A CORBA-based Multidatabase System - Panorama Project

Lou Qin-jian, Sarem Mudar, Li Rui-xuan,
Xiao Wei-jun, Lu Zheng-ding, Chen Chuan-bo

School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan 430074, China.

Abstract: Panorama is a multidatabase system developed in HUST. The project aims to achieve interoperability among existing, heterogeneous, federated database management systems such as Oracle8, Sybase, and DM2 (A database management system developed at HUST, Wuhan, China). This system is based on OMG's distributed object management architecture and it is implemented on top of CORBA compliant, namely VisiBroker, which is used as its infrastructure. Panorama can provide its users a single common data model and a single global query language named PanoSQL, which make it possible to incorporate different databases into the system. The main component of this system are interfaces for the Local DBMSs that participate in Panorama, a transaction manager, a common data model, a schema integrator, a global query language, and a global query processing and optimization.

In this paper, we, first, discuss the architecture and components of Panorama system. We also discuss the schema integration in this system. And we extend our discussion to the query language, transaction management, and the query processing developed for this system. Finally, a conclusion and the future work for our designed system have been given.

Keywords: Heterogeneity; CORBA architecture; Interoperability; Schema integration; Multidatabase system
CLC Number: TP311.13

0 Introduction

The need for integration of different database systems, making differences between these systems invisible and providing users with a uniform access to all the databases has grown rapidly during the last decade. However, database autonomy and heterogeneity still form a severe bottleneck for the development of effective interoperable information systems. In fact, the heterogeneity exists at three basic levels [1]: the first is the platforms level (i.e., different hardware, different operating systems, and different communication protocols). The second is the database management level (i.e., different database management systems based on different data models and languages). And the third level of heterogeneity is that of semantics (i.e., schema and data conflicts).

An advanced way of achieving interoperability among heterogeneous database is through a multidatabase system. A multidatabase system allows its users to simultaneously access autonomous, heterogeneous database systems using a single data definition and manipulation language.

Several approaches have been proposed to address the issues of integrating heterogeneous database systems and to achieve the interoperability among them. In the earliest prototypes, multidatabase systems are able to provide interoperability only among DBMSs and cannot handle repositories, which do not have DBMS capabilities. Recently, the use of CORBA architecture as an infrastructure of a multidatabase system has made it possible to provide interoperability of a multidatabase system with other repositories that do not have DBMS capabilities [2].

We've implemented a multidatabase system, called Panorama on a top of the latest CORBA compliant ORB, namely VisiBroker. At its first step, this system aims to achieve interoperability among different heterogeneous, distributed DBMSs such as Oracle8, Sybase, and DM2. And this paper introduces our developed system –Panorama.

Received date: 2001-04-07
Foundation item: Supported by the National High Performance Foundation of China.
Biography: Lou Qin-jian (1962-), male, Ph. D Candidate, research interests: multidatabase systems. E-mail: rxli2001@hotmail.com

1 The Architecture and the Components of Panorama

Fig. 1 illustrates an overview of Panorama architecture. As a multidatabase system, this architecture contains the following components:

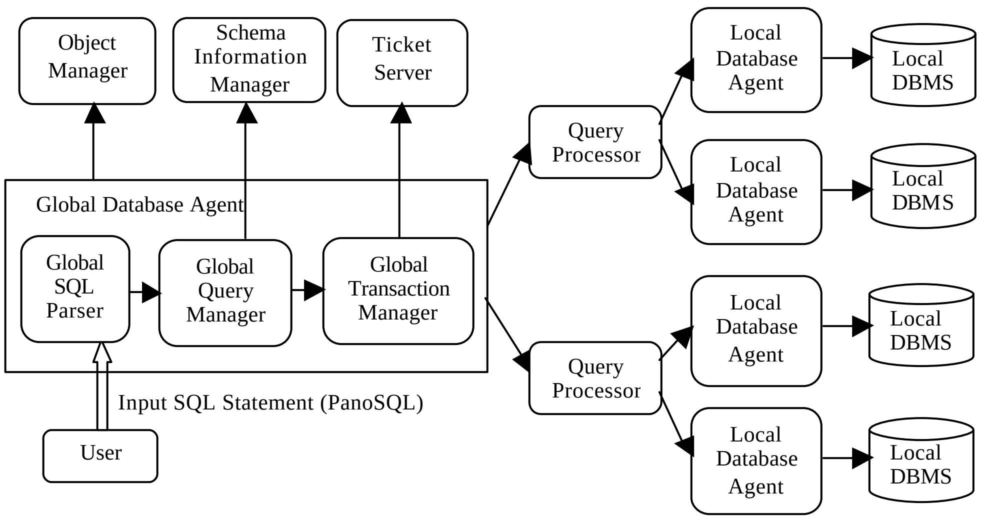


Fig. 1: An overview of Panorama multidatabase system's architecture.

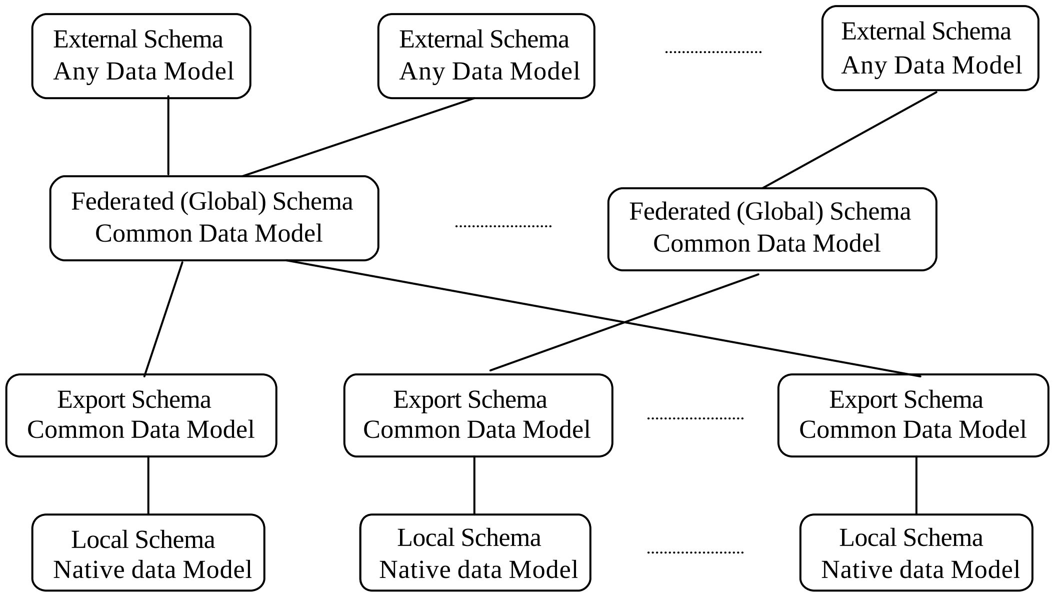
1) **Global Database Agent (GDA):** The GDA contains the following three parts:
   a) **Global SQL Parser (GSP):** The GSP is used to ensure the correctness of the SQL statements issued to the *Global Query Manager (GQM).*
   b) **Global Query Manager (GQM):** The GQM is responsible from the decomposition of global queries into subqueries. It also performs query optimization according to the information obtained from the *Schema Information Manager*.
   c) **Global Transaction manager (GTM):** The GTM is used to ensure the serializability of multidatabase transactions without violating the autonomy of local databases.

   These servers (GSP, GQM and GTM) are started on demand by the ORB using the information provided by the ORB administrator.
2) **Local Database Agents (LDAs):** Each LDA is in fact a connection between a client and the LDBMS server. This means that the LDA is an interface to the LDBMSs. The LDA is responsible from maintaining export schemas provided by the local DBMSs represented in the canonical data model. Also, it is responsible from translating the queries received in the global query language to the local query languages.
3) **Query Processor (QP):** The QP is responsible for query processing. It permits parallel execution of subqueries and this improves the performance of the multidatabase system. It also performs the necessary operations (such as join, outer join, and union) for processing partial results coming from the local DBMSs in order to get the final result of the global query. The QP is started by the ORB, as a result of a request from the GQM indicating that two partial results to be processed together are ready at the LDAs.
4) **Object Manager:** The function of this server is to create objects such as LDA and QP object. Once this server is started, it creates an object of its own implementation. It also can activate, deactivate, and delete these objects.
5) **Ticket Server:** This server provides a globally unique, monotonically increasing ticket number at each time it receives a request. These unique ticket numbers are used by the GTM to ensure the serializability of the multidatabase transaction. The ticket server is started at the initialization of the multidatabase system and it continues to serve the whole multidatabase system continuously.
6) **Schema Information Manager:** this server provides and manages the global schema information necessary for decomposition of global queries into subqueries. It is also started at the initialization phase of MDBS system and stays alive during the lifetime of the system.

In this architecture, when a global query is issued by a user to access multiple databases through the Panorama multidatabase system, first, a GSP, GTM and a GQM object are created by the Object Manager. The GSP object checks the correctness of the global query and sends it to the GQM. The GQM object obtains global schema information necessary for the decomposition of the global query from the Schema Information Manager object. Then, the GQM object decomposes the global query into global subqueries that are sent to the GTM object, which is responsible from the correct execution of global transactions. The GTM object controls the submission of global subtransactions to the LDA objects through the ORB. Next, the LDA objects control submission of operations to the LDBMSs and communicate with GTM object and GQM object to achieve atomic commitment of global transactions. Finally, the partial results returned by the LDA objects are processed by the QP after a query processor object has been created to process them.

2 The Schema Integration in Panorama

For Panorama, 4-level schema architecture is implemented to address the requirements of dealing with distribution, autonomy, and heterogeneity of the system (see Fig. 2). These schemas are:



External Schema Any Data Model
External Schema Any Data Model
External Schema Any Data Model
Federated (Global) Schema Common Data Model
Federated (Global) Schema Common Data Model
Export Schema Common Data Model
Export Schema Common Data Model
Export Schema Common Data Model
Local Schema Native data Model
Local Schema Native data Model
Local Schema Native data Model

Fig. 2: The 4-level of schema architecture for Panorama.

I) **The local schema:** this is the conceptual schema of each component database in Panorama system. It is expressed in the native data model of the component database; thus, the local schemas of different component databases may be expressed in different data models.

II) **The export schema:** this schema is derived by the translation of local schemas into a common data model. Since CORBA is used as Panorama's infrastructure, the canonical (common) data model is based on the ODMG-93 data model, which is powerful enough for supporting both relational and object-oriented technologies.

III) **The federated (global) schema:** is created by the integration of multiple export schemas. It includes the information of the mappings that generated when export schemas are integrated.

IV) **The external schema:** For customization or access control reasons, an external schema is created, in Panorama, to meet the needs of a specific group of users or applications. In this schema, additional integrity constraints can also be specified.

We've used a classification framework of the possible conflicts between export schemas in Panorama. In this classification, two types of conflicts have been included:

a) **Semantic conflicts**, which occur when the same concept is interpreted differently in different component databases.

b) **Structural conflicts**, which occur when the same concept is represented by different constructs of the data model, or although the same concept is modeled by the same constructs, the constructs used have either different structure (missing or different relation/dependencies) or different behavior (different or missing operations).

According to the classification framework, the schema translation and schema mapping are generated to translate the local schemas into the export schemas which are, then, integrated to form the global schema. Different external schemas are also defined for different groups of users.

3 PanoSQL Query Language for Panorama

In a multidatabase system, some of the key features required of a language for interoperability among its local DBMSs are as following [3]:

1) The language must have an expressive power that is independent the schema with which a database is structured.
2) To promote interoperability, the language must permit the restructuring of one database to conform to the schema of another.
3) The language must be easy to use and yet sufficiently expressive.
4) The language must provide full data manipulation and view definition capabilities, and because of the importance and the popularity of SQL in the database world, this language must be downward compatible with SQL in the sense that it must be compatible with SQL syntax and semantics.
5) Finally, the language must admit effective implementation.

So, for Panorama multidatabase system, we've designed a global query language called PanoSQL, which meets the above criteria. PanoSQL is extended from the standard SQL-92 query language, and thus support Oracle8 and Sybase Open Client. It provides constructs that perform queries involving several databases at the same time. PanoSQL statement can be decomposed to sub-statements that can be submitted to the relevant DBMSs and executed without any translation. Fig. 3 shows the architecture of implementing PanoSQL in Panorama.

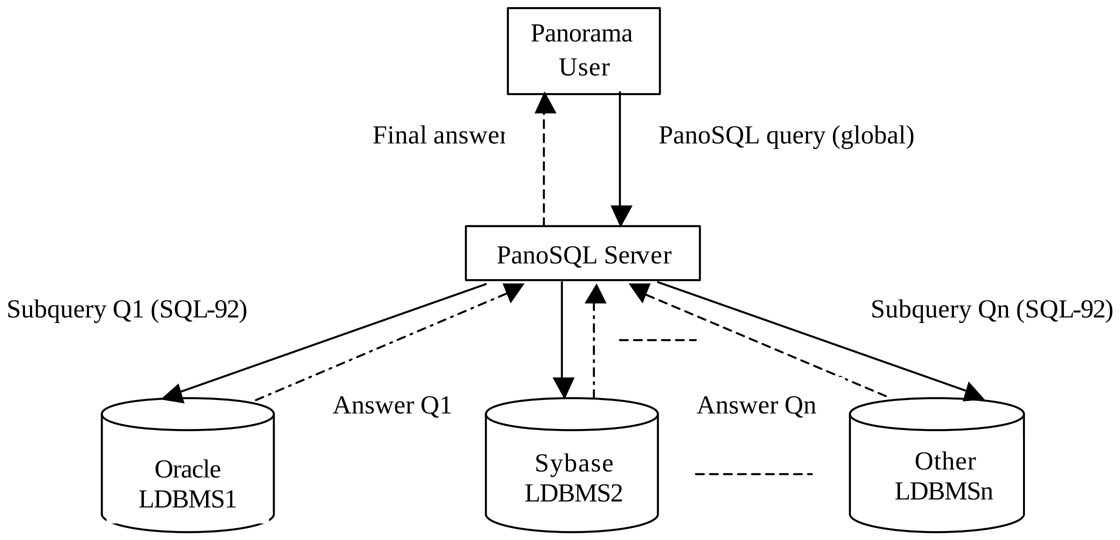


Fig. 3: PanoSQL – Implementation architecture in Panorama system.

In this architecture, the global PanoSQL queries sent by a Panorama user are submitted to the PanoSQL server. According to the Schema Information Manager implemented in Panorama, this server determines a series of local SQL-92 queries and submits them to the local databases. The answers of these local queries are sent back to PanoSQL server, which can produce the final answer of the global PanoSQL queries. In the following, we summarize some of PanoSQL features:

a) PanoSQL supports simultaneous manipulation of tables in different schemas.
b) This multidatabase language extends a standard DB language (SQL-92) so; the standard queries are executed according to their usual semantics.
c) It allows direct reference to local database objects, as well as to export schema objects that may have been defined for the sake of joining the federation.

d) Using PanoSQL, the location transparency is not enforced; i.e., a global naming scheme may be used to refer to distributed objects.
e) The multiquery processor has access to relevant data dictionary information from each LDBMS, either directly through queries or by keeping private views of the LDBMSs schemas.
f) The semantic of the multiquery are defined as a context, which extends the LDBMS object's name space by providing new names for Panorama multidatabase objects.
g) PanoSQL provides enough expressivity to allow for semantic conflict resolution at query definition time.

The above features and the implementation architecture of PanoSQL show that PanoSQL provides a great facility for interoperability among the different local DBMSs implemented in our system.

4 The Transaction Management in Panorama

In Panorama multidatabase environment, two types of transactions are supported (See Fig. 4):

I) **Local Transactions,** those transactions that are executed by the local DBMS (Oracle8, Sybase, etc.), outside of Panorama control.
II) **Global Transactions,** those transactions that are executed under Panorama control.

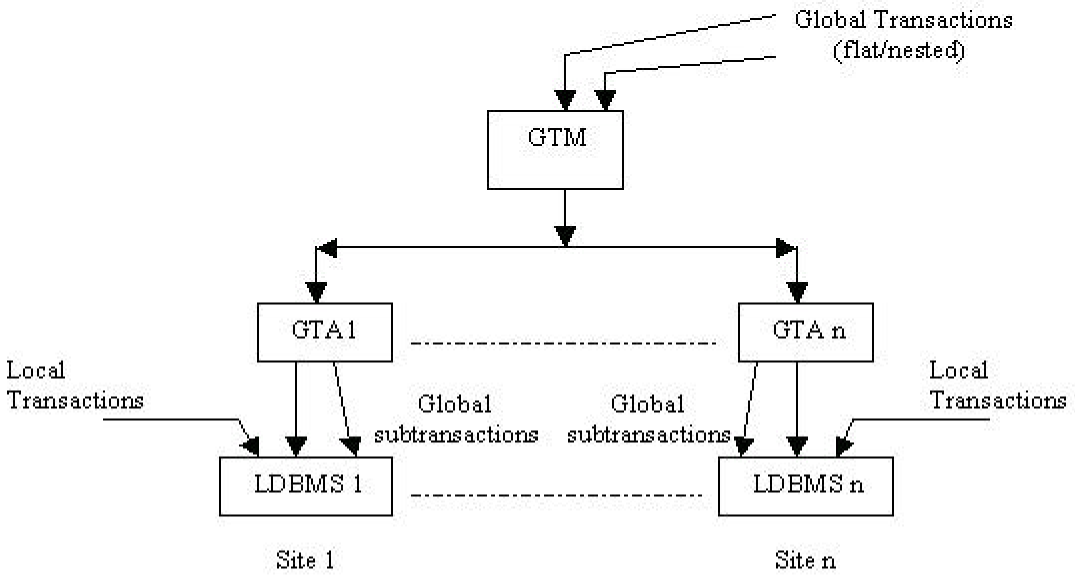


Fig. 4: Panorama transaction management model.

In Panorama project, the transaction manager implemented on top of CORBA compliant, VisiBroker has used the NTNT technique [4] since it is sufficient for ensuring global serializability of both flat and nested transactions submitted to the system. Currently, only Sybase supports nested transactions. Therefore, the parts of a global transaction submitted to Sybase server can be nested transactions; the others must be flat transactions. But, we are planning to implement nested transaction over Oracle8 and DM2 as a future work using the NTNT technique.

5 Processing and Optimizing the Global Queries in Panorama

There are four major steps in processing an MDBS query [5]:

1) **Compilation:** First, an MDBS query is compiled and transformed into an internal form.
2) **Decomposition:** In this step, the integrated data items are replaced with corresponding local data items, along with necessary inconsistency resolution functions.
3) **Optimization:** Here, the query tree obtained from the decomposition is optimized and analyzed, and all subtrees local to a single LDBS are identified.

4) **Translation:** In the final step, an executable query tree is constructed by translating each identified subtree into a subquery that is executable by the underlying LDBMS.

The compilation and translation steps are either straightforward or similar to those of traditional database system. So, the problem of query processing in a multidatabase system can be reduced to the decomposition and optimization steps.

a) **Query decomposition:** In a multidatabase system, a query is expressed in the global query language against the global schema. Since the global query may need data from various local DBMSs, it is necessary to decompose the global query into subqueries such that the data needed by each subquery are available from one local DBMS. While decomposing the global query into subqueries, the operations (such as: join, projection, selection, union etc.) that need to be performed to merge the results of these subqueries are also determined [6]. Consequently, when a global query is issued to the system, it is sent to the query decomposer. Then, two types of subqueries are produced. One is against the export schemas; the other is the post-processing queries that combine the results returned by the LDBMSs. The global query decomposition process is highly dependent on the schema integration information and on the query decomposition algorithm to be implemented for the query decomposer.

b) **Query optimization:** As we've said, two types of queries are produced by the decomposition of the global query: I) Queries against the export schemas. II) The post-processing queries. Due to the autonomy of local systems, there is no way of optimizing the first type. The queries of the second type combine the results returned by the LDBMSs and only the execution of these queries can be optimized. Thus, after a global query is decomposed, it is executed according to the execution plan dictated by the global query optimizer, which designed to optimize the post-processing queries.

Generally, the optimization part of query processing involves the following operations: I) A suitable internal representation. II) Logical transformation of the query. III) Access path selection of the alternatives. IV) Estimate costs and select best.

It's easy to note that the last operation, estimating the cost of queries, is the problem of global query optimization in multidatabase system. The challenges are due to the schema heterogeneity and local autonomy of the component local systems. The crucial one among these challenges is that some local optimization information, such as local cost parameters, local access methods, and some local table sizes, may not be known or accurately known by the global query optimizer because of local autonomy. One solution to the unknown cost problem [5] is to use a logical cost model that views the cost on the basis of the logical execution of global subqueries. This means that the cost of a given subquery is estimated based on logical characteristics of the LDBMSs, the data, and the subquery. Another challenge to be faced is the reducing of the response time. The response time of MDBS queries can only be improved by performing joins concurrently. Different algorithms have been developed for global query optimization. The previous ones have concentrated in static optimization of post-processing queries. Later, dynamic query optimizations have been developed.

When a global query is issued to Panorama system, it is sent to the query decomposer. Then, two types of subqueries are produced. One is against the export schemas. The other is the post-processing queries that combine the results returned by the LDBMSs. The global query decomposition process is highly dependent on the schema integration information and on the query decomposition algorithm, which implemented for the query decomposer.

Due to the autonomy of local systems, there is no way of optimizing the first type that mentioned above. The queries of the second type combine the results returned by the LDBMSs and only the execution of these queries can be optimized. Thus, after a global query is decomposed, it is executed according to the execution plan dictated by the global query optimizer, which designed to optimize the post-processing queries.

In Panorama, an algorithm for query decomposition is implemented to automatically transform a global query to subqueries corresponding to local databases (LDBMSs). Then, the query optimizer, developed for Panorama, produces a query execution plan according to the algebra optimization, cost-based optimization, and dynamic optimization techniques. In this paper, due to the limitation, we will not discuss the decomposition algorithm and the query optimizer developed for Panorama. After the subqueries are executed according to this plan, query processors execute the post-processing queries to merge the partial results returned from LDBMSs to form the final answer.

6 Conclusion and Future work

Panorama is a multidatabase system developed at HUST to achieve interoperability among existing distributed DBMS such as Oracle8, Sybase, and DM2. Panorama followed the trend of using CORBA architecture as an infrastructure of a multidatabase system. CORBA proved to be a highly effective architecture for the implementation of the communication layer for such systems.

At the moment, Panorama provides interfaces for Oracle8, Sybase, and DM2 database management systems. Also, in Panorama, a global schema, which is defined as the integration of the schemas exported from the underlying databases, has been implemented on the top of the existing system. Panorama provides its users a global query language called PanoSQL, which is extended from SQL-92 standard that supports Oracle8 and Sybase Open Client. And last, a global query processor and optimizer are implemented in this system.

As a future work for Panorama, we're planning to provide a new common data model since the recent one can only support the relational and object-oriented DBMSs. The new model will support other data sources, such as web and file systems, to be registered to our system. Finally, we believe that the experiences that have been learned from this system as an ongoing project can help us to improve the optimization of the query processing in the next step of Panorama.

References

[1] Dogas, A., Dengi, C., Kilic, E., et al. *METU Interoperable Database System*. ACM SIGMOD Record, 1995, 24(3): 56-61.

[2] E. Kilic, G. Ozhan, C. dengi, et al. *Experiences in Using CORBA for a Multidatabase Implementation*. The 6th Int'l Conf. on Database and Expert Systems Applications Workshop presentation, London, 1995, 223-230.

[3] Laks V. S. Lakshmanan, Fereidoon Sadri, Iyer N Subramanian. *SchemaSQL – A Language for Interoperability in Relational Multi-database Systems*. Proceedings of the 22nd VLDB Conference, Mumbai (Bombay), India, 1996, 239-250.

[4] U. Halici, B. Arpinar, A. Dogac. *Serializability of Nested Transactions in Multidatabases*. Proc of the 6th int'l Conf. on Data Theory (ICDT'97), Delphi, Greece, 1997, 321-335.

[5] Ahmed Elmagarmid, Marek Rusinkiewicz, Amit Sheth. *Management of Heterogeneous and Autonomous Database Systems*. San Francisco: Morgan Kufmann, 1999.

[6] Sena Nural, Pinar Koksal, Fatma Ozcan, et al. *Query Decomposition and Processing in Multidatabase Systems*. Object Oriented Database Symposium of the 3rd European Joint Conference on Engineering Systems Design and Analysis, Montpellier, France, 1996, 41-52.