Oganesyan Artem

Member of the itSMF association

Russia, Moscow

Theme: ANALYSIS AND COMPARISON OF CRUD OPERATIONS PERFORMANCE OF RELATIONAL AND NOSQL DATABASES

Annotation: Despite the fact that NoSQL systems have existed for quite a long time, today there are relatively few studies on the topic of comparing their performance with relational systems. The available works often do not allow us to get a complete picture, because either they describe experiments of a narrow focus (for example, comparing time spent only on data insertion operations), or they have specialized and rarely used DBMS as research objects.

As part of this work, it is proposed to consider the fairly well-known PostgreSQL and MongoDB.

Keywords: DBMS, PostgreSQL, MongoDB, efficiency

To date, it is safe to say that the process of informatization concerns all spheres of human activity, which means that certain information repositories are used almost everywhere.

A relational database is a set of interconnected tables, each of which contains information about objects of a certain type. Each row of the table contains data about one object (for example, a car, a computer, a client), and the columns of the table contain various characteristics of these objects - attributes (for example, engine number, processor brand, phone numbers of companies or customers).

As part of this work, the "online store-warehouse" structure is organized for the PostgreSQL DBMS (figure 1).

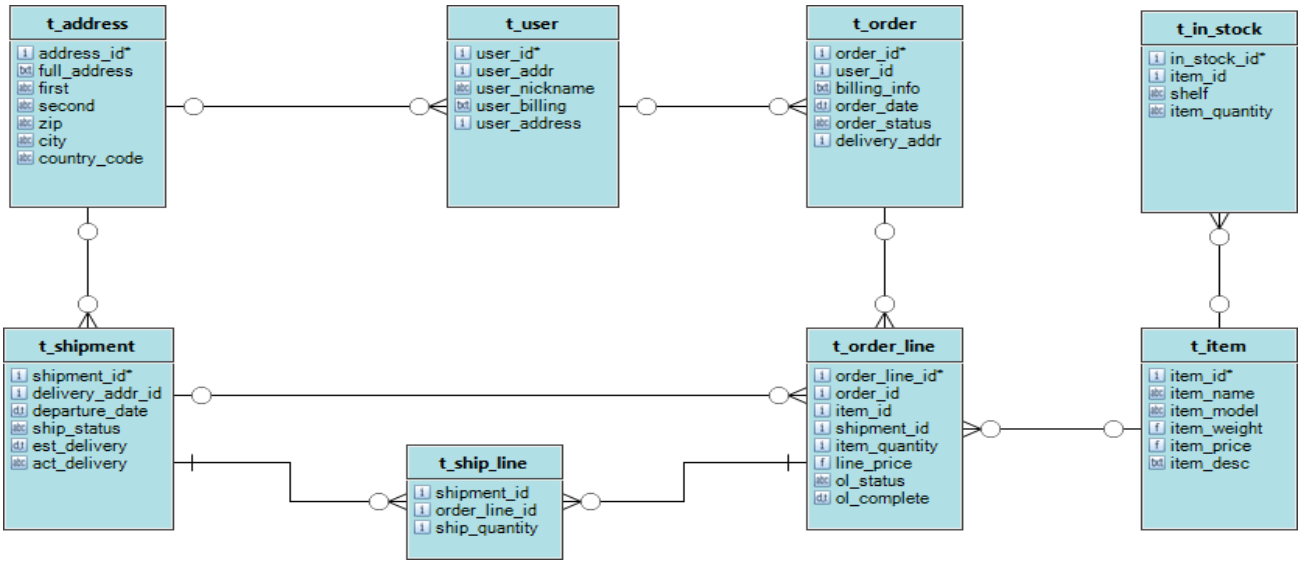


Figure 1– ER database diagram for PostgreSQL

As for MongoDB, it is a non-relational database, which means it is impossible to build arbitrary queries based on the available data. This problem is solved, as a rule, in two ways. The first of them consists in designing collections in the manner of tables from relational databases. The connection itself is carried out within the framework of the application. The second method is related to data denormalization. By placing, for example, the t_address collection inside the t_user collection (while leaving a separate copy of the t_address table), you can provide the possibility of pre-organizing join requests for these entities. This approach, however, is associated with very serious difficulties in ensuring data consistency, because changes that have occurred with a specific record in one collection must occur in all copies. Thus, one should be extremely careful when implementing "pre-join" and take into account the difficulties associated with it when analyzing the experiments described in this paper.

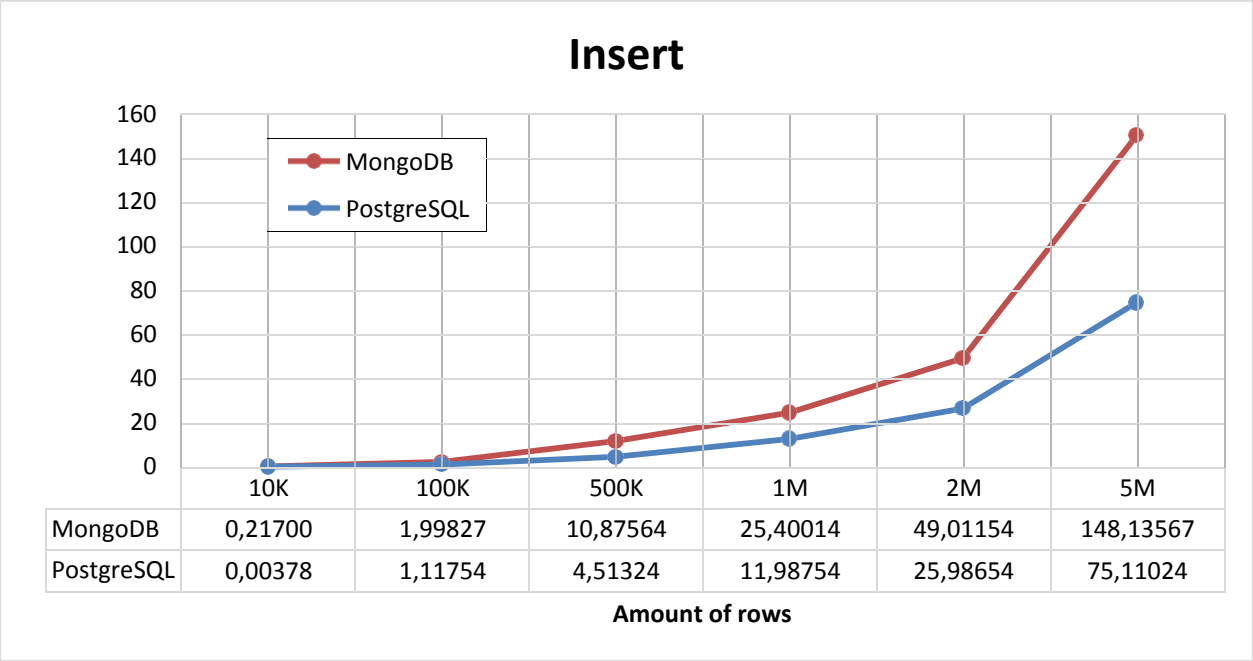
A personal computer with the following characteristics was used as a workstation:

- Operating system: Windows 10;
- processor: Intel Core i7 2.6 GHz;

• RAM: 8 GB.

PostgreSQL version is 9.6.1, MongoDB version is 3.4.2. An Internet service was used to generate data [9]. Each experiment was conducted for 10000; 100,000; 500,000; 1 000,000; 2,000,000 and 5,000,000 records with calculation of the average execution time for thirty attempts. Calculations and construction of histograms were carried out in the Microsoft Excel software product. To measure the execution time in PostgreSQL, the /timing directive was used, in MongoDB, methods of profiling the operation log were used, using the explain() method where possible, as well as placing timestamps with subsequent calculation of the difference between their values.

It is worth noting that the MongoDB internal query analyzer has an accuracy of 1 ms, so when conducting experiments with a small amount of data, it will not be possible to get accurate information about the query execution time.



	10K	100K	500K	1M	2M	5M
MongoDB	0,21700	1,99827	10,87564	25,40014	49,01154	148,13567
PostgreSQL	0,00378	1,11754	4,51324	11,98754	25,98654	75,11024

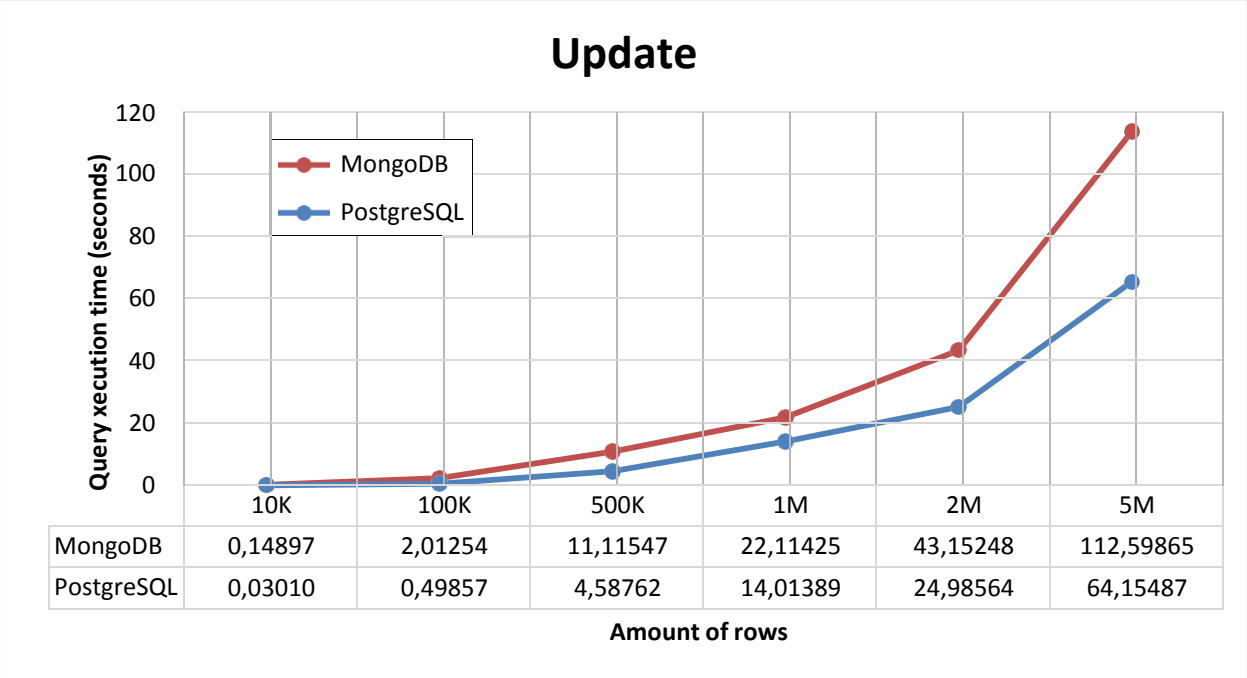
Figure 2– Comparison of insertion operation execution time

In the experiment on inserting records, the t_item entity was used, storing rows of the form:

Table 1 – Example of a row from the t_item table.

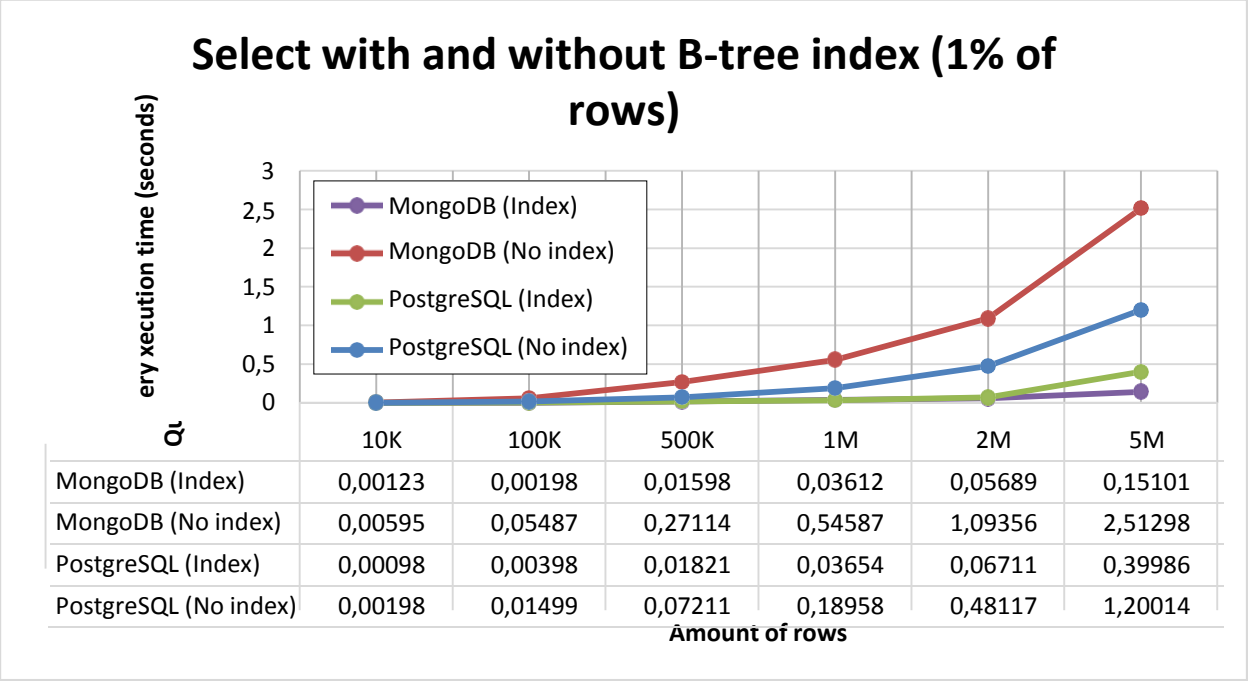
item_id (integer)	item_name varchar(30)	item_model varchar(30)	item_weight float	item_price float	item_desc text
1	vitae consectetuer	adipiscing	51,886	9869,215	ante ipsum primis in faucibus

The data update experiment was carried out within the t_address table, the zip numeric field (zip code) was changed[1,3]



	10K	100K	500K	1M	2M	5M
MongoDB	0,14897	2,01254	11,11547	22,11425	43,15248	112,59865
PostgreSQL	0,03010	0,49857	4,58762	14,01389	24,98564	64,15487

Figure 3– Comparison of the execution time of update operations

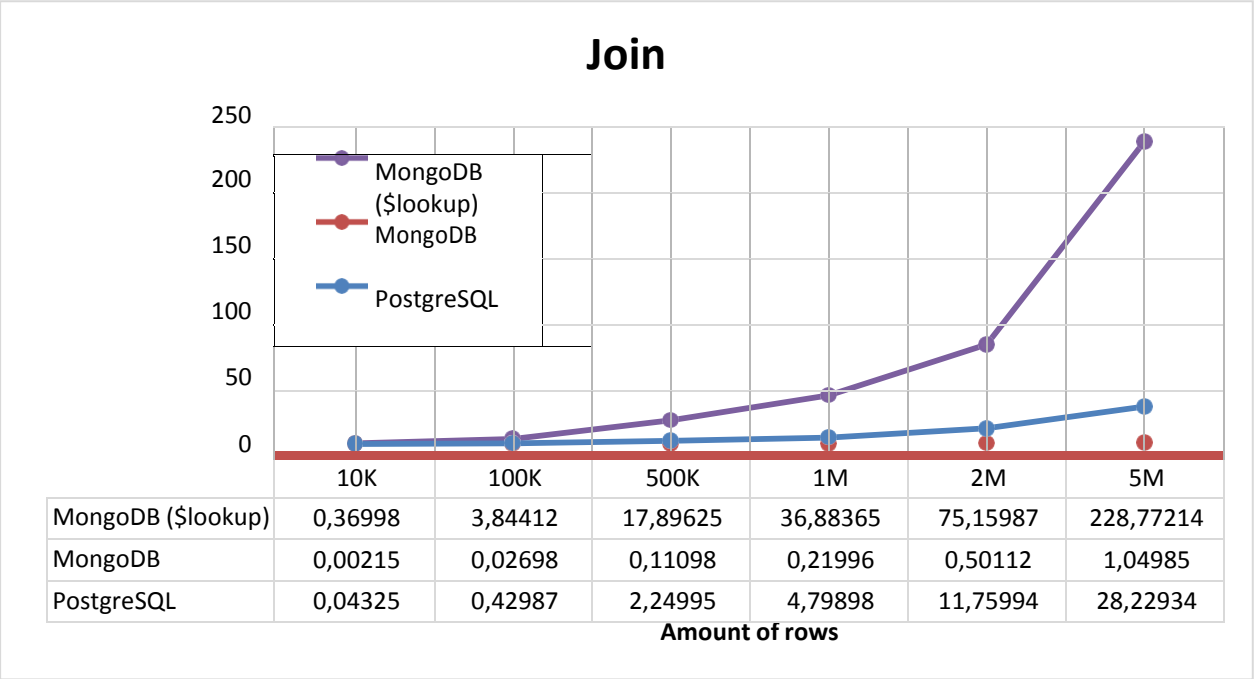


	10K	100K	500K	1M	2M	5M
MongoDB (Index)	0,00123	0,00198	0,01598	0,03612	0,05689	0,15101
MongoDB (No index)	0,00595	0,05487	0,27114	0,54587	1,09356	2,51298
PostgreSQL (Index)	0,00098	0,00398	0,01821	0,03654	0,06711	0,39986
PostgreSQL (No index)	0,00198	0,01499	0,07211	0,18958	0,48117	1,20014

Figure 4– Comparison of the execution time of sampling operations with an index and without using an index

For experiments with the sampling operation, the following scenario was used: the execution time of operations with the sampling condition for the same records with the presence of an index based on the binary search tree and without it was measured. The condition for the item_price field of the float type was used. One percent of the records satisfied the search expression[2]

The following are the results of experiments with the selection operation with table attachment (t_user and t_address by the user_address_id and address_id fields, respectively).



	10K	100K	500K	1M	2M	5M
MongoDB ($lookup)	0,36998	3,84412	17,89625	36,88365	75,15987	228,77214
MongoDB	0,00215	0,02698	0,11098	0,21996	0,50112	1,04985
PostgreSQL	0,04325	0,42987	2,24995	4,79898	11,75994	28,22934

Figure 5– Comparison of the execution time of data attachment operations

As already mentioned above, today under the word

NoSQL is understood not by those DBMSs that are managed using a language that does not belong to the SQL standard, but rather by those that are not relational. It is all the more surprising to see that in the MongoDB version -3.2 – the developer company has provided the opportunity to organize connections by common fields of the table [1]. The connection command looks like this:

```
db.t_user.aggregate([{$lookup:{

from: "t_address",

localField: "user_address_id",
foreignField: "address_id",

as: "find_address"}}]),
```

where **from** is the name of the external collection, **localField** is the attachment field from the collection in question, **foreignField** is the attachment field from the external collection, **as – alias** is the name for the resulting attachment of records.

At the moment, it is possible to join only one field and only in the left outer join format. Thus, based on the existing limitations and time characteristics of the execution of

this query, when organizing table joins in the selected NoSQL solution, other methods should be used.

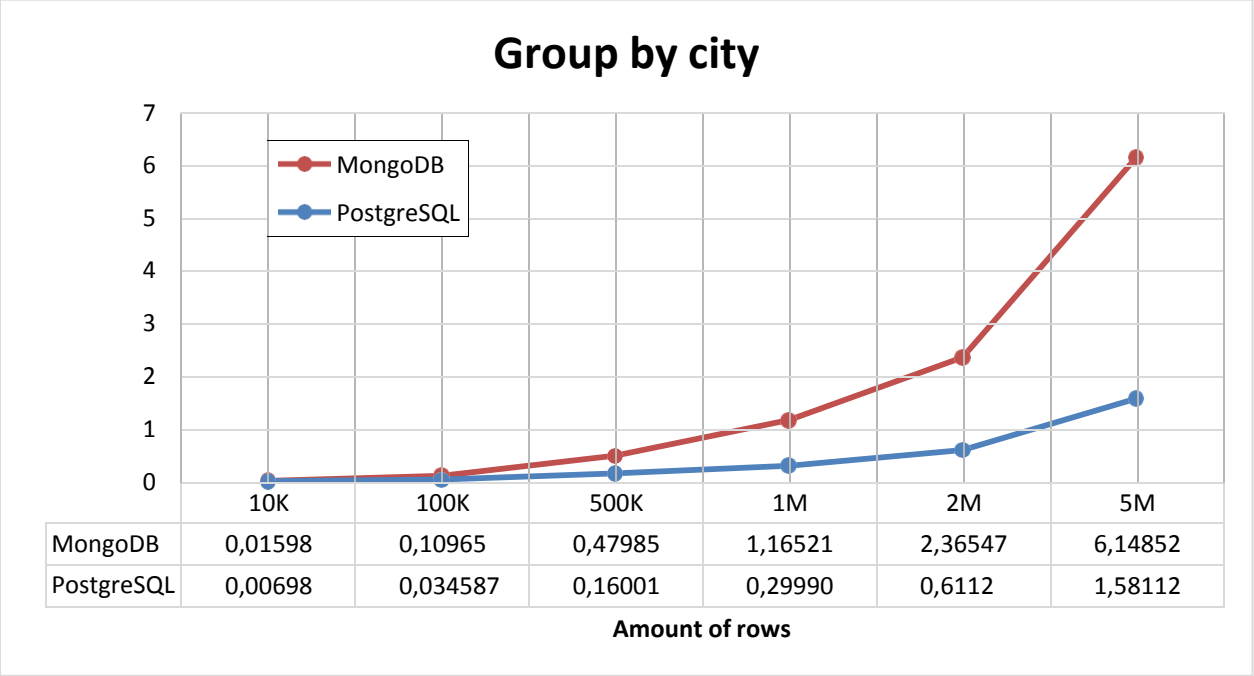


Figure 6– Comparison of the execution time of data grouping operations

This experiment is based on a query calculating how many resource users represent a particular city (based on the t_address table; sort the result in descending order of the aggregating function)[4]

The last experiment in this chapter is related to finding the maximum value of the item_price field in the t_item table.

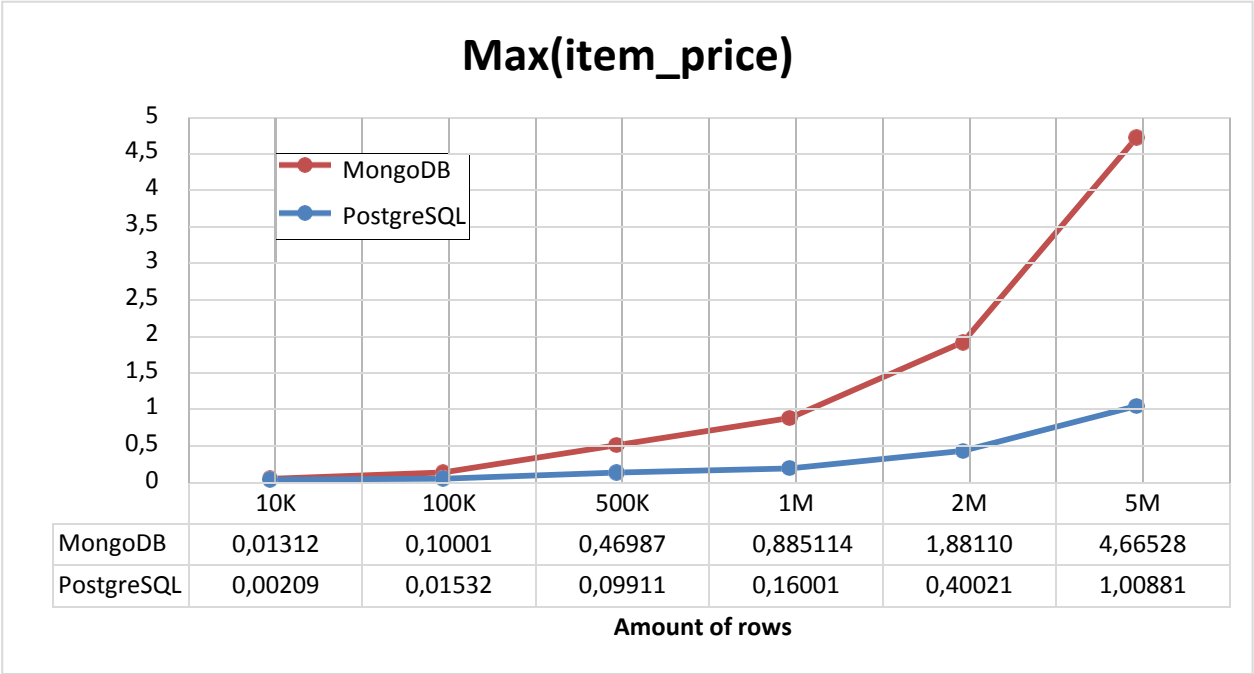


Figure 7–Comparison of the execution time of the maximum value search operations

Thus, experiments show the advantage of PostgreSQL in all tasks except indexed search. As for the joining operation, the decision on data denormalization for NoSQL DBMS directly depends on the specific task, taking into account the costs of maintaining consistency and storing redundant information.

List:

1. Release Notes for MongoDB 3.2 – MongoDB Manual 3.2 [Электронный ресурс]. URL: https://docs.mongodb.org/manual/release-notes/3.2/#aggregation- framework-enhancements

2. Rick Cattell. Scalable SQL and NoSQL data stores. ACM SIGMOD Record,Volume 39 Issue 4, December 2010. – NY: ACM New York. – p.12-27

3. Parker Z., Poe S., Vrbsky S. Comparing NoSQL MongoDB to an SQL DB, Proceedings of the 51st ACM Southeast Conference. – NY: ACM New York,2013. – 6 p.

4. Daniel J. Abadi, Peter A. Boncz, Stavros Harizopoulos. Column-oriented database systems. Proceedings of the VLDB Endowment, Volume 2 Issue 2, August 2009 (pages 1664-1665).