UDC 004.658: 652.3

R. Belous, E. Krylov, V. Anikin

METHODS OF OPTIMIZATION OF DISTRIBUTED DATABASES

Abstract: The great variety of existing databases is emphasized and their brief description is given. When analyzing a distributed database management system, it is emphasized that they consist of a single logical database, divided into a number of fragments. Each fragment of the database is stored on one or more computers (nodes, sites), which are interconnected by a communication network and each of which operates under the control of a separate database.

The advantages, disadvantages and necessary requirements to the distributed database are analyzed. The most effective ways to optimize the structure of distributed databases are listed separately, approaches to assessing their effectiveness (performance, scalability (extensibility), reliability, data protection, availability, ease of application development, the level of interaction with the user). The fragment of grammar of T-SQL language is resulted and the basic generalizing recommendations on writing of inquiries which are convenient for the optimizer and effective at execution are resulted.

Keywords: Distributed databases, optimization methods, DBMS, database «DB-DBMS».

Problem description

Among the great variety of databases (DB) we can distinguish relational and post-relational database systems. To date, the second group of databases is of the greatest interest, because they operate with complex structures, which is typical for such application systems as CAD, artificial intelligence systems, etc. Among this group of databases can be distinguished [4, c. 38]:

1. Active databases. A database is called active if the database contains a production system, the conditions and actions of which are not limited to the contents of the database or direct actions on it by the user.

2. Deductive databases. This type of database consists of two parts: extensional, containing facts, and intentional, containing rules for the logical conclusion of new facts based on the extensional part and the user's request.

3. Temporal databases. The main features of temporal databases:

– for any data object created at a time t_1 and destroyed at the time t_2, in the database are stored (and available to users) all its states in a time interval $[t_1, t_2]$;

– it is possible to sample the information stored in the database at the specified time, at the specified time interval, etc.;

– it is possible to create versions of the relationship, and their further modification is allowed, taking into account changes in the main options.

Temporal database (as well as deductive) - is an add-on to the relational system.

4. Multi-databases (integrated or federated systems). Multi-databases use special naming methods to access local database objects of distributed corporate systems. At the global level, only data sampling is allowed, which allows to preserve the autonomy of local inhomogeneous databases distributed in the computer network. As a rule, a relational data model is used for the external representation of integrated and multibases, although recently it is increasingly proposed to use object-oriented models.

5. DDB - Distributed Database - it is a set of interconnected databases distributed in a computer network. The system consists of many request receiving nodes and a non-empty set of data nodes. Data nodes have the means to store data, and request reception nodes do not; they only run programs, implement a user interface to access data stored in data nodes. Nodes are independent computers connected by a network, not processors that make up a multiprocessor configuration. Their most important distinguishing feature is the weakly connected nature of the environment, where each node has its own operating system and operates independently [2]. The database is physically distributed across data nodes through fragmentation and replication or replication of data.

6. Let's analyse the shortcomings of the distributed database. The most significant disadvantages are:

– increasing the complexity of software and hardware. Distributed DBMSs are more complex software packages than centralized DBMSs. For example, it is sufficient to point out the fact that the data can be replicated. If data replication is not maintained at the required level, the system will have a lower level of data availability, reliability, and performance than centralized systems. Distributed systems can have problems protecting not only the data that is replicated to several different sites, but also protecting the network connections themselves. In centralized systems, access to data is easily controlled.

– requirements for ensuring the integrity (correctness and consistency of data) are formulated in the form of restrictions. Compliance with the restrictions guarantees the protection of information in the database from destruction. Implementing such integrity constraints requires access to a large amount of data used during inspections.

– in distributed DBMS the increased cost of data transmission and processing can prevent the organization of effective protection against violations of data integrity.

– complicating the database design procedure. In addition to the usual problems associated with the design of centralized databases, the development of distributed databases requires a decision on data fragmentation, the distribution of fragments on individual sites and the organization of data replication procedures.

Currently, there are distributed databases with intelligent information retrieval systems. One of the modern directions of artificial intelligence is the creation of an intelligent system for searching relevant information using the principles of building multi-agent systems, the main component of which are agents [3]. The general task is divided into subtasks, and its

solution is carried out as a composition of the behavior of agents who implement the selection and implementation of a sequence of actions available to them aimed at achieving their own goals [1]. It assumes the existence of mechanisms for adapting agents, algorithms for their training, especially when it comes to finding relevant information in information environments. Of particular interest are genetic algorithms and evolutionary strategies.

In summary, it should be said that among the many properties that, according to K. Data, must satisfy a distributed database, we highlight the following:

– Local autonomy. Local data belongs to local nodes and is managed by administrators of local databases.

– Continuous operation. Removing or adding a node should not require stopping the system as a whole.

– Independence from fragmentation. Access to data should not depend on the presence or absence of fragmentation and the type of fragmentation.

– Processing of distributed requests. The system should automatically determine the methods of data connection (aggregation).

– Independence from the DBMS type. DBMS must be able to function on top of different local databases and be compatible with different data models (heterogeneity requirement).

The process of designing a database «DB-DBMS» consists of the following 4 stages:

– System analysis of the subject area.

– Construction of an infological (conceptual) model of the subject area. This model does not depend on the database (and, consequently, on the storage environment), and is presented in terms of the chosen semantic model (one of the most common semantic models is developed by P. Chen ER-model (Entity Relationship).

– Building a logical model. The data model and DBMS type are selected. The scheme of a DB and subschemes for various users is under construction. A set of possible standard queries is created. Specifications for the software are defined.

– Construction of a physical model. The location of the database on external media is selected and the DBMS used is determined. A real database is created, and then applications that will work with the database are programmed and tested. The result of the process of designing a connection «DB-DBMS» is a ready-to-fill real database and ready-to-use software.

Thus, in connection «DB-DBMS» infological, logical and physical levels are allocated. The first two levels are called external and are designed to support authorized access to database data. The third level is called internal and reflects the organization of data in the storage environment.

The indexing device is the most important tool of any relational database. And although almost all actions on the data can be performed without the participation of indexes, it is absolutely unthinkable to ignore them when creating real information systems. Only the use of indexes allows to achieve acceptable speed characteristics of data processing, because search operations (present directly or indirectly) are used in programs very widely.

Optimization of the structure

However, such indexes occupy a certain place on the disk. In addition, data entry / editing operations in the database are slowed down, because when it is supplemented with a new record, the index must be automatically rebuilt according to the new or changed data. Among the main ways to optimize the structure of distributed databases are the following:

– optimization of placement of distributed databases on the nodes of the network structure of the system, which is not currently supported by modern CASE - systems (Computer Aided System Engineering);

– network scalability of databases for geographically distributed systems;

– design and implementation (use) of the communication environment of the memory system as a basis for immersion of the distributed database;

– optimizing the placement of data in distributed databases to ensure load balance;

– synchronization of data access and parallel processing of requests to ensure the required system performance;

– maintaining copies of data across multiple network nodes to reduce forwarding when performing distributed queries;

– fragmentation of database objects and expansion of search space for query execution options;

– transaction management (for example, correct completion of distributed transactions), multilevel data protection in the database and the implementation of the possibility of obtaining a variety of structure and content of information from different types of sources.

At the present time, scientists identify the following two approaches to the analysis of the efficiency of query processing in connection with «DB-DBMS». The first approach is based on the use of a symbiosis of mathematical logic, model theory and applied theory of algorithms. The subject of his research is a mathematical model of the connection «DB-DBMS». The purpose of this approach is to construct lower estimates of the asymptotic temporal complexity of the functioning of the connection «in the worst case» or «on average». The second approach is based on the study of the real connection «DB-DBMS». As a rule, it is carried out by methods of statistical analysis. The purpose of this approach is to estimate the real-time processing of requests (possibly model) implementations of communication under certain conditions.

We emphasize that in order to obtain an adequate picture, both of these approaches to the analysis of the efficiency of query processing must be carried out both at the design stage and during the operation of the connection «DB-DBMS», especially with its structural modifications and the creation of new versions.

The word «optimization» is a mathematical term and means the choice of the best (or close to it according to a given criterion) object or algorithm that belongs formally to a certain class, objects or algorithms. It is clear that this approach is unsuitable neither from a

theoretical nor from a practical point of view for the connection «DB-DBMS» (as well as to many other application systems). Therefore, the «optimization of the processing» of database requests means strategies that are designed to improve the efficiency of request processing procedures. Such strategies are heuristic methods, the feasibility of which is justified by statistical methods, ie processing the results of experiments.

Using stored procedures in optimization processes will significantly speed up the execution of frequently used SQL statements (especially queries). This is because the SQL statement is already compiled and located on the server. The program only needs to pass to the server the name of the procedure to be performed, which will reduce the amount of information transmitted between the client machine and the server.

The solution of the problems considered above for different types of distributed databases significantly affects the quantitative and qualitative indicators of the main parameters, which with a sufficient degree of completeness characterize a particular class (type) of databases. In this case, when it comes to evaluating the parameters of a database of a specific type (class), it means a database as a whole product that performs a specific set of functions. To do this, it must contain not only the actual database (devices with all the attributes of hardware and software and its internal data content), but also the database management system, in the classical sense - DBMS, because, as the analysis of modern and promising databases, database parameters and DBMS are interconnected: specific values of the database parameters can be obtained only in the presence of a specific type of database and conversely. From this point of view, it is advisable to evaluate the parameters and solve the design problem for the database, taking into account the presence of the appropriate database.

The following are usually used as evaluation parameters of the database: productivity (Efficiency) - P, scalability - M, reliability - V, data protection - W, accessibility - Q, ease of application development - Z, level of user interaction - X. The set of these parameters is defined as a characteristic set ψ of the product (system): $\psi = (P, M, V, W, Q, Z, X)$. Productivity (P), usually provided by a combination of several complementary solutions present in the system as a whole, such as the use of operating systems focused on database support, concurrency, optimization, load balance. One way to increase the performance of database servers is parallelization: parallel input-output operations with disk, parallel utilities, and parallel query processing.

Parallel disk input-output allows the server to efficiently use multidimensional tables and partition tables. Parallel operations-utilities (sorting, index construction, loading, backup, recovery) use both parallel processing and parallel exchanges with a disk.

Scalability (M) - this is a property of the system that allows higher performance, bandwidth, etc. by adding computing resources without changing applications and administrative support. There are two ways to expand databases: horizontal and vertical. Horizontal extension can be obtained when multiple database servers operate independently and share workspace.

Vertical expansion is achieved by adding additional resources to the platform, such as faster processor elements or additional processors to reduce response time or increase

bandwidth. Supporting the database server for vertical expansion should not require the inclusion of additional software modules, as this increases administrative costs.

Ideally, a parallel and, to a lesser extent, distributed database has the property of linear extensibility and linear acceleration. Linear extensibility means maintaining the same level of performance while increasing the size of the database and at the same time proportionally increasing the processing power and memory. Linear acceleration means that with increasing CPU power and memory while maintaining the previous size of the database, productivity increases proportionally. Moreover, the expansion of the system will require minimal reorganization of the existing database.

Reliability (V) - database protection against failures and the ability to recover from them, providing access to data from both reading and writing regardless of the circumstances, including failure of individual components. Such systems are defined as fault tolerance. They contain, as a rule, hardware redundancy and redundancy according to the data. In the first case, separate platforms, processors, dual drives are used. A common phenomenon is the reflection of hardware, in which one disk is used as a copy of another and protects it from failure. Redundancy according to the data is presented in two forms of mirror mirroring and copying, which eliminates single points of failure: software mirroring and copying, which eliminates single points of failure. In this case, the failure of one node or failure of the communication line does not lead to failure of the entire system.

Data protection (W). The two most basic approaches to data protection are the most popular: applying a set of valid privileges to each protected database; defining for each user a specific set of access rights, for example, based on the use of cryptographic methods.

Accessibility (Q). The main characteristic of database availability is their maintenance in the operational mode. Ideally, maintenance utilities should support continuous operations by which the system should reduce or even eliminate planned and unplanned problems.

The physical location of the database, reorganization, memory management, protocol archiving, and system restart should have minimal impact on applications.

Easy application development (Z). The complexity and simplicity of application development are determined primarily by the presence of the developer of advanced design tools (special preprocessors and interactive tools for developing database schemas, the ability to use multi-user application development tools, various customizers), additional tools for building applications, advanced settings, the required volume of object libraries, as well as the completeness of technical documentation, source code and other components.

As a way to optimize the structure for translating queries of different DBMS, we can offer to use grammars of the most popular DBMS languages in a form close to RBNF, as well as ANTLR - parser generator, which allows you to create a parser in one of the target programming languages. Below is a snippet of T-SQL grammar:

```
grammar tsql;
tsql_file
: sql_clauses? EOF;
```

```
sql_clauses
: sql_clause+;
sql_clause
: dml_clause
| ddl_clause
| cfl_statement
| another_statement;
dml_clause
: delete_statement
| insert_statement
| select_statement
| update_statement ;
 insert_statement
 : with_expression?
INSERT (TOP '(' expression ')'
PERCENT?)?
INTO? (ddl_object |
rowset_function_limited)
insert_with_table_hints?
('(' column_name_list ')')?
output_clause?
insert_statement_value
for_clause? option_clause? ';'?
```

Recommendations for writing queries

Here are the main general recommendations for writing queries that are convenient for the optimizer and effective in performing.

1. If the request contains several conditions, they must be arranged in descending order of selectivity.

2. In a query that implements the connection of two or more tables, these tables must be in the FROM list in descending order of the number of records in them, and in the WHERE part must be the first condition for the main (parent) table.

3. If the request contains a condition with an indefinite leading part of the type (field LIKE '% ...') or (field LIKE '_ ...'), then it is necessary to supplement this condition so that the system can use the index on the field field (if it exists). Here is the condition special> 'A' does not exclude any table entry from the search, but allows the system to perform this index search, which takes up much less memory than the table itself.

4. If the query contains a condition for the indexed field of the small table, which can be read in one memory access, the query must be formulated so that the system ignores the index.

5. Should be used UNION ALL instead UNION, if there are no identical records in the combined relationship (or the presence of identical records is uncritical), because UNION is calculated by sorting, which can take a long time, and UNION ALL does not require sorting.

6. Should be used IN instead EXISTS, if EXISTS not optimized. But for the subqueries giving out the big list, the variant of request with connection (in the presence of an index on a foreign key) can appear more optimum:

SELECT DISTINCT e. * FROM Emp AS e, Children AS з WHERE

c.empNo = e.empNo;

7. If the optimizer does not optimize the operation well «OR», then it can be replaced by an operation UNION in the presence of indices. To be convinced of «bad optimization» it is possible so: to execute request on condition (Field = X) and a conditional request ((field = X) OR (field = Y)) on a large table. If the second query runs much longer than the first, the OR is not optimized.

8. The condition «is not equal to» ('<>') also inhibits the use of the index. Therefore, if the values of the indexed column are unevenly distributed, it should be replaced by a combination of conditions '<' OR '>' and, subject to the previous rule, ie implement it with UNION.

9. Some optimizers will use index scanning if the query contains a partition ORDER BY with an indexed column. To fulfill the next query will be used index on the column tabNo, even if this column is not used in section conditions WHERE:

SELECT * FROM emp WHERE depNo <3 ORDER BY tabNo;

10. Terms <expression1> op <expression2>, where op - operation also does not allow the use of the index. From expressions it is necessary to take out in the left part the field on which there is an index if possible. For example, a condition (Salary * 0.87> 30000) it is better to write so: salary> 30000 / 0.87.

When setting up commands SQL it is important to remember that by setting up one of them, you can make an impact (and not always positive) to other teams. Therefore, during the configuration it is necessary to periodically perform regression testing, ie restart the already tested commands to estimate the time of their execution. Many DBMSs allow you to view the request execution plan by administrative means. So, you can make sure that the system uses built-in indexes to execute queries.

Conclusions

Thus, the use of data replication technology poses a number of optimization problems, namely: optimal distribution of replication databases in a distributed information system in terms of minimizing maintenance costs, limitations on computing resources of nodes, minimizing data synchronization time or minimizing the average time required to search for information. lack of synchronization of replication databases.

The practice of using IP has an array of data in several databases on certain logical or technological grounds. For example, a combination of a relational database (for example, Microsoft SQL Server), which acts as a business data warehouse, and a NoSQL database (for

example, MongoDB) as a repository of service information (for example, a log file) is popular, so it is possible to place it in one nodes of the distributed information system of several databases. As a result, the study of the optimal distribution of replication databases in information systems is relevant and is the subject of further research.

REFERENCES

1. Belous R. V. Devising a method to identify an incoming object based on the combination of unified information spaces / R.V. Belous, I.V. Krylov, Y.I. Kornaga та ін. // Eastern-European Journal of Enterprise Technologies. – 2021. – С. 35–44.

2. Bobreshov-Shishov D.I. Dynamically managing the structure of a distributed database / D.I. Bobreshov-Shishov, L.A. Sayarkin // Young scientist. 2015. № 7. С. 51–53.

3. Fisun M., Dvoretskyi M., Dvoretska C. Construction of models for optimizing the structure of the database of the node in corporate information systems // ITKI, 2020. – vol 48, № 2, С. 52-60.

4. *Yartsev V.P.* Distributed databases: a textbook. K. DUT 2018. 97 с.