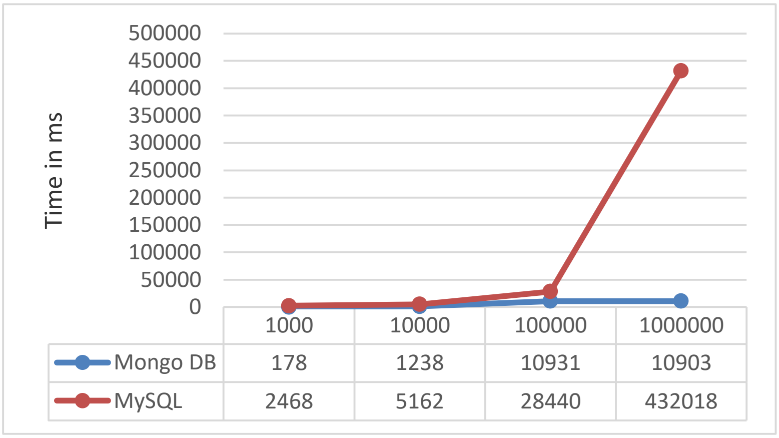


	1000	10000	100000	1000000
Mongo DB	41	63	190	1262
MySQL	21	76	415	3706

Fig. 5 Concurrent read in database populated with different number of records

4.3. Update Operation

Two update scenarios were performed. First, all records from the database were updated in single mode. Figure 6 and 7 depict the performance of each system, but it can be observed again that the difference becomes more noticeable when dealing with large amount of existing data in database.



	1000	10000	100000	1000000
Mongo DB	178	1238	10931	10903
MySQL	2468	5162	28440	432018

Fig. 6 Single mode update of all records in database when changing database size

When concurrently updating and the number of records in databases increases, both systems failed to complete the updating process, once the number of concurrent accesses achieved 100000 in a database populated with over 100000 records. Table 6 depicts such situation.

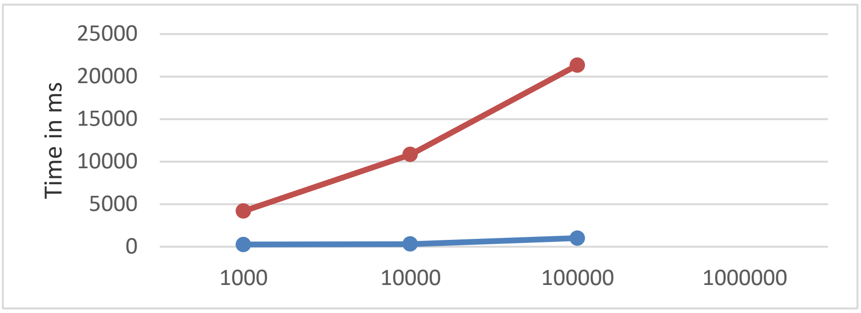
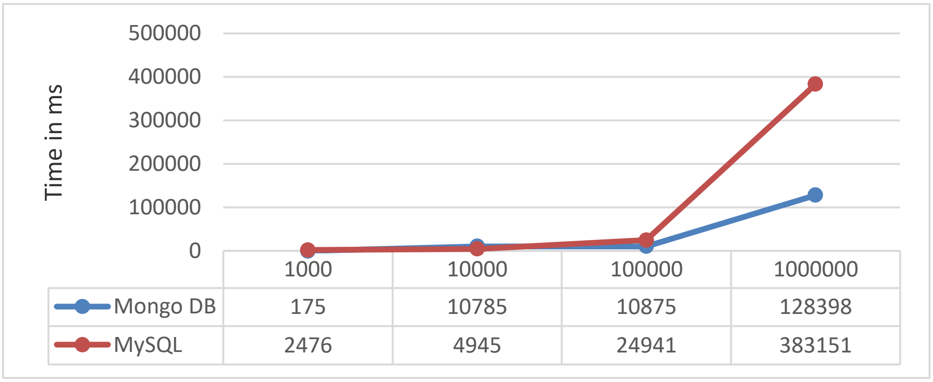


Fig. 7 Concurrent update when changing database size

4.4. Delete Operation

Similar to other operations, both MongoDB and MySQL show similar performances when working with small amount of data. The gap in performance becomes noticeable when dealing with larger amount of data. The deletion process have been tested in different database sizes when deleting all records in single mode as depicted in Figure 8.



	1000	10000	100000	1000000
Mongo DB	175	10785	10875	128398
MySQL	2476	4945	24941	383151

Fig. 8 Single mode delete of all records in database when changing database size

5. DISCUSSION OF RESULTS

SQL and document-based databases are complementing approaches rather than excluding or replacing technologies in data management processes. The analysis and comparison highlights the advantages and disadvantages of each type of concerned databases, and it tries to determine the situations when to move from traditional SQL databases to document-type.

MongoDB is a flexible schema-less database that can be implemented in distributed environment. It is well positioned in situations when the data is not strictly structured and not complex to handle. In addition, it can scale horizontally easily using sharding and can be implemented in cloud services.

If data integrity is a major concern, than MySQL would be the choice as it implements ACID transactions. Although it is announced that the latest versions of MongoDB will implement a sort of ACID, in general to achieve better scalability and performance, they have weaker concurrency model implemented known as BASE.

In document – based databases there is no real use of JOIN operation like in the relational model. Documents can be nested inside other documents, and $lookup can be used to work with aggregate data, yet it encourages data redundancy and it is not real implementation of classical JOIN operation.

MongoDB shows better performance in CRUD operations especially when working with large data and having concurrent access to it. But, the benchmark analysis was conducted in simple data scheme. It will be interesting to run similar experiments with larger and more complex schema, and implemented in distributed environment. We theorize that MongoDB will continue to show better performance compared to MySQL, however this need to be supported with benchmarks.

6. CONCLUSION

In recent years, there is a trend of adopting databases other than traditional SQL ones in order to address the challenges for efficient storage capacities, high scalability and concurrency, high availability, and reduced management and operational cost. Document based databases, such as MongoDB have gained momentum in this shift, therefore in this paper we aim to compare and contrast these emerging databases compared to traditional SQL ones in a comprehensive manner. Overall comparing, MongoDB is a serious competitive/complementary database compared to MySQL, and in some aspects and certain situations it outperforms SQL databases, and thus becomes a database of choice.

REFERENCES

[1] Elmasri, R. and Navathe, S (**2011**). Fundamentals of Database Systems, Addison-Wesley.

[2] Sullivan D (**2015**). NoSQL for Mere Mortals, Addison-Wesley.

[3] C. Győrödi, R. Győrödi, G. Pecherle and A. Olah **(2015)**, "A comparative study: MongoDB vs. MySQL," *13th International Conference on Engineering of Modern Electric Systems (EMES)*, Oradea, 2015, pp. 1-6.

[4] Soni, S. et al (**2017**). A Comparative Study: MongoDB vs MySQL. In: *International Journal of Scientific & Engineering Research Volume 8, Issue 5*, pp.120-123.

[5] Wui, C.M. et al (**2015**). Comparisons Between MongoDB and MS-SQL Databases on the TWC Website. In: *American Journal of Software Engineering and Applications*, SciencePG, pp.35-41.

[6] Parker, Z. et al (**2013**). Comparing NoSQL MongoDB to an SQL DB. In: *Proceedings of the 51st ACM Southeast Conference Article No. 5*, ACM Publishing, pp.1-6.

[7] MongoDB for GIANT Ideas | MongoDB (**2018**). https://www.mongodb.com/

[8] MySQL (**2018**). https://www.mysql.com/

[9] STONEBURNER, G (**2001)**.Underlying technical models for information technology security. Technical Report.