TASHKENT UNIVERSITY OF
INFORMATION TECHNOLOGIES
NAMED AFTER MUHAMMAD AL-KHWARIZMI

MUHAMMAD AL-XORAZMIY NOMIDAGI
TOSHKENT AXBOROT TEXNOLOGIYALARI
UNIVERSITETI

BULLETIN OF TUIT:

MANAGEMENT AND COMMUNICATION TECHNOLOGIES



ISSN:2181-1083 DOI:
Askaraliyev O.U., Normatova M.N. № 4(31)2024

Organizational and information support of NoSQL data structures in integrated information systems

Askaraliyev Odilbek
Rector of Sarbon University, Tashkent city 100 000, Uzbekistan
e-mail: oasqaraliyev77@gmail.com

Normatova Malika
Researcher Gulistan State University, Gulistan city 120 100, Uzbekistan
e-mail: normatova.2989@gmail.com

Abstract - This article presents an analysis of the organizational and information support of the NoSQL data structure in integrated information systems. For experimental studies, an architecture was developed based on virtual machines simulating the operation of a web service for data collection with a classic three-tier architecture. It was proved that differences in the structure of an unstructured database can affect the reliability and efficiency of the entire information system. The article proposes a methodology for assessing the effectiveness of the data structure of corporate information systems using the MongoDB database management system (DBMS), a methodology for information and organizational support of the database.

Keywords - unified data processing system (UDPS), NoSQL, information system, information provision, data structure, web service, efficiency, modeling, architecture, logical connection, integration.

I. INTRODUCTION

In the last decade, many data acquisition, processing, and transmission systems have become web applications. Changes in software design, human-machine interface, architecture, and application requirements have determined the main requirements for storing big data[1]. Large volumes of data pose serious problems in ensuring the scale and performance of systems based on relational databases. Document-based databases allow storing documents consisting of tagged elements, and the most popular NoSQL technologies are used in them. MBBT uses BSON (Binary JavaScript Object Notation) as a data document model. This model provides convenience and high performance for program processing.

In general, the task of choosing a structure requires special resource study in each specific case[3]. It has been found that modern methods of database design do not take into account non-functional requirements. Using the example of the system under consideration in the article, a methodology for conducting experimental research aimed at obtaining adequate evaluations in given working conditions has been developed. The obtained results and conclusions apply only to the considered system in terms of the number of users and the infrastructure configuration. However, the resulting methodology can be useful for developers of information systems to solve

the problem of choosing a method of organizing loosely structured data storage.

II. MAIN PART

This study was conducted to develop a unique digital platform in a specific computing infrastructure configuration. The corporate governance system of tax authorities is expected to collect data from more than 20,000 participants simultaneously through communication channels in conditions of limited computing resources.

Taking into account the expected nature of data exchange, ensuring the performance of write operations is set as a priority task. It is important to select possible options for data structures that should be stored in the database to perform the task in the system. Two possible options can be considered:

1) storing the results of each participant's (taxpayer's) activity as a single document in a single data set (this can also be called a denormalized database);

2) storing the results of the activities of each process participant in separate data sets by dividing them into several documents (in this case, the principles of database normalization can be applied to varying degrees, for example, to reduce the structural complexity of the data).

MongoDB uses the following components to deploy a cluster:

– *Mongod:* The main process (domain) for managing access to data. This is the database server.

– *Mongos:* The process responsible for routing between the user application and the MongoDB database.

– *config.server:* Stores metadata about the placement of documents on the database servers to efficiently process requests from user applications.

– Database servers can be grouped as follows:

– *Replica set:* A group of MongoDB processes that maintain identical copies of the data. The primary replica maintains the master copies of the documents. The secondary replicas replicate the primary. If the master fails, a voting process is initiated to determine a new master copy by majority vote.

– *Shard:* This is a set of replicas that store a portion of a database. It is used to distribute parts of a database across multiple servers.

MongoDB has two types of replication schemes: *Master-Slave Replication* and *Replica-Set Replication.*

Replica set: Provides good automation and redundancy. Regardless of the number of shard replicas, only one replica acts as the primary replica. All write and read operations are sent to the primary replica and then replicated to the secondary replicas.

The following tools were used to set up the experiment on a local virtual infrastructure:

– *VirtualBox* is a virtualization system;

– *Vagrant* is a virtual infrastructure management tool;

– *Ansanbl* is a provisioning system;

– *Ubuntu Xenial* is a guest operating system;

– *Atop* is a performance monitoring tool;

– *Gulp.js* is a tool for automating frequently used tasks.

The infrastructure preparation scheme using the listed tools is shown in Figure 1. The infrastructure deployment was carried out on the following technical tools:

– Central processor – AMD Ryzen 7 3700X 8-core processor, 3600 MHz, 8 physical cores, 16 logical cores.

– RAM – 32 GB DDR4, frequency 1600 MHz, dual-channel mode.

– Disk subsystem – Samsung SSD 970 EVO Plus.

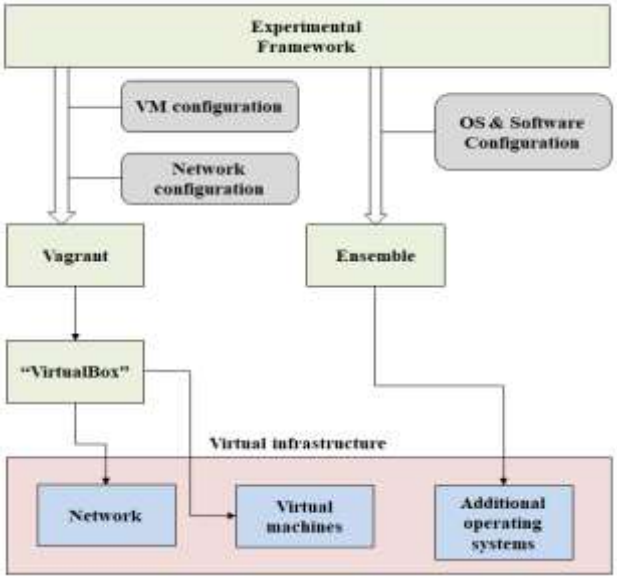


Fig.1. Virtual infrastructure preparation scheme

Table 1.

Virtual machine configuration

№	Roles	The number of cores of the central processor	RAM size	CPU limitations	Disk I/O speed limit
1	User Query Generator	4	8 Gbayt	-	-
2	Server	2	8 Gbayt	-	-
3	DBMS	2	8 Gbayt	50 %	25 MB/sec.

The prepared set of virtual machines is configured as shown in Table 1.

The experiment was conducted using the following technological tools:

1. *Node.js version 13.x* – a platform for executing query generator and server application code.

2. *Loopback.io 3* – a server application framework, the "loopback-connector-MongoDB" connector version 3.3.1 was used to interact with MBBT.

3. *MongoDB version 4.2.3 Community* – for document-oriented MBBT to store transmitted data.

The experiment algorithm consists of several stages:

1. Starting virtual machines.

2. Loading a data array into the RAM of the client query generator virtual machine.

3. Sequentially recording an invariant dataset (ResearchSubject).

4. Sequentially recording a test dataset (ResearchResult, ResearchResultAnswer).

5. Obtaining results on the number of records stored.

6. The interaction of the components of the experimental setup during the initialization of the virtual infrastructure and the execution of the algorithm is shown in Fig. 2.

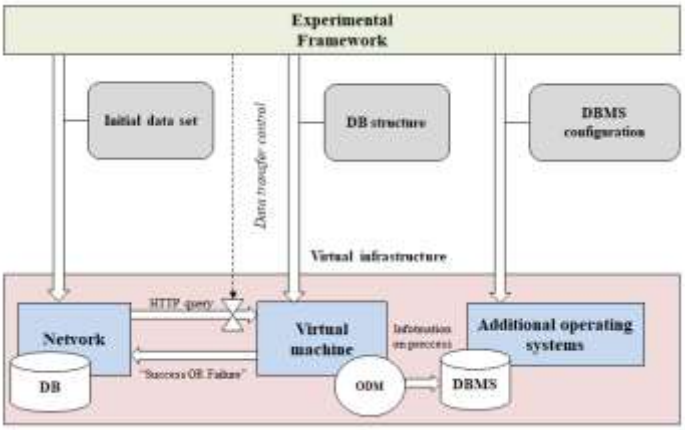


Fig. 2. Database management system, structure and data preparation

Two structures are proposed for storing the data shown in Figures 3 and 4. The structure of the ResearchSubject collection is immutable and does not require detailed consideration, while the structure of the ResearchResult collection implies two alternative solutions: storing in a single data set or dividing it into two sets (ResearchResult and ResearchResultAnswer), which have a “more than one” logical relationship.

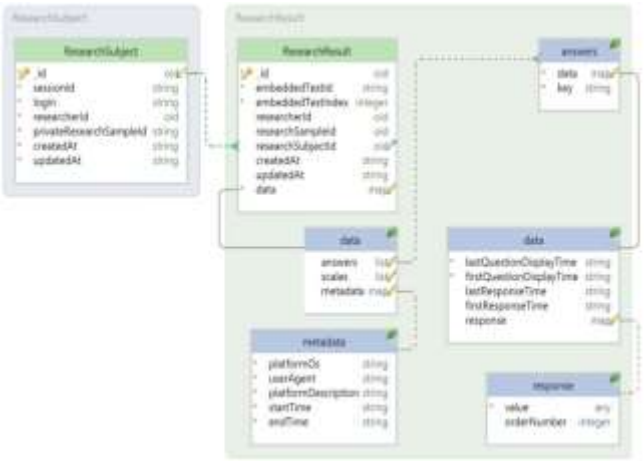


Fig. 3. IDEF1X model structure of the MongoDB database №1

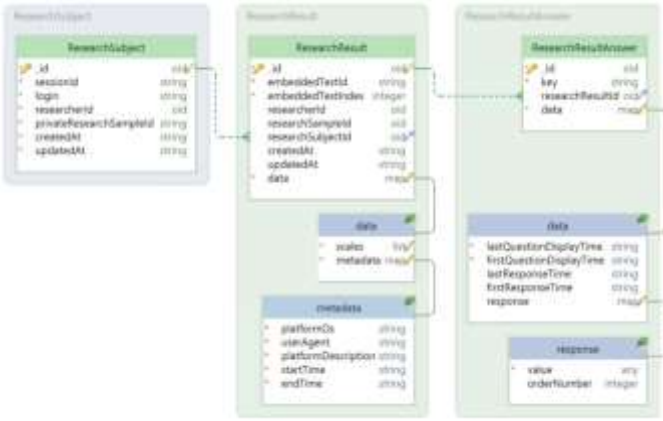


Fig. 3. IDEF1X model structure of the MongoDB database № 2

Two algorithms were used to write data to the database: algorithm 1 denormalized data stored in the ResearchResult collection, algorithm 2 divided the stored data into two collections ResearchResult and ResearchResultAnswer (Figures 5 and 6, respectively).

```
function saveResult(result) {
cont {ResearchSubject} = this.app.models;

const rrs=_.omit(result, []);

return ResearchSubject.findByID(result.ResearchSubjectID)
.then(()=>ResearchResult.create(rrs));
}
```

Fig. 4. Algorithm for writing data to MongoDB in a single batch (algorithm #1)

```
function saveResult(result) {
cont {ResearchSubject, ResearchResultAnswer} = this.app.models;

const rrs=_.omit(result, ['data.answers']);
const rras=_.map(
result.data.answers,
(answer)=>_.assignIN({}, answer, { ResearchResultID:result.ID})
);

return ResearchSubject.findByID(result.researchSubjectID)
.then(()=>ResearchResult.create(rrs))
.then(()=>ResearchResultAnswer.create());
}
```

Fig. 5. Algorithm for writing data to MongoDB in 2 separate sets (algorithm #2)

The structure of experimental research methodology for algorithms and data structures No. 1 and No. 2 is developed and presented in Figures 7 and 8.

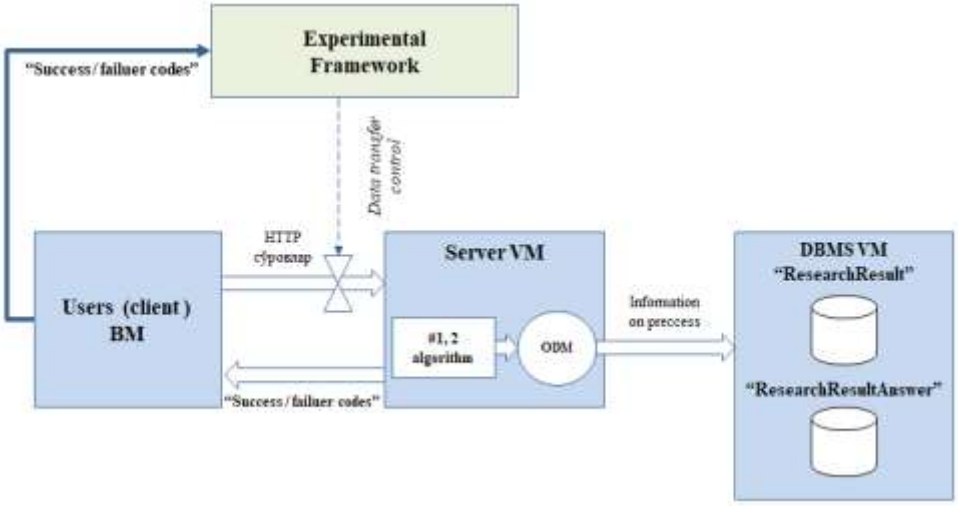


Fig. 6. Experimental structure using algorithm № 1 and data structure № 2

Requests were sent in 4 threads, each of which was executed on a separate processor core. At the same time, 1000 simultaneous requests were made in each thread (totaling up to 4000 simultaneous connections to the server) with a waiting time of 10 seconds. After 4000 requests were sent (a response was received from the server, the response waiting time expired), an additional 10-second wait was set before starting the next iteration of sending requests. An additional 60-second wait was also introduced between the experimental stages.

III. RESULTS

The experiments were carried out at the Scientific and Information Center for New Technologies under the State Tax Committee of the Republic of Uzbekistan. In this case, there are no differences in the load on the central processor of the MBBT virtual machine when storing an unchanging data set (Fig. 7) (the part of the graph before the first vertical separator, approximately 180 seconds). The graphs of the central processor load when writing different data structures differ (Fig. 9). The incomplete use of processor resources is possible due to the fact that the server application was restarted 3 times during the second experiment. After the end of sending client requests, attention should also be paid to the peak load (the second vertical separator, approximately 360 seconds). This corresponds to the long and resource-intensive processing of requests that were sent last, but did not lead to a restart of the server application and, as a result, data loss.

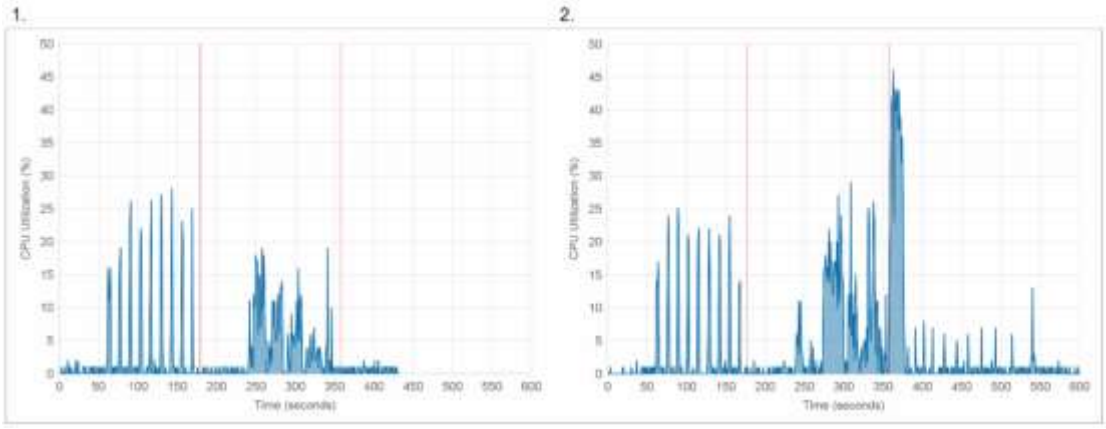


Fig. 7. CPU usage graph for the MBBT virtual machine

In none of the experiments was the disk I/O subsystem significantly loaded, so it is not important to take this indicator into account.

The amount of free RAM decreases in both the first and second experiments (Figure 8). As is known, in the standard configuration, MongoDB caches part of the data in RAM. The larger space occupied in the first experiment corresponds to the fact that more data is stored in the database. No other significant differences in this indicator were observed in the experiments.

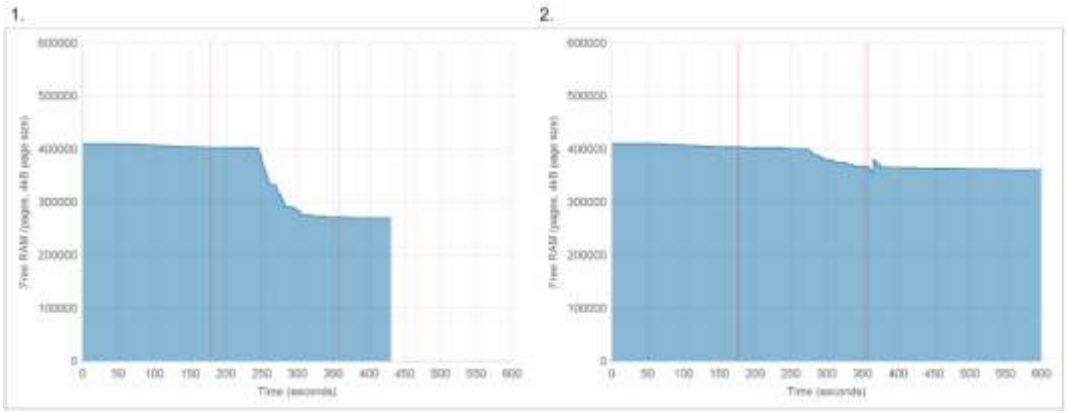


Fig. 8. Free RAM graph of the MBBT virtual machine

B. Server application virtual machine parameters

During the storage of a fixed data set (Fig. 12), there is no difference in the CPU load of the server application virtual machine (the part of the graph before the first vertical separator, approximately 180 seconds).

Both graphs (Fig. 8) show that the CPU load of the server application virtual machine is at its maximum during the storage of the main volume of data. If during the first experiment there is a period of almost complete decrease, then in the second it remains constantly high. In addition, the second experiment shows a long and resource-intensive processing of the last sent requests.

The graph of the second experiment (Figure 8) clearly shows the 3-time reboot of the server application, which is characterized by intensive access to the disk I/O system. Otherwise, the results obtained are as expected.

For the second experiment (Figure 9), 3 peaks are clearly visible when the RAM is

released due to the server application being restarted. This fully corresponds to the results shown in the graph in Figure 9. Otherwise, the expected pattern is visible: the amount of free memory decreases after receiving client requests and increases after they are processed and garbage collection begins.

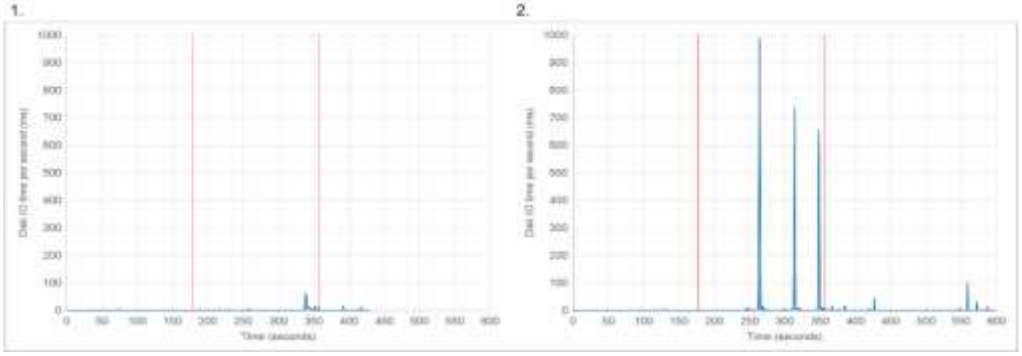


Figure 9. Time graph of disk I/O operations for a server application virtual machine

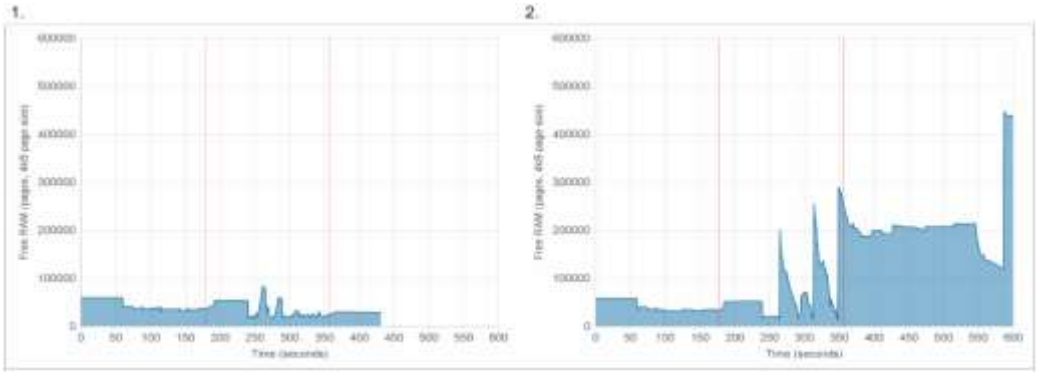


Fig.10. Free RAM graph of a virtual machine server application

Table 2 show the numerical values of experimental studies. The results presented in Table 2 and the identical resource utilization indicators (Figures 7-10, the parts of the graphs up to the first vertical axis, approximately 180 seconds) confirm the equal conditions for conducting experiments.

Table 2.

Efficiency of recording experimental data.

№	Indicatiors	Results for experiment № 1	Results for experiment № 2
1	Experiment time (seconds)	252	423
2	Documents saved by *ResearchSubject*	7034 (41,66%)	2497 (14,79%)
3	*ResearchSubjectAnswers* бўйича сақланган ҳужжатлар	- (41,66%)	398064 (8,07%)
4	Server messages about successful writing of *ResearchResult* and *ResearchResultAnswer* documents	3619(21,44%)	21(0,12%)
5	Server error message when writing *ResearchResult* and	0 (0%)	0 (0%)

	ResearchResultAnswer documents		
6	Timed out waiting for server response while writing *ResearchResult* and *ResearchResultAnswer* documents	13265(78,56%)	16862(99,88%)

III.CONCLUSION

The experiments were carried out at the Scientific and Information Center for New Technologies under the State Tax Committee of the Republic of Uzbekistan. The article developed a methodology for analyzing computing resources for two variants of storage structures for denormalized data in the MongoDB document-oriented NoSQL system. Experimental results were obtained under simulated conditions of large data flows and limited computing

resources. The experimental results (Table 2) show the following:

- Writing data divided into two sets takes more time than writing objects to a single set;
- Data loss is significantly lower when writing to a single set, and the probability of writing denormalized data is low;
- It was found that the quality of service for client requests when dividing data into two sets is lower than when writing to a single set.

When the server program fails, some data may be lost or lost. In summary, differences in the data structures used, based on the methods used for them, can create problems and affect the performance of software systems.

REFERENCES

[1]. W. Hendricks, "Review of NoSQL Data Stores: Using a reactive threetier application for software developers to achieve a high availability application design architecture," In 2019 Open Innovations (OI), 2019, pp. 71-77.

[2]. Мусаев М.М. Современные методы цифровой обработки речевых сигналов.// Вестник ТУИТ 2(42)/2017. -с. 2-13.

[3]. Рахматуллаев М.А. Мировые информационные ресурсы. –Т.: «Fan va texnologiya», 2017. 196 с.

[4]. A. Corbellini, C. Mateos, A. Zunino, D. Godoy and S. Schiaffino, "Persisting big-data: The NoSQL landscape," Information Systems, vol. 63, pp. 1-23, 2017.

[5]. P. Kolyasnikov et al., "Analysis of software tools for longitudinal studies in psychology," International Journal of Advanced Compyuter Science and Applications, vol. 10, no. 8, pp. 21–33, 2019.

[6]. N.Q. Mehmood, R. Culmone and L. Mostarda, "Modeling temporal aspects of sensor data for MongoDB NoSQL database," Journal of Big Data, 4(1), 82017

[7]. Roy-Hubara, N., Shoval, P., & Sturm, A. "A Method for Database Model Selection," In Enterprise, Business-Process and Information Systems Modeling, 2017, pp. 261-275.

[8]. Mehmood, E., & Anees, T. (2019). Performance Analysis of Not Only SQL Semi-Stream Join Using MongoDB for Real-Time Data Warehousing. IEYEE Access, 7, 134215-134225.

[9]. Chopade, R., & Pachghare, V. (2020). MongoDB Indexing for Performance Improvement. In ICT Systems and Sustainability (pp. 529-539). Springer, Singapore.

[10]. B.M. Basok, A.N. Rozhanskaya and S.L. Frenkel, "On web-applications usability testing," Russian Technological Journal, vol. 7, no. 6, p. 9- 24, 2019. DOI: 10.32362/2500-316X-2019-7-6-9-24

[11]. R. Cattell, "Scalable SQL and NoSQL data stores," Acm Sigmod Record, vol. 39, no. 4, pp. 12-27, 2011.

[12]. V. Herrero, A. Abelló and O. Romero, "NOSQL design for analytical workloads: variability matters," In International Conference on Conceptual Modeling, 2016, pp. 50-64.

[13]. N. Roy-Hubara and A. Sturm, "Design methods for the new database era: a systematic literature review," Software & Systems Modeling, vol. 19, pp. 297–312, 2020.

[14]. Atzeni, P., Bugiotti, F., Cabibbo, L., & Torlone, R. (2020). Data

[15]. modeling in the NoSQL world. Compyuter Standards & Interfaces, 67, 103149.

[16]. J. K. Chen and W. Z. Lee, "An Introduction of NoSQL Databases based on their categories and application industries," Algorithms, vol. 12, no. 5, p. 106, 2019

[17]. Mahruqi, R. S. A., Alalfi, M. H., & Dean, T. R. (2019, November). A semi-automated framework for migrating web applications from SQL to document oriented NoSQL database. In Proceyedings of the 29th Annual International Conference on Compyuter Science and Software Engineering (pp. 44-53).