Schema Mapping for Interoperability in XML-Based Multidatabase Systems

Ruixuan Li Zhengding Lu Weijun Xiao Bing Li Wei Wu
College of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, 430074, P. R. China
rxli@public.wh.hb.cn

Abstract*

The main aim of a multidatabase system (MDBS) is to achieve data integration and interoperability among distributed and heterogeneous database systems. But data model heterogeneity and schema heterogeneity make this a challenging task. Multidatabase users can only view the global schemas whose real data come from local database systems. Thus, mappings from global schemas to local schemas should be established. At the same time, the recent emergence of XML has shown great attractability as a new standard for data representation and exchange on the web. Data model based on XML becomes more suitable for integrating different types of systems. This paper firstly introduces a multidatabase common data model based on XML, named XML-based Integrated Data Model (XIDM). Then, an approach of schema mappings based on XIDM in MDBSs has been presented. Finally, the illustration and implementation of schema mappings in a multidatabase prototype – Panorama are also discussed.

Key words: *Interoperability, Schema mapping, Integrated data model, Multidatabase systems, eXtensible markup language (XML)*

1. Introduction

Nowadays, due to the increasing development of web data management and collaborative work, the traditional single database system cannot satisfy the requirements of data sharing and interoperation in web environment. At the same time, the existing databases cannot be discarded totally, so developing some collaborative software that can access and process the data of multiple pre-existing databases is a necessary trend. A multidatabase system (MDBS) is a layer of software that can integrate a collection of pre-existing, heterogeneous, distributed database systems called local database systems (LDBSs) [1]. It mainly solves the problem that how to achieve the schema integration and data interoperability among multiple LDBSs. Due to the data model heterogeneity and schema heterogeneity, there may exist kinds of differences and conflicts in different local database schemas. In order to access the data in these multiple database systems transparently, an approach that can reduce these differences and resolve these conflicts must be needed [2]. A general way of integrating these schemas is through a global schema that is constructed by schema transformation from local schemas of participating database systems. Global schema is a virtual knowledge base in MDBSs. Multidatabase users can only query the global schemas whose data come from local database systems. So, schema mappings from global schemas to local schemas should be employed.

To represent and deal with multidatabase export schemas and global schemas, common data model is needed. In the old fashion multidatabase systems, such as Pegasus [3], Interbase [4], CORDS [5], MIND [6], relational data model, object-oriented model and object relational model played the most important roles. The growing popularity of the web has increased the need to search, display, manipulate and exchange information among different data sources, including database systems and file systems. An attempt to achieve this through data standardizing representation gives rise to the advent of XML (eXtensible Markup Language) techniques. As a data description language, XML can describe not only structured data, but semi-structured data. Thus, a common data model based on XML is a better selection for MDBSs.

Researches on data models and heterogeneous information integration over the last several years have focused on handling semi-structured data, mapping between XML and relational databases, and querying web data or XML models, such as OEM model and LoreL query language in Lore [7], YAT XML algebra and YATL query language in YAT system [8], XML data model and XMAS query language in MIX mediator system [9], X-Ray approach for integrating XML with relational database systems [10]. These researches have made great contributions to modeling semi-structured or XML data, storing and managing semi-structured data, wrapping relational data sources as XML views, and extending relational algebra for efficient query evaluation in XML integration systems. However, most of these existing research efforts haven't shown how to integrate

* This work was supported by National High Performance Computing Foundation of China under grant 99319.

these different data sources into one common data model and make a comprehensive approach of schema mapping between the common data model and other pre-existing data models. In our approach, besides providing a common data model based on XML (named XIDM, XML-based Integrated Data Model) to integrate different data models in MDBSs, we present a methodology for schema mappings between XIDM model and other data models, such as relational data model, object-oriented data model, and even file systems.

The remainder of the paper is organized as follows. Section 2 gives the motivation of schema mapping in MDBSs. Section 3 introduces an XML-based Integrated Data Model (XIDM) as the common data model for MDBSs. Section 4 presents an in-depth analysis and description of schema mappings in MDBSs. The illustration and implementation of schema mappings in Panorama, a MDBS prototype developed in HUST, are discussed in Section 5. Finally, Section 6 concludes the paper with a summary and gives an outlook to the future work.

2. Motivation

Before beginning the detailed discussion, we motivate the problem of schema mapping for interoperability in MDBSs. In most of the existing MDBSs, 4-level schema architecture is introduced [1]. These four schema levels are:

- *Local schema level*: Local schema is expressed in the native data model of the local database, such as relational model, object-oriented model.
- *Export schema level*: For each local database, MDBSs should provide tools to translate the relevant parts of its local schemas into export schemas expressed in common data model. This translation will create mappings between local schemas and export ones.
- *Global schema level*: A global schema based on global data model presents the global views of MDBSs and the mapping information about the distribution of global data.
- *External schema level*: External schemas are created for special group of users or applications.

As described in Figure 1, there are three database layers in MDBSs: global database layer, export database layer and local database layer. Global database, created for multidatabase global users, is a virtual database, containing global schemas and no actual data. Export database is also a virtual one containing export schemas. There are different export schemas corresponding to different local schemas of local database systems. Local databases, in a broad sense, are database systems, file systems or HTML/XML documents, which contain the actual data. Different local databases have different local schemas and representations.

Multidatabase global users can only view global objects in global schema. Since these global objects are empty and the data for global queries can only come from local databases, schema mappings from global schemas to export schemas and from export schemas to local schemas should be developed. But how can we get these schema mappings? As we know, the multidatabase administrator translates local schemas into export schemas, and then integrates these export schemas into global schema. Schema mappings could be drawn out during the translation and integration. Figure 1 illustrates the integration process and schema mapping architecture. In an MDBS, there are two types of schema mappings: global mapping (GM) and local mapping (LM). The global mappings, a portion of the global schema, relate global objects in global database with export ones in export database, while the local mappings in each export schema relate its export objects with corresponding local objects.

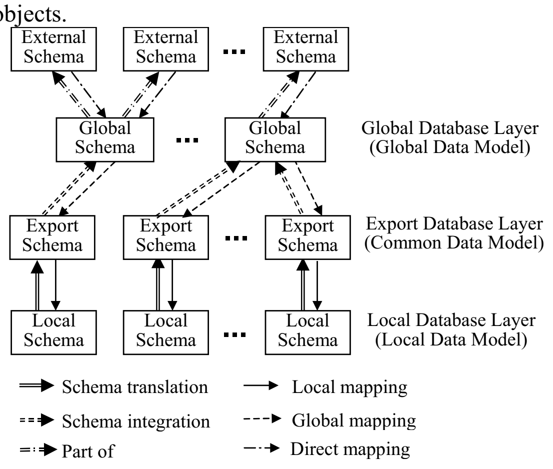

Figure 1. Schema mapping architecture in MDBSs

3. XML-based Integrated Data Model

Most of the existing MDBSs are using object-oriented model as their common data model [1]. However, with the advent of an increasing of applications in various requirements, it is necessary for traditional object-oriented model to be extended to integrate multiple heterogeneous data sources. XML, a meta markup language extending HTML greatly, is now fast emerging as the dominant standard for representing various kinds of data, especially for web-based information systems. As a self-describing language, 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. A data model, called XML-based Integrated Data Model (XIDM), which is based on XML and serves as a common data model for integrating and interoperating heterogeneous data sources in multidatabase prototype - Panorama system, is presented as follows.

Definition 1. In XIDM model, an Element Cluster (EC) is an aggregation of all the elements with the same description using Document Type Definition (DTD) or XML Schema.

Element cluster, which models a type of elements, is similar to the concept of class in object-oriented methodology, while a concrete element is similar to an instance (i.e. object) of a class. Here, we introduce the concept of element cluster because the global schemas in MDBSs only include the element definitions, not the real element contents. There are two types of element clusters in an MDBS, global element cluster (GEC) and export element cluster (EEC), which correspond to global schemas and export schemas respectively.

Definition 2. In XIDM model, an XML document is modeled as a labeled, ordered graph named XML graph, which is a pair $G = <Vertex, Edge>$, where Vertex is a set of nodes and Edge is a set of edges.

Definition 3. Each node in graph G, which represents an element cluster, is a group $EC = <K, A, S, Q, M>$, where K is a list of key attributes, corresponding to either ID attribute of XML element in DTD or identity constraint definition components in XML Schema, A is an ordered list of attribute names, S is an ordered list of sub element clusters, Q is the set of qualifications forced on the element cluster EC, and M is the set of schema mappings for element cluster EC.

Definition 4. Each edge in graph G is a group $B = <EC1, EC2, Label>$, where EC1 is the starting element cluster, EC2 is the ending element cluster and Label is the symbol on the edge.

In an XIDM graph, edges are directed and labeled. Edges are classified as two types: tag edges and reference edges. A tag edge, labeled by the tag of the child element cluster, represents nested relationship between the element cluster and its child element cluster, pointing to the child from its parent. A reference edge, generally labeled by the referencing attribute (named keyref attribute) of the element cluster, represents reference relationship between different XML element clusters, pointing to referenced element cluster from referencing element cluster.

4. Schema Mapping in MDBSs

4.1. Global Mappings

Schema mappings in MDBSs will enhance the interoperability among different database systems. In Panorama, the same data model, XIDM, was hired to describe global schema and export schema. Thus, global mappings between global schema and export schema don't need to deal with schema translation between different data models. There are two types of global mappings: horizontal mapping and vertical mapping.

Definition 5. The horizontal mapping *HM* of a global element cluster $GEC = <K, A, S, Q, M>$ is an operation, which transforms *GEC* into a group of export element clusters, $EEC_1(K_1, A_1, S_1, Q_1, M_1), \ldots, EEC_n(K_n, A_n, S_n, Q_n, M_n)$, according to a set of given conditions $P_1, \ldots, P_n$, where (1) $K_1 = K_2 = \ldots = K_n = K$; (2) $A_1 = A_2 = \ldots = A_n = A$; (3) $S_1 = S_2 = \ldots = S_n = S$; (4) $Q_i = (Q \wedge P_i)$; (5) $M_i = LM_i$.

Here, K, $K_1, \ldots, K_n$ are the key attributes of *GEC*, $EEC_1, \ldots, EEC_n$ respectively. $P_1 \vee \ldots \vee P_n = True$, $LM_1, \ldots, LM_n$ are local mappings, and $i \in \{1, \ldots, n\}$.

Definition 6. The vertical mapping *VM* of a global element cluster $GEC = <K, A, S, Q, M>$ is an operation, which transforms *GEC* into a group of export element clusters, $EEC_1(K_1, A_1, S_1, Q_1, M_1), \ldots, EEC_n(K_n, A_n, S_n, Q_n, M_n)$, according to a set of given pairs $<R_1, T_1>, \ldots, <R_n, T_n>$, where (1) $K_1 = K_2 = \ldots = K_n = K$; (2) $A_i = R_i \cup K_i$; (3) $S_i = T_i$; (4) $Q_1 = Q_2 = \ldots = Q_n = Q$; (5) $M_i = LM_i$.

Here, K, $K_1, \ldots, K_n$ are the key attributes of *GEC*, $EEC_1, \ldots, EEC_n$ respectively. $R_i \subseteq A$, $T_i \subseteq S$, $R_1 \cup \ldots \cup R_n \cup K = A$, $T_1 \cup \ldots \cup T_n = S$, $LM_1, \ldots, LM_n$ are local mappings, and $i \in \{1, \ldots, n\}$. To ensure correct reconstruction of global element clusters, the key attributes should be mapped into each export element cluster.

The horizontal mapping defines how global element cluster will be breadthwise mapped into export element clusters via a set of given conditions, while the vertical mapping dose that lengthwise on attributes and sub element clusters. Global schema mapping in MDBSs is similar to partition operation in distributed database system (DDBS), but not quite the same. For instance, vertical partition in DDBS is based on the attributes of the relation, while vertical mapping in MDBS is not only based on the attributes, but sub element clusters.

4.2. Local Mappings

As we have described above, local mappings will be built during the translation from local schemas to export schemas. In Panorama system, we hired object-oriented database system (OODB), relational database system (RDBS) and file systems (especially HTML/XML documents) as the local component systems. We will discuss the local mappings between these local schemas and export schemas in the following paper.

4.2.1. Mappings Between OO Model and XIDM Model. As a local database schema, some concepts in OO model, such as class aggregation (CA), classes, attributes (including key attribute, simple type attribute and complex type attribute), methods and interclass relationships, should be concerned. Table 1 shows the mappings between OO model and XIDM model.

Table 1. Mappings between OO model and XIDM

No	Concepts in OO Model	Concepts in XIDM
(1)	Class	Element cluster (EC)
(2)	Key attribute (or OID)	Key attribute
(3)	Attribute with simple type	Attribute
(4)	Attribute with simple type	Sub-EC
(5)	Attribute with complex type	Sub-EC
(6)	From class to its attribute	Tag edge (for (4), (5))
(7)	Interclass relationship	Keyref attribute
(8)	From class to its relationship	Reference edge
(9)	Key attribute of its superclass	Keyref attribute
(10)	From subclass to superclass	Special reference edge
(11)	Method	Sub-EC
(12)	Parameter of method	Attribute of sub-EC
(13)	From class to its method	Special tag edge
(14)	CA	EC
(15)	From CA to class	Tag edge

4.2.2. Mappings Between Relational Model and XIDM Model. Similar to OO model, database, relations (tables) and attributes (including primary key and foreign key attribute) should be taken into account in relational model (RM). Table 2 shows the mappings between RM and XIDM model.

Table 2. Mappings between RM and XIDM

No	Concepts in RM	Concepts in XIDM
(1)	Relation (Table)	Element cluster (EC)
(2)	Primary key attribute	Key attribute
(3)	Attribute	Attribute
(4)	Attribute	Sub-EC
(5)	From relation to its attribute	Tag edge (for (4))
(6)	Foreign key attribute	Keyref attribute
(7)	Relation referenced by foreign key	Independent EC
(8)	From relation to referenced relation	Reference edge (for (6),(7))
(9)	Foreign key attribute	(None)
(10)	Relation referenced by foreign key	Sub-EC
(11)	From relation to referenced relation	Tag edge (for (9),(10))
(12)	Database	EC
(13)	From database to table	Tag edge

4.2.3. Mappings Between File Systems and XIDM Model. As we know, most of the file systems are semi-structured or non-structured and have no complete schemas. To integrate file system to MDBSs, data pre-processing, which will pick up the user-interested and pivotal data, should be needed. Some frame data, such as "<HTML>""<HEAD>""<TITLE>" in HTML web pages, "<No>" defined by "<! ELEMENT No (#PCDATA)>" in XML documents, can be drawn out as the key and structure information. These structure data, named schema information, will be modeled by XIDM and form a local schema, which will be convenient for integration.

There are many types of file systems, but, at present, Panorama system mainly integrates XML/HTML-based file systems. Because XIDM model itself is oriented to XML Schema, there are almost no needs to deal with the schema transformation when we establish schema mappings between XML/HTML document model and XIDM model.

5. Implementation in Panorama System

In Panorama system, XIDM acts as the common data model and provides a universal data representation for multidatabase global users. In [11] and [12], we gave an architecture for Panorama system based on CORBA and XML, including Global Agent, Schema Information Manager, Ticket manager and Local Agent, and discussed how to register different database systems to Panorama with CORBA. Here, we will give some implemented examples to illustrate schema mappings among different schemas in Panorama system, which can integrate object-oriented database systems, relational database systems and HTML/XML file systems.

Example 1. In a local schema of object-oriented database ObjectStore participating in Panorama, there are two classes, *Staff* and *Department*, which are defined by object definition language (ODL):

```
module Company {
  interface Staff (key No){
     attribute integer EmpNo;
     attribute string Name;
     attribute float Salary;
     relationship Staff Director inverse Staff::Direct;
     relationship Set<Staff> Direct inverse
            Staff::Director;
     relationship Department Dept inverse
            Department::Contain;
     boolean SetSalary(in float NewSal);
  };
  interface Department (key Number){
     attribute string Number;
     attribute string Name;
     attribute string Struct Addr {string Street, string
            City} Address;
     relationship Set<Staff> Contain inverse
            Staff::Dept;
  };
}
```

Here, *Company* is the class aggregation of *Staff* and *Department*. Figure 2 illustrates the export schemas based on XIDM model for the above class schemas, where each node (i.e. element cluster) represented by number 1, 2, 3, …, tag edges denoted by solid line and reference edges denoted by dashed line. Sub element clusters of each node are represented by tag edges and attributes are given by the enclosing curly bracket parentheses. Special tag edges, e.g. *Method::SetSalary*, are used to represent methods of local class. Note that some name transformation should be needed when we translate the local schemas into export schemas. For example, class *Staff* is translated into element cluster *Employee* and attribute *Name* of class *Staff* is transformed into element cluster *EName*.

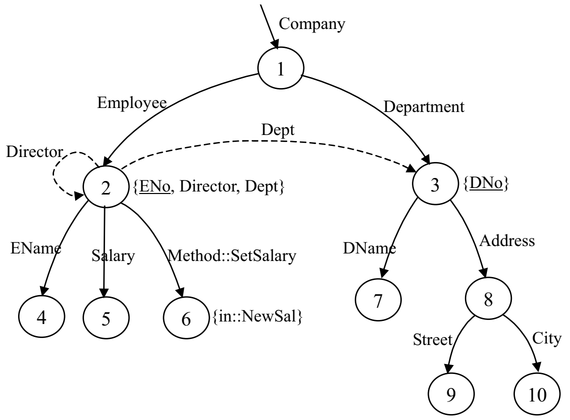

Figure 2. XIDM-based export schemas for class schemas in Example 1

As a demonstration, the schema mappings and qualifications of node 2 and node 4 are shown in Figure 3. "&2" and "&4" are used to represent the local concepts corresponding to node 2 and node 4 respectively. We use double dashed line to denote schema mapping and the symbol on the edge is comprised of local concepts corresponding to the element cluster and its attributes. Qualifications are given via a pair of square brackets. Suppose we just want element cluster Employee to include those staffs whose salary is more than $1000, then the qualification will be "[Salary > $1000]".

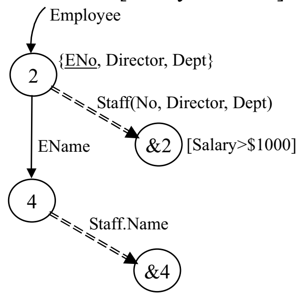

Figure 3. Local mappings for node 2 and 4 in Figure 2

Example 2. In an instance *Corporation* of relational database Oracle, there are three tables: *Person*, *Branch* and *Address*, whose relational schemas are presented in Figure 4 and corresponding XIDM-based export schemas are given in Figure 5. The same with Example 1, we also need to process schema transformation between local schemas and export schemas. For example, the relation *Address* is transformed into sub element cluster *Address* of *Department* because *Address* is referenced by the relation *Branch* through *AddrID*. The way to represent schema mappings and qualifications is similar to Example 1.

Database instance: Corporation

Person				
No*	Name	Age	Salary	Branch^

Branch			
Number*	Name	AddrID^	Manager^

Address		
AddrID*	Street	City

Where "*" denotes Key attributes and "^" represents Foreign attributes.

Figure 4. Relational schemas for database *Corporation*

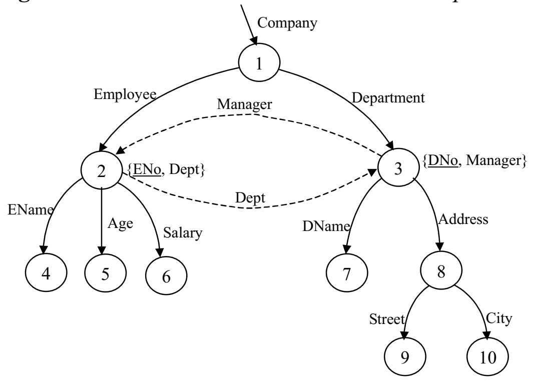

Figure 5. XIDM-based export schemas for relational schemas in Figure 4

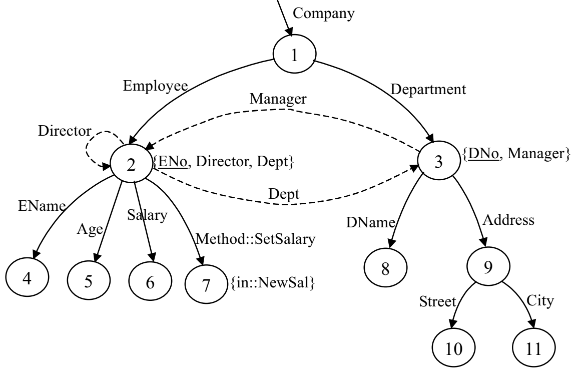

Figure 6. XIDM-based global schemas

After integrating the XIDM-based export schemas represented in Figure 2 and Figure 5, we can get the global schema in panorama system (see Figure 6). The schema mappings for global schema can be obtained through analyzing the local systems during the integrating process. For instance, the global element cluster *Department* may be horizontally mapped into export element cluster *Department* in Figure 2 and Figure 5 respectively via conditions "[DNo < 301]" and "[DNo > 210]" and vertically mapped on "<{DNo}, {DName, Address}>" and "<{DNo, Manager}, {DName, Address}>", while the global element cluster *Employee* will be vertically mapped on "<{ENo, Director, Dept}, {EName, Salary, Method::SetSalary}>" and "<{ENo, Dept}, {EName, Age, Salary}>".

6. Conclusions and Future Work

Schema mappings are part of the schema information in MDBSs. They show how local schemas will be transformed into global schemas through schema translation and integration. On the other hand, they also indicate how global queries will get their result from local database systems. Hence, schema mappings are the basis of global query decomposition and processing in MDBSs. With the rapid development of new protocols, approaches and techniques, more and more application requirements demand new methods for data representation and efficient strategies for information integration. Recently, there has again been great interest in integrating heterogeneous data sources. Since schema mapping is tightly related with common data model in MDBSs, this study firstly presents an XML-based Integrated Data Model (XIDM) as a common data model, which can integrate structured data and semi-structured data. Then, an approach of schema mapping based on XIDM is given. We also show the illustration and implementation of these mappings among global schemas, export schemas and local schemas in a multidatabase prototype – Panorama system.

As a future work for Panorama, we intend to enhance the semantic representation of XIDM model to integrate more complicated data sources and process more complex semantic queries. We are developing a query language complying with XIDM model and designing GUI tools to maintain the global schema information including schema mappings. We are also extending the CORBA-based Panorama system to support Web Services using XML, SOAP and UDDI. Furthermore, we are planning to provide more efficient approaches for query processing, transaction management and security administration in Panorama system coping with the environment distribution, data representation and manipulation heterogeneity and local autonomy. Finally, we believe that the experiences that have been learned from this system as an ongoing project can help us to improve the overall performance and optimization in the next step of Panorama.

References

[1] A. Elmagarmid, M. Rusinkiewicz, and A. Sheth, Management of Heterogeneous and Autonomous Database Systems, Morgan Kufmann publishers, 1999.

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

[3] R. Ahmed, P. De Smedt, W. Du, W. Kent, M.A. Ketabchi, W. Litwin, A. Rafii, and M.-C. Shan. The Pegasus Heterogeneous Multidatabase System. IEEE Computer, 1991, 24(12): 19-27.

[4] O.A. Bukhres, J. Chen, W. Du, A.K. Elmagarmid, R. Pezzoli. InterBase: An Execution Environment for Heterogeneous Software Systems. IEEE Computer, 1993, 26(8): 57-69.

[5] G.K. Attaluri, D.P. Bradshaw, N. Coburn, P.-A. Larson, P. Martin, A. Silberschatz, J. Slonim, and Q. Zhu. The CORDS multidatabase project. IBM Systems Journal, 1995, 34(1):39-62.

[6] A. Dogac, C. Dengi, E. Kilic, G. Ozhan, F. Ozcan, et.al. METU Interoperable Database System, ACM SIGMOD Record, 1995, 24(3): 56-61.

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

[8] V. Christophides, S. Cluet, and J. Sim'eon. On Wrapping Query Languages and Efficient XML Integration. Proceedings of ACM SIGMOD Conference on Management of Data, Dallas, Texas, SIGMOD Record, 2000, 29(2): 141-152.

[9] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based Information Mediation with MIX. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMod-99), SIGMOD Record, New York, June, 1999, 28(2): 597-599.

[10] G. Kappel, E. Kapsammer, S. Rausch-Schott, W. Retschitzegger, X-Ray - Towards Integrating XML and Relational Database Systems, Proceedings of the 19th International Conference on Conceptual Modeling, (ER'2000), Salt Lake City, LNCS, Springer, October, 2000, pp. 339-353.

[11] Li Bing, Lu Zhengding, Xiao Weijun, Li Ruixuan, Zhang Wei, and Mudar Sarem. Architecture for Multidatabase Systems Based on CORBA and XML. In Proceedings of the 12th International Workshop on Database and Expert Systems Application (DEXA'2001), Munich, Germany, September, 2001, pp. 32-37.

[12] Mudar Sarem, Li Ruixuan, Xiao Weijun, and Lu Zhengding. Registering Different DBMSs to Panorama with CORBA. In The International Software Engineering Symposium 2001 (ISES'01), Wuhan, China, appears in Wuhan University Journal of Natural Sciences, March 2001, 6(1-2): 423-431.