A Comparative Study: MongoDB vs. MySQL

Conference Paper · June 2015

DOI: 10.13140/RG.2.1.1226.7685

4 authors:

Cornelia Győrödi
University of Oradea
62 PUBLICATIONS **453** CITATIONS
SEE PROFILE

Robert Gyorodi
University of Oradea
60 PUBLICATIONS **509** CITATIONS
SEE PROFILE

George Pecherle
University of Oradea
24 PUBLICATIONS **266** CITATIONS
SEE PROFILE

Andrada Olah
University of Oradea
2 PUBLICATIONS **204** CITATIONS
SEE PROFILE

A Comparative Study: MongoDB vs. MySQL

Cornelia GYŐRÖDI, Robert GYŐRÖDI, George PECHERLE, Andrada OLAH

***Abstract*— In this paper we will try to present a comparative study of non-relational databases and relational databases. We mainly focus our presentation on one implementation of the NoSQL database technology, namely MongoDB, and make a comparison with another implementation of relational databases, namely MySQL, and thus justifying why MongoDB is more efficient than MySQL. We will also present the advantages of using a non-relational database compared to a relational database, integrated in a forum in the field of personal and professional development. The NoSQL database used to develop the forum is MongoDB [6, 7], and was chosen from a variety of non-relational databases, thanks to some aspects that we will highlight in this article. The database integration in the framework will also be presented.**

***Index Terms*— MySQL, MongoDB, NoSQL, RDBMS, non-relational databases**

I. Introduction

IF a few years ago an application normally only used to have thousands of users to tens of thousands of users in the most extreme case, currently there are applications that have millions of users and who are connected 24/7, 365 days per year. It is important to use an appropriate database, which supports simultaneous connection of hundreds of thousands users.

Relational databases are widely used in most of the applications and they have good performance when they handle a limited amount of data. To handle a huge volume of data like internet, multimedia and social media the use of traditional relational databases is inefficient. To overcome this problem the "NO SQL" term was introduced. The NoSQL term was coined by Carlo Strozzi in 1998 and refers to non-relational databases, term which was later reintroduced in 2009 by Eric Evans. More recently, the term has received another meaning, namely "Not Only SQL", which is a milder variant of defining the term, compared to its previous significance, the *anti-relational* [3].

NoSQL, is not a tool, but a methodology composed of several complementary and competing tools [1]. The primary advantage of a NoSQL database is that, unlike a relational database it can handle unstructured data such as documents, e-mail, multimedia and social media efficiently [1, 10]. Non-relational databases do not use the RDBMS principles (Relational Data Base Management System) and do not store data in tables, schema is not fixed and have very simple data model. Instead, they use identification keys and data can be found based on the keys assigned.

There are four strategies for storing data in a non-relational database, as shown in [2], and they are as follows:

1. Key-Value - Key-Value databases [1, 2], such as Riak, are conceptual distributed dictionaries and do not have a predefined schema; they are schema less. The key can be synthetic or self-generated, and the value can be anything: string, JSON, BLOB and others.

2. Document - Couchbase and MongoDB [2] are the most popular document based databases. They are flexible in the type of content because they do not have a predefined schema. Conceptually, they work with documents of different types: JSON, BSON, XML and BLOBs [2, 8]. Basically they represent only a specialization of key-value databases. A document is written/read using a key. Besides the functionality Key-Value, document based databases add different functionalities to find documents based on their content.

3. Column – Databases from BigTable category, such as HBase and Hypertable are columnar type and must have a predefined schema [2, 4]. Data is stored in cells grouped in columns, and the columns are logically grouped into families of columns. Theoretically, they can contain an unlimited number (limited depending on the implementation) of columns that can be created at runtime or at schema definition.

4. Graph-Oriented – This strategy can support complex data queries which are also performed in a relatively smaller period of time compared to other databases using the strategies mentioned above.

Also, non-relational databases provide high flexibility at addition or deletion of an attribute from the database because they do not have a fixed database schema. Depending on the requirement of the application, we can use different type of NoSQL database and each NoSQL database has its own features, data model and architecture choice of the database depends on the application.

In this paper we focus on one of the NoSQL technologies, namely MongoDB, and make a comparison with MySQL to highlight why MongoDB is more efficient than MySQL. In addition, we will present the advantages of using a non-

Cornelia GYŐRÖDI is with the Department of Computer Science and Information Technology at University of Oradea, str. Universitatii nr. 1, Romania (e-mail:cgyorodi@uoradea.ro).

Robert GYŐRÖDI is with the Department of Computer Science and Information Technology at University of Oradea, str. Universitatii nr. 1, Romania (e-mail:rgyorodi@uoradea.ro).

George PECHERLE is with the Department of Computer Science and Information Technology at University of Oradea, str. Universitatii nr. 1, Romania (e-mail: gpecherle@uoradea.ro).

Andrada OLAH is currently a computer science master student with an interest in database management systems, web technologies and data mining. She graduated Computer Science at University of Oradea (andra.olah@gmail.com).

relational database in a forum application, using MongoDB as the NoSQL database.

II. Application development using MongoDB vs. MySQL

We realized a comparative study between MongoDB and MySQL based on their concepts and commands used for different operations.

This comparative study is based on the development of a forum that has a dynamic structure, depending on the preferences of its users. Using a relational database such as MySQL [9], the structure would have been static and each user would have had to follow a structure set implicitly in the database (ex.: forum - subforum - discussion - comments). Thus, each user is forced to comply with the structure, exemplified in Fig. 2-1.

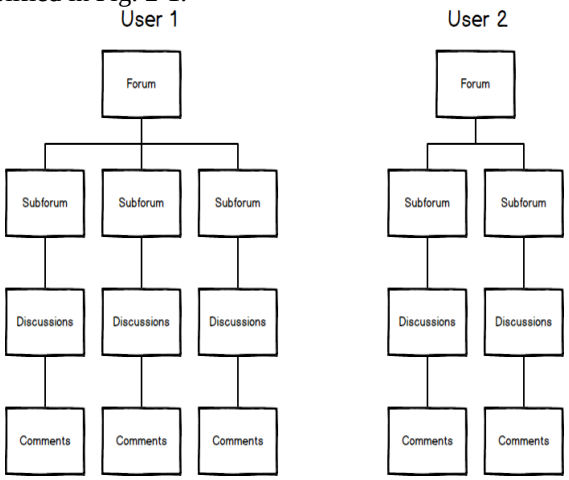

Fig. 2-1 Forum structure using MySQL

In the Fig. 2-1 each user has a different number of subforums in a forum, but the method of implementation that each user must meet is the same: first, they need to create a forum, then the subforums, followed by the creation of discussions and finally by the addition of comments in the discussion.

Within this application, a non-relational database, MongoDB, was chosen – a database based on JSON documents with dynamic schemas, written in C++. MongoDB originally launched in 2009, is currently in development, and is expanding [6, 7]. MongoDB is a database that can be used both in small projects that have several thousands of users, but primarily for products and applications that contain hundreds of thousands of users, including Craigslist, eBay, Foursquare or New York Times [3, 4].

MongoDB database holds a set of collections. A collection has no predefined schema like tables, and stores data as BSON documents (binary encoded JSON like objects) [1, 10].

A document is a set of fields and can be thought of as a row in a collection. It can contain complex structures such as lists, or even an entire document. Each document has an ID field, which is used as a primary key and each collection can contain any kind of document, but queries and indexes can only be applied on collections [1].

Using a non-relational database, such as MongoDB, for development of a forum, allows the structure to be specific to each user, thus each user has the possibility to organize its forum in a unique way, as shown in Fig. 2-2.

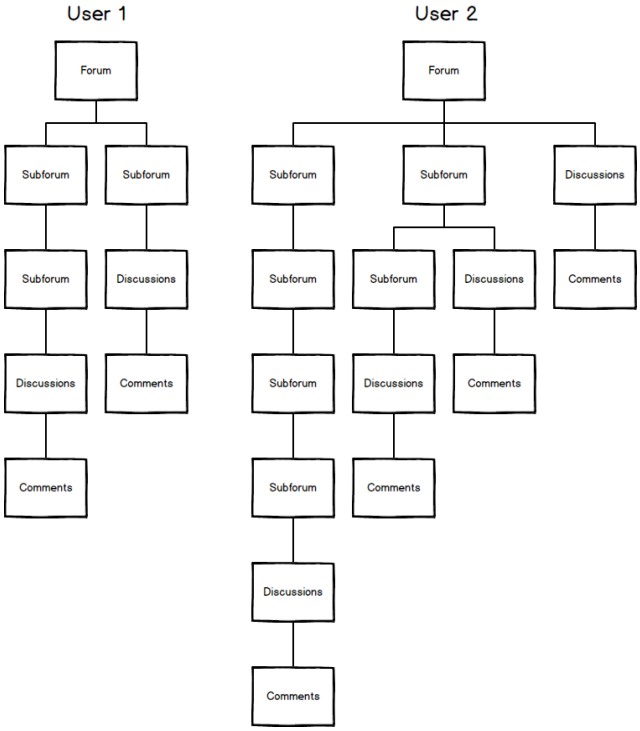

Fig. 2-2 Forum structure using MongoDB

Figure 2-2 shows that using a non-relational database, the static structure in which a forum contains subforums, the subforums contain discussions and discussions contain comments, is not mandatory to meet. Thus, a subforum may contain another subforum, which may contain another subforum, which contains another subforum and so on. There is also the possibility of attaching discussions directly to a forum, without the need to create a subforum for opening a new discussion.

The application was developed using Symfomy2 [5], an open source PHP framework that allows easy integration of MongoDB database. The integration of MongoDB database within the framework was made via a bundle, an auxiliary package that has to be registered in the list of packages used within the application.

After the bundle has been integrated into Symfony2 [5], the database configuration was made (server choice - which in this case is a local server, and database name choice - choosing a non-existent database does not lead to error, but the creation of the database), as follows:

```
# app/config/config.yml
doctrine_mongodb:
    connections:
     default:
         server: mongodb://localhost:27017
         options: {}
    default_database: my_database
    document_managers:
        default:
```

```
auto_mapping: true
```

III. Comparative Study

To further understand how to use MongoDB, Table 3.1 shows a comparison between the terms used in MySQL and MongoDB respectively.

As shown in Table 3.1, in MongoDB, some MySQL terms, such as table or row, get another name, namely collection, respectively BSON document. In other words, we can say that MongoDB contains collections, collections contain documents and a document contains multiple fields.

MySQL	MongoDB
Database	Database
Table	Collection
Index	Index
Row	BSON Document
Column	BSON Field
Join	Embedded documents and linking
Primary key	Primary key
Group by	Aggregation

Table 3.1 MySQL vs. MongoDB terms

Unlike MySQL, where the database is presented graphically in the form of a table, in MongoDB, a database has the following graphic structure:

```
{
    "_id": "d4acaf3a76e4378b853eb15fde216722",
    "username": "andra",
    "email": "andra@gmail.com",
}

{
    "_id": "d4rvgf3a76e4378b853tr15fde216722",
    "username": "ioana",
    "email": "ioana@gmail.com",
}
```

The example above shows a database for users, each user having an id that is unique and automatically generated, a username and an email address.

The application will have 3 classes of users, namely the administrators, the moderators and the regular users. Each user has the right to create a private forum/subforum and has full administrator rights on that particular forum/subforum. Within a subforum, the moderators have the right to edit/delete the subforum and they can also moderate other users' discussions, while regular users are only allowed to post discussions and leave comments. If a relational database has been used, the columns for forums and subforums should have appeared at all forum users, although normal users will never have the right to create, modify or delete them, unless of course, they are the administrators of that particular forum. Using MongoDB, these fields regarding the forum and the subforum will appear only to users who have that right (moderators and administrators), thus significantly reducing storage space, which is much higher using MySQL.

In fact, these roles are not assigned to the users, but to the forums themselves. This is because a user might never be an administrator on a forum, but only a regular user, and there is no point to assign the administrator field to that user (which will always be null), whereas a forum will always have this field, because its administrator is practically its author.

As in relational databases, MongoDB also has *one-to-many* relationships, but in this case the concept of foreign key is not used; instead, the concept of *annotations* is used. Thus, in this case, regarding a forum, the connection between the forum and its subforums is as follows: in the *forum* document, the subforums are referenced using the annotation *@MongoDB\EmbedMany (targetDocument = "Subforum")*, and in the *subforum* document, the forum is referenced by the following annotation that connects the two documents: *@MongoDB\ReferenceOne (targetDocument = "Forum", inversedBy="Subforums").*

```
class Forum
{
/**
 * @MongoDB\EmbedMany(targetDocument="Subforum")
 */
 public $subforums = array();
}

Class Subforum
{
/**
 * @MongoDB\ReferenceOne(targetDocument="Forum",
inversedBy="subforums")
 */
  protected $forum;
}
```

In order to create a flexible structure for the forum, the actual connection between the forum and its subforums, and also between forum – discussions, subforum – subforum, subforum – discussions is made using their IDs that are unique and which allow a dynamic link between the elements of the forum. So, a discussion can easily be attached both to a forum and to a subforum, thanks to the dynamic connection between the collections.

Also, if in the future it is desired to insert a new field in the database, this can be easily achieved by inserting the name and the type in the corresponding document without affecting the actual data stored in the database and without the need of redefining the entire database structure.

In following chapter we will present the differences between the two databases, MongoDB and MySQL, in terms of performance, after executing different operations on both databases.

IV. Performance tests

To highlight the advantages of using the non-relational database MongoDB compared to the relational database MySQL, various operations were performed on the two databases. These operations are the four basic operations that can be performed on any database, namely:

1. Insert
2. Select (query)
3. Update
4. Delete.

Because the test results depend on the computer on which these tests are carried out, it is important to note that all the results presented below were obtained from studies conducted on a computer with the following characteristics: Windows 7 Ultimate 64-bit, processor Intel Core i3 (2.4 GHz), 4 GB RAM memory.

A. *Insertion operation*

We began testing with the creation of databases without any content, both in MongoDB and MySQL. The structure of the two databases is similar and they have about the same number of columns/fields and tables/documents. The common elements of the two databases are:

- Table/document *User* with the columns: id, username, password, email.
- Table/document *Forum* with the columns: id, title, author, info (short description).
- Table/document *Subforum* with the columns: id, title, author, info, created, updated.
- Table/document *Discussion* with the columns: id, title, author, created, updated, content.
- Table/document *Comments* with the columns: id, author, created, content, approved.

In addition to these common fields, there also exist other fields linking the tables in MySQL and the documents in MongoDB, and they generally are foreign-keys that identify e.g. a subforum that belongs to its parent forum or the author of a certain comment.

Data insertion began in both databases with users' insertion. For each database, 10,000 users were inserted. Their ID was generated automatically by both databases and for the username, password and email address we have used PHP functions such as *md5*, *rand*, *substr* and *str_shuffle*.

For recording the time required to insert the elements in the database, we used the PHP function *microtime*, which recorded the time from the beginning of the script runtime and until its completion.

Thus, after running the two scripts, one for MySQL and one for Mongo, Figure 4.1 shows that 10,000 users were inserted into MySQL in 440 seconds, while in MongoDB, time was just 0.29 seconds.

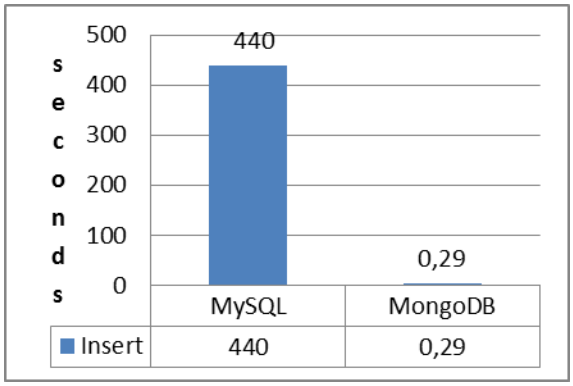

Fig. 4-1 MySQL vs. MongoDB insert

Once users have been inserted into the database, inserting data was passed to the forums, subforums, discussions and comments. All IDs of all the tables were generated automatically in both MySQL and in MongoDB, and to generate the other fields we used the functions described above. As a subforum cannot exist if a forum or another subforum is not created first, and as no comment can exist on its own as it must be owned by a discussion, we started by inserting the forums, then the subforums, followed by the insertion of the discussions, and finally the comments.

For testing, it was decided to insert 5000 rows for each table in part, as follows: 5000 forums, 5000 subforums, respectively 5000 discussions and 5000 comments.

Thus, after inserting data in the two databases, the following insertion times resulted:

MySQL: 1010 seconds, MongoDB 3.3331 seconds.

We notice that MongoDB spends less time than MySQL, for insertion of large amount of information in the database.

B. *Query operations*

To test the performance of query operations, we made two *select queries*, namely:

- Selection of all discussions a user attended and with a date different than a certain one;
- Selection of all users from the database and the number of discussions started by each user.

The first select has the following MySQL syntax (Select 1-MySQL):

```
mysql_connect('localhost','root','');
mysql_select_db('routinie');
$query = "SELECT d.username, d.dtitle, d.dcontent,
d.created, s.id, f.id
FROM discussion d
inner join subforum s on s.id = d.subforumid
inner join forum f on f.id = s.forumid
where d.username = `andra`
          and d.created <> '2014-11-27'";
  $query_run = mysql_query($query);
```

and the following MongoDB syntax (Select 1- MongoDB):

```
$m = new MongoClient();
$db = $m->selectDB('routinie');
```

```
$collection = new MongoCollection($db,
'categories');
$date = '1417091683';
$q = array(
'subcategories.topics.discussions.dauthor' =>
'andra',
'subcategories.topics.discussions.created' =>
array( '$ne' => new MongoInt32($date))
);
$cursor = $collection->find($q);
```

The second select has the following MySQL syntax (Select 2- MySQL):

```
mysql_connect('localhost','root','');
mysql_select_db('routinie');
$query = "select u.id, u.username, count(d.dtitle)
         from users u
         left outer join discussion d on d.userid
= u.id
         group by u.id, u.username";
$query_run = mysql_query($query);
```

and the following MongoDB syntax (Select 2- MongoDB):

```
$m = new MongoClient();
$db = $m->selectDB('routinie');
$collection = new MongoCollection($db,
'categories');
$user =
'subcategories.topics.discussions.dauthor';
$find_users = $collection->distinct($user);
foreach($find_users as $find_user) {
$discussions = $collection-
>count(array('subcategories.topics.discussions.dau
thor' => $find_user ));
```

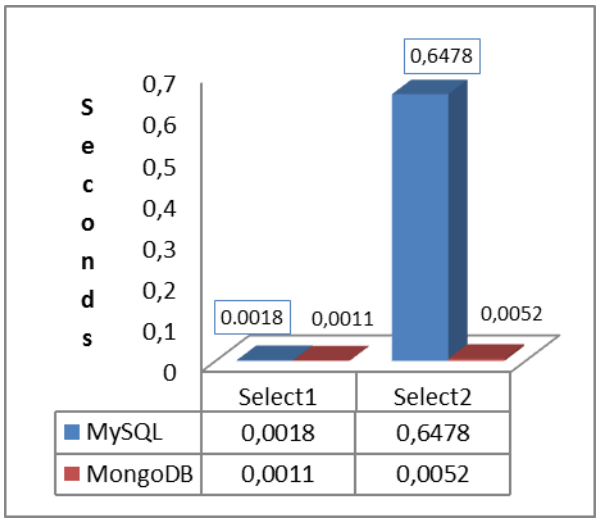

Fig. 4-2 MySQL vs. MongoDB *select*

The first select was executed in MySQL in 0.0018 seconds, while in MongoDB it was executed in 0.0011 seconds and the second select was executed in 0.6478 seconds in MySQL and 0.0052 seconds in MongoDB. From the graph, we notice that MongoDB spends less time than MySQL, for performing query operations as shown in Fig. 4-2.

C. Update operation

In order to test the update operation in the databases, the following two queries were executed:

- Updating a comment written by a specific user (Update 1).
- Updating the email address of a user, (Update 2).

The first update has the following MySQL syntax (Update 1- MySQL):

```
mysql_connect('localhost','root','');
mysql_select_db('routinie');
$query = "update `comments`
          set content = 'This is the new content.'
          WHERE id = 10594
          and username = `andra`";
$query_run = mysql_query($query);
```

and the following MongoDB syntax (Update 1- MongoDB):

```
$m = new MongoClient();
$db = $m->selectDB('routinie');
$collection = new MongoCollection($db,
'categories');
$criteria = array(

'subcategories.topics.discussions.comments.cauthor
' => 'andra',
   'subcategories.topics.discussions.comments.$id'
=> new
MongoId('5c936263f3428a40227908d5a3847c0b');
);
$collection->update(
    $criteria,
array('subcategories.topics.discussions.comments.c
content' => "new content")
);
```

The second update has the following MySQL syntax (Update 2- MySQL):

```
mysql_connect('localhost','root','');
mysql_select_db('routinie');
$query = "update `users`
set email = 'andra.olah@gmail.com'
         WHERE id = 1012";
$query_run = mysql_query($query);
```

and the following MongoDB syntax (Update 2- MongoDB):

```
$m = new MongoClient();
$db = $m->selectDB('routinie');
$collection = new MongoCollection($db,
'categories');
$criteria = array(
'users.$id' => new
MongoId('92cc227532d17e56e07902b254dfad10');
);
$collection->update(
```

```
        $criteria, array('users.email' =>
"andra.olah@gmail.com")
 );
```

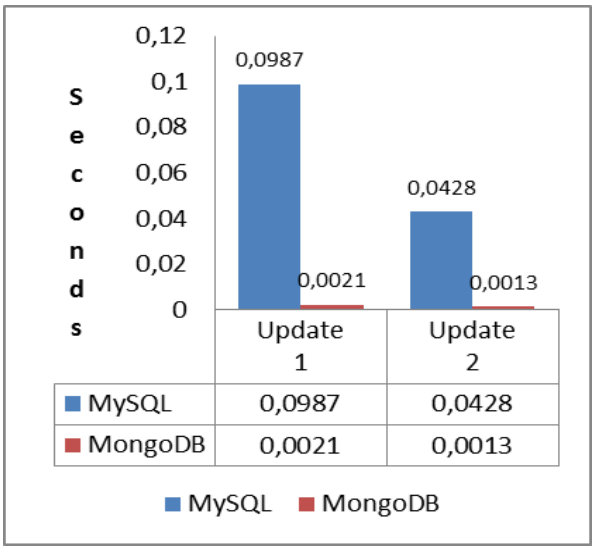

Fig. 4-3 MySQL vs. MongoDB *update*

The first update was executed in MySQL in 0.0987 seconds, while in MongoDB it was executed in 0.0021 seconds and the second update was executed in 0.0428 seconds in MySQL and 0.0013 seconds in MongoDB. From the Figure 4-3, we notice that MongoDB spends less time than MySQL, for performing update operation as shown in figure Fig. 4-3.

D. Delete operation

As the other operations described above, we have executed two delete queries for each database.

The first query deleted all the comments posted by a user (Delete 1). The second query deleted all the forums created by a specific user (Delete 2). It is important to mention that with the deletion of a forum, all the subforums, comments and discussions contained in the specified forum were also automatically deleted, in both MySQL and MongoDB.

The first delete has the following MySQL syntax (Delete 1- MySQL):

```
mysql_connect('localhost','root','');
mysql_select_db('routinie');
$query = "delete from comments where username =
`andra`";
$query_run = mysql_query($query);
```

and the following MongoDB syntax (Delete 1- MongoDB):

```
$m = new MongoClient();
$db = $m->selectDB('routinie');
$collection = new MongoCollection($db,
'categories');
$criteria =
array('subcategories.topics.discussions.comments.c
author' => 'andra');
$collection->update(
    $criteria,
    array('$unset' =>
array('subcategories.topics.discussions.comments'
=> true)),
    array('multiple' => true)
);
```

The second delete has the following MySQL syntax (Delete 2- MySQL):

```
mysql_connect('localhost','root','');
mysql_select_db('routinie');
$query = "delete FROM `forum` WHERE username
=`andra`";
$query_run = mysql_query($query);
```

and the following MongoDB syntax (Delete 2- MongoDB):

```
$m = new MongoClient();
$db = $m->selectDB('routinie');
$collection = new MongoCollection($db,
'categories');
$q = array('admins.username' => 'andra');
$cursor = $collection->remove($q);
```

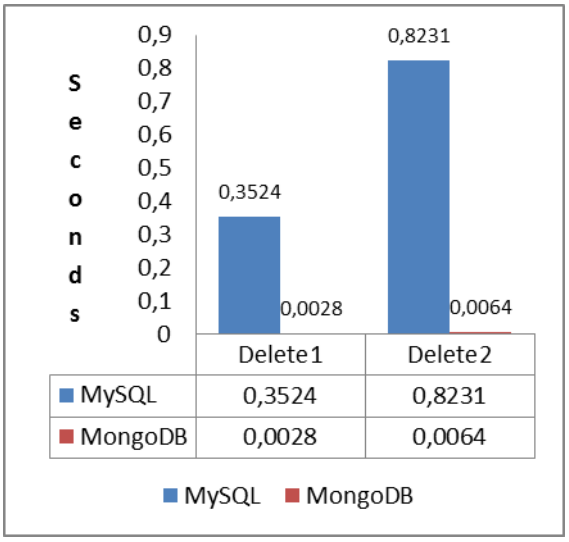

Fig. 4-4 MySQL vs. MongoDB *delete*

From the figure Fig. 4-4, we notice that the first delete was executed in MySQL in 0.3524 seconds, while in MongoDB it was executed in 0.0028 seconds and the second delete was executed in 0.8231 seconds in MySQL and 0.0064 seconds in MongoDB.

MongoDB provided lower execution times than MySQL in all four basic operations, which is essential when an application should provide support to thousands of users simultaneously. Thus, the above comparison, proves that for large amounts of data MongoDB has a good performance and it is preferred over MySQL.

V. Conclusions

Switching from a relational database to a non-relational database can be a challenge in many ways, among which the need to study carefully all possible types of non-relational databases and finding the optimal solution for the specific application, and the creation of a non-relational database which can provide exactly the same features and querying operations as the database that will replace. But the use of relational database will never come to an end, because it provides us with an unparalleled feature set, by maintaining data integrity and scalability. Mainly, it is the developers' job to decide which database should be used to meet the application' requirements.

For this application, i.e. the forum, the most suitable non-relational database was MongoDB, because the forum will have thousands of users, or even tens of thousands and MongoDB has enabled its customization to suit each user by creating private forums, each with its own flexible and dynamic structure.

Although it is known that MongoDB is a database that does not allow join operations, there are alternatives to this problem, the most common being the use of references, which is a convention for creating documents. For example, when inserting a comment in the database, the comment will necessarily contain a reference to the document that it is linked to, namely the discussion, and it will also contain the id of the discussion it belongs to. Thus, the connection between the two documents is made through references.

The advantage of using MongoDB was further highlighted by conducting tests and interpreting their results, which were presented in the previous chapter. MongoDB provided lower execution times than MySQL in all four basic operations, which is essential when an application should provide support to thousands of users simultaneously.

We can choose MongoDB instead of MySQL if the application is data intensive and stores many data and queries lots of data.

It is also important to note that within the same application, sometimes each user is likely to need its own custom settings and relational databases do not allow total customization that is based solely on the needs of users. Thus, more and more applications are beginning to use a non-relational database because they provide a more flexible structure that can shape after each user' needs; they are designed to store large amounts of data and they are denormalized databases, which increases performance.

References

[1] K. Sanobar, M. Vanita, "SQL Support over MongoDB using Metadata", *International Journal of Scientific and Research Publications*, Volume 3, Issue 10, October 2013

[2] T. Frătean, Bazele de date NoSQL – o analiză comparativă, *To Day Software Magazine*, number 10 [Online]. Available: http://www.todaysoftmag.ro/article/304/bazele-de-date-nosql-o-analiza-comparativa, acceseed oct. 2014.

[3] S. Hoberman, "Data Modeling for MongoDB**"**, Publisher by Technics Publications, LLC 2 Lindsley Road Basking Ridge, NJ 07920, USA, ISBN 978-1-935504-70-2, 2014.

[4] S. Hall (2014, Jul) MySQL vs MongoDB, jul, 2014, Available: http://www.scriptrock.com/articles/mysql-vs-mongodb, acceseed nov. 2014

[5] F. Potencier, "The symfony 1.3 & 1.4 Reference Guide", Publisher: Sensio SA (November 4, 2009), ISBN: 978-2918390145, pp 278.

[6] MongoDB, Inc. (2015, Aprilie), *MongoDB Ops Manager Manual Release 1.6*, [Online]. Available: https://docs.opsmanager.mongodb.com/current/opsmanager-manual.pdf

[7] R. P Padhy, M. R. Patra, S. C. Satapathy, "RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's", *International Journal of Advance Engineering Sciences and Technologies*, Vol. 11, Issue No. 1, 015-030, 2011.

[8] J. Clarence, M. Tauro, S. Aravindh, A. B. Shreeharsha, "Comparative Study of the New Generation, Agile, Scalable, High Performance NOSQL Database", *International Journal of Computer Applications*, ISSN 0975 – 888, Volume 48– No.20, June 2012.

[9] M. Irfan **(**March 14, 2014**)** *Tools and tips for analysis of MySQL's Slow Query Log Oracle Corporation***.** [Online]. Available: http://www.percona.com/blog/2014/03/14/tools-and-tips-for-analysis-of-mysqls-slow-query-log/

[10] Z. Wei-Ping, LI Ming-Xin, H. Chen, "Using MongoDB to Implement Textbook Management System instead of MySQL", IEEE 3rd International Conference on Communication Software and Networks (ICCSN), ISSN 978-1-61284-486, 2011.