Performance evaluation of SQL and MongoDB databases for big e-commerce data

Seyyed Hamid Aboutorabi[a], Mehdi Rezapour[b], Milad Moradi[c], Nasser Ghadiri[d]
Department of Electrical and Computer Engineering
Isfahan University of Technology
Isfahan, Iran
[a]h.aboutorabi@ec.iut.ac.ir, [b]mm.rezapour@ec.iut.ac.ir, [c]milad.moradi@ec.iut.ac.ir, [d]nghadiri@cc.iut.ac.ir

***Abstract*—With the advent of big data phenomenon in the world of data and its related technologies, the developments on the NoSQL databases are highly regarded. It has been claimed that these databases outperform their SQL counterparts. The aim of this study is to investigate the claim by evaluating the document-oriented MongoDB database with SQL in terms of the performance of common aggregated and non-aggregate queries. We designed a set of experiments with a huge number of operations such as read, write, delete, and select from various aspects in the two databases and on the same data for a typical e-commerce schema. The results show that MongoDB performs better for most operations excluding some aggregate functions. The results can be a good source for commercial and non-commercial companies eager to change the structure of the database used to provide their line-of-business services.**

Keywords—NoSQL; SQL; Big data; E-commerce; Database; MongoDB; Performance evaluation

I. Introduction

Traditional databases are based on the relational model for storing data, and they were named the SQL databases after that the SQL query language was used for querying them [1]. However, in recent years non-relational databases that are known as NoSQL databases, have been highly regarded.

With the increasing use of the Internet and the availability of cheap storage, massive amounts of structured, semi-structured and unstructured data are created and stored by a variety of applications. Usually such data in large-scale is known as big data [2]. Processing large amounts of data requires fast machines, flexible database schemas and distributed architectures that do not fit into relational databases. NoSQL databases claim that they provide easy access, high speed and development capabilities for working with large data. On the other hand, there are many options proposed as commercial and open-source NoSQL databases [3]. The variety of options available, brings us to the question of the difference between these databases and their capabilities for different applications. Good review articles [4] and [5] have dealt with this matter, as well as online resources of sites and blogs on the subject that show the emerging need for higher performance.

In general, when the dataset has no specific structure and occupies a huge amount of storage, the NoSQL database engines are a good choice. But for the structured datasets of moderate size, it cannot be said with certainty the NoSQL databases perform better than their SQL counterparts. The following is a brief overview of the differences between NoSQL databases and SQL that describes and specifically addresses the differences between MongoDB and SQL Server.

The data records in a relational model database are represented as a schema. The related data recorded in this structure are grouped in tables as rows and columns, and each row has the same number of columns.

The relational database tables help the designer to avoid data redundancy when the tables are normalized. The normal result is a multi-table design. The queries usually require a combination of the tables and merging the information in the tables. For this reason, the join operation should be used. Join operations need to define foreign keys and this imposes a large overhead to the database. The larger database schemas and higher number of tables will take much more time for responding to a query and returning the results.

NoSQL databases come to our help where their SQL counterparts face performance problems. First, there is no need for unstructured data schema definition to be predetermined. The second feature is that the ACID transaction properties of the traditional SQL databases are ignored or receive less attention in NoSQL databases. This will lead to higher performance in such databases. In MongoDB engine, the ACID properties are replaced by the BASE architecture. More description of this architecture item can be found in [6]. On the other hand, the concepts of joins and transactions are not supported in NoSQL databases due to their specific architectures.

Our focus in this paper is on the comparison between a well-known document-oriented type of NoSQL databases i.e. MongoDB with the Microsoft SQL Server as a relational database. We investigate the fundamental differences between these two database management systems in terms of performance of processing the queries. Our selected queries are run on the same dataset with the same number of records

978-1-4673-9181-8/15/$31.00 ©2015 IEEE

(record in SQL is identical to document in MongoDB) and the operations are Read, Write, Delete, and Select, as well as aggregate functions.

MongoDB is selected since its functionality is good, despite the fact that it has been assumed as a new market entry of NoSQL databases, with many projects and products based on this engine such as the MTV network, the Craiglist (one of the most well-known sites in online advertising), Disney Media Group, Sourceforge, which is a repository of open source codes, the Wordnik dictionary site, the Guardian news, Forbes business magazine, the New York Times, the GitHub project, the FourSquare mobile social network as a discovery and search service and the blogs and comments managed with Disqus as well as many other projects. We can add to this list the EA Sports, Yandex, ebay, Square Enix, Sailthru, AHL, Qihoo, Global Financial Services Company, McAfee, Adobe, CARFAX and also many of the most successful and innovative web companies. Each of them have justified the use of MongoDB by specific reasons. For example, the Guardian News uses MongoDB because by using MongoDB it would be much easier to store documents. And many of them use MongoDB for its scalability and its usability in distributed environments such as clusters. MongoDB tackles the scalability problem in three dimensions i.e. cluster scale, performance scale and data scale.

It is generally believed that the BigTable database [3], which was created and used by Google, was the first NoSQL database. The successful operation of BigTable in the database domain has lead organizations to future use of other NoSQL databases. A feature that most of such databases use is the key-value pairs for storing data. The key-value databases are divided into two groups of column-oriented databases and document-oriented databases. The MongoDB database is document-oriented and is placed in this category. Just like other databases in this category, MongoDB is capable of storing objects. Objects can be in well-known XML, JSON, or BSON (binary coded form of JSON) formats.

The MongoDB database, which its first version was released in 2009, has been written in C ++ [7]. The objects are stored in the BSON format and do not require the same structure and schema to store objects. This database is capable to set the data to automatically partitioning mode. By using this feature, a large dataset like an e-commerce company data can be segmented into several parts and each part is stored in a separate server to distribute the load evenly. MongoDB database focuses on four characteristics of flexibility, strength, speed and ease of use. As well as, several drivers are available to work with the database by a variety of programming languages. To see the syntax of the commands and how to work with the database one can check the website of the database.

The rest of the paper is organized as follows. In Section II, the related works that have been done are discussed. Section III presents the design of our experiments and Section IV shows the results and explains the issues that have been evaluated. Finally, the conclusion of our discussions and experiments is presented in section V.

II. Related Works

In recent years, there have been a lot of general discussions in blogs and non-scientific articles about the comparison of SQL and NoSQL databases. As far as we know, very few research papers have worked on such comparison with simple experiments and small to moderate datasets. In this section we will refer to some of the related works. In one of the related works [1] the authors performed their testing in a C# console application. They perform insert operations in four different levels for three tables and different number of records. The test for the update has used three different types of updates. Also, some kind of query that has been considered at both simple and complex levels has been performed on tables. The results of running on a high number of records show that MongoDB is better with a high difference compared to SQL. Also, in the update for the indexed mode and non-indexed results obtained the reverse fact i.e. the superiority of SQL in the indexed mode. In [2], authors investigate a comparison with the functional view of the implementation of the management of digital books using the databases MySQL and MongoDB. The results also show the superior performance of MongoDB for insert and select operations. In [3] to compare the speed of the databases in several areas of the insert and delete and update, tests have been done. For performing these operations in each phase of test, a table with three columns with different scales has been used. The final results in all three areas has shown a higher performance with MongoDB.

One of the existing works in this area refers to the lack of benchmark tools for NOSQL databases [4], and suggests a benchmark for MongoDB and MySQL databases to evaluate their performance. Tests for measuring database performance has been listed three different scenarios, including operations OLTP, OLAP, and operations of the web 2.0 as the basis of its assessment. The process of generating initial data for evaluating the DBMS of interest, generating commands to read and write operations and measuring the time taken to perform each of these procedures has been done by means of Visual C# for each of the databases and they have been compared individually and the results have been recorded. The results of the benchmark represent MongoDB is a better engine for both read and write operations.

In [5] a comparison between several NoSQL and SQL databases in specific applications and for the implementation of key-value storage is done. While NoSQL databases are generally designed to optimize the storage of key-value, it was shown in their paper that NoSQL databases in this area do not perform better than SQL databases. The comparison was between the MongoDB, RavenDB, CouchDB, Cassandra, HyperTable and CouchBase databases that all are from the NoSQL family and SQL Server as a relational database. The operations were Instantiate, Read, Write and Delete. The NoSQL engines provided different results for different operations, and none of them was superior in all aspects. In this comparison, However, MongoDB has shown more stability and relatively good performance in four types of operations.

In another related work [8], a comparison between MongoDB and MySQL has been done and the authors have shown how to integrate these two databases by a middleware layer between the application and database layers. The paper also compares the basic concepts and expressions used by two databases, as well as, performs some tests for evaluating the speed of these two databases in terms of insert operation, simple and complex queries. This paper suggests use of MongoDB for those applications that are data-intensive and store and query large amounts of data.

With emerge of some needs such as maintaining huge volume of unstructured data, high availability and scalability, and with emerge of NoSQL databases and their proven superiority, large businesses and enterprises have to migrate from traditional models, particularly relational model, to new NoSQL models. These new models must be identical to original models in terms of semantic and conceptual structure. [9] addresses the diversity of data models in the world of NoSQL databases and investigates common NoSQL databases in terms of their types (i.e. key-value, column-oriented, document-based, etc.), their data model and their architecture. Identifying available new data models can be helpful for lossless migration from relational model to NoSQL models. In [10], authors present a framework named NoSQLayer to perform data and model migration from MySQL (as a relational database) to NoSQL databases, more specifically MongoDB, in a transparent and automatic manner. Their proposed layer maintains the entire structure of the original database and stores all data as a NoSQL model. All requests for performing SQL operations, converting them to the NoSQL requests, performing them on the NoSQL database, converting the results to the SQL standard format and returning the results are handled by the abstract layer (i.e. NoSQLayer). In section III, we express that how we carried the database from relational data model to document-oriented data model by a simple manner. This migration was appropriate for our experiments and we didn't lose original data and original conceptual structure. However, used migration depends on underlying conceptual structure, source model and destination model and might be inappropriate for some other applications.

In this paper, our focus is on evaluating the performance of the SQL Server compared to MongoDB for a relatively large database with a real-world schema that is common to e-commerce applications.

III. The Evalution Method

As discussed in section I, the aim of this research is to answer the questions that many of the users of relational and non-relational databases have been encountered with them, after the advent of NoSQL databases or even those have raised before that. Here are some important points to keep in mind. The first is that the structural nature of each of these databases should be considered. The NoSQL databases, and specifically MongoDB, do not follow the relational model. MongoDB can consider the all of stored data as a collection, without facing with some problems like empty or NULL fields. But on the other hand, there are a concept and procedure in relational databases called normalization, which considered for resolving some anomalies like redundancy and empty fields. Therefore, if we want to compare these two database engines, we must evaluate each one under its standard circumstances and under the data model that the database is designed for it. This means that the used database for evaluation of MongoDB must follow the common cases of data volume that are used in [7, 11]. And on the other hand, the used database for evaluation of relational database must follow the standard definition for such databases, i.e. normalized database (if needed up to 3NF or even more). This important issue, unlike those comparisons were done in [1, 2, 3 and 5], is essential for the accuracy and reliability of study.

The second important point is that in order to our experiments be identical to actual mechanism of large businesses and enterprises as much as possible, we must consider an actual simulation of real model of transaction registration used by these companies' database engines in comparisons that are performed between the two databases in terms of basic operations such as INSERT, DELETE and UPDATE. In fact, when we know the data usually do not insert together into their database's tables, we also must avoid using optimal instructions such as Bulk Insert to insert data at once, which are more efficient in terms of execution time, and only we must use the standard INSERT command for each row (or record). We have same argument for another basic commands. It should be noted that a server can be exist and be used simultaneously by thousands of clients.

Finally, the third important point is that in researches that their goal is comparison between two or more systems, it should be considered that in order to putting down any ambiguity related to different processing overhead of interaction interfaces of target systems, a common interface should be used for all systems. Notice that no matter this interface will work with how much overhead, because the task is comparison and is not measuring the strength of each individual system. Since we are trying to study more realistic statistics as well, the standard JDBC interface is used for both data storage engines. Of course, other well-known standard interfaces (such as ODBC) also could be an option for use in this study.

Considering the above, we can explain how the comparison of MongoDB and SQL Server is done. In this comparison and in the relational database, we have used a symbolic dataset of a business enterprise. This database has been achieved in the standard mode with eight tables after normalization to 3NF level. The tables and the relationships between them are shown in Fig. 1. As can be seen, we consider various and different links between tables in order to use queries with different levels of complexity.

For the INSERT, DELETE and UPDATE we use one of these tables, and we apply the commands with the scale of 100, 1,000, 10,000, 100,000, and 1,000000 records. We consider two experiments for the queries. In the first experiment, we use five mentioned scales to impact of difference in the scale range of commands be fully specified. The second experiment is performed using the comprehensive queries, as in each query a certain number of tables are included. It causes that the execution time differences relevant to number of joins between tables that may be exist in queries be considered in the obtained results from these comparisons. As well as, in the second experiment we have divided the problem into two parts, including queries without aggregate functions and queries with aggregate functions. We investigate aggregate functions individually in several queries with using three functions Count(), SUM() and AVG(). The used queries will be listed in the experiment results.

To evaluate the MongoDB database, we perform our desired operations on the same data used in the evaluation of Ms SQL Server database. The difference is that the data were transferred to the database in the format of JSON files. We converted each table of relational database to a collection in MongoDB and selected the fields of each collection exactly same as the fields of each table in relational database. For example, JSON format of one record in the Customer table is shown in Fig. 2.

Similar to experiments were performed on SQL Server database, the Java programming language was used to run the desired operations and measure the time taken to perform them. Also we used the framework that JDBC provides for communicate with various databases, along with the Java driver for connecting to MongoDB, in order to perform desired operations.

Because there is no join concept in MongoDB and programmer is responsible for establish communication between the collections, we could use two approaches to bring

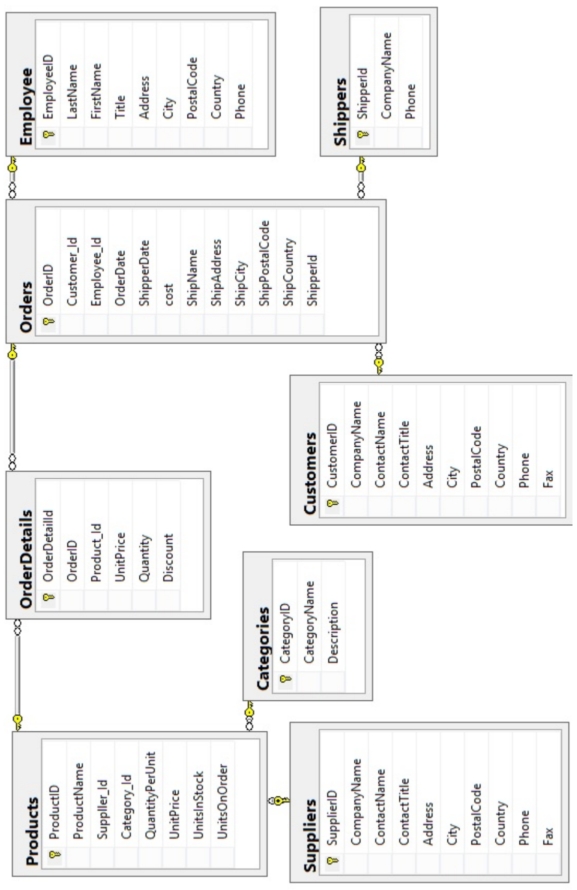

Fig. 1. The schema used for MS SQL Server database

```
{
  "CompanyName":"Tambee",
  "ContactName":"Fowler",
  "ContactTitle":"Graphic Designer",
  "Address":"75461 Di Loreto Street",
  "City":"Erfurt",
  "PostalCode":"99089",
  "Country":"Germany",
  "Phone":"4-(301)503-4931",
  "Fax":"3-(520)813-5586"
}
```

Fig. 2. An example of one record in the Customer table in JSON format

the data from SQL to MongoDB. The first was that all the tables be stored in a collection and all operations be performed on that collection. This method is good, but in communication between tables in queries that performed on more than one table, this method could be faced with some problems. For this reason, we have used the second approach, as we create a collection in MongoDB for each table in SQL in order to implement the join concept in multi-table queries easily. It should be noted that, with using this approach for bringing database from SQL to MongoDB, we didn't lose any data or conceptual structure of original database. We will show in section IV that the number of participated tables in such queries do not affect the processing time in MongoDB experiments.

Experiments include the insert, update, delete, simple select and comprehensive queries on the database. First, the SQL language queries were designed to run on Ms SQL Server database. Then the queries were converted to same queries in MongoDB syntax in order to run in Java driver for MongoDB.

In the insert operations, we inserted the files with size of 100, 1,000, 10,000, 100,000 and 1,000,000 records into a collection, as each record was read separately from the data file and was inserted into the collection in the form of a document. We have measured the total time taken to insert all desired number of records and will compare the results in next section.

As noted above, because there is no join concept in MongoDB database, we manually established in Java code the relationship between collections in those queries that need the relationship between several tables. In such queries that several tables communicate with each other, one query must be performed for each collection, and desired results from joining the tables can be achieved by common fields that these collections have with each other. So we provide the ability of performing complex queries that several tables are participate in them. As a simple example, in order to communicate between the Products table and Suppliers table, so as to connect these two collections by a SupplierID, two queries must be performed in such a manner that is shown in Fig. 3.

In order to evaluate the performance of the query operations, several queries with different degrees of complexity have been performed on MongoDB database, which equivalent queries in SQL language have been performed on Ms SQL Server too. We will show the comparison of the time taken to run the queries on both databases in experiment results.

```
DBCollection coll1 = db.getCollection("product");
DBCollection coll2 = db.getCollection("supplier");

BasicDBObject query1 = new BasicDBObject(
        "ProductID", 30);
DBCursor cursor1 = coll1.find(query1);

   while(cursor1.hasNext()) {
       obj1=cursor1.next();
       Number supid = (Number) obj1.get(
               "SupplierID" );
       BasicDBObject query2 = new BasicDBObject(
               "SupplierID", supid);
       DBCursor cursor2 = coll2.find(query2);
   }
```

Fig. 3. A sample Java code for manually joining two collections in MongoDB

IV. Experiment Results

In order to evaluate the databases we used a system with an Intel Core i5 4200m CPU with 6GB of DDR3 1600 main memory and 1TB 5400 rpm hard disk and Windows 8.1 Professional 64bit operating system.

As mentioned in previous section, we used JDBC for connecting to both databases and prevent the graphics to have a negative impact on results. The used schema and data are fully meaningful, so meaningful queries can be designed. Each of queries and basic operations such as insert, delete and update performed 100 times in most cases, and 10 times in some cases, and the average of those runs for each operation or query is reported, in order to illusive results be avoided. In overall, we have provided circumstances in our study so that commercial and non-commercial companies interested in the results of this research would have a relatively comprehensive source for their future decisions.

In the following, we have assigned one subsection to each experiment, which desired operation or query is expressed and the results are illustrated in that.

A. INSERT Operations

Insert operations have been done in both databases in five scales of 100, 1,000, 10,000, 100,000 and 1,000,000 records. To insert data, only one of tables (collection in MongoDB) named Customers have been used. The results show that MongoDB performs INSERT operations with higher speed in all five scales, and with the increase in number of records this superiority would be more dominant. The results are shown in Fig. 4.

B. DELETE Operations

In this subsection, we perform delete operations with the five scales that mentioned in the previous subsection. As can be seen from the results, MongoDB database engine has absolute superiority. The results are shown in Fig. 5.

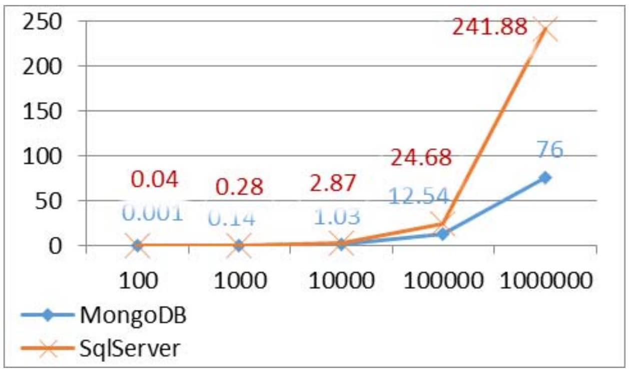

Fig. 4. Comparing the performance of inserting the records at different scales (times are in milliseconds)

C. UPDATE Operations

The update operations is also have evaluated in these experiments. The results in Fig. 6 show that MongoDB outperforms SQL by an order of about 4 when the number of records reach a million.

D. SELECT Operations

In this subsection, you can see the results of SELECT operations on the Customers table in Fig. 7. MongoDB could record less time than SQL Server in these experiments too. Keep in mind that most of these tests are performed 100 and some 10 times and the averaged results are shown in Fig. 7.

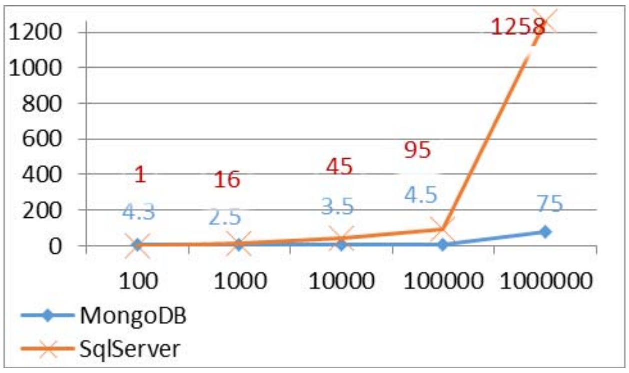

Fig. 5. Comparing the performance of deleting the records at different scales (times are in milliseconds)

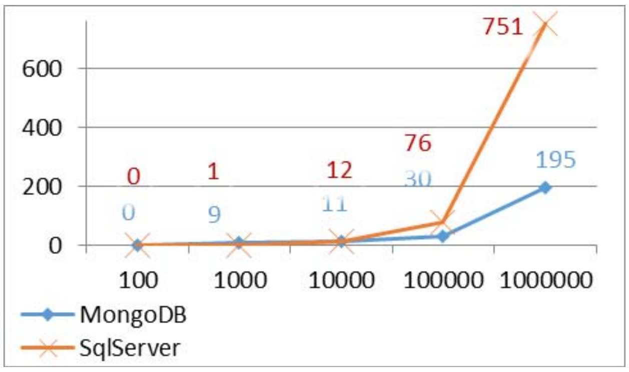

Fig. 6. Comparing the performance of updating the records at different scales (times are in milliseconds)

E. *Querying in different modes*

With regard to the query on the data, as mentioned earlier, we have divided the queries (without aggregate functions and with aggregate functions), and will review the results.

1) *Queries without aggregate functions*

In this queries we have tried as much as possible the different levels of complexity. In fact, with considering the different number of joins and output records as well as some other issues, we provide different levels of complexity and would see wider results. The English queries identical to the performed queries are as follows:

Q1. Name and country of brands that their "UnitsInStock" for this type of product is more than 100 units.

Q2. Name and address of vendors who have customers from Iran.

Q3. Name, address, city and country of vendors who their sold product belongs to the Russia.

Q4. Name and telephone number of shipping companies that have not had any instances of "lovenox" product for transmission.

By comparing the results obtained from non-aggregate queries in Fig. 8, we can say almost there is no doubt that MongoDB has higher performance than SQL Server in queries with such structure.

In the next test we will see how the results will be converse and SQL Server will win against MongoDB in aggregate operations. Of course, the weakness of MongoDB in such queries has been shown also in [12].

2) *Queries with aggregate functions*

The trial against the previous experiments has obtained interesting results that indicate superiority of SQL Server in applying three aggregation functions. We performed three queries in this experiment using three aggregation functions separately, i.e. count, sum and avg. the goal of this experiment is evaluation of the two databases in terms of aggregation operations. Queries raised in this experiment are as follows:

Q1. The total number of employees of the trading system that have Iranian nation.

Q2. The total number of products in stock that they have been ordered more than 500 units.

Q3. Average discounts given to different orders of products, for each product separately.

The results of running the queries can be seen in Fig. 9.

Times that shown in Fig. 9, indicate the result of 100 times running each query, that we have shown the average of them.

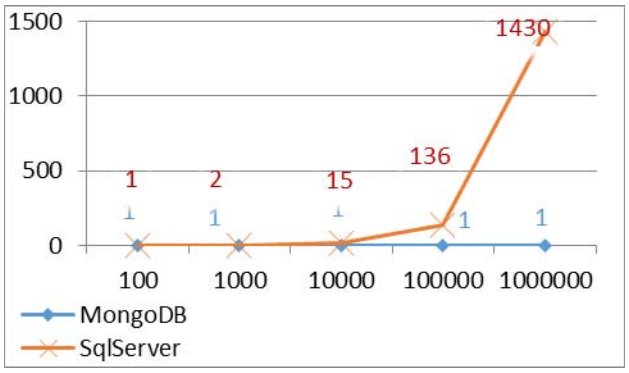

Fig. 7. Comparing the performance of selecting the records at different scales (times are in milliseconds)

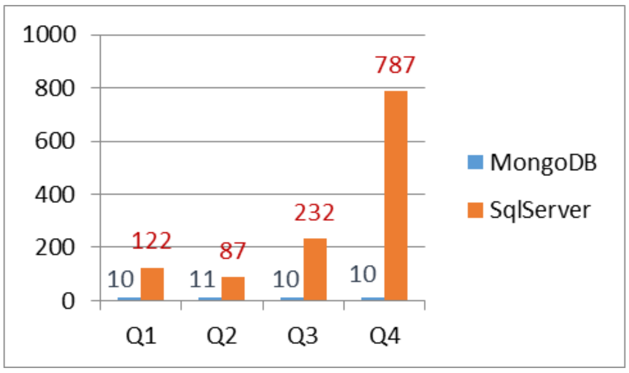

Fig. 8. Comparing the processing time of four queries in two databases (times are in milliseconds)

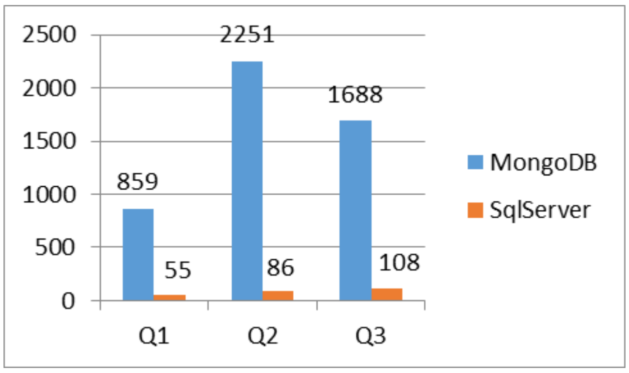

Fig. 9. Comparing the processing time of four aggregate queries in two databases (times are in milliseconds)

V. CONCLUSION

MongoDB provides good flexibility in database development process and in this database engine the data environment is able to be developed horizontally. Data with different complexities can be inserted into this database in the form of one field. MongoDB can be used in circumstances that data are very large or database structure is changed continuously. In fact, MongoDB can easily handle dynamic schemas such as document management systems that consist of several dynamic fields and only a few searchable fields.

For those users that use a relatively stronger and more structured schema, MongoDB still presents better results in terms of performance and speed in many cases. However, it should be noted that the weakness of MongoDB occurs when aggregate functions are done on non-key attributes. The type of data still needs further discussion as well as the strategy of the company that will use the database.

In future works, we plan to evaluate the application of MongoDB and SQL in a distributed environment to evaluate their performance, especially when the in-memory mode of relational databases is used.

REFERENCES

[1] Z. Parker, S. Poe, and S.V. Vrbsky, "Comparing NoSQL MongoDB to an SQL db," In Proceedings of the 51st ACM Southeast Conference, p. 5. ACM, 2013.

[2] Z. Wei-ping, L. Ming-Xin, and C. Huan, "Using MongoDB to implement textbook management system instead of MySQL," In Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, pp. 303-305. IEEE, 2011.

[3] A. Boicea, F. Radulescu, and L.I. Agapin, "MongoDB vs Oracle database comparison," In 2012 Third International Conference on Emerging Intelligent Data and Web Technologies, pp. 330-335. IEEE, 2012.

[4] I. Lungu, and B.G. Tudorica, "The development of a benchmark tool for nosql databases," Database Systems Journal BOARD (2013), 13.

[5] Y. Li, and S. Manoharan, "A performance comparison of sql and nosql databases," In Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on, pp. 15-19. IEEE, 2013.

[6] D. Pritchett, "Base: An acid alternative," Queue 6, no. 3 (2008): pp. 48-55.

[7] X. Wang, H. Chen, and Z. Wang, "Research on improvement of dynamic load balancing in MongoDB," In Dependable, Autonomic and Secure Computing (DASC), 2013 IEEE 11th International Conference on, pp. 124-130. IEEE, 2013.

[8] S. Khan, and V. Mane, "SQL support over MongoDB using metadata," International Journal of Scientific and Research Publications 3, no. 10 (2013).

[9] R.P. Padhy, M.R. Patra, and S.C. Satapathy, "RDBMS to NoSQL: Reviewing some next-generation non-relational databases," International Journal of Advanced Engineering Science and Technologies 11, no. 1 (2011), pp. 15-30.

[10] L. Rocha, F. Vale, E. Cirilo, D. Barbosa, and F. Mourão, "A framework for migrating relational datasets to NoSQL," Procedia Computer Science 51 (2015), pp. 2593-2602.

[11] A. Kanade, A. Gopal, and S. Kanade, "A study of normalization and embedding in MongoDB," In Advance Computing Conference (IACC), 2014 IEEE International, pp. 416-421. IEEE, 2014.

[12] A. Faraj, B. Rashid, and T. Shareef, "Comparative study of relational and non-relations database performances using Oracle and MongoDB systems," International Journal of Computer Engineering and Technology 5, no. 11 (2014), pp. 11-22.