## РОЗДІЛ 2

## ПРОГРАМНІ ЗАСОБИ УПРАВЛІННЯ, ЗБОРУ, ОБРОБКИ І ПЕРЕДАЧІ ІНФОРМАЦІЇ

UDC 656.078.1

# DEVELOPMENT OF AN INFORMATION SYSTEM TO JUSTIFY THE CHOICE OF DATABASES WHEN USING CRM SYSTEMS

M. Alekseev, S. Ochkur, I. Hnennyi
(Ukraine, Dnipro, NTU «Dnipro Polytechnic»)

**Introduction.** This paper is about the development and research of a method for evaluating the performance of databases that are used in CRM systems. The development of a separate CRM system for each organization is determined by the inability to solve the tasks on the basis of ready-made existing platforms. Finished products are designed to solve typical problems, not suitable for all companies. While individual development allows us to solve the main issues of the company related to internal document management and customer relations.

The main goal is to develop an information system for the method of evaluating and justifying the choice of a database model for CRM systems.

The main problem is that any database model can be used for each CRM system, which can adversely affect the performance of the system as a whole.

**The aim of this paper** is to develop an information system to justify the choice of databases when using CRM systems.

**Ready-made solutions.** While NoSQL databases have the speed and scalability advantage, there have a number of drawbacks compared to traditional relational databases. Leavitt lists these challenges [1]. He notes that NoSQL databases, even though fast for simple tasks, are time-consuming for complex operations. Besides queries for complex operations can be hard to form. The other drawback is the lack of native support for consistency. Leavitt also notes that NoSQL is a technology that many organizations are yet to learn and there is a lack of support and management tools to help.

Bartholomew gives a tutorial introduction to the history of and differences between SQL and NoSQL databases [2]. Sakr et al. discuss data management solutions, including NoSQL, for cloud-based platforms [3]. They discuss the challenges data management solutions face in the light of the cloud.

Hecht and Jablonski provide a use-case oriented survey of NoSQL databases [4]. They identify the difficulties in choosing a NoSQL database to fit a particular use-case, and therefore focus their paper to address this. They use as the basis for their comparison the data model, support for queries, partitioning, replication, and concurrency controls. They compare in this light fourteen NoSQL databases, including MongoDB, CouchDB, Cassandra and HBase.

Boicea et al. compare a NoSQL database against a SQL database. They choose Oracle for the SQL implementation and MongoDB for the NoSQL implementation [5]. They report that, with a large number of records, insertion time is a factor more in Oracle and update and delete times are several factors more in Oracle.

Yahoo! Cloud Serving Benchmark is an open-source work-load generator tool for comparing key-value stores [6].

**Solve.** In the study of performance of database management systems, the following criteria will be the benchmarking criteria: adding, finding, modifying, and deleting from a single table. All listed operations are present in MySQL and MongoDB.

Several tests were compiled for the study. You must create two tables before you start testing. The first table will consist of 30,000 entries with identical text and a random foreign key in the range from 1 to 100,000. The second table will consist of 100,000 entries, the keyCol and valueCol fields of each row will be equal to each other and take values from 1 to 100,000.

The text in the first table is only needed to increase the amount of data recorded. The keyCol attribute of the second table is named to emphasize that it is not a primary key. It is not the primary key for the reason that it will slow down the speed of MySQL operations through additional integrity checks. The keyCol and valueCol fields accept the same values just to simplify perception, conceptually it does not affect anything.

It should be noted that the second table is larger than the first table. This is done for several reasons. First, it should make searching for keyCol a little more complicated. Secondly, due to the first fact, it will make it somewhat difficult to join the two tables.

The first test will be adding up to 100,000 rows to the keyCol_valueCol table. The time commit is 1000 entries. The second test is to link two tables. In MySQL is used LEFT JOIN, in MongoDB - $ lookup. In this test, the size of the textCol_outsideKeyCol table will vary from 10,000 to 30,000 records.

The third test is to search up to 10,000 records in the keyCol_valueCol table.

The fourth test changes the value of the valueCol column by keyCol to 10,000 rows. The last test is to delete entries in the keyCol_valueCol table with keyCol for up to 10,000 rows.

Since all tests, except the first, require a keyCol attribute search, the tests are performed twice: with indexing of the attribute and without indexing.

For ease of testing, a Python program is written in three parts: the main program, the module with the MySQL test function, the module with the MongoDB test function.

Linux Ubuntu 18.04.3 LTS distribution was used as the server operating system. MySQL 8.0.17 and MongoDB 4.2.0 are installed as local databases and can be accessed through the computer's internal address. A computer with the following configuration was used as the server:

- processor: Intel Pentium CPU B950 clocked at 2.10 GHz 2 cores;
- RAM: DDR3 6 Gb;
- Video Adapter: GeForce GT 540M 1 Gb.

**Test results.** The first test (using indexes) is to perform line insertion. Line insertion runs in stages from 10,000 to 100,000 records. The test results are shown in Figure 1.


Fig. 1. Chart of time of operation of INSERT operation on number of records

Based on the results above, we can conclude that the insertion operation runs faster in MySQL almost 2 times than in MongoDB. It is also worth noting that as the number of MySQL product lines increased, it declined less rapidly than MongoDB.

The second test (using indexes) is the binding operation. The test uses the left link in stages from 10,000 to 30,000 rows. MySQL DBMS is a great time-consuming bind operation.

In other tests, MySQL was also better than MongoDB. But when tested without using indexes in a relational model, MongoDB showed an advantage in search, update, and delete operations.

The result of the search operation test is shown in Figure 2.

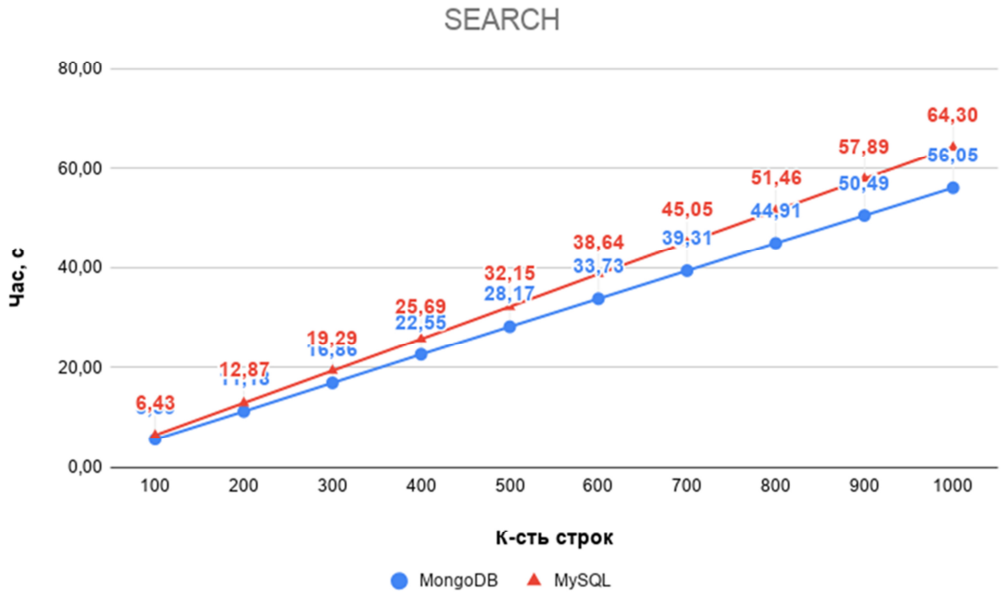


Fig. 2. Diagram of the execution time of a SELECT operation (not index) on the number of entries

According to the results of this test, we can conclude that the search time without indexing is almost the same in the two DBMSs, but MongoDB showed the best time for this test.

The result of the upgrade operation test is shown in Figure 3.


Fig. 3. UPDATE (not index) operation time chart versus number of records

According to the results of the fourth test, MongoDB showed the best time. It should be noted that with the increase in the number of records, the execution time of the operation in MySQL increased more intensively than in MongoDB.

The result of the removal test is shown in Figure 4.

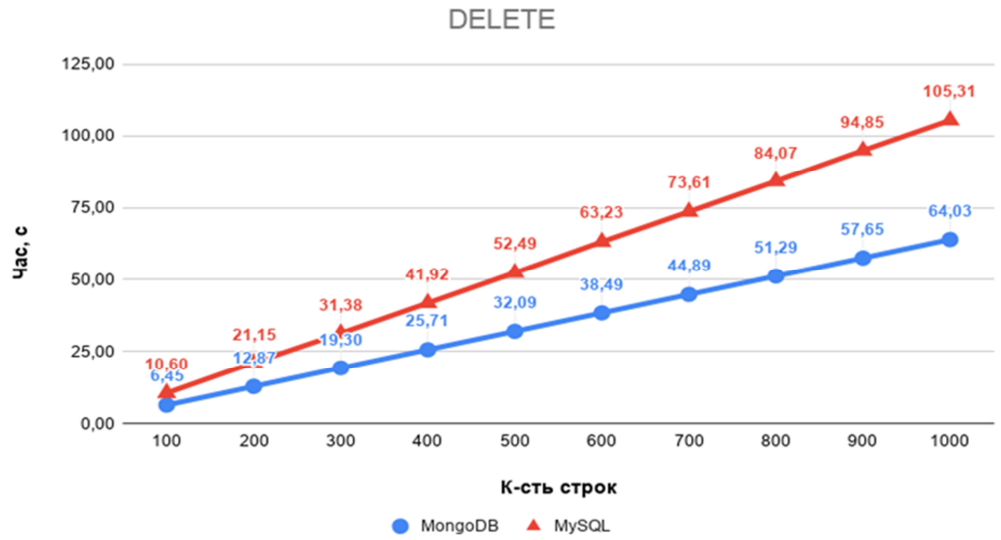


Fig. 4. Chart of DELETE operation time (not index) on the number of entries

In the operation of deleting records without using indexes, MongoDB obtained a time advantage.

**Conclusions**. MySQL is more suitable for CRM systems where the database must be rigorously structured without the dynamic appearance of tables or schemas. A prerequisite is to use indexes in tables. MongoDB is more suitable for CRM systems where the database is dynamic in terms of data change, flexible, performs a large number of searches and has no rigid structure. In this case, the indexes do not play a big role.

**REFERENCES:**


1. N. Leavitt, "Will NoSQL databases live up to their promise?" [Text], Journal Computer, - IEEE Computer Society Press Los Alamitos, CA, USA, vol. 43, no. 2, pp. 12 –14, feb. 2010.
2. D. Bartholomew, "SQL vs. NoSQL," [Text], Linux Journal, - Department of Computer Science, Maharaja Surajmal Institute of Technology, Janakpuri, N.delhi 110058, Indiano. 195, July 2010.
3. S. Sakr, A. Liu, D. Batista, and M. Alomari, "A survey of large scale data management approaches in cloud environments," [Text] Communications Surveys Tutorials - IEEE, vol. 13, no. 3, pp. 311–336, 2011.
4. R. Hecht and S. Jablonski, "NoSQL evaluation: A use case oriented survey" [Text], in Cloud and Service Computing (CSC), - 2011 International Conference on, dec. 2011, pp. 336 –341.
5. A. Boicea, F. Radulescu, and L. I. Agapin, "MongoDB vs Oracle – database comparison" [Text], in Emerging Intelligent Data and Web Technologies (EIDWT), - 2012 Third International Conference on, sept. 2012, pp. 330 –335.
6. B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb" [Text], in Proceedings of the 1st ACM symposium on Cloud computing, - ser. SoCC '10. ACM, 2010, pp. 143–154.



УДК 004.056.5: 004.414.22


## ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ МЕТОДІВ ВИЯВЛЕННЯ ШКІДЛИВИХ ПРОГРАМНИХ ЗАСОБІВ


М.Д. Даценко, С.В. Машурка
(Україна, Дніпро, Національний ТУ «Дніпровська політехніка»)


**Постановка проблеми.** Щороку в світі створюється величезна кількість комп'ютерних програм. Разом з цим зростає кількість комп'ютерних вірусів. Згідно зі звітом McAfee Labs [1] за перший квартал 2019 року в світі з'явилося понад 65 млн одиниць шкідливого програмного забезпечення (далі ШПЗ). Дані приведені на Рис. 1. Це на 18% більше, ніж за останній квартал 2018 року. При цьому загальна кількість ШПЗ практично досягло 1 млрд. Однією з причин цього явища є можливість автоматизованої розробки шкідливих програм. Про це свідчить і дані Data Breach Investigation Report за 2016 рік [2]. У звіті сказано, що більше 99% шкідливих програм існують у незмінному вигляді протягом 58 секунд і менше. При цьому більшість програм виявляються лише одного разу.