2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus) | 978-1-6654-0476-1/20/$31.00 ©2021 IEEE | DOI: 10.1109/ElConRus51938.2021.9396363

Research and Development of the Method of Investigating the Possibility of Transformation Relational Databases to NoSQL

Tatyana S. Ivanova
National Research University of Electronic Technology
Moscow, Russia
tatyana.skorbilina@yandex.ru

Evgeniy A. Ivanov
National Research University of Electronic Technology
Moscow, Russia
jonny-evildream@yandex.ru

***Abstract*—Relation database has been a universal solution for any system, allowing manipulating various types of data and creating complex relationship between them for over forty years. RDBMS are not adapted to work with unstructured and semi-structured data and the increase the number of such data has led to the emergence of NoSQL (Not only SQL) database models. The article presents the method of investigating the possibility of transformation relational databases to a configuration from a relational and document-oriented database, based on an assessment of the speed of query execution. The main problems of using relational databases and prerequisites for the transition to NoSQL databases are highlighted.**

Keywords—relation database; NoSQL; DBMS; linear programming

I. Introduction

Currently, almost any information system needs to store, process and analyze a large amount of data. The best tools to carry out these operations are databases. The most common is the relational database model, in which data is presented in the form of tables and relationships between them. This organization provides flexibility in use and design, which has facilitated the spread of this model. Until recently, the relational model was a universal solution for any system, allowing to manipulate various types of data and create complex relationships between them.

But there are specific tasks for which the representation of data in a relational model is suboptimal, complex, or impractical. For example, you need to store and use a large list of products in an online store. The product has a name and number, which in the relational model will be represented as one table of two columns. The relational model can be used to solve this problem, but the system runtime will be greatly increased due to the complexity of the relational model, which is not used for the system. Therefore, other database models were developed, which are called NoSQL (not only SQL).[1] Table 1 lists the main database models and examples of the most popular database of this type.

TABLE 1. Database Models and Examples

Model	Examples
Relation	Oracle, MySQL
Key-value store	Redis, React
Wide Column store	Cassandra
Document store	MongoDB
Graph	Neo4J

Key-value store represents data as an associative array. Such a system does not require the construction of a data schema and strong data typing, and there are no relationships between values. This simplicity enables scalability and performance. Key-value store is the simplest of all the models under consideration and therefore very rarely a relational database can be transformed into this model without data loss.

In wide column databases data belonging to one column is stored side by side and it follows that adding a new column is cheap and done line by line. In terms of structure, wide column databases occupy an intermediate position between relational DBMS and Key-value storage.

A document-oriented database is a system for storing hierarchical structures of documents that can be represented as a tree. The tree structure starts at the root node and can have multiple internal and leaf nodes. The data does not have strong relationships with each other, but this model allows creating complex structures. The document model is a kind of analogue of a hash, in which there is a unique identifier field, and the value can be arbitrary data type, including other hashes.

For a graph model, data is represented as graph nodes, and the links between them are represented as edges. It allows creating strong relationships between objects without restricting their types or creating a rigid structure.

Each of the models has its own advantages and disadvantages, and it is impossible to say unequivocally which one is the most effective and powerful. It is important to understand that when choosing, you should always rely on a specific task, priority criteria for the system and parameters that are not so essential. Each model is suitable for its field of tasks. But some problems can be solved using different models and it is not always obvious which solution will be the most optimal. And a method is needed through which it will be possible to determine which database is optimal for a given task.

Since relational databases are currently the most widespread, the greatest need exists to transform a relational database into a non-relational one. This transformation can improve the speed and efficiency of the system in view of the

simplest data structure in NoSQL databases. The most compatible with the relational model is the document-oriented model, in which data is presented as a hierarchy of documents. This way of presenting data is less flexible than relational representation in the form of tables and the relationship between them, but it has a significantly higher query processing speed[2].

However, not every relational structure can be converted to a document-oriented model without duplication of data, which can lead to a decrease in operational efficiency[3]. Large structures can contain both complex relationships between tables, which would be more expedient to leave in a relational database, and large reference books or tables with a simple structure, but a large amount of data that will be easier to process in a document-oriented model. Therefore, to improve performance, it makes sense to transform into a document-oriented database not the entire database from the relational model, but only some of its tables. And it becomes necessary to determine the transformation of which blocks of information will help achieve the greatest efficiency.

To solve this problem, a method was developed that allows transforming a relational database into a configuration from a relational and document-oriented database[4], taking into account the speed of query execution.

II. Main Part

Before moving on to migrating data from a relational database to a non-relational database, you need to assess whether such a solution will be effective. Following from the definition of efficiency as the ratio between the achieved result and the resources used, a criterion will be chosen that will reflect the relationship between the achieved gain in any parameter and the resources expended for this achievement.

Thus, initially it is necessary to determine the gain by which parameter in the context of working with database management systems can be considered the most significant. Among such parameters, the most significant ones can be singled out: speed, reliability and minimization of resources spent on work. The following are explanations for each of these criteria.

This method evaluates the efficiency based on the speed of the DBMS. In relation to working with databases, the system performance should be considered based on the parameters latency and throughput.

Latency, also known as response time or access time, is a measure of how long it takes for a database to respond to a single request. This parameter makes it clear how long a particular user will wait for a response to his request, regardless of the load on the system.

Throughput is a measure of the number of queries that a database can respond to in a unit of time. Throughput indicates how many concurrent requests, and therefore users, can be served by the database management system.

As a result, the criterion for evaluating the effectiveness of the database can be understood as the least execution time. To begin with, we present a description of this assessment of temporary effectiveness. In general, the estimate of time will be the estimate of the execution time of database queries. Thus, data migration will be performed efficiently if inequality 1 is true.

$$T_{new} < T_{old} \quad (1)$$

Where T_{new} is the total time for executing queries in the new database, T_{old} is the total time for executing queries in the original database.

The following is formula 2 showing how time T is calculated. In the above expression, n is the number of query types, q_i is the frequency of execution of a query number i, t_i is the average execution time of query number i.

$$T = \sum_{i=0}^{n} (q_i \cdot t_i) \quad (2)$$

When substituting the expression 2 into the expression 1, the inequality is formed:

$$\sum_{i=0}^{n} (q_{new\,i} \cdot t_{new\,i}) < \sum_{i=0}^{n} (q_{old\,i} \cdot t_{old\,i}) \quad (3)$$

It should be noted that modern relational databases can use methods of parallel execution of queries, and thus the total query execution time will be less than in the presented formula. But, since modern NoSQL databases also support parallel query execution, after transforming tables into a non-relational court, queries can also be executed simultaneously, and therefore, in this work, the possibility of parallel query execution will not be taken into account.

Thus, the problem is reduced to finding the minimum of the function T_{new}. From this it follows that it makes sense to try to reduce this problem to a problem of the linear programming type[5].

The predictive method uses linear programming techniques to minimize the function of consumption of time resources by requests. To construct the basic equation and boundary conditions of the linear programming problem, it is necessary to introduce the conventions for the variables used:

Q – requests to SQL database;

Q* – requests to NoSQL database;

n – number of types of requests to NoSQL database;

m – maximum number of types of requests to NoSQL database;

X – the required number of requests for migration;

R – time resources consumed by all requests;

r – time resources consumed by one request;

X_{min} – minimum number of requests allowed.

Next, you should introduce the basic relationships between the described parameters that will be used in the method. The first one should indicate that the amount of consumed resources is greater than or equal to the sum of the consumed resources by each of the requests:

$$r_1 + r_2 + \ldots + r_n \leq R \quad (4)$$

The following is the formula by which resources are calculated:

$$r_i = c_i \cdot x_i \quad (5)$$

In expression 5, c_i denotes the amount of resources for a request of type x_i.

The next step is to classify queries that access a relational database into two types: dependent and independent. In the chosen terminology, independent queries are those that, in the From and Where fields, refer to only one table from the database. Such queries are most easily converted to a system in which the selected table was transformed into a non-relational database, since in this case there is no need to investigate relationships between tables and additional software tools to perform relationships between several tables.

On the other hand, there are dependent queries that access columns from different tables in the From and Where fields and work with disparate data using relational relationships between tables. This type of query requires an initial analysis of the relationships between tables.

The following is an inequality for a relational database before including NoSQL tables. Inequality 6 describes resource indicators as the product of the request time and the coefficient:

$$c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_i \cdot x_i + \ldots + c_n \cdot x_n \leq R \quad (6)$$

The above inequality shows all types of queries that access a given relational database. Further, with this inequality, the following transformations should be performed sequentially for each query:

1. Analyze the request: is it dependent or independent.

2. If the query is independent, transform the only table to which it refers to a non-relational database and determine the coefficient Ci*.

3. Make up inequality 7 by substituting the coefficient calculated in paragraph 2 into inequality 6:

$$c^*_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_u \cdot x_i + \ldots + c_n \cdot x_n \leq R \quad (7)$$

4. If the query is dependent, determine the list of tables it accesses, and migrate all of them to a non-relational database, and then calculate new coefficients of the type for each of them Ci*.

5. Make inequality 8 by substituting all the coefficients calculated in paragraph 4 into inequality 6:

$$c_1 \cdot x_1 + c^*_2 \cdot x_2 + \ldots + c^*_u \cdot x_i + \ldots + c_n \cdot x_n \leq R \quad (8)$$

6. Repeat steps 1 - 5 for each request.

7. After performing the transformation for each query, combine inequalities 7 and 8 obtained in points 3 and 5 and draw up a general inequality, the fulfillment of which guarantees the efficiency of the transformation:

$$c^*_1 \cdot x_1 + c^*_2 \cdot x_2 + \ldots + c^*_u \cdot x_i + \ldots + c^*_n \cdot x_n \leq R \quad (9)$$

After drawing up inequality 6, on its basis, the objective function of the linear programming problem is constructed, which must be minimized to find the optimal transformation method.

$$F(x) = X_1 \cdot C^*_1 + X_2 \cdot C^*_2 + \ldots + X_i \cdot C^*_i + \ldots + X_k \cdot C^*_k \rightarrow min \quad (10)$$

In two additions to the objective function, it is necessary to introduce a system of boundary conditions:

$$\forall i < k \; x_i \geq x_{min} \quad (11)$$

Where x_i is the number of requests and x_{min} is the minimum number of requests.

The solution to this problem will allow you to determine which tables should be transformed into a document-based database, and which should be left in the relational one. This technique solves the problem of increasing the efficiency of data storage and processing by using NoSQL databases in conjunction with relational.

The developed algorithm for converting a relational database to a non-relational one is relevant, since it will automate the assessment of the efficiency of data migration, select the source database and migrate only part of the data, combining the advantages of relational and NoSQL models.

The developed algorithm makes it possible to increase the efficiency of data storage and processing, which will make it possible to create storages of greater volume without losing speed and reliability.

III. Conclusion

The presented work is devoted to the problem of converting relational databases to non-relational in order to improve the efficiency of DBMS. This problem is justified by the increase in the amount of Big Data, for which relational database models are not optimal.

The presented technique allows not only transforming a relational database into a non-relational one, but also evaluating the effectiveness of this transformation[6]. The greatest efficiency is achieved when the following conditions or some of them are met:

1. A large number of independent queries accessing only one table.

2. Large amount of stored data.

3. Static data schema of the source database, in which no changes are planned

The advantage of this algorithm over other existing ones is the ability to move only part of the tables from the relational database to the non-relational model. This approach allows you to simultaneously take advantage of different storage models based on data processing speed.

REFERENCES

[1] S. Kashyap, S. Zamwar, T. Bhavsar, S. Singh, "Benchmarking and analysis of nosql technologies", International Journal of Emerging Technologies, Adv Eng 3, 2013, pp. 422-426.

[2] M.A. Likht, A.A. Revo, A.M. Andrianov, T.S. Skorbilina, "Problems of designing the DBMS interface for the development environment of accounting systems PK VIKTA", Actual problems of modern science, 2017, vol. 3, pp. 43-47

[3] L. Wang, J. Yu, T. Song, "Research on Big Data processing and analysis architecture based on MongoDB", Acta Technica CSAV (Ceskoslovensk Akademie Ved), 2016, vol.4, pp 185-195.

[4] A. Gadkari, V.B. Nikam, B.B. Meshram, "Implementing Joins over HBase on Cloud Platform", Proceedings of IEEE International Conference on Computer and Information Technology (CIT), 2014, pp. 547-554.

[5] A.F. Chuvenkov, M.M. Belokonsky, "Conditionality of solving computational problems of mathematical models in linear programming", Russia and the EU: ways of development and prospects. Materials of the International Scientific and Practical Conference, 2016, pp. 862-867.

[6] T.S. Ivanova, "Development of a software module for testing databases", All-Russian scientific and practical conference "Actual problems of informatization in science and education - 2018", 2018, pp. 53-57