UDC 004.02
IRSTI 20.53.15

INTERNAL WORKLOAD OF NOSQL RELATIONAL DATABASE

Mirzakhmet Syzdykov
al-Farabi Kazakh National University, Almaty, Kazakhstan
mspmail598@gmail.com
ORCID ID: https://orcid.org/0000-0002-8086-775X

Abstract. In this article we present the scheme of internal workload of typical relational database supporting NoSQL human interaction protocol, we state that the production consumption of query complexity cannot be avoided and modern techniques like explain plan play vital role in bringing the functionality of database server to the end-user with support of open data storage like CSV file format which, in turn, is textual – thus, the question of open storage is discussed in this work within the unlimited support of table sheet formats like CSV, as to the modern Big Data trends we propose the solution used supposedly empirically by popular SQL database systems – this is a plan to run the incoming query and transform the result to the format which can be adopted for final output; we also develop the end programmable product using C# programming language and .NET framework which are the modern programming environments for proper development of platform independent software – the proof of important theorems of relational database query complexity processing and optimization is also given.

Keywords: NoSQL, data processing, relational database, open storage.

Introduction

In the modern era the structured query language (SQL) became a standard for purpose of querying data to obtain the results as per giving the limits and if-statements to get the more precise result of seeding values. The NoSQL (not only SQL or no SQL) trend today is emerging as an alternative to the general approach and gives the possibility of probable out-performance for the previous generations of SQL database management systems (DBMS).

The work of addressing the OLAP feature in NoSQL databases [1] is solved by our definition of structures along which we achieve the universality of using relational data over production system, the definition of which is give later in this work.

From the other side [2] the NoSQL is giving more broad perspectives for Big Data, according to the authors of referenced work the modern data storages approximate to petabytes. We give the solution according to the optimizing explain plan for the relational database tuple to make the right decision of choosing the data to be processed on server or specially developed RISC-machine.

The comparison of SQL and NoSQL databases is presented in [3]. As previously described we prove by definition of tuple that SQL and NoSQL databases working time observes to be equivalent to the theorem proof.

Java Script object notation (JSON) is discussed in [4]. For this time MongoDB or other systems adopt this format for formal clauses which are equivalent to the SQL queries, the difference is in how they are processed by the “black box” or database engine for query processing and assembling of finite data.

In [5] the way of optimization is presented with respect to the document and relational model. Our approach based upon the law proof which, in turn, is an outcome defined by the main theorem and, thus, the optimization is only a path through tuple elements giving better or best throughput.

The comparison of document-based MongoDB and relational MySQL is given in [6] – it turns out that in some cases the non-relational approach is better for Big Data, however, we define that relational and non-relational databases adhere to the same theorem of the complexity equivalence of this classes of data processors.

For the purpose of our proof of concept we have developed the hybrid database system based on the relational model with the support of non-relational features like NoSQL and typical join operations [7]. In our model the open textual storage is used like comma-separated values (CSV).

SQL DBMS review

SQL database systems are widely used in the modern age, especially for the web applications and cloud processing of streams of data. They use typical relational database model for operational purposes and to define the final human computer interaction with the end user. The SQL or query by example (QBE) is used as a query language in order to get the results which are seeded by the predicates of the formed query string.

The bottleneck of these systems is that they are highly inefficient and require special hardware resources to process the query and store indexes and data – thus, the question of optimization arises and is solved by introducing explain plan or other alternatives. Most of them are open-source and gives us the possibility to fork the new database engine with the support of extended or NoSQL features.

Thus, the SQL database systems cover approximately more than fifty percents, or half, of the Internet segment of the on-line services and are limited to the gigabytes of data to be stored and queried on budget and on time.

NoSQL DBMS review

Modern trends like NoSQL arise to be an alternative to the industry standard like SQL database. According to our research, most of them adapt JSON notation or any other document-oriented format like XML, for example, which, in turn, is accomplished by the XSL assertion.

Thus, the NoSQL databases which are more efficient than its predecessors are available today in the most common document formats – we consider this as a good addendum, meanwhile, the other side is that the relational model which is simple for understanding cannot be achieved and the broad perspectives of data modeling are omitted.

In the other hand, the relational model is very strict which gives the priority to the non-relational databases to convert data to documents and operate on the higher level rather than typical database schemes and constructed data models.

We define the paradigm between relational database and object-oriented programming (OOP) to be equivalent according to the modern standards of workload of the today developed systems by the use of persistence of objects in the relational model as a single separate table in the database system.

Main theorem and its proof. We define the theorem for the relational data and the data which are document-oriented which are subject of the joining different set of entities in one pre-defined tuple.

By this tuple we define the following expression:

$$<T, Q, R>, (1)$$

where *T* is a set of tables or document formats in non-relational database and *Q* is a set of queries to the database engine incoming from end user, and *R* is a set of optimization rules – these rules can be met as explain plan hints in Oracle database.

Thus, by defining the common model for both SQL and NoSQL databases which can be also divided as relational or non-relational, we get the main theorem which states that the production rule for internal joins adheres to the multiplication of all the tables for the given complexity of each query in *Q* cannot be avoided and is supposed to be processed by the cursor:

$$O(Q) = |T_1| * |T_2| * \ldots * |T_n|. \qquad (2)$$

While the cursor is to be defined as a state in the production rule (2):

$$C(Q) = (t_1, t_2, \ldots, t_n): t_i = 1..|T_i|, \qquad (3)$$

where t_i is a current step in the cursor for production consumption within the defined complexity and |...| is a number of rows in the table or document to be processed by the query.

The proof of the theorem for any type of database, which can be both relational or document-based, follows from the fact that the set *R* of optimization rules is limited to the set of complexity of cursor tuple in (2).

By this proof we get that the join factor of several tables cannot be uniformly optimized in the worst case of evaluation of the truth of the query predicate in *Q* by simple paths of explain plan, meanwhile, the whole structure is to be preserved, so that any of the acceptable data are covered by each iteration of querying process.

As the cursor is common for both models, we can state that the optimization rules in set *R* are to be constructed according to non-index approach – as we use open data storage in pure textual format. These optimization rules reduce the production explosive complexity to the limited set of operations for which the tuple evaluates to true and the query predicates hold true at the same time. This technique is effectively used in the modern DBMS like Oracle, where user has the possibility to tune the execution plan of the query according to hints which appear in comments for the query string; the explain plan feature is also present in Oracle with respect to the user evaluation and tuning.

Proof of concept software description

For our purpose to prove the product-rule (2) explosion in (1) while evaluating the query, we have developed the software available from GitHub [7].

In this manner we define some SQL-like clauses like logical operators as well as joining operator.

While evaluating the query expression, the optimization rule *r* in *R* can be applied to the cursor and the order of acceptance or reject of current cursor can be determined.

Conclusion

Thus, we have given the proof of product explosion for relational model or document-based model where relational operator cannot be avoided in general, if even we would consider the NoSQL database system.

We have also developed the software for evaluation purposes which gives the outcome as the relational model leads to the product explosion for complexity of processing the query.

We also give the solution for Big Data by applying the optimization rules in definition of tuple. By this fact, we define the set of separated rules which are to minimize the set of elements in cursor and, thus, leading to the more effective process of query evaluation within the product model for both relational or non-relational databases within the entity join concept. This concept leads us to the fact that relational model cannot be avoided as per the complexity of cursor manipulation over the whole set of data stored in table or document storage.

Acknowledgements

The author expresses gratitude to the Free Software Foundation (FSF) for providing with all the necessary software in order to prepare this work for evaluation.

We have also experienced a good work-around using the tools and cloud services described in this work.

References

[1] Banerjee, Shreya. Bhaskar, Sourabh. Sarkar, Anirban. Narayan, C. Debnath. A Formal OLAP Algebra for NoSQL based Data Warehouses. Annals of Emerging Technologies in Computing. 2021. 5. 154-161. 10.33166/AETiC.2021.05.019.

[2] Erraissi, Allae. Hadoop Storage Big Data layer: meta-modeling of key concepts and features. International Journal of Advanced Trends in Computer Science and Engineering. 2019. 8. 646-653. 10.30534/ijatcse/2019/49832019.

[3] Chang, Ming-Li. Chua, Hui Na. SQL and NoSQL Database Comparison. Proceedings of the 2018 Future of Information and Communication Conference (FICC). 2019. 1. 10.1007/978-3-030-03402-3_20.

[4] Lv, Teng. Yan, Ping. He, Weimin. Wang, Tao. On Approximate Querying Large-Scale JSON Data. Journal of Physics: Conference Series. 2020. 1575. 012066. 10.1088/1742-6596/1575/1/012066.

[5] Ha, Muon. Shichkina, Yulia. Translating a Distributed Relational Database to a Document Database. Data Science and Engineering. 2022. 1-20. 10.1007/s41019-022-00181-9.

[6] Matallah, Houcine. Belalem, Ghalem. Bouamrane, K. Comparative Study Between the MySQL Relational Database and the MongoDB NoSQL Database. International Journal of Software Science and Computational Intelligence. 2021. 13. 38-63. 10.4018/IJSSCI.2021070104.

[7] Syzdykov, Mirzakhmet. CSVdb: NoSQL CSV Database System. https://github.com/mirzakhmets/CSVdb. (accessed May 7th, 2022).

Internal workload of NOSQL relational database
Mirzakhmet Syzdykov
Әл-Фараби атындағы Қазақ ұлттық университеті, Алматы, Қазақстан
mspmail598@gmail.com
ORCID ID: https://orcid.org/0000-0002-8086-775X

Аңдатпа. Бұл мақалада біз NoSQL пайдаланушысының өзара әрекеттесу протоколын қолдайтын типтік реляциялық Дерекқордың ішкі жүктеме схемасын ұсынамыз, біз сұраныстардың күрделілігін өндірістік тұтынудан аулақ бола алмайтынымызды және explain plan сияқты заманауи әдістер дерекқор серверінің функционалдығын ашық деректер қоймасын қолдайтын соңғы пайдаланушыға жеткізуде маңызды рөл атқаратынын мәлімдейміз. CSV файл пішімі сияқты, ол өз кезегінде мәтіндік болып табылады – осылайша, ашық репозиторий мәселесі CSV сияқты кесте форматтарын шексіз қолдау аясында осы жұмыста талқыланады, қазіргі заманғы үлкен деректер тенденцияларына келетін болсақ, біз SQL-дің танымал дерекқор жүйелері эмпирикалық түрде қолданатын шешімді ұсынамыз-бұл кіріс сұрауды іске қосу және нәтижені түпкілікті шешім қабылдау үшін қабылдануы мүмкін форматқа айналдыру жоспары шығару; сондай – ақ, біз C# бағдарламалау тілін және.NET framework көмегімен бағдарламаланатын өнімді жасаймыз, олар бағдарламалық жасақтама платформасынан тәуелсіз дұрыс әзірлеу үшін заманауи бағдарламалау ортасы болып табылады-сонымен қатар реляциялық мәліметтер базасына сұраныстардың күрделілігін өңдеу және оңтайландыру үшін маңызды теоремалардың дәлелі.

Түйін сөздер: NoSQL, деректерді өңдеу,реляциялық деректер базасы, ашық сақтау.

ВНУТРЕННЯЯ РАБОЧАЯ НАГРУЗКА РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ NOSQL

Мырзахмет Сыздыков
Казахский национальный университет имени аль-Фараби, Алматы, Казахстан
mspmail598@gmail.com
ORCID ID: https://orcid.org/0000-0002-8086-775X

Аннотация. В этой статье мы представляем схему внутренней рабочей нагрузки типичной реляционной базы данных, поддерживающей протокол взаимодействия с пользователем NoSQL, мы заявляем, что нельзя избежать производственного потребления сложности запросов, а современные методы, такие как explain plan, играют жизненно важную роль в доведении функциональности сервера баз данных до конечного пользователя с поддержкой открытого хранилища данных, такого как формат файла CSV, который, в свою очередь, является текстовым – таким образом, вопрос открытого хранилища обсуждается в этой работе в рамках неограниченной поддержки форматов таблиц, таких как CSV, что касается современных тенденций в области больших данных, мы предлагаем решение, предположительно эмпирически используемое популярными системами баз данных SQL – это план запуска входящего запроса и преобразования результата в формат, который может быть принят для окончательного вывода; мы также разрабатываем конечный программируемый продукт, используя язык программирования C# и .NET framework, которые являются современными средами программирования для правильной разработки независимого от платформы программного обеспечения – также приводится доказательство важных теорем обработки и оптимизации сложности запросов к реляционным базам данных.

Ключевые слова: NoSQL, обработка данных, реляционная база данных, открытое хранилище.

Сведения об авторе:
Анг.: Syzdykov Mirzakhmet - al-Farabi Kazakh National University, Almaty, Kazakhstan
Каз.: Сыздықов Мырзахмет- әл-Фараби атындағы Қазақ ұлттық университеті, Алматы, Қазақстан.
Рус.: Сыздыков Мырзахмет- Казахский национальный университет имени аль-Фараби, Алматы, Казахстан.