Data Utility Assessment while Optimizing the Structure and Minimizing the Volume of a Distributed Database Node

Mykhailo Dvoretskyi[1], Svitlana Dvoretska[2], Yuriy Nezdoliy[3] and Svitlana Borovlova[4]

[1-4] Petro Mohyla Black Sea National University, Mykolaiv, Ukraine
m.dvoretskiy@gmail.com[1], svetag603@gmail.com[2],
nezdoliy.yura@gmail.com[3], svetlya1952@gmail.com[4]

Abstract. While using universal accounting systems, appears a set of problems, among which can be mentioned at least central database overload and low reliability. But shift to the several specialized solutions leads to existence of the set of databases, some data in which needs to be synchronized. Another cause of database splitting can be company branches, removed from the central office geographically in case, when cloud approach cannot to the solution. It all leads to distributed database structure. The article highlights the difficulties, connected with distributed queries and distributed transactions while getting data from relational database, hosted on several remote nodes. It is suggested to minimize the number of such queries by placing valuable data for the node locally. To do this, firstly SQL-queries parser was created. It presents SQL-query as a hierarchical tree, in nodes of which can be subqueries, tables, table columns and tuples. Based on the parsing results, the queries to database (DB) were analyzed in the analytics of the application, user, location, list of used tables, attributes, tuples and other possible dimensions. It allows to determine valuable for the node data based on list of needed applications, node location and other available input parameters. The problem of new and modified data classifying and ways of its solving also is presented. The paper shows the results of testing of the research while the structure of the database of the automated retail store were designing.

Keywords: distributed transaction, database management system, the consolidated accounting, distributed database, distributed SQL-query, data replication, text parsing, parse tree, profiling, ANTLR, multidimensional analysis, classification, neural network, perceptron.

1 Introduction

While creating the database, the user desires to organize information by some characteristics. It allows obtaining needed information quickly, by giving a set of search criteria. The necessity of automation of different company accounting types (warehouse, accounting, staff, etc.) in some cases may cause usage of "universal" accounting systems. If we choose this approach, we need to be ready to face set of difficulties. It can

be mentioned central database overload, low reliability, system vulnerability and far from perfect mechanisms of accounting automation [1].

If the specialized accounting solutions are used, many of mentioned misfortunes can be solved. But this approach leads to existence of different database management system platforms in measure of one company. So, their data need to be synchronized. That is why the information systems development moves from the local databases usage to the side of distributed databases (DDB). The main tasks of distributed database management system (DBMS) are to provide access control to the data for many consumers as well as the integrity and consistency of data while several storage nodes are used. So, such a DBMS needs to coordinate the simultaneous work of many users with shared information [2-4].

2 Topicality

While solving the problem of distributed database users autonomy, we van face many issues, connected with database structure. There are several approaches to present data in distributed databases, among which combined data sharing strategy unites approaches related to non-duplication and duplication of data in remote nodes to take advantage of all of them. But when it is used, besides the problem of synchronization of duplicate information, the task of optimal design of the remote node database structure becomes urgent. The question is to determine if the data belongs to the particular node of the distributed database. Besides, the system performance directly depends from the decision of partial or complete duplication of data in remote nodes.

Information systems review in solving the problems of automation of different accounting types within the same company, and the disadvantages of using universal accounting systems allowed to make the conclusion of need to optimize the structure of remote database (DB) nodes by identifying useful data and minimizing the volume of the distributed database node [5, 6].

3 Purpose of publication

The research aims to increase the level of data availability in the remote node of the distributed database. It also aims to increase the software usage efficiency, which is connected with the database data. All this achieved by reducing the number of distributed requests in the way of optimizing the structure of the distributed database node and minimizing the amount of data stored in it.

To achieve the goals of the research, it is needed to solve several tasks. We need to perform the research of methods of formal grammars creation and usage. As the result, the subsystem of SQL-query code parsing need to be developed [7]. We need to develop the model and create the technology of accounting user activity based on SQL-queries profiling with the set of analytical characteristics [8]. Next, it is needed to detail user's SQL-queries to database tables in the level of relation attributes and tuples. Finally, the information technology for decision support [9-10] in designing the structure of the

distributed database node and determining the amount of data required needs to be designed.

4 The main part

When using a combined approach to presenting data in the distributed database, the choice of replication type will depend on several factors [5-7]. Among them, one of the important is whether the request type is remote or distributed. In the case of a remote request, when the interaction with other nodes is required only to ensure data replication, it is considered appropriate to use asynchronous replication. Such a decision is justified by the possibility to exclude from the transaction all "unnecessary" nodes. Only the node that stores the data remains (see Fig. 1).

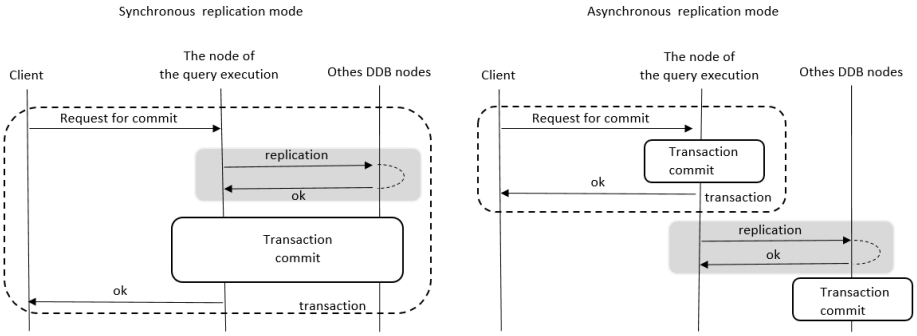

Fig. 1. Replication modes comparison in case of remote query.

If we need to synchronize data in the case of the distributed request, the interaction with the remote nodes cannot be made outside the transaction. This is due to the fact that the distributed transaction in any case includes pieces of data stored on other nodes and not presented at the node of the request execution. In view of this, carrying out the replication process outside the transaction does not seem quite appropriate. This is due to the fact that the reduction of transaction execution time is insignificant compared to the possible data conflicts, related to non-synchronous updating (see Fig. 2).

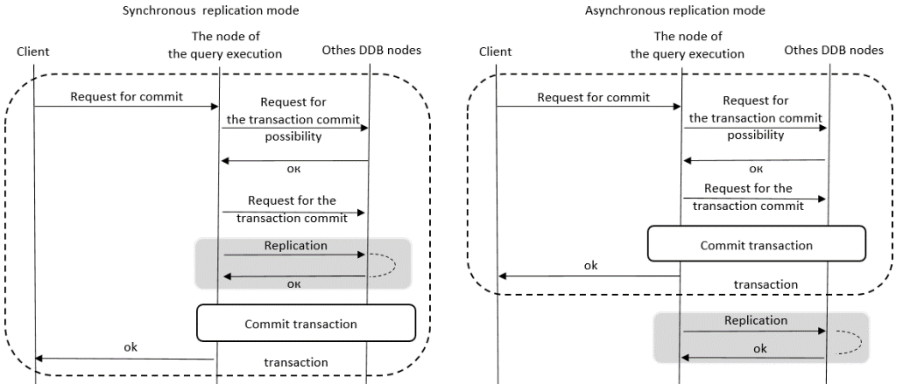

Fig. 2. Replication modes comparison in case of distributed query.

Let's try to formalize the main task on the way to increase the level of general availability of data and efficiency of software systems usage, that work with database data, when designing a remote node of a distributed database. Based on the given above, it is reducing the number of distributed queries in which the data of multiple database nodes are involved and to replace them with local ones [2-3].

To minimize the number of distributed transactions within the remote node of distributed database, the subsystem of the user requests accounting was created. It includes queries classification based on belonging to one or another automated workplace, geographical location, user role or other criteria, which can be dynamically added to the system, depending on one or another subject area. The datalogical model of developed technology is shown in Fig. 3.

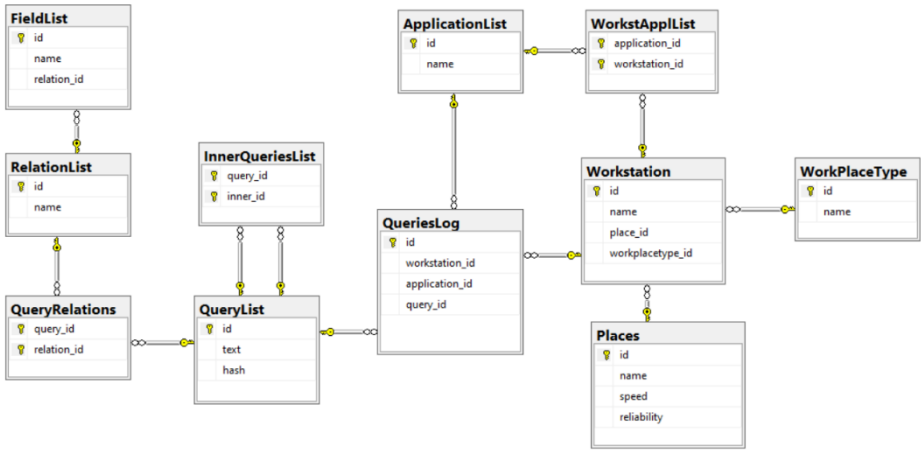

Fig. 3. The datalogical model of the users' SQL query profiling subsystem.

In the center of the model is users' queries log and users' query catalog (QueriesLog and QueryList tables). If a query is nested or has multiple levels, its hierarchical structure is described using the InnerQueriesList table. Queries are classified by the type of software, workstation, user, and location, from wich they come from, binding to future nodes in the distributed database (Workstation, Places, ApplicatioList, WorksAppList, and WorkPlaceType tables). Based on the structure, given in fig.3, it is also evident that including the additional analytical characteristics that can be required depending on a particular subject area, will not require major changes to the database. After parsing the SQL query code (parsing is the process of converting the source code to a structured view), they are also classified by the list of database tables, which are present in the query, and, after doing more in-depth analysis, by the list of relation attributes and tuples (QueryRelations tables , RelationList, and FieldList).

The populating of the developed model with the input data is realized with the help of DBMS profiling mechanisms. While making the research, the SQL Profiler software were used. This product is included to the standard SQL Server installation package. This choice is due to a sufficient list of functionality that application provides and compatibility with the DBMS version that is used in the company, where research results were tested. If the input conditions change, the choice may be different in favor to another application.

The subsystem has the user interface, consisting of data input and editing forms for the main entities. There are mechanisms for importing data from text and csv files so that they can interact with third-party software (for example, while importing user list, software, or workstation tasks). Some entities can also be filled, based on the list of users' SQL queries.

To perform further analytical analysis of the accumulated statistics, it is necessary to determine specific relationships (as well as links to their attributes and tuples) from the text of user's queries. Tools for parsing and linguistic analysis of the text were used to solve this problem. SQL, like every programming language, has rules that define the syntactic structure of the program. The syntax of programming language constructions can be described using context-free grammars or BNF notation (Backus-Naur Form).

In the research, the ANTLR parser generator was used. It is LL (*) and has been in existence for more than 20 years. In 2013, it released its 4th version. Now its development is underway on GitHub. At the moment, it allows to generate parsers in Java, C#, Python and JavaScript. The initial step in implementing the T-SQL query parsing subsystem is to create a grammar. Further, the generated lexer and parser classes allow presenting the query code as a tree model. Then it can be used to determine the list of relationships and attributes that were used in the query. Figure 4 shows an element hierarchy tree for the "select (select number from groups where id = students.id), name, age from students" query, which consists from multiple attributes and has one subquery.

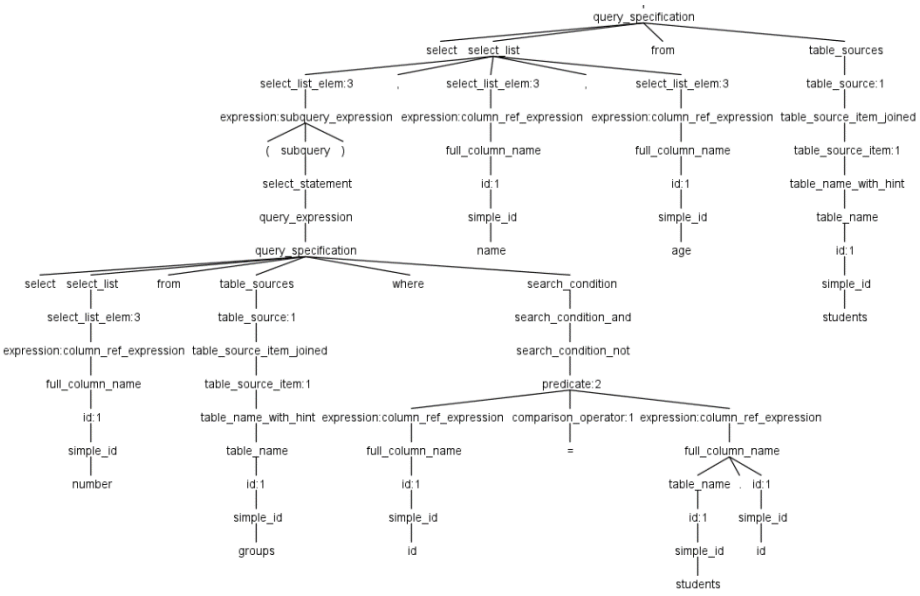

Fig. 4. T-SQL query parsing tree.

Then, each request is presented as an object that has the following attributes: the workstation, the user, and the software from which the request came; the text of the request itself; the collection of tables, which the query requested; and a collection of subqueries, if there are any. In this case, the "table" (TsqlRelation) entity is a subset of the database table and contains the table name, the collection of attributes that are involved in the query, and the collection of table primary key values, according to the tuples set that are returned as the result of the query. The fragment of the classes diagram of the SQL query parser information technology is shown in Fig. 5.

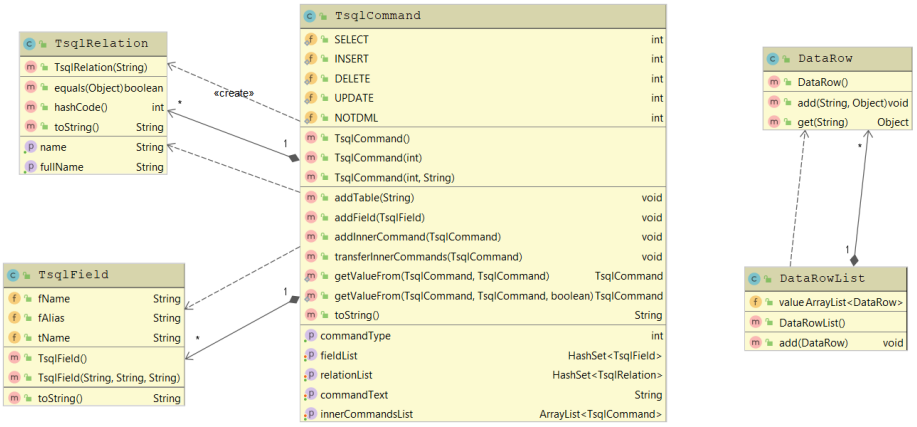

Fig. 5. Classes for presenting queries in the model.

So at this step we have the set of relationships with the attributes used in the query. This is sufficient to determine the list of tables represented in the node of the distributed database and to limit the total amount of data by applying the projection operation to them. However, this may not be enough, as there are usually some tables in the accounting system database that have a lot of tuples. Such tables take a lot of disk space and create extra workload when querying them. At the same time, in one or another remote node of the distributed database, rather a small percentage of the total data amount included in them is actually used.

The next step is to determine the set of tuples used in a particular query. To do this, we extend the TsqlRelation class with the TsqlField type collection that will determine the relation primary key. If the is no primary key, or the key contains a large number of non-numeric data type attributes, and as a result, it has a large volume, the unique auto-incremental not null field can be created at the DB level, which will be used later to identify the tuple.

Further, if the request has an nested queries, each of them is processed separately. An additional database query for each table in the query relationship collection is executed, the attribute list in which is replaced with the relation primary key. The query result is stored as the set of table attributes used by the current query. For example, if the request selects the students' names with student group numbers from two tables, using the nested query to filter by student specialty (Fig. 6), it will be divided into the next queries set (Fig. 7).

```
select s.name, g.number
from students s
inner join groups g
on s.gr_code = g.code
where g.special_code in(
    select code
    from specialties
    where name = 'computer science'
)
```

Fig. 6. Incoming SQL-query example.

```
select distinct s.code from students s
inner join groups g on s.gr_code = g.code
where g.special_code in(
    select code from specialties
    where name = 'computer science'
);
select distinct g.code from students s
inner join groups g on s.gr_code = g.code
where g.special_code in(
    select code from specialties
    where name = 'computer science'
);
select code from specialties
where name = 'computer science';
```

Fig. 7. Query list for retrieving the tuple set, used by the input query.

After identifying attributes set, it is available to determine the subset of valuable data for the remote database node. The determining of the subset is made using table filtering by the value of the primary key. It allows to place the most of data, that node needs, locally and, accordingly, reduce the number of distributed queries. At the same time, the amount of data in the local node is also minimal, which in turn minimizes the need to synchronize different versions of the same data.

However, in the process of operating the described technology, there is a problem, connected with modifying the data in database, when new data is added to the tables or the existing ones are modified or deleted. In this case, the simplest solution of the problem is to replicate to the remote database node any new or changed data (since we do not know the value of its usefulness to the node). When the volume of this data reaches a critical point, for example, based on server load while retrieving or synchronizing data, it is necessary to re-analyze the SQL queries and update the base of attributes and tuples of the database tables, that are used by a remote node.

It should be mentioned, that performing the complete analysis of the database tables attributes and tuples evaluation is a very resource-intensive operation and cannot be performed frequently. So the approach, connected with regular recalculations is unacceptable for large and frequently changing databases. Therefore, it was suggested to present the evaluation problem of new or modified data in the form of a data mining classification task [11]. As an input we have the name of the table and the list of values of its attributes (new or changed row), and as an output - the decision about its value for the remote database node. To solve this problem it is proposed to use the classical neural network perceptron with one hidden layer. Network training is based on the data, obtained from the SQL query parsing analysis results. Then, after being trained, the network classifies new and changed data.

According to need of performing the analysis of the stored data in terms of multiple dimensions, as well as likely large volumes, the data required for the analysis was presented in the form of multidimensional database or, in other words, on-line analytical processing (OLAP) cube [12]. To do this, a number of views were added at the relational database level. They implements the fact table and the dimension tables in the form of a star model. The structure of the obtained multidimensional cube is shown in Fig. 8.

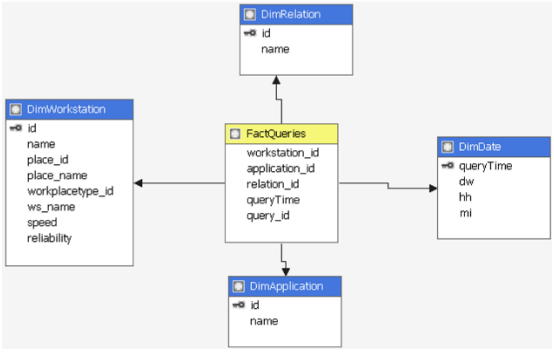

Fig. 8. The OLAP-cube structure.

The figure shows the base structure with one measure, which is time of the query execution and four dimensions – workstation, relation (table), application (where the query came from) and standard datetime dimension. It is obvious, that we can easily expand the quantity of dimensions by adding, for example, hierarchy of fields and tuples into the "relation" dimension, or user, department and branch into workstation dimension. Also, if queries profiler application allows to accumulate other useful data, for example i/o bytes volume, it also can be added to the fact table.

The results of the research were tested while the structure of the database of the automated retail store were designing. It also was used in the implementation of synchronization subsystem with the central database. Using local and remote DB versions, some indicators were collected. The results are shown in Fig. 9.

	Remote database	Local database
DB volume, mb	29025,63	174,88
Tables quantity	460	34
DB backup time	7:58	0:13
Average number of connections	59,07	1,02
Average requests per minute	817	44
Average processing time, ms	5665	99
Synchronization time, sec	0	18

Fig. 9. Indicators comparison while using remote and local version of the database.

Therefore, the result is an increase of the read/write operations speed at the remote node information system and unloading of the central database. This result is achieved by optimizing the DDB structure, minimizing the number of distributed transactions, and using asynchronous data replication mode.

5 Summary and conclusion

While the research was done, several achievements were made. For the first time, it is proposed to use multilevel catalogs of analytical characteristics of user SQL queries with data relevance level estimation. The SQL query parsing subsystem has been developed. It allows to present user's query as a set of relations, with the sets of used attributes and tuples. For the first time, to analyze user's SQL queries, the multidimensional data model was used and based on it, the decision support system was implemented. While designing the remote workplace DB structure, the decision support system allows to perform on-line analytical analysis of SQL queries by dimensions, quickly and efficiently perform operations of consolidation, detail, rotation and slice. When modifying the central database data, it is suggested to use the neural network to

solve the classifying task of new or changed data according to their value for the remote node of the distributed database.

References

1. Mahon, E.: Transitioning the Enterprise to the Cloud: A Business Approach, Cloudworks Publishing Company, 178 p. (2015).
2. Tamer Özsu, M., Valduriez, P.: Principles of Distributed Database Systems. 3rd ed., Springler, , 845 p. (2011).
3. Kimball, R., Caserta, J.: The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data/ Willey Publishing, Inc., 528 p (2014).
4. Fisun, M., Shved, A., Nezdoliy, Yu., Davydenko, Y.: The experience in application of information technologies for teaching of disabled students. In: Proc. of the 8th IEEE Int. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). Warsaw, Poland, pp. 935-939, September 2015 IEEE (2015). DOI: 10.1109/IDAACS.2015.7341441.
5. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems: The Complete Book. Upper Saddle River, New Jersey: Prentice Hall; Dep. of Computer Science Stanford University (2002) (in Russian).
6. Transaction Management in ORACLE: http://www.novsu.ru/file/96492/, last accessed 10/11/2019.
7. Dvoretskyi, M. Davydenko, Y.: Creating the structure of DDB based on SQL-queries pasring results. Science works "MNU named by P.Mohyla". Series: Computer technologies, (2016). (in Ukrainian).
8. Fisun, M., Dvoretskyi, M., Shved, A., Davydenko, Y.: Query Parsing in Order to Optimize Distributed DB Structure. In: Proc. of the 9th IEEE Int. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017. Vol. 1, pp. 172-178. IEEE (2017). DOI: 10.1109/IDAACS.2017.8095071.
9. Larichev, O. I. ,Petrovskiy, A. V.: Decision support systems. The current state and prospects for their development. Vol.21., p. 131-164, (1987). (in Russian).
10. Tikhonov, A. N., Tsvetkov, V.Y.: Methods and systems of decision support. MAX Press, 312 p., (2001). (in Russian).
11. Oreshkov, V.I.: Intellectual data analysis as the most important tool for the formation of the intellectual capital of organizations. Creative Economy, 12, 84-89 (2011). (in Russian).
12. Barsegyan, A. A., Kupriyanov, M. S., Stepanenko, V. V., Holod, I. I.: Methods and models of data analysis: OLAP and Data Mining. St. Petersburg: BHV-Petersburg, 336 p, (2004). (in Russian).