An Architecture for Multidatabase Systems Based on CORBA and XML

Li Bing, Lu Zheng-Ding, Xiao Wei-Jun, Li Rui-Xuan, Zhang Wei and Mudar Sarem

College of Computer Science & Technology
Huazhong University of Science & Technology
WuHan, HuBei, China, 430074
libingyk@public.wh.hb.cn

Abstract

A multidatabase system is an effective approach to implement data sharing and interoperability among many distributed and heterogeneous data sources. In this paper, a CORBA-based architecture model of multidatabase system is firstly introduced. Then, an XML-oriented common data model, named XIDM, is presented. These models conform well to the characteristics of multidatabase systems such as autonomy, distribution and heterogeneity. Panorama, a prototype system implemented based on these models, is introduced at the end of the paper.

1. Introduction

With the rapid development of computer networks, the need for a uniform access to information stored in different databases has grown increasingly during the last decade. In order to solve these problems, the integration of different data management systems is needed, making differences between the existing data management systems invisible and providing users with a uniform and transparent access to all the databases. This integration of several existing heterogeneous database systems and file systems is multidatabase system (MDBS) [1,2]. All these database systems and file systems, which have their own local database management systems，(LDBMS) are called local database systems (LDBS). A MDBS constructs global system management level and provides interface towards global users so those users can access all heterogeneous database systems and file systems transparently.

As a new type of DBS, MDBS have common characteristics of general DBS. For example, a MDBS must have ACID properties of transaction. Furthermore, MDBS have special characteristics such as pre-existence, autonomy, distribution and heterogeneity [3,4]. Distribution means that data are stored in disperse fields that can

intercommunicate. In fact, because LDBSs locate at different nodes of networks, MDBSs are usually distributed. A MDBS is also faced with distributive transaction processing and field transparence, etc. But the LDBSs comprising MDBS are pre-existing. Main problem for MDBS is not to divide data but to achieve schema integration. This also means that some conflicts may exist among schemas and data of each LDBS and integrality restriction defined on each LDBS may be contradictory. So every LDBS doesn't correspond with each other in logic. MDBS must iron out all differences in global so that its users can access consistent and reasonable data.

Autonomy of MDBS means each LDBS is free from the influence or control of others. Even though LDBSs have been integrated into MDBS, their intrinsic application programs are still performed in each LDBS that has its own DBMS to manage data.

The heterogeneity of MDBS exists at two aspects: environment and data. Environment heterogeneity means different data sources have self-governed platforms and use different measures to share data. This includes diverse hardware, different operating systems, different communication protocols, and different request for integrality and security.

Data heterogeneity means different systems describe their data with different manners. In fact, even in the same system, there are varied methods describing their data. Although MDBSs are not always heterogeneous, for most applications, especially MDBS including file systems, heterogeneity is much in evidence. Thereby, it is difficult for heterogeneous MDBSs to achieve query processing and transaction processing.

In a word, an effective and practical MDBS must completely solve problems on distribution, autonomy and heterogeneity. This paper discusses some implement technologies based-on CORBA (the Common Object Request Broker Architecture) and XML（eXtensible Markup Language）, and then introduces Panorama, a MDBS prototype based on these technologies under development in HUST.

2. MDBS architecture based on CORBA

An architecture model based-on CORBA is applied in Panorama (see figure 1.) [5]. The model consists of three levels: MDBS application level, MDBS system level and LDBS level. MDBS system level is also composed of two sub-levels:

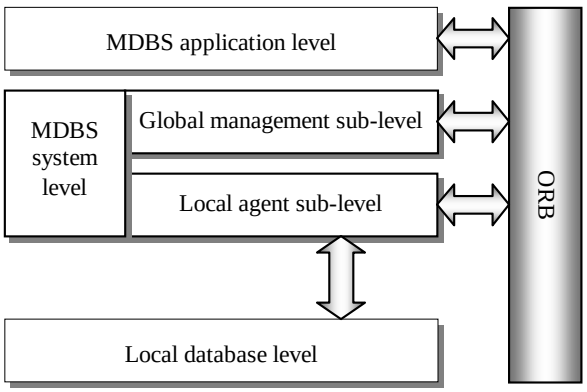

Figure 1. A model of MDBS architecture

global management sub-level and local agent sub-level.

CORBA specification, adopted by the OMG (Object Management Group), provides favorable specification and practical standard for distributed object applications so as to promote development of client/sever architecture. Because CORBA doesn't relate concrete implements in low level but need only to define an interface to high level. Thus, designers need not to deal with a great lot of details about implementation while constructing a MDBS and users can access data through an identical interface. This means CORBA-based model has adequate flexibility, and isn't excessively complex. Because the main aim of CORBA is to achieve distributing computation in heterogeneous environment, some factors such as different connection of networks, different communication protocols and supporting from different operating systems need not take into account. Therefore, through adopting CORBA specification to implement a MDBS, not only problems about distribution and environment heterogeneity can be solved, but also the complexity of the whole system can be reduced. Such MDBSs are able to be more open and scalable.

In this model, MDBS application level includes variant applications that need access to global integrated data. These applications can access data stored in every LDBS through invoking relevant APIs (Application Program Interfaces) to MDBS as if in one DBS. MDBS system level is the core of the whole system. According to corresponding integrated information, global management sub-level decomposes global query requests received from MDBS application level into many local sub-queries that will be sent to local agent sub-level and processes results from local agent sub-level that will be sent back to MDBS application level. Moreover, global management sub-level must manage integrated information and ensure correctness and coherence of global transaction processing.

Local agent (LA) sub-level is composed of many local agents. There is one-to-one correspondence between each LA and LDBS. On the one hand, a local agent sub-level converts every local sub-query request sent from global management level into some format that its LDBS can accept, and then sends these sub-queries to its LDBS. On the other hand, a local agent sub-level converts every result of sub-query into the format required by global management level and returns these results to global management level.

Local database level consists of LDBSs participating in MDBS. Both MDBS application level and global management sub-level communicate with each other through ORB (Object Request Broker), and so do global management sub-level and local agent sub-level. A LDBS communicates with its local agent through various special interfaces provided by the LDBS, such as CLI (Call Level Interface) and ODBC (Open Database Connectivity).

3. Common data model based on XML

3.1. Common data model in MDBS

The main difference between MDBS and traditional distributed DBS is the definition of the global schema. Global schema of traditional distributed DBS, which originated from global logic integration, present global conceptual view.

However, global schema of MDBS, which originated from incompactly integration, expresses the set of the shared data in each LDBS. In other words, data that global users access in the MDBS consists of the shared data in each LDBS, and other private data are provided for local applications. That is to say, the global DBS of a traditional distributed DBS is a union set of each local DBS and the global DBS of a MDBS is a subset of this union set. So a special common data model (CDM) is needed to define global conceptual schema.

In addition, due to the heterogeneity existing between data models of LDBSs, a heterogeneous MDBS must present mappings between concepts in different model. A CDM is usually created so that model of every LDBS can create mappings to the CDM. So CDM is the base of integrating heterogeneous data in MDBS. Currently, while selecting a CDM, two principles must be complied with:

- CDM should be as easy as possible so that it convert with data models of LDBSs.
- Common data language of the CDM should be convenient for expression of data processing.

3.2. XIDM: a common data model oriented to XML

Currently, OO (Object Oriented) model is usually used as CDM of MDBS [6]. However, in order to integrate file systems into MDBS, it is necessary for traditional OO model to be extended. Because data in FS, often called semistructured or non-structured data, [7,8,9] lack explicit structure and store with meta-data, a CDM that can be use to integrate FS should has capability of describing structures of multiple files. In fact, a powerful common data model is not only the foundation on which export schemas are built, but also an advantage to query component systems efficiently.

XML（eXtensible Markup Language）is a meta markup language extending HTML greatly[10]. People can define their own set of tags and make it possible for other parties (people or programs) to know and understand these tags. This makes XML much more flexible than HTML. Furthermore, as a self-describing language, XML orient content of data completely. XML can describe various data structures such as linear list, tree and graph. So XML is becoming a general specification of data interface among various application systems.

At present, a valid XML document must comply with the constraints expressed in an associated document type declaration (DTD) that restrict the logical structure of the document as a grammar definition and support the use of predefined storage units. DTD is similar to the schema of a traditional database in many aspects. It is helpful to add structure on semistructured data in various applications. However, there are some defects in DTD:

- Grammar of DTD is special different from XML;
- DTD provides limited facilities for applying datatypes to document content;
- DTD doesn’t support namespace;
- Content model in DTD is not an open model.

Accordingly, a new content model named XML schema [11,12], which is itself represented in XML_1.0, provides a superset of the capabilities found in DTDs for specifying datatypes on elements and attributes. This means that document

authors, including authors of traditional documents and those transporting data in XML, can achieve a higher degree of type checking to ensure robustness in document understanding and data interchange. A schema can be viewed as a collection (vocabulary) of type definitions and element declarations whose names belong to a particular namespace called a target namespace. The target namespace enables us to distinguish between definitions and declarations from different vocabularies. XML Schema also enables us to indicate that any attribute or element value must be unique within a certain scope.

Explicit structure of a XML document is a tree, but depending on some attribute types XML can represent graph structure. Consequently, we provide a data model, called XIDM (XML-based Integrating Data Model), which bases on XML schema and serves as a common data model for integrating data in file systems. Next, we give several simple definitions useful for describing XIDM.

Comments:

1) An XML graph may be generated not only from parsing a XML document, but also from transforming or querying an existing XML graph.

2) XIDM is an ordered model. Because DTD or XML schema can govern the appearing order for elements and sub-elements, nodes in an XML graph can be arranged in order. A order model has many advantages such as more complex semantic expression and more exact querying. Provided no order in DTD or XML schema, an manual order could be assigned to the XML graph.

3) XML graph does not have a unique representation as an XML document. It is obvious for the XML graph whose XML document doesn't assign a special order in its DTD or XML schema. Although element order is given by DTD or XML schema, on account of different search strategy for selecting successor node such as depth-first or breadth-first, XML graph is different also.

4. Implement of the Panorama

Architecture of Panorama is based on above models (see figure 2.). Functions of each component in the system are explained as follows:

4.1. Schema Information Manager (SIM)

4-level schema architecture is implemented in Panorama MDBS:

- **Local schema:** Local schema is expressed in the native data model of the local database. Thus, the local schemas of different local databases may be expressed in different data models, such as OO model, relation model, etc. If local system is file system that lacks schema, a schema based on XIDM can be added to the file system.
- **Export schema:** For each local database, Panorama provides tools to translate the parts of its local schema into a schema expressed in XIDM automatically, which is called the export schema. This translating process creates a mapping between class of local schema and class of export schema.

- **Global schema:** A global schema, created by the integration of multiple export schemas and based on XIDM, presents mapping information about distribution of global data. Query vs. global schema can be transmitted to corresponding export schema.

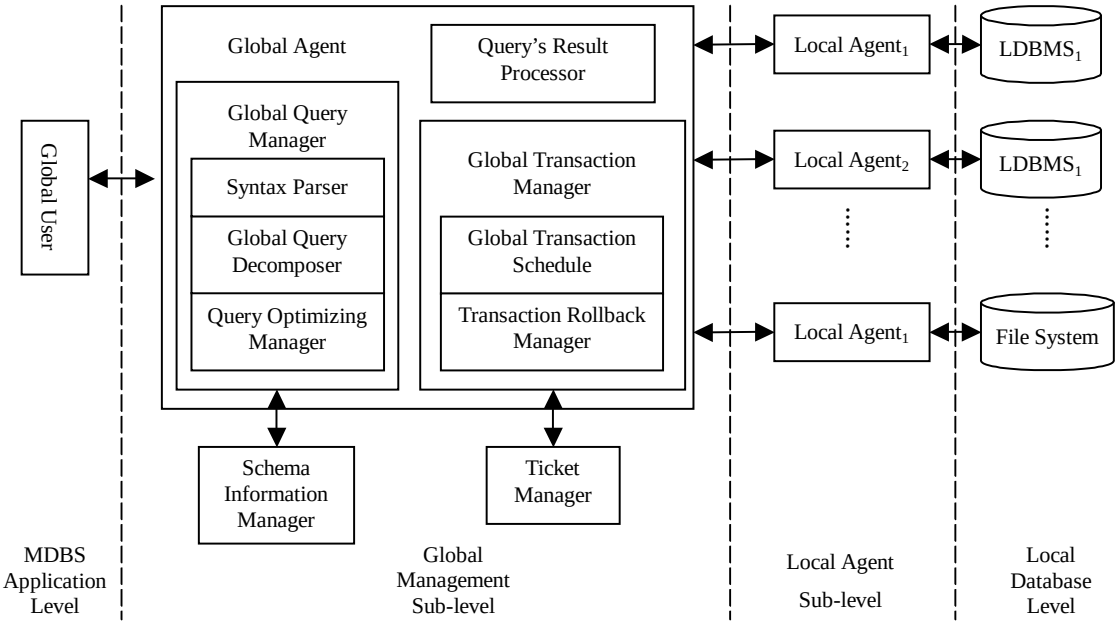

Figure 2.Panorama's Architecture

- **External schema:** For customization or access control reasons, an external schema is created to meet the needs of a specific group of users or applications.

Schema Information Manager integrates export schemas into a global schema and provides and manages the global schema information necessary for decomposition of global queries into sub-queries. SIM is started at the initialization phase of MDBS system, present a series of serves and stays alive during the lifetime of the system.

4.2. Ticket manager

Because serializable is the guarantee of the correct execution for parallel transactions, [13,14,15] arithmetic for concurrent control about MDBS transactions must ensure that execution for transactions possesses global serializable. In MDBS, nesting transaction and flatness transaction differ in correctness. Apropos of nesting transaction, it is necessary for not only global transactions but also sub-transactions in every LDBS to be consistent in executive sequence.

Panorama adopts nesting Ticket arithmetic for nesting transaction. The arithmetic set a Ticket item at every field, which decides the order in which local sub-transactions of global transaction are executed in LDBS. [16,17] Ticket is logic sign according to time and stored as general data in LDBS. Each sub-transaction of global transaction must read and increase value of the Ticket, then writes the vale into LDBS. Each value of the Ticket with operation toward it is committed to local concurrent control. Because of Ticket, order of

global transaction is introduced into LDBS so that local concurrent control protocols ensure global serializable. The ticket manager is started at the initialization of the MDBS and it continues to serve the whole MDBS continuously.

4.3. Global Agent (GA)

Panorama system creates a global agent object (GAO) when a global client creates a connection with MDBS. GAO reseives global query requests from client, creates global query manager object (GQMO) and global transaction object (GTMO) to process global query statement and return final results to client. Because system ensure global serializable, each GAO, which is executed concurrently, increases the parallelism degree of executing global transaction.

GQMO contains the following three parts:

Global query Syntax Parser (GSP): GSP compiles query statement and stores the result into special data structure. These results are sent to global query decomposer.

Global Query Decomposer (GQD): According to global schema information obtained from requesting serve of SIM, GQD decomposes global queries into sub-queries and prepares execution plan for query. All sub-queries are sent to query optimizing manager.

Query Optimizing Manager (QOM): QOM optimizes the execution plan for query according to scheduled optimizing strategy.

In query processing, results produced by each sub-query are called middle results waiting for processing. When plural middle results are generated, GQMO creates an object named query's result processor to combine these results in one result sent to GQMO. These servers (GSP, GQD and GQM) are started on demand by the ORB using the information provided by the ORB administrator.

Every transaction needs a GTMO that obtains Ticket from Ticket manager and schedules both global transactions and sub-transactions according to Ticket. After transactions processing, GTMO that sends sub-queries to local agent commits reports about results of transactions processing to global agent. In addition, in case of failure of global transactions GTMO requests to create a transaction rollback manager to undo any changes made in response to the global sub-transactions.

4.4. Local Agent (LA)

In MDBS, all material database operations are executed by LDBMSs. As interface between global management level and LDBS, local agents ensure autonomy of each LDBS while achieving global transactions. In addition, LA maintains export schemas presented by LDBS, translates receiving sub-queries into expression of query language in local system and sends results of query to GQMO after transforming these results into CDM.

Panorama achieves LA with special interface of database. [18] In this way, not only common data types and public characteristics of all LDBS but also special data types of each LDBS can be supported well. Private characteristics of characteristics, such as data models, stored methods, transaction protocol, concurrent control and user interfaces, are fully used so that advancing efficiency of the MDBS. It is in fact easy for file systems to achieve LA.

5. Conclusion and Status

In this paper, we have provided an effective implement technology based on CORBA and XML, which can be used to integrate various database systems and file systems, which are distributed, autonomic and heterogeneous. By this technology, we've developed a MDBS prototype named Panorama into which some database systems such as Oracle, Sybase and DM2 (a database management system developed at HUST, Wuhan, P. R. of China.) can be integrated. For the moment, Panorama processes ability to achieve essential transaction processing and support general operations such as query and modifying. Currently, we are extending the global query language that supports query languages of Oracle8 and Sybase to support XML.

References

[1] Bright M.W. et al, "A taxonomy and current issues in multidatabase systems", IEEE *Computer*, 1992, 25(3), pp. 50-59.

[2] Shirley A.Becker, Rick Gibson and Nancy L.Leist, "A Study of a Generic Schema for Management of Multidatabase Systems", *Journal of Database Management*, 1996, 7(4) pp. 14-20.

[3] Sheth, A.P. and Larson, J.A., "Federated Database System for Managing Distributed, Heterogeneous, and Autonomous Database", ACM *Computing Surveys*, 1990, 22(3) pp. 183-236.

[4] Witold Litwin, Leo Mark, Nick Roussopoulos, "Interoperability of Multiple Autonomous Database", ACM *Computer Survey*, 1990, 22(3) pp. 267-293.

[5] Object Management Group. "The Common Object Request Broker: Architecture and Specification Revison 2.2", OMG Doc. 1998,2.

[6] Evaggelia Pitoura, Omran Bukhres, Ahmed Elmagarmid, "Object Orientation in Multidatabase systems", ACM *Computing Surveys*, 1995, 27(2) pp. 141-195.

[7] Serge Abiteboul. "Querying semistructured data", In *Proceedings of 6th International Conference on Database Theory (ICDT)*, Delphi, Greece, 1997 pp.1-18.

[8] Paolo Atzeni, Giansalvatore Mecca, Paolo Merialdo. "Semistructured and Structured Data in the Web: Going Back and Forth", *SIGMOD Record*, 1997, 26 (4) pp. 16-23.

[9] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. "Lore: A Database Management System for Semistructured Data", *SIGMOD Record*, 1997,26(3) pp. 54-66

[10] W3C, Extensible Markup Language (XML) 1.0 (Second Edition), http://www.w3.org/TR/2000/REC-xml-20001006

[11] W3C, XML Schema Part 1: Structures, http://www.w3.org/TR/2000/WD-xmlschema-1-2000 0922/

[12] W3C, XML Schema Part 2: Datatypes, http://www.w3.org/TR/2000/WD -xmlschema- 2-20000922/

[13] U. Halici, B. Arpinar, and A. Dogac. "Serializability of Nested Transactions in Multidatabases", In *Proceedings* of *6th International Conference on Database Theory (ICDT)*, Delphi, Greece, 1997 pp. 321-335.

[14] Kung,H. T. and Lehman et al , "Concurrent manipulation of binary search trees", ACM *Transactions on Database Systems (TODS)*,1980, 5(3) pp. 354-382.

[15] Jean-Marc Cadiou, "On semantic issues in the relational model of data", In *Proceedings of 5th Mathematical Foundations of Computer Science (MFCS)*, Gdansk, Poland, 1976, pp.23-38

[16] D. Georgakopoulos, M. Rusinkiewicz, and A. P. Sheth. "Using Tickets to Enforce the Serializability of Multidatabase Transactions", IEEE *Transactions on Knowledge and Data Engineering*, 1994, 6(1) pp. 166-180.

[17] Lu Zheng-ding, Xiao Wei-jun, Yi Tong, Li Rui-ruan, "A Multidatabase Transaction Model Based on Integrated Database Systems and File Systems", *Journal of Huazhong university of science and technology*, 1998 , 26(6) pp. 66-68,75.

[18] Lu Zheng-ding, Li Rui-ruan, Xiao Wei-jun. "Design and Implementation of the Local Agent in a Mulitidatabase Transaction Model", *Computer research & development*, 1998, 35(12) pp.1130-1134