World Applied Sciences Journal 26 (6): 728-732, 2013
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2013.26.06.13509

DISGO: Data Integration System for Distributed Corporate and Inter-Corporate Networks

Alexander Alexeyevich Bukatov and Alexander Vladimirovich Pyhalov

Southern Federal University, Rostov-on-Don, Russia

Submitted: Oct 12, 2013; **Accepted:** Nov 18, 2013; **Published:** Nov 23, 2013

Abstract: The article concerns the task of integrating initially isolated independent databases (DB), which store the data of some common domain. Techniques and tools for solving this task are discussed. Main subtasks and disadvantages of existing solutions emerging when they are used in distributed corporate network are concerned. New techniques for solving these subtasks are suggested. These techniques are used in DISGO data integration system intended for integrating data in distributed corporate network.

Key words: Databases · Data integration · Schema mappings · Datalog · Query optimization · Error processing

INTRODUCTION

This paper concerns techniques and tools for integrating initially independent heterogeneous databases containing data of some common domain.

A necessity of providing unified access to the overall set of functionally similar data emerges during integration of several enterprises. Such access may be provided by special user-oriented software, interacting with data by means of data integration systems, such as [1,2]. This paper describes architecture and algorithms used in DISGO data integration system.

Nowadays there is a noticeable trend of mergers and acquisitions in several fields (e.g. education, software development). Data integration systems utilization looks attractive concerning this trend and a necessity for quick access to all data of united organization. The important requirements to data integration system are scalability with respect to the growing number of connected data sources [3] and ability to return incomplete answers during temporary unavailability of several databases due to network connectivity problems.

Another use case arises during data exchange between several independent organizations. Taking into account independence of collaborating organizations real merging of their databases is impossible. The only possible approach is creating virtual integrated database by means of some data integration system. Meanwhile the requirements of scalability with respect to the growing number of connected databases and ability to return incomplete answers are also actual.

This paper treats suggested data integration system. We start by discussing basic tasks of data integration systems and highlighting tasks in which we are particularly interested. Then suggested DISGO data integration system is described. Its advantages are shown in comparison with available commercial data integration software. We conclude by stating our results.

Data Integration System Tasks: The main purpose of data integration system is to provide unified API for accessing multiple independent data sources. A user (application programmer) submits queries to the set of data sources in terms of some virtual data schema (by data schema (or briefly schema) we mean formal data structure definition) which is defined by means of data integration system (so called global schema). This virtual data schema is mapped to local data schemas of data sources, which are called foreign data sources. Queries to global schema are translated into queries to local schemas.

Data integration tasks in enterprise networks has the following peculiarities:

Corresponding Author: Bukatov, Southern Federal University, Stachki 200/1, Building 2, 344090, Rostov-on-Don, Russia.

- Large number of data sources;
- Only small subset of existing data is shared (i.e. each data source provides small data volume);
- Several data sources are likely to be unavailable (due to distributed network organization);
- Relatively high cost of querying data source (e.g. foreign data sources may be connected by VPNs over Internet);
- Data sources are managed by different groups of administrators, however, coordinated administration is possible;
- There is no necessity (or even possibility) to change data of foreign data sources.

Let's call answer X for user query to data integration system canonical, if it is obtained when all data sources are available. Let's call answer Y for user query sound if $X \subseteq Y$ and complete if $X \supseteq Y$. Notice, that canonical answer is sound and complete. When some data sources are unavailable, system can't always get complete answer for user query. In this case it is necessary to return the nearest to complete sound answer Y_1, i.e. such answer Y_1 that for every other sound answer X which can be received, $X \subseteq Y$. Notice, that in any case it is possible to get sound answer represented by empty set.

Modern data integration techniques utilize one of three basic approaches to description of schemas mappings (or their modifications). These approaches are GAV (Global As View) [4], LAV (Local As View) [5] or GLAV (Global-Local As View) [6]. Due to complexity of LAV and GLAV approaches most of the practical techniques are based on GAV approach to description of schemas mapping (or GAV extensions). This fact significantly restricts possible relations between global and local schemas.

Research projects devoted to data integration in Internet often use semi-structured data models, such as XML [7], RDF, OWL [8] or some other models from knowledge representation area. However, the fact that relational data model is the basic one for enterprise networks creates problems of efficient query mapping, let's say, from RQL or XQuery to SQL language. This justifies utilization of relational model and similar in data integration techniques for enterprise TCP/IP networks.

Basic techniques for query optimization in the field of data integration are targeted on minimization of traffic between data sources and data integration system (this also includes exclusion of data sources based on global constraints or their properties) [9,10]. In distributed network natural optimization is parallelizing subquery execution in different data sources. Meanwhile in case of data source unavailability data integration system should return users nearest to complete sound answers. Authors unaware of existing parallel query execution techniques taking into account data source unavailability.

DISGO Data Integration System: In this section we discuss DISGO data integration system, its basic principles and adopted approaches to solving above-mentioned tasks.

DISGO data integration system is intended for integrating information of multiple independent relational data bases in distributed network. While developing DISGO the following factors were taken into consideration: high probability of data sources unavailability, possible inconsistency of different data sources, incompleteness of data sources (there is no data source which contains all information available from other data sources). Query optimization techniques are based on the following heuristic. Data integration system returns small amount of data querying relatively large amount of independent data sources. It is related to inability of a man to process large amounts of data and so there is no need to process queries returning thousands tuples. The main way of query optimization in DISGO data integration system is to rise efficiency of retrieving information from available data sources.

Global schema in DISGO is represented by a set of explicit predicates mapped to queries to data sources and rules determining implicit predicates on the base of earlier defined. User queries to DISGO and global schema's rules are expressed in specially developed Datalog dialect - DISGO QL.

DISGO operations may be described as follows. The system translates user queries to the set of SQL queries. While translating queries and getting information about referenced data sources DISGO queries its dictionary which is stored in so-called central database. We name central DBMS those one which is used to manage central database. To receive necessary data DISGO queries data sources. Intermediate answers are temporary stored in central database. The system forms final answer from received data issuing a set of queries to the central DBMS. Then DISGO returns to user a status code determining if all necessary data sources were successfully queried. If query was successfully executed

the system also returns a binding table (binding table in logic programming languages is a set of query variables' values). The binding table contains values of variables which make logical statement of user query true.

DISGO employs special techniques which allow existing applications after new data sources being added to the data integration system to use their information transparently. These techniques are based on utilizing an extended Datalog version (DISGO QL) as a data model and a query language and in particular on predicates with named arguments (PNA) suggested by authors. Global schema and data are represented by predicates and rules expressed in DISGO QL. The main peculiarity of DISGO QL is PNA concept.

In DISGO QL predicate structure description defines not only data types of predicate arguments, but also their names. It allows significantly change semantic of rules defining implicit predicates. PNA can be used in global schema definitions or in user programs. All PNA arguments (in contrast to usual Datalog predicate) are referenced by name. When such predicate is found in the tail of rule (i.e. in its right part), the system is looking in the namespace of the PNA for the predicates having the same name and providing all arguments with specified names. Then the PNA is evaluated using found predicates. When a PNA is used in a head of user program rule (i.e. in its left part), it is added to the PNA search space. Arguments of all explicit global schema predicates are named.

Besides increasing query language expressiveness PNA significantly influence mapping creation procedures. First of all, notice that we receive necessary flexibility in mapping creation when using GLAV approach. When using GAV approach query answering algorithms are trivial but we loose information about attributes provided by data source, which is missed in global schema. Query answering algorithms used by GLAV and LAV approaches are more complex and they can lead to incomplete answers even when all data sources are available [5]. This is why suggested techniques use extended GAV approach based on PNA. The approach allows to process queries efficiently and to return complete answers when all referenced data sources are available.

Consider the following scenario of using PNA (Fig.1). Suppose, an application selects data about lectures read by university staff using PNA *Lecture* with arguments *course, group, prof* defined in *studdb* namespace.

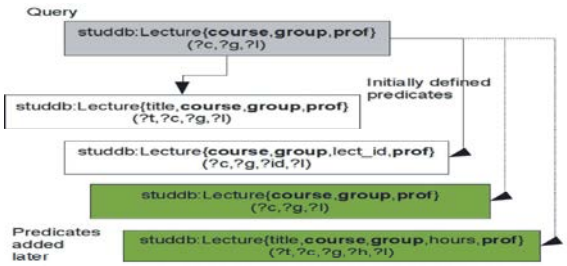

Fig. 1: PNA usage example

Suppose, that initially there are two types of data sources. First one provides information about course title, course and group numbers and professor reading a lecture.

The second provides info about course and group numbers, lecture internal identifier and professor.

Then it is possible to define two PNA *Lecture* in *studdb* namespace having attributes *title, course, group, prof* and *course, group, lect_id, prof* respectively.

Also suppose that necessary mappings for these PNA to SQL queries to different DBMS are defined. In this way the application has access to information provided by both types of data sources. If later additional data source types appear (for example one type - providing only required information and another providing additional information about number of lectures in a course and course title), then by defining in *studdb* namespace two additional PNA and creating necessary mappings to SQL queries it is possible to make new information available for our application without modifying it.

Using PNA it is possible to determine predicates (and corresponding data sources) used by application in run time. This allows existing applications to use data of new data sources even if these data sources provide additional information unexpected by application and new applications to use all available data.

Query processing in DSIGO involves following steps: 1) building a data access plan, 2) data access plan optimization, 3) building a query execution plan based on the data access plan, 4) executing query according to the query execution plan. Internally query is represented as expression in relational algebra (RA) dialect.

Data access plan is a list of data sources and SQL queries to these data sources, used in definitions of explicit predicates in user query. During data access plan optimization step queries to the same data sources in different subqueries can be merged. Statistical estimation is used to determine advisability of the transformation.

On base of data access plan a set of data retrieval, relational calculus programs execution and local transformation operations is constructed. To execute these operations concurrently we introduce an *operation group* term.

An operation group is a set of data retrieval and transformation operations which is to be executed for getting answer for subquery of processed RA expression, such that all operations in the group can be executed concurrently. Each operation group has a fault tolerance attribute, which determines the maximum number of errors detected in this operation group which is not fatal for getting incomplete answer.

Dependencies are formed between operation groups. These dependencies determine the order of operations execution. They are formed on the base of query's operation tree, common data access operations and recursive predicates defined in user query. During query execution number of errors in each operation group is monitored. When this number is more than fault tolerance of the group, the group is marked as faulted and number of errors in depended operation groups is increased. All operation groups on which the faulted (but no other non-faulted) group depends are also marked as faulted, because their further execution is unjustified.

Suggested techniques of determining and executing operation groups allows parallel execution of independent operations and processing errors arising during information retrieval. This technique also allows determine and eliminate unjustified operations. E.g. operations corresponding to processing the right part of *JOIN* operation can be eliminated when it is impossible to get any data for relations in the left part.

DISGO advantages in comparison with existing solutions are the following. When using PNA 3-level scheme is formed: used PNA => defined PNA => queries to data sources. This allows combine GAV simplicity when defining and processing mappings meanwhile creating rather complex mappings which otherwise can be expressed only using GLAV approach. DISGO considers all data source incomplete and equal in rights. DISGO in contrast to other data integration systems working with incomplete data sources [9,10,11] doesn't use global constraints. All variables values making query predicate true are allowable and can not exclude each other. In contrast to Xiao system [8] and Piazza system [7] DISGO is a system with central organization which is more appropriate in corporate network. Such organization and flexible rights system allows delegate mapping creation to data sources administrators, exclude their automatic generation and make them more robust.

During performance tests DISGO was compared to data integration subsystem of a popular commercial DBMS in local and distributed network. Results of the tests showed rather good DISGO efficiency. In local network DISGO performance was higher on 30% when query answer size was not too large (<=3000 tuples). When query answer size was less then 5000 tuples DISGO showed comparable performance. It should be noted that most of human-processed answers (for example, Web store search results) contains 1000 records or less. (The dependency on answer size is mostly related to XML-based communication protocol. Translation of answers from binary form to XML created high load on server CPU). In distributed environment DISGO could receive incomplete answers when at least one way of query processing existed. Query processing speed increased with growing number of inaccessible data sources. Commercial solution couldn't work in such conditions.

In distributed network when data sources provide relatively small volume of information main delays during query processing were network delays (but not network speed limitations) and delays introduced by DBMS. Query parallelism allows decrease total delay during information retrieval. This is why robust parallel query processing techniques are important.

CONCLUSION

The paper describes DISGO data integration system providing means of creating middleware integrating a set of independent data bases distributed in corporate or inter-corporate network. The main results discussed in this paper are the following.

Firstly, DISGO allows quickly create middleware for integrating data bases with possibility of system scalability with respect to the growing number of connected data sources. Fast speed of middleware creation is accounted for DISGO query language utilization and provided query optimization and processing techniques which include native error processing support. Scalability is achieved by PNA utilization. While adding new data sources all schema mappings are set in data dictionary transparently for applications.

Secondly, DISGO provides means to retrieve nearest to complete sound answers for queries in case of data source unavailability. Query answering speed decreases with growth of unavailable data source number.

REFERENCES

1. Ensemble product specifications. Date Views 10.10. 2013 www. intersystems. com/ ensemble/ product-specifications. html.
2. Introduction to InfoSphere Federation Server. Date Views 10.10.2013 pic.dhe.ibm. com/infocenter/ iisinfsv/v8r5/index.jsp?topic=%2Fcom.ibm.swg.im.iis.productization.iisinfsv.overview.doc%2Ftopics%2Fcisofedoverview.html
3. Hentschel, M., L. Haas and R.J. Miller, 2010. Just-in-time Data integration in Action. Proceedings of the VLDB Endowment, 3(1): 1613-1616.
4. Garcia-Molina, H., V. Papakonstantinou and Y.D. Quass, 1997. The TSIMMIS approach to mediation: Data models and Languages. Journal of Intelligent Information Systems, 8(2): 117-132.
5. Duschka, O.M., M.R. Genesereth and A.Y. Levy, 2000. Recursive Query Plans for Data Integration. In: Journal of Logic Programming, 43(1): 49-73.
6. Friedman, M., A. Levy and T. Millstein, 1999. Navigational Plans For Data Integration. National Conference on Artificial Intelligence (AAAI): 67-73.
7. Halevy, A., Z. Ives and P. Mork, 2003. Piazza: Data Management Infrastructure for Semantic Web Applications. 12th international conference on World Wide Web., pp: 556-567.
8. Xiao, H. and I. Cruz, 2006. Ontology-based Query Rewriting in Peer-to-Peer Networks. 2nd International Conference on Knowledge Engineering and Decision Support (ICKEDS 2006), pp: 11-18.
9. Arens, Y., C.Y. Chee and C.N. Hsu, 1993. Retrieving And Integrating Data From Multiple Information Sources. International Journal of Intelligent and Cooperative Information Systems, 2(2): 127-158.
10. Levy, A.Y., 1998. The Information Manifold Approach to Data Integration, IEEE Intelligent Systems, 13(5): 12-16.
11. Motro, A., 1999. Multiplex: A formal model for multidatabases and its implementation, 1999. Technical Report ISSE-TR-95-103, Department of Information and Software Engineering, George Mason University.