A Comparative Study Between the Capabilities of MySQl Vs. MongoDB as a Back-End for an Online Platform

Cornelia Győrödi
Department of Computer Science and Information Technology, University of Oradea
Oradea, Romania

Ioana Andrada Olah
Department of Computer Science and Information Technology, University of Oradea
Oradea, Romania

Robert Győrödi
Department of Computer Science and Information Technology, University of Oradea
Oradea, Romania

Livia Bandici
Faculty of Electrical Engineering and Information Technology, University of Oradea
Oradea, Romania

***Abstract*—In this article we present a comparative study between the usage capabilities of MongoDB, a non-relational database, and MySQL's usage capabilities, a relational database, as a back-end for an online platform. We will also present the advantages of using a non-relational database, namely MongoDB, compared to a relational database, namely MySQL, integrated in an online platform, which allows users to publish different articles, books, magazines and so on, and also gives them the possibility to share online their items with other people. Nowadays, most applications have thousands of users that perform operations simultaneously thus, it takes more than one operation to be executed at a time, to really see the differences between the two databases. This paper aims to highlight the differences between MySQL and MongoDB, integrated in an online platform, when various operations were executed in parallel by many users.**

***Keywords*—MySQL; relational database; MongoDB; non-relational database; comparative study**

I. Introduction

Nowadays, an application must be accessible to its users 24 hours a day, 7 days a week, so it is important to implement an appropriate database, which supports simultaneous connection of hundreds of thousands of users [6]. Also, more and more complex requirements from users appeared, and companies were forced to find different solutions to meet the needs of their customers. Thus, the applications must support millions of users simultaneously and handle a huge volume of data and a relational database model has serious limitations when it has to handle that huge volume of data. Because each customer has his own needs and requirements, it might be possible that within the same application, there is a need for a different customization for each user. Relational databases do not allow a complete configuration that can shape after their needs. These limitations have led to the development of non-relational databases, also commonly known as NoSQL (Not Only SQL) [11]. 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 [2, 6].

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 [6]. Their main advantage is represented by their flexible structure, but also because they are designed in a way that can store large amount of data. In addition, they are denormalized databases, which leads to increased performance [1].

In this paper, we will try to present the advantages of using MongoDB compared to MySQL, integrated in an online platform, which allows users to publish different articles, books, magazines and so on, and gives them the possibility to share online their items with other people. At the same time, we will show a comparison between the two databases, MongoDB and MySQL, in terms of execution times when many users executed various operations in parallel.

II. Presentation of the Application Database

Databases to be presented are part of an online application that allows its users to create interactive digital flipbooks. Basically, the application could be called an online library, where users can create digital books that can be accessed by users worldwide.

Such an application can be used by many types of users, such as writers (who want to submit previews of their books), teachers (who can publish educational tutorials for their students), companies or supermarkets (that wish to advertise online or to submit weekly brochures) and so on.

NoSQL databases provide you with ways of storing and retrieving the data that is not modelled as the relational databases are modelled. Mainly, NoSQL databases are designed to allow us insertion of data for which we do not have a predefined schema, as the structure of our data is not set [7].

This work was performed through the Partnerships Program in priority areas, PN-II-PT-PCCA-2013-4-2225 - No. 170/2014 developed with the support of MEN - UEFISCDI, "Electromagnetic methods to improve processes wine".

A database like MongoDB does not a have the concept of a "row"; instead, we have a more flexible model [4]. There are four strategies for storing data in a non-relational database, as shown in [5]. They are not based on a single model (e.g. relational model of RDBMSs) and each database, depending on their target-functionality, adopt a different one [9]. The design of NoSQL databases depends on the type of database, called store. Document Stores pair each key identifier with a document which can be a document, key-value pairs, or key-value arrays. Graph Stores are designed to hold data best represented by graphs, interconnected data with an unknown number of relations between the data [10].

As a MongoDB database does not actually have a graphical representation, the tabular representation for the MySQL database will be presented. The two databases have the same number of columns and the same data type for each field. At a structural level, the only difference between the two databases is the way their data is represented.

The database can be divided into 3 parts, namely:

- users' area;
- items' area;
- orders' area.

A. *Users' area* – lists tables containing information about all the users, such as address, country, city or phone number as shown in Fig. 1.

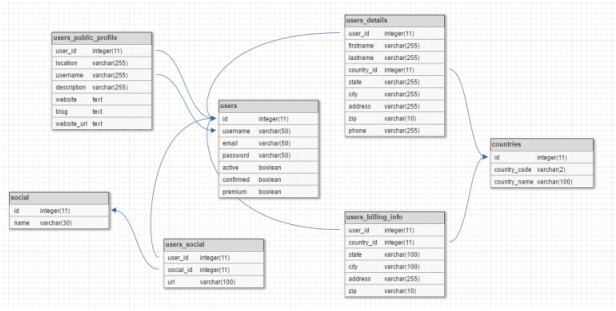


Fig. 1. Users' tables

As reflected in the figure Fig. 1, the users area contains 7 tables, 2 of which are static: *countries* (which contains a list of all existing countries) and *social* (which contains a list of names of social networks, such as Facebook and LinkedIn, which users can add to their public profile, to promote themselves).

The other tables, listed below, contain the following fields:

- *users* table, with the fields *id*, *username*, *email*, *password*, *active*, *confirmed* and *premium* (this table contains the user's data required for registration);
- *users_billing_info* table, with the columns *user_id*, *country_id*, *state*, *city*, *address* and *zip*, which stores the user's billing info data. When a user registers on the platform, it has by default the Free Package. The user may subsequently change the package, buying a new one;
- *users_details* table, with the optional fields *firstname*, *lastname*, *country_id*, *state*, *city*, *address*, *zip*, *phone*, which the user can set or not;
- *users_public_profile*, which contains information about the user's public profile, such as personal website or blog page;
- *users_social*, which stores data about the user's social networks (Facebook, Google+, LinkedIn etc.).

Also, the users' area is closely linked to 2 static tables, which are the *countries* and the *social* tables. The link between the tables is made by foreign keys, which are either the country id, or the social network id.

B. *Items' area* – contains information about the items ploaded by a user (books, magazines, catalogs), as well the SEO categories to which they belong as shown in Fig. 2.

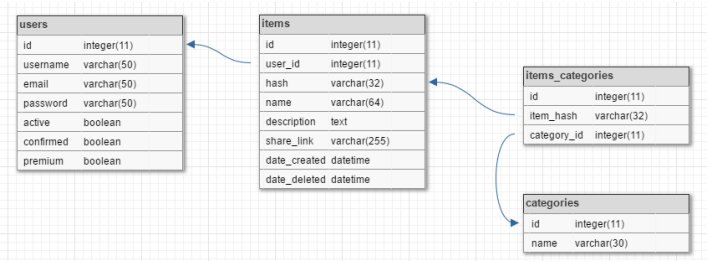


Fig. 2. Items' tables

In the figure Fig. 2 are presented the tables containing data about the items a user that can be upload on the platform.

The *items* table contain explicit data about each item, such as the name and description set by the user, the date on which it was created, respectively the date on which it was deleted (which is *null* by default), a hash (a string representing the surest way of identifying an item from the platform, because it is unique) and a *share link* that is automatically formed from the name of the platform and the item hash.

Also, as the database described in this application can be integrated into a promotional platform, each user has the possibility to add different categories for each item, in order to be found faster by search engines. The categories are static and are found in the *categories* table.

C. *Orders' area* – contains information about the users' rders and transactions (the premium package bought by a user, the amount of the package, the date the order was made and so on) is shown in Fig. 3.

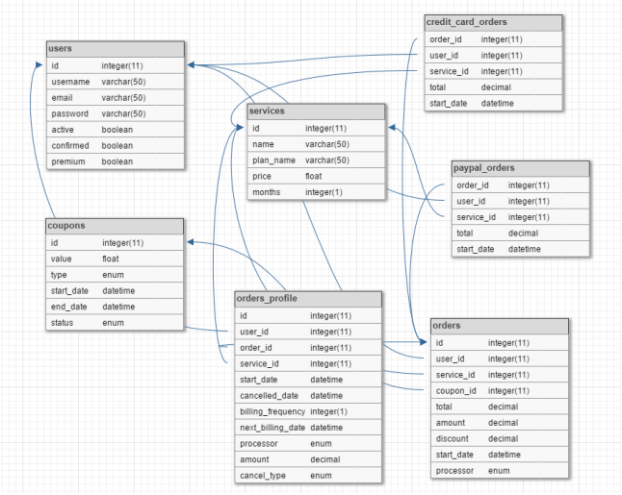


Fig. 3. Orders' tables

The orders area is a rather complex part in any application, because it must communicate with all other parties. In this application, an order refers to a user switching from one package to another (packages will be listed later). Thus, the *orders* table, that effectively represents all the transactions made by users, contains the following fields:

- *id* – a unique field, automatically generated, that identifies the order;
- *user_id* – id of the user who made the order (the user id is also unique);
- *service_id* – the type of the package that the user has bought (the types of packages that exist in an application are varied and they depend greatly on the type of application concerned). For this case study, we used the following packages: User Pro, User Business and User VIP (each package can be purchased either for a period of one month, or for a period of 12 months);
- *coupon_id* – most applications (websites, online platforms, shops and many others) offer various discounts to its users. These discounts are usually provided with discount coupons, which can be applied in most cases only once, and can be of two main categories: numerical coupons and percentage coupons;
- *amount* – the amount of the package chosen by the user, and which is calculated by multiplying the number of months selected with the package price (usually, the annual packages are cheaper than the monthly ones, and offer a discount by default);
- *discount* – the amount of discount that is given (or not) to a user, when they conduct a transaction (when they purchase a package);
- *total* – the total amount that the user has to pay, and which represents the difference between the package and the discount offered;
- *start_date* – the date on which the package was bought;
- *processor* – the payment method chosen by the user (the most commonly used payment methods are PayPal and Credit card).

Also, depending on the payment method chosen by the user, all the data related to the transaction will be inserted either in the *credit_card_orders* table, if the payment was made by credit card, or in the *paypal_orders* table, if payment is made through PayPal.

Another important table is the table *orders_details.* This table contains information about the next date when user will be billed, i.e. the date when the current subscription will expire and when the current package will have to be automatically renewed. In most applications, once a user has purchased a package, it will be renewed automatically, once a month or every 12 months, depending on the type of package that the user owns, but only if the user does not manually cancel the subscription.

III. PERFOMANCE TESTS

To highlight the advantages of using a non-relational database, MongoDB, compared a relational database, MySQL, various operations were performed on the two databases in parallel by many users. These operations represent the four elementary operations that can be performed on any database, namely: insert, select, update and delete [3].

All the tests to be presented were conducted on a computer with the following configuration: Windows 7 Pro 64-bit, processor Intel Core i3 (2.4 GHz), 4 GB RAM memory.

To have data on which to carry out operations, some data had to be inserted. Because in any application, the users are the most important part and without them, the application would not exist, the test started with the insertion of users in both databases (MySQL and Mongo, respectively).

To generate user's data such as username, email address, and password, various PHP functions [8], such as *md5*, *rand*, *substr* and *str_shuffle* were used. In fact, the functions listed above were used to generate all data for the databases, such as city, address, telephone number, personal website, item name or description. In order to record the time required to insert the elements in the database, it was used the PHP function *microtime*, which recorded the time from the beginning of the script runtime and until its completion.

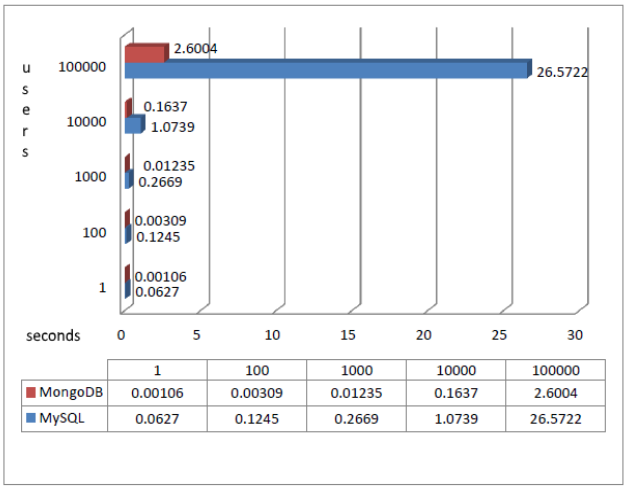
Most applications and websites give users the possibility of creating a public profile, after they successfully register on the site. Since not all users who make an account in an application complete all the necessary data for a public profile, and they only create a simple account, we conducted two types of tests for inserting users.

The first test refers only to a simple user registration on the website, which is the creation of an account only by setting a

username, *email address*, and *password*. These data are listed in the table of *users*.

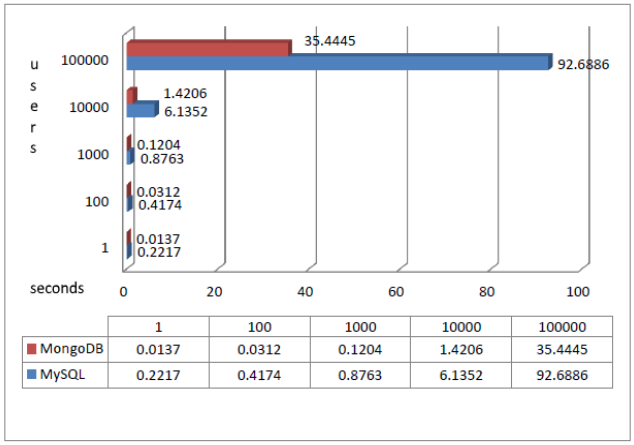
The second test includes the creation of a public profile and the insertion of some additional data, such as country, city, phone number, billing details or links to various social networks. These data are entered, as appropriate and as described in the previous chapter, in the tables *users_billing_info*, *users_details*, *users_public_profile* and *users_social*.

To test the performance in terms of speed differences between the two databases, 5 tests were performed for each insert case (5 tests for users who only register on the platform, and 5 tests for users who also completed some other details). The number of users has varied, starting from 1 user and continuing with 100, 1.000, 10.000 and 100.000 users respectively.



	1	100	1000	10000	100000
MongoDB	0.00106	0.00309	0.01235	0.1637	2.6004
MySQL	0.0627	0.1245	0.2669	1.0739	26.5722

Fig. 4. Insert users without details



	1	100	1000	10000	100000
MongoDB	0.0137	0.0312	0.1204	1.4206	35.4445
MySQL	0.2217	0.4174	0.8763	6.1352	92.6886

Fig. 5. Insert users with details

As is showed in the graphs from Fig. 4 and Fig.5, we can see that MongoDB has a good performance; it was faster than MySQL in all insert cases. However, this is best proven by the insertion of 100.000 users, where MongoDB was 2 times faster when the users only registered on the platform, and 10 times faster, when users also had completed some other personal details.

Following the results from Fig. 4 and Fig. 5, we can be easily seen that MongoDB has a much higher 'working speed' compared to the speed of MySQL. However, although the data obtained above is true and accurate, those tests are not applicable in everyday life. Tests carried out above represent one operation at a time. Although there are small applications and websites that have a relatively small number of users, nowadays, most applications have thousands of users that perform operations simultaneously. In other words, it takes more than one operation at a time, to really see the differences between the two databases.

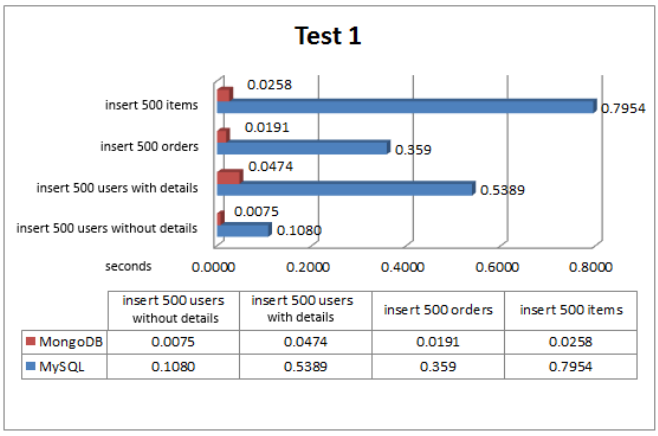
This paper aims to highlight the differences between MySQL and MongoDB, using the databases described above, in a real application, where various operations (such as: data is inserted, data is deleted or data is modified) were executed in parallel by many users.

Thus, we will describe further three tests, each with different difficulty level in order to accurately calculate the differences between the two databases.

A. *Test 1*

For the first test, 1.000 users were previously inserted in both databases. Then, the test actually begins with the insertion of 1.000 users in the databases, of which 500 users only register on the platform, and the remaining 500, also complete other details (described in a previous test). At the same time, there are 500 orders (half of the existing users buy a premium package) and other 500 users create (upload) different items. All tests in this paper were achieved using the PHP programming language, and their execution was carried out in parallel using the PHP *exec* function [8], that executes these operations in parallel.

The results obtained after carrying out the first test described above are illustrated in the figure Fig. 6.



	insert 500 users without details	insert 500 users with details	insert 500 orders	insert 500 items
MongoDB	0.0075	0.0474	0.0191	0.0258
MySQL	0.1080	0.5389	0.359	0.7954

Fig. 6. The results of Test 1

As shown in the figure Fig.6, MongoDB was faster than MySQL in this case, when various operations were executed in parallel. However, the differences between the two databases

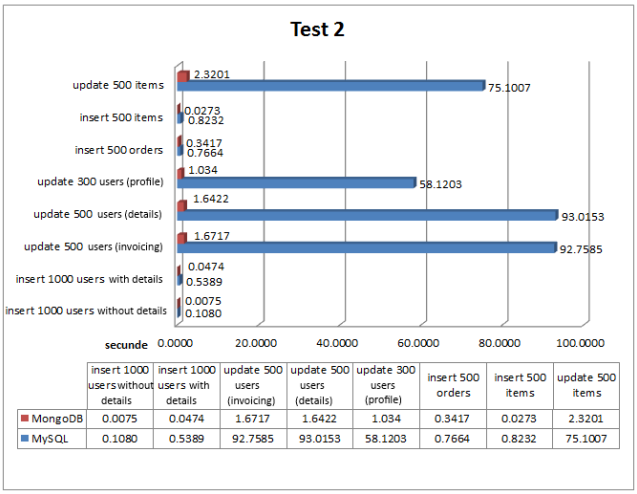
are relatively small, since no operations exceeded one second, in none of the databases.

B. Test 2

The second test includes even more parallel operations, namely:

- insertion of 1.000 users without details (*users* table);
- insertion of 1.000 users with details (*users* table);
- update billing info data for 500 users (*users_billing_info* table);
- update of some users' details, for 500 users (*users_details* table);
- update some users' public profile, for 300 users (*users_public_profile* table);
- insertion of 500 orders (*orders* table);
- insertion of 500 items (*items* orders);
- update of 500 items (*items* table).

The test described above requires simultaneous access to 6 tables, and the results are shown in Fig. 7.



	insert 1000 users without details	insert 1000 users with details	update 500 users (invoicing)	update 500 users (details)	update 300 users (profile)	insert 500 orders	insert 500 items	update 500 items
MongoDB	0.0075	0.0474	1.6717	1.6422	1.034	0.3417	0.0273	2.3201
MySQL	0.1080	0.5389	92.7585	93.0153	58.1203	0.7664	0.8232	75.1007

Fig. 7. The results of Test 2

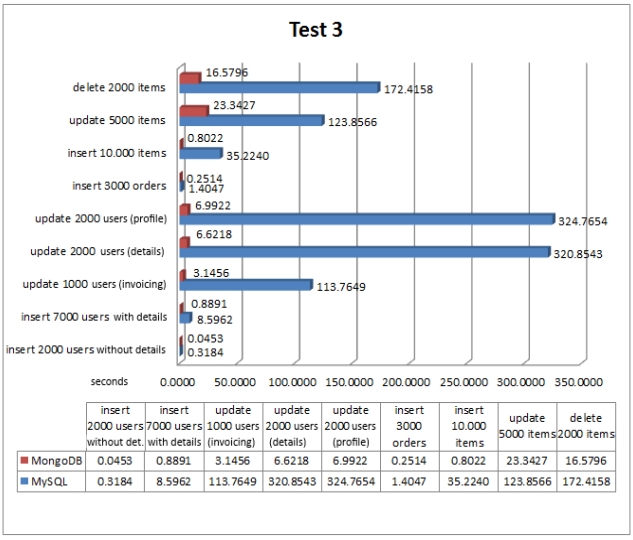
From the results of the second test (Fig 7), we can easily see that the greatest differences between the two databases arise in the *update* operation. While 500 users (with details) have been updated in MySQL in 93 seconds, as many users have been updated in MongoDB in less than 2 seconds. This is due to the different ways of accessing the two databases. For the update operation, the users were randomly selected in both databases.

C. Test 3

Although the operations described in the two tests above highlight the differences between the two databases to conclude we will present further a third test which includes the following:

- insertion of 2.000 users without details;
- insertion of 7.000 users with details;
- update of 5.000 users
  - 1.000 for modifying some billing info data;
  - 2.000 for modifying some general users' details;
  - 2.000 for updating users' public profiles.
- insertion of 3.000 orders;
- insertion of 10.000 de items;
- update of 5.000 items (modify name and description);
- delete of 2.000 items.

The results obtained due to performing this ultimate test are represented in Figure 8 and detailed in the below rows.



	insert 2000 users without det.	insert 7000 users with details	update 1000 users (invoicing)	update 2000 users (details)	update 2000 users (profile)	insert 3000 orders	insert 10.000 items	update 5000 items	delete 2000 items
MongoDB	0.0453	0.8891	3.1456	6.6218	6.9922	0.2514	0.8022	23.3427	16.5796
MySQL	0.3184	8.5962	113.7649	320.8543	324.7654	1.4047	35.2240	123.8566	172.4158

Fig. 8. The results of Test 3

The biggest differences between the two databases have emerged again in the *update* operation, which modified data in both databases. Thus, if 2.000 users change simultaneously different details, MongoDB is faster than MySQL for about 46 times. This is particularly important for applications that use databases that perform constant data update operations, such as Populations Records. For an application that does not require such amount of data to be modified often, such as an online shop, where the insert and delete operations are the most important, the update test is not as relevant as for the application described in this paper.

In a study conducted by authors in 2015 and that was presented in [6], during a conference, the presentation of various case studies in the field of programming and databases, different speed tests in terms of integrating a non-relational database in a forum, were performed. Besides the higher speed that MongoDB had a compared to MySQL, in all the operations that were performed, MongoDB has provided a very important benefit, namely, customizing the application. Thus, due to its use as a database in a forum, the application's structure became very variable, being different for each user. MongoDB allowed modeling the application to the needs of

users, thanks to the fact that a database in MongoDB does not have a predefined data structure, while MySQL has.

IV. CONCLUSIONS

From tests performed and presented for this application, an online publishing platform, the most suitable database was the non-relational MongoDB database. As such, a platform typically has thousands of users or even tens of thousands, MongoDB offered the best solution in terms of the speed at which different operations have to be performed in parallel by many users. If an application that does not require such amount of data to be modified often, such as an online shop, where the insert and delete operations are the most important, the update test is not as relevant as for the application described in this paper.

The advantage of using MongoDB database was further highlighted by conducting the tests and interpreting their results, which were presented in the previous chapter of this paper. MongoDB' query times were much lower than MySQL ones, which is essential when an application has to support thousands of users and multiple operations simultaneously.

We can choose MongoDB instead of MySQL if the application has thousands of users or even tens of thousands, which perform various operations at the same time and the application has to handle a huge volume of data. More and more applications are beginning to use a non-relational database because they provide a more flexible structure; they can support thousands of users and multiple operations simultaneously; they are designed to store large amounts of data and they are denormalized databases, which increases performance.

Switching from a relational database to a non-relational database can be a challenge in many ways, and in the end, the developers have the responsibility to decide which database should be used in a particular application, depending on its requirements and finding the optimal solution for the specific application.

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] 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.

[3] H. Martin, "REST and CRUD: the Impedance Mismatch". Developer World. InfoWorld, 29 January 2007, http://www.infoworld.com/article/2640739/application-development/rest-and-crud--the-impedance-mismatch.html.

[4] Kristina Chodorow, "MongoDB: The Definitive Guide, Second Edition" Published by O'Reilly, May 2013, pp 3-4.

[5] 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

[6] C. Győrödi, R. Győrödi, G. Pecherle, A. Olah, " A comparative study: MongoDB vs. MySQL", 13th International Conference on Engineering of Modern Electric Systems (EMES), 2015 , Oradea, Romania, 11-12 June 2015, pag. 1-6, ISBN 978-1-4799-7650-8.

[7] C. Győrödi, R. Győrödi, R. Sotoc, "A Comparative Study of Relational and Non-Relational Database Models in a Web- Based Application", International Journal of Advanced Computer Science and Applications, Volume 6 Issue 11, 2015, pag. 78-83, ISSN : 2158-107X(Print), ISSN : 2156-5570 (Online).

[8] The definitive guide to PHP's DocBook Rendering System Available: http://www.php.net/, accessed July 2016.

[9] R. P. Padhy, D. Panigrahy, "NoSQL Databases: state-of-the-art and security challenges", International Journal of Recent Engineering Science (IJRES), ISSN: 2349-7157, volume 16, October 2015, http://ijresonline.com/archives/volume-16/IJRES-V16P106.pdf

[10] Yingjie Shi, Xiaofeng Meng, Jing Zhao, Xiangmei Hu, Bingbing Liu, Haiping Wang, "Benchmarking cloud-based data management systems", CloudDB '10 Proceedings of the second international workshop on Cloud data management, pages 47-54, ACM New York, NY,USA 2010, ISBN: 978-1-4503-0380-4, doi: 10.1145/1871929.187193.

[11] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, D. Gosain, "A survey and comparison of relational and non-relational databases", International Journal of Engineering Research &Technology (IJERT) ISSN: 2278-0181, Vol 1, Issue 6, August 2012, pp. 1-5.