A comparative study: MongoDB vs. MySQL

Conference Paper · June 2015

DOI: 10.1109/EMES.2015.7158433

4 authors:

Cornelia Győrödi
University of Oradea
54 PUBLICATIONS **239** CITATIONS
SEE PROFILE

Robert Gyorodi
University of Oradea
54 PUBLICATIONS **224** CITATIONS
SEE PROFILE

George Pecherle
University of Oradea
23 PUBLICATIONS **91** CITATIONS
SEE PROFILE

Andrada Olah
University of Oradea
2 PUBLICATIONS **70** CITATIONS
SEE PROFILE

A comparative study: MongoDB vs. MSSQL

Cornelia GYŐRÖDI, Roxana Sotoc

Abstract - As technology nowadays is tireless and evolves more and more, every day, we need to keep up with it. This is why we need to be really carefully when we want to choose a database for the application that we will want to create. We should take in consideration main factors like the amount of data, the budget, the amount of transactions that will be made and how frequent they are called. In this paper we present a comparative study between relational databases and non-relational databases. To accomplish this, we created an application about *population records*. This will let us to manipulate a big amount of data. For the non-relational database we used MongoDB and for the relational database we used MSSQL 2014.

Index Terms – MongoDB, MSSQL, NoSQL, non-relational database.

I. Introduction

These days, the companies, depending on the application that they want to develop, they have the possibility to choose the most suited database from a wide range of databases. Generally, for smaller and medium applications an SQL database will be chosen and for big applications which use and manipulate a large quantity of data, a NoSQL database will be chosen. Of course, these are not the only criteria for choosing a database, but it depends on each company and the scope of the application that needs to be developed.

According to the article written by Matt Asay, “NoSQL databases eat into the relational database market”, the NoSQL databases, especially MongoDB, occupy more and more space on the market, but with all these Oracle and SQL Server are still constant. The popularity of MongoDB can also be seen in the next image where is represented its evolution from 2014 to 2015.

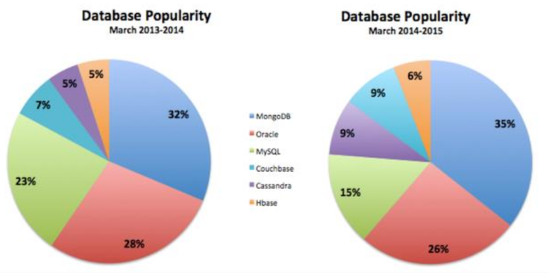

Fig. 1-1 Database Popularity [1]

In Fig. 1-1 we can see a growing of 3% for MongoDB from 2014 until 2015 and a decrease of 2% for Oracle and 8% for MySQL. This means that now the companies are oriented to NoSQL databases, so they can manipulate more data at a low price and that is exactly what MongoDB can offer and more than that. [1]

The NoSQL can be categorized in 4 types:

1. Key-Value databases – which are the simplest NoSQL data stores to use, from an API perspective. The most popular are Redis, Riak, etc.

2. Document databases – which stores and retrieves documents as XML, JSON, BSON and so on. The most popular document database is MongoDB, which provides a rich query language.

3. Column family stores – these databases store data in column families as rows that have many columns associated with a row key. One of the most popular is Cassandra.

4. Graph Databases – which allows you to store entities and relationships between these entities. Between this type of NoSQL database we can mention OrientDB, FlockDB, etc. [2]

A big advantage of non-relational databases is that they are more scalable and provide superior performance and their data model addresses several issues that the relational model is not designed to address like large volumes of structured, semi-structured and unstructured data, agile sprints, quick iteration, frequent code pushes, object-oriented programming, efficiency, monolithic architecture and so on. [3]

II. APPLICATION DEVELOPMENT USING MongoDB VS. MSSQL

We created a comparative study between relational databases, namely MongoDB, and non-relational databases, namely MSSQL. The study is based on the implementation of a website for population records, which needs to manipulate a big amount of data.

Through the application we make operations of SELECT, INSERT, UPDATE and DELETE on the registered people. These operations were made on 1, 100, 500, 1.000, 5.000, 10.000, 25.000 and 50.000 of records. The application have been developed in ASP.NET MVC 4 with C# programming language and we made parallel methods for both MSSQL and MongoDB databases. For MongoDB we used the MongoDB C# Driver which is the officially supported C# driver for MongoDB. The version of the driver is 2.0 and the version of MongoDB is 3.0.

For connecting to a running mongod we used a *MongoClient*. After that, in code we added two usings for MongoDB.Bson and MongoDB.Driver and we created a connection string in the web.config file like this:

```
  <connectionStrings>
  <add name="MongoDB"
connectionString="mongodb://localhost:27
017/"/>
  </connectionStrings>
```

We called this connection string through this code:

```
public static string connString =
System.Configuration.ConfigurationManage
r.ConnectionStrings["MongoDB"].Connectio
nString;
```

And after this we made methods for calling the collections that we need like the one bellow:

```
public static
IMongoCollection<BsonDocument>
ConnectToServerPeopleDoc()
{
   var client = new
MongoClient(connString);
   var db =
client.GetDatabase("Dizertatie");

IMongoCollection<BsonDocument>
collection =
db.GetCollection<BsonDocument>("People")
;
    return collection;
}
```

In the method above, we call the database called “Dizertatie” and we get the collection called “People” and the results that will be returned will be a

IMongoCollection of BsonDocument type which is a specific format for MongoDB.

Next, we created asynchronously methods to work with the data from MongoDB. For example, for deleting all the registered people from the collection we created a method like this:

```
public static async Task<DeleteResult>
DeleteAllConsumers()
{
     var collection =
ConnectToServerPeopleDoc();
     var filter = new BsonDocument("_id", new
BsonDocument("$exists", true));

     var result = await
collection.DeleteManyAsync(filter);

     return result;
}
```

The MongoDB method that we call for deleting the registered people, *DeleteManyAsync*, will delete multiple documents inside the collection that we are connected to. The number of the deleted documents depends on the filter that we need to create and provide when we want to call the method. In our case, we will delete all registers from this collection.

In the next chapter we will mainly focus on the performance results for both databases, MongoDB and MSSSQL, that we obtained after we executed operations of SELECT, INSERT, UPDATE and DELETE.

III. MongoDB vs MSSQL: Comparative study

Generally, depending on the scope of the application that we want to develop, when the project is in the planning stage, each company will establish the resources and the limits for the project. We also need to choose the database for the application that will be developed. Here, we should consider the amount of data that will be manipulated, the rapidity that the project needs and the budget.

Considering these factors, for an application that will need to store a large amount of data. If the application lives and breathes through the stored data, then we should think how we could accomplish this in an efficient way or how to allocate a lot of resources for this scope, in generally it's about the financial part. [4]

For example, in the application that we created, we should consider the fact that we need a huge space for storing the data and at the queries that will be made every day like adding new data, deleting, updating and so on. All these queries are expensive and we should also think about the rapidity at which they are processed. Another important fact that we need to consider is the number of users that will access the application, like the employees from all the country, plus the usual users which will need different information.

Considering all these facts, now is a good time to think about the *performance* of the application. The performance of the database that we've chosen can be a very important fact because of the storage space, all the hardware and other components that we need.

In order to satisfy the needs of a big application we executed operations of SELECT, INSERT, UPDATE and DELETE on both databases. All these operations have been made on 1, 100, 500, 1.000, 5.000, 10.000, 25.000 and 50.000 of records. After we executed these operations we obtained the following results:

INSERT operation:

Insert	MongoDB - sec	SQL - sec
1 user	00:00:00:003	00:00:00:402
100 users	00:00:00:005	00:00:00:096
500 users	00:00:00:018	00:00:00:183
1.000 users	00:00:00:033	00:00:00:387
5.000 users	00:00:00:162	00:00:00:736
10.000 users	00:00:00:521	00:00:01:085
25.000 users	00:00:00:816	00:00:03:378
50.000 users	00:00:01:835	00:00:08:306

Table 3.1 – Insert operation results

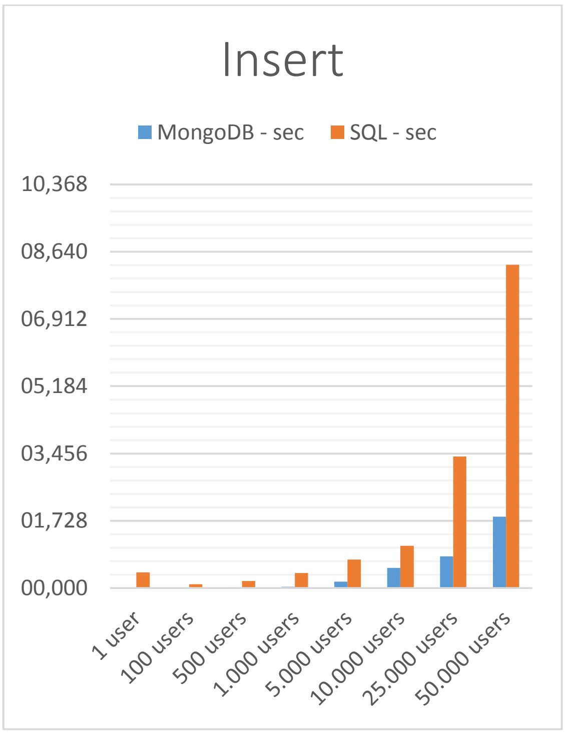

Fig. 3.1 – The graphic obtained based on Table 3.1

Until 1.000 records, we obtained a maximum difference of 200 milliseconds, but we can see that the difference is emphasized after 1.000 records and for 50.000 records we obtained a difference of 7 seconds and half.

SELECT operation:

Select	MongoDB - sec	SQL - sec
1 user	00:00:00:003	00:00:00:083
100 users	00:00:00:004	00:00:00:002
500 users	00:00:00:017	00:00:00:005
1.000 users	00:00:00:031	00:00:00:006
5.000 users	00:00:00:206	00:00:00:028
10.000 users	00:00:00:291	00:00:00:052
25.000 users	00:00:00:830	00:00:00:190
50.000 users	00:00:01:616	00:00:00:327

Table 3.2 – Select operation results

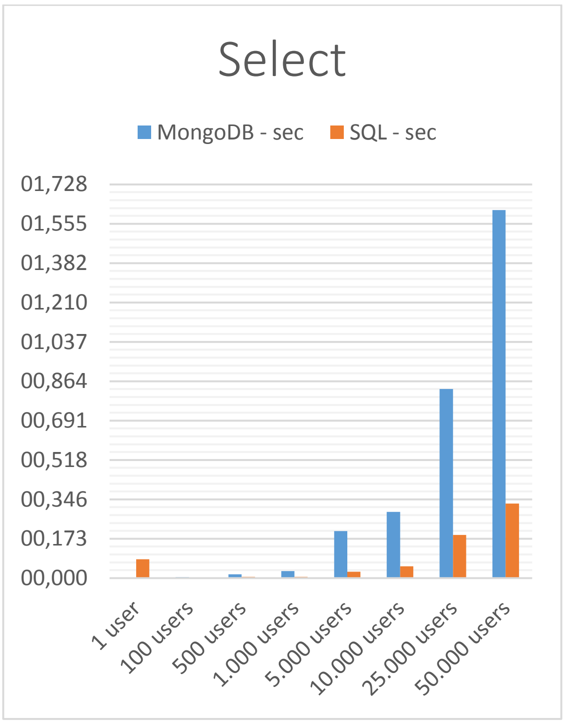

Fig. 3.2 – The graphic obtained based on Table 3.2

For the select operations we can observe that they are more fast and efficient for MSSQL than for MongoDB. We also see a difference from 1.000 records, which is 26 milliseconds. This difference grows up to 1:300 seconds when we select 50.000 records.

UPDATE operation:

Update	MongoDB - sec	SQL - sec
1 user	00:00:00:005	00:00:00:039
100 users	00:00:00:007	00:00:00:048
500 users	00:00:00:059	00:00:00:059
1.000 users	00:00:00:042	00:00:00:159
5.000 users	00:00:00:245	00:00:02:219
10.000 users	00:00:00:463	00:00:04:634
25.000 users	00:00:01:294	00:00:19:946
50.000 users	00:00:02:224	00:00:31:205

Table 3.3 – Update operation results

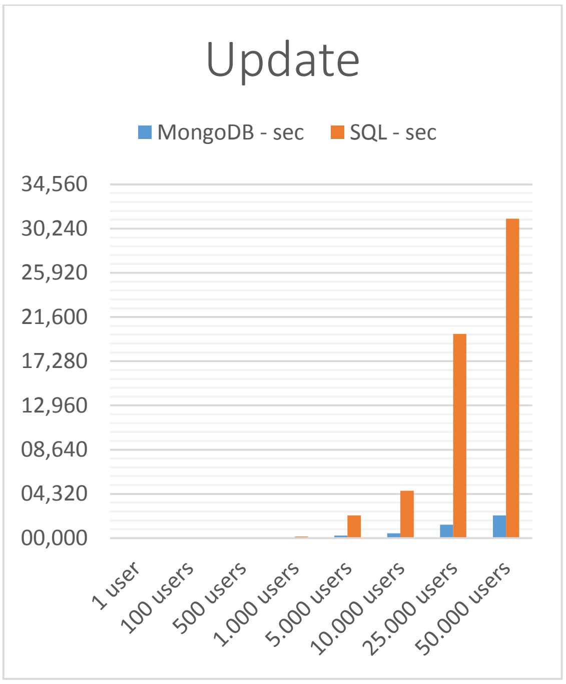

Fig. 3.3 – The graphic obtained based on Table 3.3

For the update operations we can see a bigger difference from 5.000 records, which is approximatively 2 seconds and until 50.000 records it grows up to 29 seconds and gives MongoDB an advantage.

DELETE operation:

Delete	MongoDB - sec	SQL - sec
1 user	00:00:00:004	00:00:00:081
100 users	00:00:00:003	00:00:00:019
500 users	00:00:00:007	00:00:00:063
1.000 users	00:00:00:017	00:00:00:082
5.000 users	00:00:00:053	00:00:00:143
10.000 users	00:00:00:106	00:00:00:200
25.000 users	00:00:00:317	00:00:00:350
50.000 users	00:00:01:508	00:00:01:787

Table 3.4 – Delete operation results

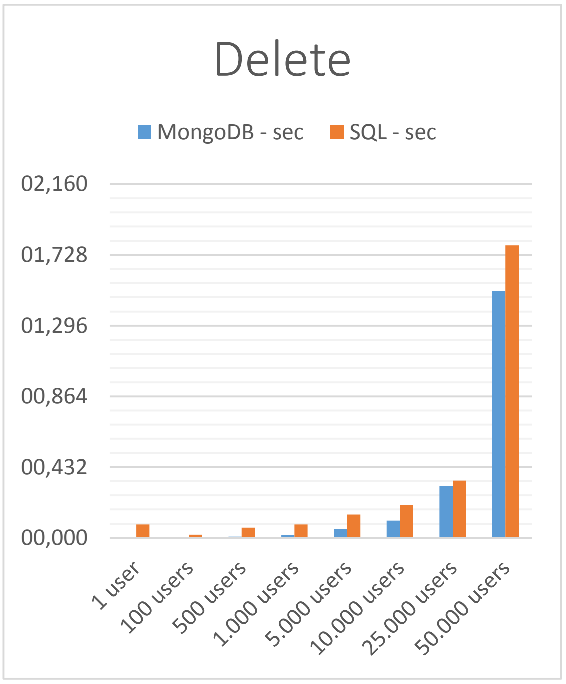

Fig. 3.4 – The graphic obtained based on Table 3.4

For the delete operations the performance difference is not that emphasized as it was for the rest of the operations. The maximum difference was obtained at 50.000 records and it was approximatively 200 milliseconds. Until 1.000 records the difference is only about few dozens of milliseconds.

IV. Conclusions

In this paper we showed the results of different operations that have been applied on MongoDB and MSSQL databases. The only time when the MSSQL obtained an advantage was with the SELECT operations, the other ones gave advantages to MongoDB.

We can also note that the difference between the results of each database was not noticeable until around 1.000 records. With this information in mind we can say that MSSQL is suitable for small and medium applications.

Depending on what each application needs, we can choose the most suitable database, a non-relational or a relational database. Considering that in our days the request for storing more and more data at a low price or inexistent is bigger every day we tend to choose a NoSQL database. Also, MSSQL being commercial at a pretty big price and MongoDB is an open source, it is a big disadvantage for MSSQL.

In the end, for choosing the correct database that can satisfy all the needs that an application demands in order to be developed, all the things discussed above should be taken in consideration before the start of developing the project.

REFERENCES

[1] Matt Asay, NoSQL databases eat into the relational database market, Available: http://www.techrepublic.com/article/nosql-databases-eat-into-the-relational-database-market/ , accessed july 2015.

[2] Pramod Sadalage, NoSQL Databases: An Overview, Available: http://www.thoughtworks.com/insights/blog/nosql-databases-overview , accessed july 2015.

[3] MongoDB, NoSQL Database Explained, Available: https://www.mongodb.com/nosql-explained , accessed july 2015.

[4] Michael Kennedy, MongoDB vs. SQL Server 2008 – Performance Shutdown, Available: http://blog.michaelckennedy.net/2010/04/29/mongodb-vs-sql-server-2008-performance-showdown/, accessed july 2015.