Performance Evaluation of NoSQL and SQL Queries in Response Time for the E-government

Ana Flores
Dirección Nacional de TIC'S
Función Judicial del Ecuador
Ambato,Ecuador
avfloresp@gmail.com

Javier Vargas
Facultad de Ingeniería en Sistemas, Electrónica e Industrial
Universidad Técnica de Ambato
Ambato, Ecuador
js.vargas@uta.edu.ec

Stalin Ramírez
Facultad de Ingeniería en Sistemas, Electrónica e Industrial
Universidad Técnica de Ambato
Ambato, Ecuador
dramirez8774@uta.edu.ec

Renato Urvina - Barrionuevo
Dirección Académica. Sistema de Nivelación.
Universidad Técnica de Ambato
Ambato, Ecuador
r_urvina@uta.edu.ec

Renato Toasa
School of Technology and Management, Computer Science and Communication Research Centre (CIIC)
Polytechnic Institute of Leiria
Leiria, Portugal
2162342@my.ipleiria.pt

José M. Lavin
Facultad de Jurisprudencia y Ciencias Sociales
Universidad Técnica de Ambato
Ambato, Ecuador
josemaria.lavin@uta.edu.ec

***Abstract*—Society demands accurate, right, quickly and easy access information. Therefore, governments are forced to implements or improve information systems with modern technologies. This research work is focused on the research of responses times on relational and non-relational data base models in an experimental data base system provided by the Función Judicial del Ecuador. The data is migrated for load in NoSQL allotment, and the performance tests is obtained a document-oriented data model. With these tests we could know the best database, focusing all on response time of data queries that have been used on performance tests. In addition, we evaluate the performance before and after the relational database administration system (RDBMS), obtaining as a result that the first test of the queries made to MongoDB substantially surpass the queries made to SQL Server, but in the subsequent tests the times they tend to be similar.**

Keywords— SQL, NoSQL, database, test, queries, response time, Document-Oriented Data Model

I. INTRODUCTION

The interaction between technology, e-participation processes, e-democracy, electronic government innovations and citizen satisfaction should be improved from the revolution in database systems. Nowadays, most public and private organizations use databases to manage and present their information in the best way. Technological advances in terms about Database Management Systems (DBMS) provide multiple benefits in different aspects such as speed of processing, scalability among others, which does not mean that they completely replace the systems that have been working until moment. But it is necessary to know the benefits of these tools in computer applications.

These needs are shown when it's necessary to extract information from the public dots databases of each government. Sometimes, these open databases have problems of slowness, due to overloads, an excessive number of users, size of files etc., which implies that it is necessary to increase the speed of extraction to give a better service to citizens. This situation happens on the website of Función Judicial Del Ecuador, which houses the records of all judicial cases in the country. The number of files and their size sometimes complicate access to them.

With the increasing accessibility to the Internet and the growing concept of cloud computing through the availability of web storage, large amounts of structured, semi-structured and unstructured data can be captured and stored for a variety of applications. These data are commonly known as Big Data [1]. The processing of this large amount of data requires speed, flexible and distributed schemes. One of the main structures for this purpose are the NoSQL bases that offer great advantages to operate in Big Data, leaving the SQL databases only for certain processes. [2].

This research makes a comparative analysis and evaluation of response times in queries between a NoSQL system and another relational SQL, using the model of documentary data, demonstrating the superiority of NoSQL or SQL systems when dealing with large amounts of data. This process is carried out in a test database containing judicial information, focusing on the web consultation system of the Judicial Council of Ecuador and using different operating systems.

The paper is organized as follows. Section II describes the context and state of the art. Section III presents the NoSQL and SQL evaluation. It is also shown Document Oriented Data Model in the evaluation. Then, in Section IV, the tests and results are detailed, and in Section V, we presented conclusions.

II. CONTEXT AND LITERATURE REVIEW

With the current emphasis on "Big Data", SQL and NoSQL queries are very important for large enterprise systems. Queries of substantial amounts of data are one of the best ways to get information through interaction between the user and computer systems. For this reason, we consider important to understand how SQL and NoSQL queries work in computer systems and when we must use them.

978-1-5386-2521-7/18/$31.00 ©2018 IEEE

Currently there are few SQL and NoSQL evaluation reports using data models and in a real system [3], but a broader search helped to find relevant documents, which are detailed below.

In [4] we can see how the performance of some SQL and NoSQL databases focused on key-value stores is investigated. The final results show that not all NoSQL databases work better than SQL databases.

In the business area, there are many systems that use SQL and NoSQL for their daily tasks, especially in web analysis and the support for large websites due to their high availability and scalability [5].

There are also research works on API standardization and the use of SQL with NoSQL systems. A common programming API was described that allows consulting Redis, MongoDB and HBase in [6]. While an SQL engine was integrated into HBase in the investigation [7], it used the Apache Derby query engine to process unions and other operators not compatible with HBase.

In [8] we can see how a SQL interface on SimpleSQL was defined. Particle [9] provides a generic SQL interface on key-value stores. These are some studies that can be mentioned with respect to Apis in SQL and NoSQL

In data modeling, NoSQL databases coexist with relational databases (SQL) in the enterprise data architecture. A conceptual data model is a description of the part of the real world. However, NoSQL data management currently lacks methods and mature tools to manage NoSQL data, as well as relational data [10].

Most of the existing NoSQL databases are designed with more consideration to the performance of applications, but no in high-level business models, data integration and data standardization. There is a gap between data modeling and physical data aspects of NoSQL databases [11]. The document-oriented data model formally describes a primitive query operator that does not retain redundancy and association, which is always preserved in the model. Analyzing the distinction of objects in response times and the equality of queries between NoSQL and SQL.

III. COMPARISON BETWEEN NOSQL AND SQL

The databases are containers that allow you to store large amounts of information in an organized, related and grouped way. From the computer point of view there are two types of relational (SQL) and non-relational (NoSQL) databases. Talking about NoSQL databases, it really means that these databases do not use the relational model. SQL is considered a standard in the way all the data in an application must be managed. The emergence of this relational model led to the emergence of new concepts such as tables, relationships, keys, rows.

For NoSQL, omitting the standard model can complicate the development and data management task. But focusing on the terms of use of this model, NoSQL facilitates the management of information in certain aspects, such as capturing or supporting scalability and access. As for the horizontal scale, it responds better than others and is also capable of storing large amounts of data. The following table shows a comparison between SQL and NoSQL.

TABLE I. SQL VS NOSQL [12][13].

Description	*No Relational*	*Relational*
Data structure.	It is a flexible structure, it is not necessary to define a data structure.	Assume that a well defined data structure should be uniform and the properties of this data can be defined in advance
Relationship between tables database.	There is no relationship between collections, however, it may depend on the data model.	It has to be very well established and systematically referenced.
Transaction	Use the CAP Theorem Consistency, availability Partition tolerance The CAP theorem states that it is impossible to guarantee all three characteristics simultaneously.	Use ACID: •Atomicity •Consistency •Isolation •Durability
Indexes and queries.	Decrease the use of indexing and the power of queries. They do not offer SQL as standard query	

The objective of this research is to perform a comparative analysis between relational and non-relational databases that will allow measuring the performance of each of the systems, with these results it is possible to know how optimal the databases are.

A. EXPERIMENTAL DATABASE

For the experiment / analysis, a database was used in SQL Server of the Función Judicial del Ecuador, which stores all the results of pending court cases. In each of the instances, there is a database called *Queries* that contains two flat tables *TableActor* and *TableDemand* where the purified causes are stored according to the type of litigant. Fig. 1 represents the architecture of the database.

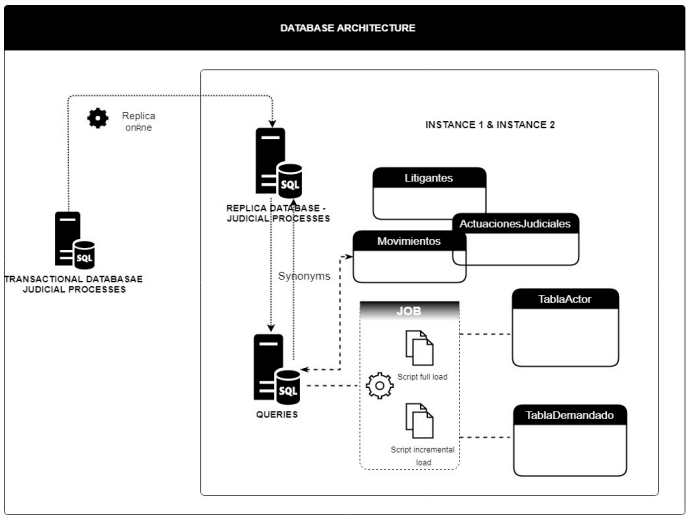

Fig. 1. Data Base Architecture – Causes Query.

From the architecture mentioned in the previous point, the table that will serve for our study is Judicial Actions. Table 2 establishes the characteristics of the table at the time of the test. The operation of the table is as follows: given a process number, you will obtain all the legal actions taken.

TABLE II. CHARACTERISTICS- SQL SERVER.

Characteristics	*Detail*
Number of records	29410990
Table Size	59314 MB
Types of data	char, varchar, text, int, datetime

The table contains a field called *Action of type* of text in SQL Server and stores the plain text of the legal action. Therefore, it is necessary to emphasize that an attempt is being made to analyze the effect produced by the variation in the consultations with respect to the size and quantity of judicial actions in the response times of the data management systems under study.

With the knowledge of how the data that will be migrated is structured, we will focus on the NoSQL document systems, with the following arguments:

- Allows complex queries to a set of data.
- They are hierarchical structures and provide means to retrieve records based on their actual content.
- Allows you to nest more complex structures.
- It has a semantic structure and its use is simple.

After defining the category of the data model that will be used in the NoSQL system, we select the data management system MongoDB 3.4 due to its characteristics [14]:

- It has a strong segmentation in the technological market
- Enhanced & Simplified Security Management
- Tunable Consistency Controls

The SQL server has its own tools to perform the migration to NOSQL, the same ones that were used in this investigation, so it is not necessary to deepen this process.

B. DATA MODEL ORIENTED TO DOCUMENTS IN THE EVALUATION

In the document-oriented data model, we work with complex query objects, analogous to relational database systems. On the other hand, relational systems should be an object-oriented DBMS, adaptable to natural query languages within the development architecture [15].

Inside of an analysis of the current architecture of the development, implementation and database of the query system, we must understand its operation and the interaction between the elements that compose it. The query systems in an application are divided into three layers: presentation, business logic and data access, which represent the architecture used for the evaluation.

The application sends read requests to the database server and due to its high concurrency, it works in an online replica of the database located in two different database instances, with two query nodes for two instances. NoSQL handles unstructured data, the information is not stored in tables, since they are stored differently (see Fig. 2), generally as key-value, it has a dynamic schema so that you can change the structure of the information without having to redesign it again [16].

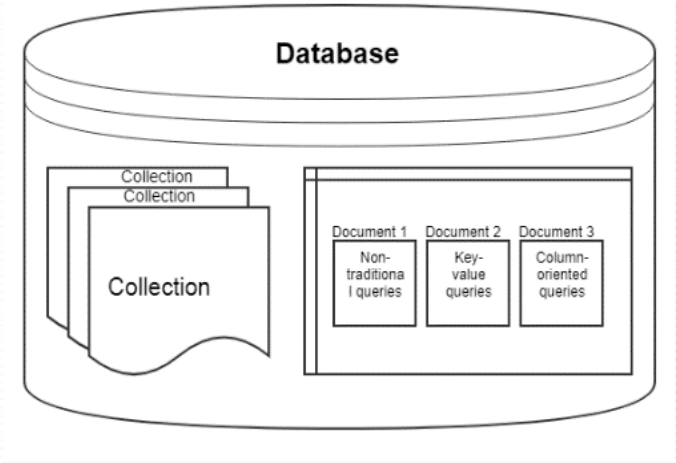

Fig. 2. Support to Database-Query.

The query is compatible with a relational database, in this case Microsoft SQL Server 2008 R2 and a SQL database engine, which considers not only the type of database, but the quick reference. To solve this, we propose the execution of bulk load tests and query of data on different types of SQL and NoSQL technologies:

- Non-traditional consultations
- Key value queries
- Queries oriented to columns.

In each of the instances, there is a database called *Queries* that contains two *TableActor* tables and *TableDemanded* tables where the objects are stored according to the type of Actor belongs to Defendant. The insert in these tables is done through queries execution of increments load jobs that process. For which the query process is normally performed, the response times may vary according to the number of matches entered by the parameter.

Thinking about a data model for NoSQL, we must analyze the structure that the documents will have and how the relationships between the data in the application. There are two tools that allow applications to represent those relationships: documented references and synonyms

IV. TEST AND RESULTS

To carry out the tests of this research, it is important to bear in mind the scenarios and test conditions detailed in Table 3. With the results obtained in each of these, the performance between SQL and non-SQL was contrasted:

TABLE III. CHARACTERISTICS - TEST TOOLS.

Characteristics	*Detail*
Operative System	Microsoft Windows Server 2008 R2 Enterprise 64-bits CentOS 7 64-bits
Database Management System	Microsoft SQL Server 2012 (SP3) MongoDB Community Edition 3.4
Evaluation software	JMeter 2.13

For the evaluation process it is important to describe the operation of the architecture (see Fig. 3) and the test plans used.

The experiment was carried out in a virtualization environment, *Proxmox Virtual Environment*. The virtual machines used to perform the load tests, were managed in this enviroment. Each of the data management systems were installed in a virtual machine with the following characteristics Table 3. In addition, a client station was created from which query requests will be made to validate the performance and behavior of the data management systems, for this the JMeter tool will be used.

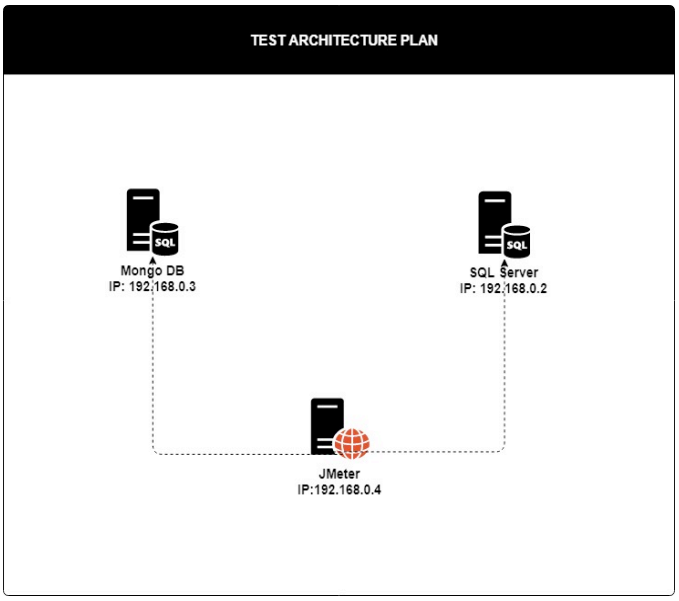

Fig. 3. Test Architecture Plan

According to IT infrastructure of the Función Judicial Del Ecuador, two evaluation testing environments were installed:

- Microsoft SQL Server 2012 and MongoDB 3.4 on Microsoft Windows Server 2008 R2 Enterprise 64-bit.
- MongoDB 3.4 over CentOs7 64-bits

Now, one of the important steps before the tests was the extraction, transformation and loading of data from Microsoft SQL Server 2008 to MongoDB 3.4 and to Microsoft SQL Server 2012. For this process it was necessary to create an ETL using the Microsoft SQL Server tool Integration Services Designer (SSIS).

The first process is to make query requests to validate the performance and behavior of the SQL and NoSQL databases. For the execution of the test plan we use the JMeter tool that allowed us to validate the performance and behavior of the concurrency management of a database server. For both SQL Server 2012 and MongoDB 3.4, the following scenario was configured (see Table 4).

TABLE IV. PERFORMANCE TEST

Test.- All legal proceedings registered in the database regardless of the amount or weight of the associated judicial proceedings	
Operation	***Selection***
Objective	Get the average response times of SQL Server and MongoDB using to complete the search process with 99 user numbers and X process.
Process	JMeter will carry out the Test Plan to randomly obtain 99 legal proceedings, which will then be used for each of the 99 users to trigger a query to the database every 2 seconds with a judicial process number.
Description	It runs 5 times each test and the average of the values obtained will be used to perform the analysis of results
Procedure	
1. Generate a list of 99 court proceedings random 2. Implement the corresponding query. 3. Execute the test plan programmed in JMeter with the query corresponding to each user. 4. Repeat the process of point 3	
SQL Server Code	
SELECT IdProceso, FechaActuacion, HoraActuacion ,TipoActuacion, NombreActuacion, Actuacion , Judicatura, Visible, loginT, FechaSistema FROM CLEX1.ActuacionesJudiciales WHERE IdProceso =?	
MongoBD Code	
db.getCollection('actuacionesjudiciales').find({"IdProceso" : "ProcesoJudicial"})	

The test was performed on Windows and Linux and the total values obtained by the JMeter Aggregate Graph compiler are shown in the following tables:

TABLE V. SQL SERVER ON WINDOWS

Label	*Samples*	*Average*	*Median*	*90 %Line*	*Min*	*Max*	*Error %*	*Throughput*	*KB/sec*
SQL Server Request1	99	1183	1446	1462	217	1510	0,00%	64,1	91145,7
SQL Server Request2	99	1048	1229	1250	172	1330	0,00%	73,7	102731,1
SQL Server Request3	99	920	1129	1192	120	1226	0,00%	79,4	110722,2
SQL Server Request4	99	835	1059	1080	250	1167	0,00%	83,7	116712,2
SQL Server Request5	99	792	863	909	70	937	0,00%	1045	145797,8

TABLE VI. MONGO ON WINDOWS

Label	*Samples*	*Average*	*Median*	*90 %Line*	*Min*	*Max*	*Error %*	*Throughput*	*KB/sec*
MongoDB Request1	99	1319	1344	2291	73	2531	0,00%	39,1	27,3
MongoDB Request2	99	1186	1194	2152	34	2422	0,00%	40,8	28,5
MongoDB Request3	99	1149	1142	2079	39	2319	0,00%	42,0	29,7
MongoDB Request4	99	1157	1156	2038	38	2290	0,00%	43,2	30,1
MongoDB Request5	99	1173	1164	2103	38	2381	0,00%	41,5	29

TABLE VII. MONGODB ON LINUX

Label	*Samples*	*Average*	*Median*	*90 %Line*	*Min*	*Max*	*Error %*	*Throughput*	*KB/sec*
MongoDB Request1	99	958	957	1722	23	1905	0,00%	52	36,3
MongoDB Request2	99	940	929	1671	20	1889	0,00%	52,3	36,5
MongoDB Request3	99	920	912	1681	22	1908	0,00%	51,9	36,2
MongoDB Request4	99	909	904	1657		1861	0,00%	53,2	37,2
MongoDB Request5	99	908	900	1657	21	1858	0,00%	53,3	37,2

For the data obtained from the analysis of the results of the test performed on SQL Server and MongoDB, we proceed to work with the average field information, which represents the average time in milliseconds (ms) of the execution of each of the tests to respond to the requests of 99 concurrent users consult each a different judicial process, of these results 5 samples were taken. The following tables show the average values.

TABLE VIII. SQL SERVER ON WINDOWS

Label	*Averange*
SQL Server request1	1168
SQL Server request2	953
SQL Server request3	882
SQL Server Request4	858
SQL Server Request5	642
Total average:	900,6

TABLE IX. MONGODB ON WINDOWS

Label	*Averange*
MongoDB Request1	1222
MongoDB Request2	1155
MongoDB Request3	1127
MongoDB Request4	1141
MongoDB Request5	1200
Total average:	1169

TABLE X. MONGODB ON LINUX

Label	*Averange*
MongoDB Request1	919
MongoDB Request2	902
MongoDB Request3	876
MongoDB Request4	898
MongoDB Request5	888
Total average:	896,6

The following figure indicates the response time in milliseconds used by the SQL Server and MongoDB database systems to respond to 99 concurrent requests to a table or collection that contains respectively 29410990 records or documents.

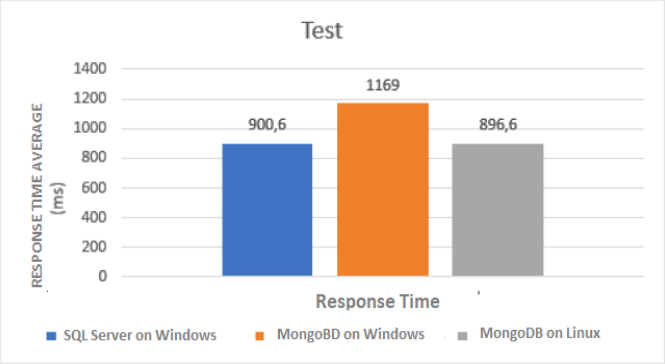

Fig. 4. Response Time Test.

Several load tests were performed similar to the previous ones, which show that the difference between SQL Server and MongoDB is high. The average times to obtain query results are constant in MongoDB and the time used to resolve the first queries is less than SQL Server.

MongoDB offers better performance over Linux operating systems. While SQL Server tends to increase response times when it comes to quantity. It should be emphasized that the consultations do not have any complexity, but it focuses on the amount of information with which one works.

V. Conclusions

We observed in the results of the tests that the SQL Server response times in the first consultations are very high, but after several queries, the times tend to decrease. This is because the data is no longer processed on the disk; otherwise, MongoDB with respect to SQL Server takes less time to resolve the first queries, maintaining a small variable response time that tends to decrease.

One of the most important aspects that can be highlighted is the variation in the response times presented by the MongoDB NoSQL data management system, with respect to the operating system where it is installed, observing a better performance in Linux operating systems despite that MongoDB is a multiplatform system.

SQL Server has a table structure to store the data and MongoDB uses the JSON document format, in order to migrate the information contained in SQL Server to MongoDB, an extraction, transformation and loading (ETL) process is necessary, within which the JSON documents later stored inside the non-relational engine. For this process, a previous data analysis for the migration was considered, concluding that for this case of study the structure of the JSON documents presented a complexity for the implementation. For future work, the efficiency of other data storage architectures (key-value, graphs, columns) for NoSQL will be tested. In addition, clusters will be implemented in the physical architecture

References

[1] 1 Warden, P.: Big Data Glossary. O'Reilly Media, Inc. (2011).2. Chen, M., Mao, S., Liu, Y.: Big Data: A Survey. Mob. Networks Appl. 19, 171–209 (2014).

[2] 2 SKIENA, Steven S. Big Data: Achieving Scale. En The Data Science Design Manual. Springer, Cham, p. 391-421 (2017).

[3] 3 Moniruzzaman, A. B. Hossain S. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison (2013).

[4] 4. Li, Y., Manoharan, S.: A performance comparison of SQL and NoSQL databases. In: 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM). pp. 15–19. IEEE (2013).

[5] 5. Lawrence, R.: Integration and Virtualization of Relational SQL and NoSQL Systems Including MySQL and MongoDB. In: 2014 International Conference on Computational Science and Computational Intelligence. pp. 285–290. IEEE (2014).

[6] 6. Atzeni, P., Bugiotti, F., Rossi, L.: Uniform Access to Non-relational Database Systems: The SOS Platform. Presented at the (2012).

[7] 7. Vilaça, R., Cruz, F., Pereira, J., Oliveira, R.: An Effective Scalable SQL Engine for NoSQL Databases. Presented at the (2013).

[8] 8. Calil, A., dos Santos Mello, R.: SimpleSQL: A Relational Layer for SimpleDB. Presented at the (2012).

[9] 9. Tatemura, J., Po, O., Hsiung, W.-P., Hacigümü\cs, H.: Partiqle: An Elastic SQL Engine over Key-value Stores. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. pp. 629–632. ACM, New York, NY, USA (2012).

[10] 10. Wang, A.: Unified Data Modeling for Relational and NoSQL Databases, https://www.infoq.com/articles/unified-data-modeling-for-relational-and-nosql-databases.

[11] 11. Jing Han, Haihong E, Guan Le, Jian Du: Survey on NoSQL database. In: 2011 6th International Conference on Pervasive Computing and Applications. pp. 363–366. IEEE (2011).

[12] 12 MEIER, Andreas. Unterschied zwischen SQL-und NoSQL-Datenbanken. En Werkzeuge der digitalen Wirtschaft: Big Data, NoSQL & Co. Springer Vieweg, Wiesbaden, p. 9-12, (2018).

[13] 13 KIM, Ho-Jun, et al. Migration from RDBMS to Column-Oriented NoSQL: Lessons Learned and Open Problems. En Proceedings of the 7th International Conference on Emerging Databases. Springer, Singapore, p. 25-33 (2018).

[14] 14. Moniruzzaman, A.B.M., Hossain, S.A.: NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. (2013).

[15] 15. Dhande, S., Bamnote, G.R.: Improving query processing performance using optimization techniques for object-oriented DBMS. 380, 119–127 (2016).

[16] 16. Hecht, R., Jablonski, S.: NoSQL evaluation: A use case oriented survey. Presented at the (2011).