Journal of Systems Engineering and Electronics, Vol. 16, No. 2, 2005, pp. 437～444

XML-based integration data model and schema mapping in multidatabase systems*

Li Ruixuan, Lu Zhengding, Xiao Weijun & Wu Wei

College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, P. R. China

(Received October 20, 2003)

Abstract: Multidatabase systems are designed to achieve schema integration and data interoperation among distributed and heterogeneous database systems. But data model heterogeneity and schema heterogeneity make this a challenging task. A multidatabase common data model is firstly introduced based on XML, named XML-based Integration Data Model (XIDM), which is suitable for integrating different types of schemas. Then an approach of schema mappings based on XIDM in multidatabase systems has been presented. The mappings include global mappings, dealing with horizontal and vertical partitioning between global schemas and export schemas, and local mappings, processing the transformation between export schemas and local schemas. Finally, the illustration and implementation of schema mappings in a multidatabase prototype - Panorama system are also discussed. The implementation results demonstrate that the XIDM is an efficient model for managing multiple heterogeneous data sources and the approaches of schema mapping based on XIDM behave very well when integrating relational, object-oriented database systems and other file systems.

Key words: multidatabase systems, common data model, schema mapping, extensible markup language (XML).

1. INTRODUCTION

A multidatabase system (MDBS) is a layer of software that integrates 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. A MDBS allows a global application to access distributed objects located at different types of LDBSs, such as relational databases, object-oriented databases, multimedia databases and even file systems, using a uniform data definition and manipulation language. However, there may exist kinds of differences and conflicts in different local database schemas[2]. In order to access the data in these multiple database systems transparently, an approach can reduce these differences and resolve these conflicts should be needed.

A general way of integrating these database schemas is through a global schema that is constructed through a set of transformations of local schemas in order to reflect the context of local databases[3]. Since any database schema is based on the application requirements of the real world, the global schema in MDBSs should comply with the same situation. That is, only the local database information benefiting the global application requirements will be picked up and transformed. There are four schema levels in a MDBS: local schema, export schema, global schema and external schema[1]. Global schema is a virtual knowledge base for MDBSs. Multidatabase users can only query the global schemas whose data come from LDBSs. So schema mappings from global schemas to local schemas should be developed.

To represent and deal with multidatabase export schemas, a common data model is needed. In the old fashion MDBSs, relational data model, object-oriented model and object relational model played the most important roles[4]. 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 also semi-structured data. Thus, a common data model based on XML is a better selection for MDBSs[5]. In our approach, besides provid-

* This project was supported by the National High Performance Computing Foundation of China (99319) and the National Key Technologies R&D Program of China (2002BA103A04).

ing a common data model based on XML (named XIDM, XML-based Integration 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. Then we give the implementation examples to illustrate the data model and schema mappings in a multidatabase prototype - Panorama system.

2. XML-BASED INTEGRATION DATA MODEL

2.1 Common Data Model

In most of the existing MDBSs, 4-level schema architecture is introduced[1]. These four schema levels are below.

(1) Local schema level 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 relational model, object-oriented model, etc. Similarly, XML documents, whose schema named DTD (document type declaration) or XML schema, are using document-based data model.

(2) Export schema level For each local database, MDBSs should provide tools to automatically translate the relevant parts of its local schemas into those schemas expressed by common data model, which is called the export schemas. This translation creates a mapping between concepts of local schemas and those of export schemas.

(3) Global schema level A global schema, constructed by integrating multiple export schemas and based on global data model, presents the global views of MDBSs and the mapping information about distribution of global data.

(4) External schema level For customization or access control reasons, external schemas are created for special group of users or applications.

The main difference between MDBSs and traditional distributed database systems is how to define the global schema. Global schemas of a traditional distributed database system, derived from global logic integration, give a global conceptual view. However, multidatabase global schemas, coming from loose integration, only express the set of the shared data in each LDBS. In other words, data that global users manipulate in a MDBS consist of the shared data in each LDBS and other private data are provided for local applications. That is, the global schema of a traditional distributed database system is a union set of all local schemas of each LDBS, and the global schema of a MDBS is a subset of this union set[6]. So a special common data model is needed to define global conceptual schema in MDBSs. In general, common data model is oriented to export schema, while global data model oriented to global schema. For example, object oriented model is used for common data model, and relational data model is used for global data model. Here, the same data model is introduced to global and common data model in the following paper for simplicity reason.

In addition, due to the heterogeneity existing among data models of LDBSs, a heterogeneous MDBS should provide mappings among concepts in different models. A common data model is usually created so that translation and mappings between local data models and the common data model could be done. Therefore, common data model is the foundation of integrating heterogeneous data in MDBSs. Currently, while selecting a common data model, two principles should be complied with as fallows

(1) Common data model should be as simple as possible so that it is easy to convert its schema to schemas of local data models or vice versus.

(2) Common data language conformed to the common data model should be convenient for data representation and processing in MDBSs.

2.2 XIDM: XML-Based Integration Data Model

Most of the existing MDBSs are using object-oriented model as their common data model[4]. 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 gen-

eral 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[7], is presented as follows.

Definition 1 In an XIDM model, an element cluster (EC) is the 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. For example, the following several rows give a definition of an element cluster *Employee*.

```
< xsd:schema xmlns:xsd = " http://www.w3.org/2001/XMLSchema ">
< xsd:element name = " Employee " type = " EmployeeType ">
< xsd:complexType name = " EmployeeType ">
< xsd:sequence >
< xsd:element name = " Name " type = " xsd:string "/ >
< xsd:element name = " Salary " type = " xsd:float "/ >
</ xsd:sequence >
< xsd:attribute name = " No " type = " xsd:string "/ >
< xsd:attribute name = " Dept " type = " xsd:string "/ >
< xsd:complexType >
```

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 a MDBS, global element cluster (GEC) and export element cluster (EEC), which correspond to global schemas and export schemas respectively. The following descriptions of XIDM model will base on the concept of element cluster.

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

Note, we take an XIDM model equally as an XIDM graph in most cases.

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 or identity constraint definition of the element, 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.

Here, the sub element cluster has the same definition with the element cluster. That is, there is a nested relationship between the element cluster and its sub element clusters. The schema mappings will be discussed later.

Definition 4 Each edge in graph G is a group $B = < EC1, EC2, Label >$, where EC_1 is the starting element cluster, EC_2 is the ending element cluster and *Label* is the symbol on the edge.

In an XIDM model, 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 of the element cluster, represents reference relationship between different element clusters, pointing to the referenced element cluster from its referencing element cluster. The directions of the edges make the XIDM an ordered model, which may have many advantages, such as more complex semantic expressions and more effective querying.

3. SCHEMA MAPPING IN MULTIDATABASE SYSTEMS

3.1 Basic Concepts

As we have discussed above, there are four schema levels in MDBSs: external schema level, global schema level, export schema level and local schema level. Since the external schemas, which are beyond the global database layer in MDBSs, are designed for special, different user groups and applications, the remaining three schema levels are concerned in this paper.

As described in Fig. 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, based on common data model, is also a virtual database, containing export schemas. There are different export schemas corresponding to different local schemas of LDBSs. 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. Objects in global database and export database are virtual ones, while local databases have real ones.

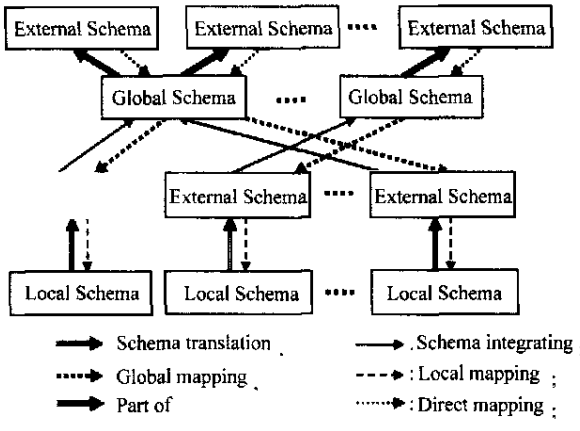

Fig. 1 Schema mapping architecture in multidatabase systems

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. Fig. 1 illustrates the integration process and schema mapping architecture. In a MDBS, there are two types of schema mappings: global mapping and local mapping. The global mappings, which are a portion of the global schema, relate global objects in global database with export objects in export database, while the local mappings in each export schema relate its export objects with corresponding local objects.

3.2 Global Mappings

In Panorama system, the same data model, XIDM, was employed to describe global schema and export schema. So global mappings between global schema and export schema don't need to deal with schema translation between different data models. In reverse, that is, when XIDM-based export schemas are integrated into XIDM-based global schemas, we just need to merge these export schemas without any changes. There are two types of global mappings: horizontal mapping and vertical mapping. Some global mappings may contain both these two mappings. We can split them into two cases: horizontal mapping first and then vertical mapping, or vice versus. Here, we give the definitions of these two mappings.

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, $EEC_1(K_1, A_1, S_1, Q_1, M_1)$, ⋯, $EEC_n(K_n, A_n, S_n, Q_n, M_n)$, according to a set of given conditions P_1, ⋯, P_n, where (1) $K_1 = K_2 = \cdots = K_n = K$; (2) $A_1 = A_2 = \cdots = A_n = A$; (3) $S_1 = S_2 = \cdots = S_n = S$; (4) $Q_i = (Q \wedge P_i)$; (5) $M_i = LM_i$.

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

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, $EEC_1(K_1, A_1, S_1, Q_1, M_1)$, ⋯, $EEC_n(K_n, A_n, S_n, Q_n, M_n)$, according to a set of given pairs $< R_1, T_1 >$, ⋯, $< R_n, T_n >$, where (1) $K_1 = K_2 = \cdots = K_n = K$; (2) $A_i = R_i \cup K_i$; (3) $S_i = T_i$; (4) $Q_1 = Q_2 = \cdots = Q_n = Q$; (5) $M_i = LM_i$.

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

ment 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, there may be some intersections among qualifications of different export element clusters, that is, reduplicate elements may exist among different export schemas after horizontal mapping. The reason is that these objects, records or elements have already existed in different local databases before they are integrated into MDBSs. Another example is 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.

3.3 Local Mappings

As we know, local mappings will be built during the translation from local schemas to export schemas. In Panorama system, we use 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.

3.3.1 Mappings between object-oriented model and XIDM model

As a local database schema, some concepts in object-oriented database system, such as class aggregation, classes, attributes (including key attribute, simple type attribute and complex type attribute), and interclass relationships, should be concerned. Table 1 shows the mappings between object-oriented (OO) model and XIDM model.

Here, we use class aggregation to indicate the concept, which has some certain functionality and contains several classes. For example, we model an object-oriented database for a company using some classes, such as *Employee*, *Department*, *Address*, *Finance*, *Production*, *Sales*. So, we may use class aggregation *Company* to include these classes. Class aggregation is similar to the concept of database instance in relational data model.

Table 1 Mappings between OO model and XIDM model

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

3.3.2 Mappings between relational model and XIDM model

Similar to OODB, database, relations (tables) and attributes (including primary key and foreign key attribute) should be taken into account in relational database system. Table 2 shows the mappings between relational model and XIDM model.

Table 2 Mappings between relational model and XIDM model

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

3.3.3 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, "<Name>" defined by "<!ELEMENT Name (#PCDATA)>" in XML documents, operation flow, process definition, action and state transition in work flow management system, file names, media type and file size in multimedia system, etc., 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. Limit to space, we don't give the details and examples in this paper.

4. IMPLEMENTATION IN PANORAMA SYSTEM

In Panorama system, XIDM acts as the common data model and provides universal data representation and exchange platform for multidatabase global users. It can integrate object-oriented database systems (such as ObjectStore, O2), relational database systems (such as Oracle, Sybase, DB2/UDB), HTML/XML documents, file systems and other semi-structured data[7]. The following examples will illustrate schema mappings among different schemas in Panorama system.

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 De-
    partment::Contain;
    boolean SetSalary(in float NewSal) raises
    (SalaryNotSet);
  };
  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*. Fig. 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. Node 9 gives the implementation part of the method, while node 10 denotes the exception processing for the method. The qualifications and schema mappings are not presented in Fig. 2. Note that some name transformation should be needed when we translate the local schemas into XIDM-based export schemas. For example, class *Staff* is translated into element cluster *Employee* and attribute *Name* of class *Staff* is transformed into element cluster *EName*.

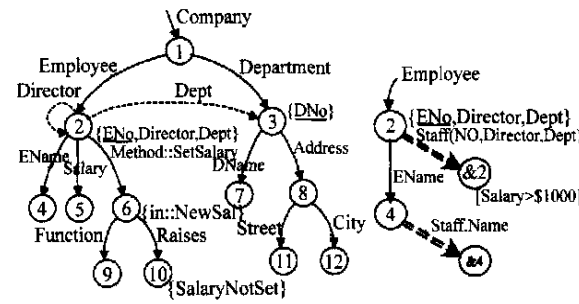

Fig.2 XIDM-based export schemas for class schemas in Example 1

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

As a demonstration, the schema mappings and qualifications of node 2 and node 4 are discussed here (see Fig. 3). "&2" and "&4" are used for representing the local concepts corresponding to node 2 and node 4 respectively. We use double dashed line to de-

note 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. For instance, the local concepts corresponding to element cluster *Employee* and its attributes *ENo*, *Director* and *Dept* are class *Staff* and its attributes *No*, *Director* and *Dept* respectively. So, the symbol on the edge is "*Staff* (*No*, *Director*, *Dept*)". For element cluster *EName* the corresponding local concept is attribute *Name* of class *Staff*, therefore, the symbol is "*Staff*. *Name*". Suppose we just want element cluster Employee to include those staffs whose salary is more than $1 000, then the qualification will be "[Salary > $1 000]".

Example 2 In a local database instance *Corporation* of relational database Oracle joining in Panorama, there are three tables: *Person*, *Branch* and *Address*, whose relational schemas are presented in Fig. 4 and corresponding XIDM-based export schemas are given in Fig. 5. The same with Example 1, we also need to process schema transformation between local schemas and export schemas. For example, table name *Person* is converted into element cluster *Employee*, and attribute *Name* of *Person* is translated into element cluster *EName* in XIDM model. Another illumination is 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 qualifications and schema mappings in Example 2 is similar to Example 1.

After integrating the XIDM-based export schemas represented by Fig. 2 and Fig. 5, we can get the global schema in panorama system for the local schemas in Example 1 and Example 2. Fig. 6 shows the integrated global schemas that also bases on XIDM model. 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 Fig. 2 and Fig. 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}>".

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.

Fig. 4 Relational schemas for database *Corporation*

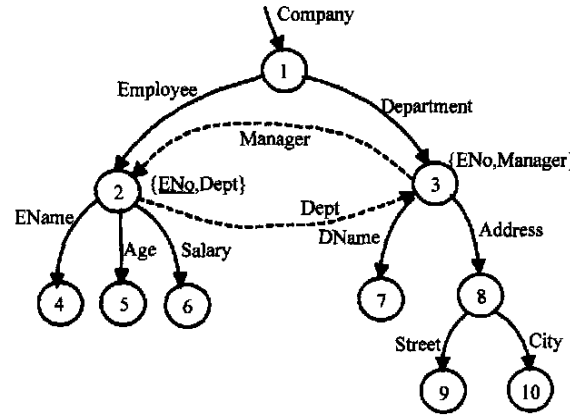

Fig. 5 XIDM-based export schemas for relational schemas in Example 2

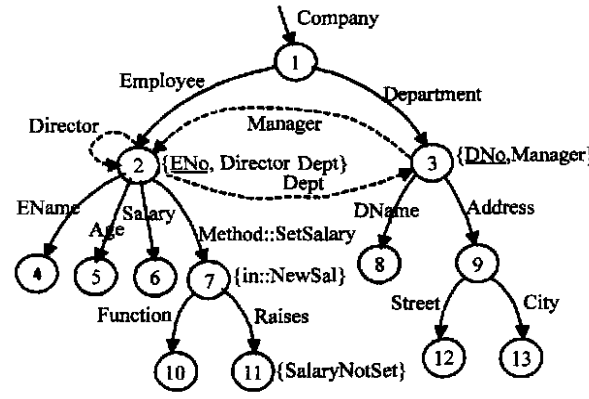

Fig. 6 XIDM-based global schemas

5. CONCLUSION

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 suggest how global queries will get their results from LDBSs. 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 want new data description methods and efficient information integration strategies. There has been much interest recently in integrating heterogeneous data sources. Since schema mapping is tightly related with common data model in MDBSs, this study firstly presents an XML-based Integration 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. We are developing a query language complying with XIDM model and designing GUI tools to maintain the global schema information including schema mappings in Panorama system. We are also 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] Bouguettaya A, Benatallah B, Elmagarmid A. An overview of multidatabase systems: past and present. management of heterogeneous and autonomous database systems. San Francisco: Morgan Kaufmann Publishers, 1999. 1～32.

[2] Roantree M, Muphy J, Hasselbring W. The OASIS multidatabase prototype. SIGMOD Record, 1999, 28(1): 97～103.

[3] Becker S A, Gibson R, Leist N L. A study of a generic schema for management of multidatabase systems. Journal of Database Management, 1996, 7(4): 14～20.

[4] Pitoura E, Bukhres O, Elmagarmid A. Object orientation in multidatabase systems. ACM Computing Surveys, 1995, 27(2): 141～195.

[5] Christophides V, Cluet S, Simeon J. On wrapping query languages and efficient XML integration. ACM SIGMOD Record, 2000, 29(2): 141～152.

[6] Li Bing, Lu Zhengding, Xiao Weijun, et al. Architecture for multidatabase systems based on CORBA and XML. Proceedings of the 12th International Workshop on Database and Expert Systems Application (DEXA'2001), Munich, Germany, 2001. 32～37.

[7] Mudar Sarem, Li Ruixuan, Xiao Weijun, et al. Registering different DBMSs to panorama with CORBA. The International Software Engineering Symposium 2001 (ISES'01), 2001, 6(1～2): 423～431.

Li Ruixuan was born in 1974. He received B. S. degree from Huazhong University of Science and Technology (HUST) in 1997 and M. S. degree in 2000, respectively. Now he is a lecturer and also a Ph. D candidate in College of Computer Science and Technology, HUST. His current research interests include heterogeneous data management, semantic web and ontology.

Lu Zhengding was born in 1944. He is a professor in College of Computer Science and Technology, HUST. His main research interests include distributed system, computer integrated system and database system.

Xiao Weijun was born in 1974. He received Ph. D. degree from Huazhong University of Science and Technology in 2002. Now he is an associate professor in College of Computer Science and Technology, HUST. His current research interests include distributed and heterogeneous system, database system and artificial intelligence.

Wu Wei was born in 1975. He is a Ph. D candidate in College of Computer Science and Technology, HUST. His current research interests include web data management, semantic web and peer-to-peer computing.