Virtual Database Technology for Distributed Database

Yuji Wada
Department of Information Environment
Tokyo Denki University
Inzai, Chiba, Japan
yujiwada@sie.dendai.ac.jp

Yuta Watanabe
*Department of Information Environmen*t
Tokyo Denki University
Inzai, Chiba, Japan

Keisuke Syoubu
Department of Information Environment
Tokyo Denki University
Inzai, Chiba, Japan

Jun Sawamoto
Faculty of Software and Information Science
Iwate Prefectural University
Morioka, Iwate, Japan
sawamoto@iwate-pu.ac.jp

Takashi Katoh
Faculty of Software and Information Science
Iwate Prefectural University
Morioka, Iwate, Japan

Abstract— In this paper, our research objective is to develop a database virtualization technique so that data analysts or other users who apply data mining methods to their jobs can use all ubiquitous databases in the Internet as if they were recognized as a single database, thereby helping to reduce their workloads such as data collection from the Internet databases and data cleansing works. In this study, firstly we examine XML scheme advantages and propose a database virtualization method by which such ubiquitous databases as relational databases, object-oriented databases, and XML databases are useful, as if they all behaved as a single database. Next, we show the method of virtualization of ubiquitous databases can describe ubiquitous database schema in a unified fashion using the XML schema. Moreover, it consists of a high-level concept of distributed database management of the same type and of different types, and also of a location transparency feature. Finally, we develop a common schema generation method and propose the virtual database query language for use in a virtualized ubiquitous database use environment.

Keywords-component; database virtualization; data mining; XML schema; ubiquitous databases; database integration;database query

I. INTRODUCTION

Nowadays, massive amounts of data are collected daily in ubiquitous sensor network environments. With such data available and elaborately structured, it is more important than ever to locate and access knowledge and trends from it using data mining techniques. Those data are valuable to support analyses and decision-making in businesses, for example. Such data normally exist in databases of various types—called ubiquitous databases hereinafter—that might usually be distributed and placed anywhere. A salient problem, however, is that a person who engages in data mining using ubiquitous databases would have to spend much time for database selection and data collection, for example, which would be merely a preparatory step to the actual data mining tasks. What a person really should want must be instead to concentrate on the work of analysis and rule extraction.

In our study, the primary objective is therefore to develop a virtualization technique so that the data analyst or other user can use all ubiquitous databases as if they were recognized as a single database, thereby helping to reduce the user's workload.

II. ASSOCIATED STUDIES

Some earlier reports in [1], [2] have described the study of database virtualization technology.

One report [1] proposed development of a system to pass information actively to all users in a mobile computing environment without fail, as sourced from various types of database groups connected by a wide-area network. By image-copying of the data of the local database group to a meta-database through the basic search and build operations, for example, it is intended to combine data and include different types of the local database group.

The data integration technique, *teiid*, which is described in [2], enables virtualization of various types of databases; through such virtual databases, one can access such data sources as relational databases, web databases, and application software such as ERP and CRM, for example, in real time. They can all be integrated for use. In fact, teiid has a unique query engine. Furthermore, the real-time data integration is accomplished by connecting business application software through the JDBC/SOAP access layer with data sources which are accessed through the connector framework.

In our study, we considered the metadata, UML, E–R model, and the XML schema as candidates for use to accomplish database virtualization. Thereby, ubiquitous databases can be used as if they were a single database. We then compared the advantages and disadvantages of each to analyze them as follows.

978-0-7695-4019-1/10 $26.00 © 2010 IEEE
DOI 10.1109/WAINA.2010.38

(1) The use of metadata presents many advantages for creation that are irrelevant to what database model the metadata are based on. On the other hand, an important disadvantage is that they require a great workload to create them in their initial stage. Moreover, no definition and manipulation language to manage metadata has been standardized yet.

(2) The UML and the E–R model have similar fundamental characteristics; each has an advantage that its database design concept structure is irrelevant to what data model it is based on and with what DBMS product it is associated. However, those are only a few design techniques. No specific DBMS and definition manipulation language are provided.

The matters described in (1) and (2) above subsume a structured static schema for their use. Therefore, they have a difficulty in use with databases of various kinds that are available in the Internet in a flexible fashion.

(3) The XML scheme is now widely used to exchange information in the Internet environment. Additionally, now that more studies and further developments of XML database management systems are made than ever before, it is advantageous to use it because its definitions and manipulations are well standardized in [3]. However, from a data model perspective, it presents the problem that it does not go well with any object-oriented data model that is now associated with multimedia. Even given that fact, now that the object-oriented concept is incorporated into the extended standardization of the SQL language, it has been improved in its affinity level with the associated XML scheme being converted to a relational database scheme. In addition to that, because the XML schema is a semi-structured dynamic one, it is still advantageous because of the fact that it is useful in a flexible fashion with various databases available on the Internet.

In the relevant literature, several studies [4–8] of the XML scheme conversion have been reported. One of those reports [4] proposes an XML-to-relational mapping framework and system that provides the first comprehensive and end-to-end solution to the relational storage of XML data. Another report [5] offers a flexible mechanism for modifying and querying database contents using only valid XML documents, which are validated over the XML-Schema file's rules. Another report [6] proposed cost-based XML storage mapping engine explores the space of possible XML-to-relational mappings and selects the best mapping for a given application. Another report [7] describes the integration of XML with a relational database system to enable the storage, retrieval, and update of XML documents. A common data model based on XML has been introduced and schema mapping based on that approach has been presented in [8].

Our study [9] examined XML scheme advantages and proposes a virtualization method by which such ubiquitous databases as relational databases, object-oriented databases, and XML databases are useful, as if they all behaved as a single database.

On the other hand, studies of recovery techniques from database trouble are now well underway at a practical level with respect to central databases or distributed databases, and are widely used in the field of Online Transaction Processing (OLTP). However, few practical studies have been undertaken in environments associated with database virtualization.

Therefore, we proposed a means to recover the associated databases by allowing users to examine the virtual environment only for ubiquitous databases without having to examine real databases, and to ensure the integrity between a virtual database and an associated real database in [10].

In summary, the method of virtualization of ubiquitous databases proposed in our study describes ubiquitous database schema in a unified fashion using the XML schema. Moreover, it consists of a high-level concept of distributed database management of the same type and of different types, and also of a location transparency feature.

In the following chapter, we are going to describe a common schema generation method and propose the virtual database query language for use in a virtualized ubiquitous database use environment.

III. DATABASE VIRTUALIZATION

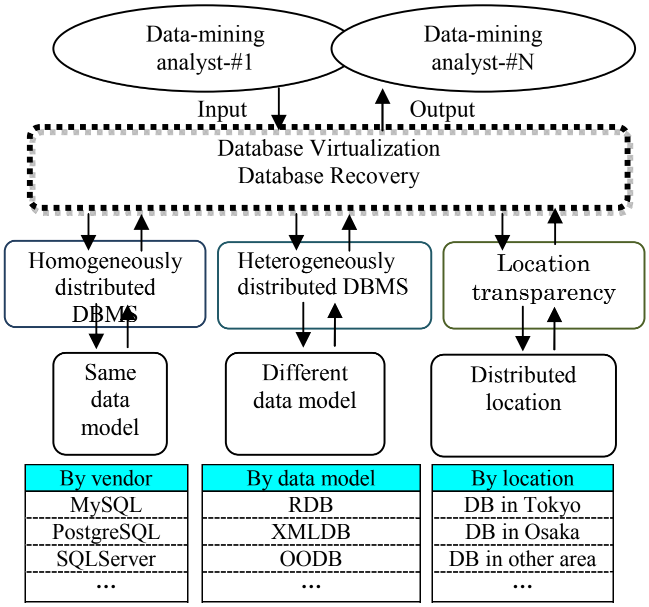
Databases of many kinds exist in terms of their associated data model differences and vendor differences. Regarding differences among data models, each has different data representation, and unique associated manipulation. Some typical examples include the table type of relational databases (RDB), XML-representation type of XML databases (XMLDB), and object-oriented databases (OODB). Even the same model database might have different features among vendors. Regarding RDB for example, there might be some differences in SQL and/or data type representation. The typical example is that we have MySQL, PostgreSQL, and SQLServer, each of which has a different vendor.

These differences according to the model and vendor bring some undesired results. For example, we might end up spending more time and labor during application system development because of the different data models that must be confronted. For example, we might need to acquire the right API to handle data of every different type of database. Virtualization of such different types of modeled databases to unify the procedures for all of them would probably impart less of workload and cost, and facilitate their management in a more flexible manner. Consequently, virtualization of databases, if it could be done, would facilitate application system design and database management as well.

To have a virtualization feature, we will consider the inclusion of features to manage distributed databases of similar types, the distributed databases of different types, and provide location transparency for users, such that they notice no differences of database structure or location and become able to use databases of all kinds in a flexible fashion. Fig.1 portrays a comprehensive view of a database virtualization technique.

For virtualization of ubiquitous databases in our study, we will describe the schema information of the real databases, of which more than one always happens to exist, by creating and using one common XML schema. We also provide functionality of data search and update with the XML-based common data manipulation API.

In the following, we will begin discussion of virtualization of the same type of distributed database followed by virtualization of different types of distributed databases.



By vendor	By data model	By location
MySQL	RDB	DB in Tokyo
PostgreSQL	XMLDB	DB in Osaka
SQLServer	OODB	DB in other area
...	...	...

Figure 1. Database virtualization technique

A. *Visualization of Homogeneous Distributed Databases*

Described next, as the first step of database virtualization, is a method of building a virtual database management system for RDBs provided by different vendors.

1) XML conversion program: Considering virtualization not only for the RDB, but also for the different data models such as XML DB and object-oriented DB, which will be required in the next phase, we will use an XML scheme that provides a flexible representation capability and a high transparency capability.

To do so, we will produce such a virtualization concept in which the user would feel as if he or she were locally manipulating the remote-site RDB from a local RDB process environment. That can be accomplished by converting the schema information and data information of the local RDB into the XML schema, and then storing that information into the RDB that the user would like to operate.

We developed an XML conversion program, XMLExport/Import, as depicted in Fig. 2. We then used such different vendor RDBs as MySQL, PostgreSQL, and SQLServer2005 because they are available in the RDB virtualization system creation environment. We used PHP as the programming language to develop an XML Export/Import system.

2) RDB schema conversion into XML: The following describes how the RDB schema is converted into XML. Fig.3 presents results of reading the schema information from the RDB and converting it into XML. The RDB schema information that is converted into an XML format includes “table names”, “field names (associated data types and default values)”, and “constraints (primary key constraint, unique constraint, check constraint, NOT NULL constraint, and foreign key constraint)”.

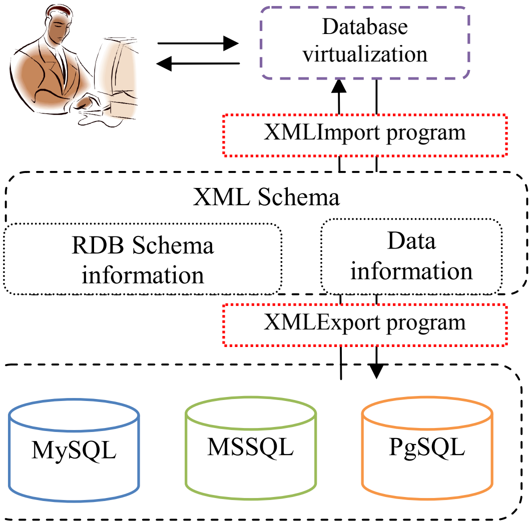


Figure 2. Virtualization technique for homogeneous distributed databases.

Regarding the XML tree structure, we described the table information in the table_structure node with its elements of Field=”column name”, Type=”data type”, Null=”TRUE or FALSE” (NOT NULL constraint), as shown in Fig. 3. We described the schema information in the schema node with its elements of TYPE=”constraint name”, Table=”table name”, Column=”column name”, ReTable=”referenced table name”, ReColumn=”referenced column name”, and Check=”rule”.

3) RDB data conversion into XML: The manner in which the RDB data are converted into XML is described next. Fig.4 portrays results of reading the data information from the RDB and conversion into XML. Because of the XML tree structure, we had dbname=”database name”, tblname=”table name”, and the succeeding column name=”actual data”.

```
<?xml version=”1.0” encoding”UTF-8” standalone=”yes”
-<root>
-<rdb Name=”mysql”>
-<database Name=”questionnaire” >
-<table_structure Name=”member”>
<field Field=”samplenum” Type=”integer”
Null=”FALSE”  Default=”  />
<field Field=”answerday” Type=”text” Null=”FALSE”
Default=”  />
….
</table_structure>
-<schema>
<constraint Type=”PRIMARY KEY”
Table=”member”  Column=”samplenum”  />
….
<constraint Type=”FOREIN KEY”
Table=”questionnaire” Column=”samplenum”
Retable=”member”  ReColumn=”samplenum” />
….
</schema>
</database>
</rdb>
</root>
```

Figure3. Example of RDB schema information conversion into XML.

```
-<root>
-<dataset dbname=”mysql”>
<data tblname=”member” samplenum=”10001”
answerday=”’2007/7/6’”
answertime=”’ 13:07:19:499’” />
<data tblname=”member” samplenum=”10002”
answerday=”’2007/7/6’”
answertime=”’ 13:10:33:507’” />
….
</dataset>
</root>
```

Figure4. Example of actual RDB data conversion into XML.

B. Virtualization of Heterogeneous Distributed Databases

The second step we discuss is the virtualization of modeled DBs of different types. For virtualization of different types of modeled DB, we describe the schema information of each model using a single common schema. The common schema we will use is an XML Schema. Around it, we will perform virtualization. Fig.5 presents a virtualization method for different database types.

To accomplish schema conversion from a different modeled database, we first get the schema information from an RDB to work on. Then we convert it into the correct XML schema for that RDB.

Table 1 presents schema conversion correspondences between the two. Because any XML DB is already described in the XML format, we extract the schema information without conversion. For an OODB, we will use the object-oriented functionality that is available because it is fundamentally an extension of RDB.

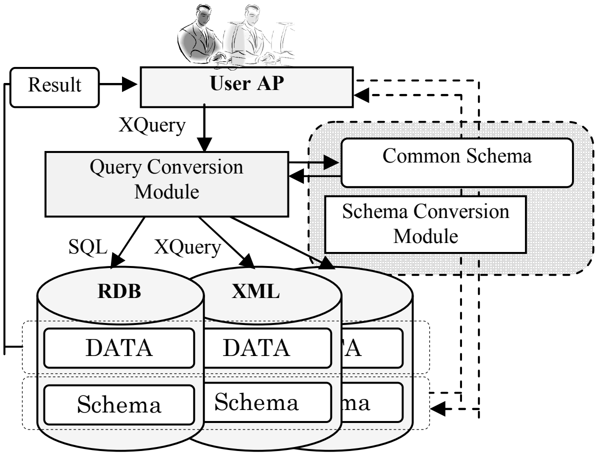


Figure 5. Virtualization technology for heterogeneous distributed databases

Table 1 SQL and associated XML

	SQL	XML
Table definition	CREATE TABLE table name…	<xsd: element name=”table name”…
Column definition	CREATE TABLE… column name...	<xsd: element name=”column name”…
Data type definition	CREATE TABLE… data type..	<xsd: element… type=”data type”…
Default values	CREATE TABLE… column name DEFAULT value	<xsd: element… default=”value”…
Primary key constraint	PRIMARY KEY	<xsd: key…
Unique constraint	UNIQUE	<xsd:unique …
Foreign key constraint	FOREIGN KEY	<xsd: keyref … refer =…
NOT NULL	NOT NULL	<xsd:… nillable=”false”...
Method	CREATE METHOD	
Inheritance	CREATE TABLE… UNDER upper level table name	<xsd: complexType …

C. Common Schema Generation

The common schema provides the virtualized database structure for the application programs, as shown in Fig. 5. This schema is used to examine the syntax of query sentences and the constraints. In this research, we developed the common schema generation program which converts the RDB schema into the common schema. In Fig.6, we show the SQL/CREATE to converted. In Fig.7, we display the conversion example of the following schema created by the SQL/CREATE.

```
        CREATEDB  EmployeeDB;

        CREATE TABLE   EmployeeTable (
          EmployeeID  int  PRIMARY KEY,
          Name  varchar(50)  NOT NULL,
         Salary  int  CHECK(0 < Salary),
         AffiliationID  int  REFERENCES
          AffiliationTable(AffiliationID)
        ON UPDATE CASCADE ON DELETE CASCADE);

        CREATE TABLE AffiliationTable (
      AffiliationID  int  Primary Key,
      Affiliation  varchar(5) UNIQUE);
```

Figure 6. Example of SQL/CREATE.

```
<!—    -->
      <xs:element name="AffiliationTable"  type="
AffiliationTable Type"/>
      <xs:element name="EmployeeTable" type="
EmployeeTable Type"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name=" AffiliationTable Type">
   <xs:annotation>
      <xs:appinfo>
         <r:index index-key="AffiliationID" primary="yes"/>
         <r:index index-key="Affiliation" unique="yes"/>
      </xs:appinfo>
<!—     -->
      <xs:complexType>
         <xs:sequence>
            <xs:element minOccurs="1" name="AffiliationID"
r:nullable="false"
            <xs:element minOccurs="0" name="Affiliation"
r:sqltype="varchar"
<!—    -->
<xs:complexType name=" EmployeeTable Type">
   <xs:annotation>
      <xs:appinfo>
         <r:index index-key="EmploeyeID" primary="yes"/>
         <r:check check-column="Salary" rule="0 &lt;
&quot;Salary&quot;"
         <r:fkey fkey-column="AffiliationID" ref-
column="AffiliationID" ref-table="Affiliation
```

Figure 7. Example of the common schema.

D. Query Conversion

We are now under development of the query language to access the virtual databases. As shown in Fig.5, we are planning the extension of the existing XQuery and the query conversion program from the XQuery language into SQL language or XQuery language Here, we show the sample of the query program using XML which we are currently investigating, as shown in Fig.8.

```
for $employee in common-schema()/DB1
          /sample_db1/employee
 for $manager in common-schema()/DB2
        /sample_db2/affiliation.xml
       where $employee/EmplyeID
          = $manager/Affiliation/
         return $employee/Name
```

Figure 8. Sample of the virtual database query

IV. CONCLUSION

We developed the common schema conversion program for RDB schema into XML schema. Especially, we showed the schema constraints (such as PRIMARY KEY, CHECK, NOT NULL, UPDATE CASCADE ON DELETE, UNIQUE) can be converted.

In the future research, we are going to develop the integration program of XML DB schema into the common schema. In addition, we plan to implement the common data manipulation API (for example, extension of the existing XQuery modules) to access the virtual databases and we are going to incorporate location transparency functions to this API.

ACKNOWLEDGMENT

This research was supported by a Grant-in-Aid for Scientific Research C (Subject No. 20500095: Study of Data utilization with Ubiquitous Database Virtualization Technology).

REFERENCES

[1] K. Mori, S. Kurabayashi, N. Ishibashi, and Y. Shimizu, Method of Sending Information Actively Reducing User Information Load Dynamically in a Mobile Computing Environment, DEWS2004 (March 2004).

[2] teiid, http://www.jboss.org/teiid, Red Hat

[3] S. Abiteboul, P. Buneman, and D. Suciu, Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufmann Series in Data Management Systems (1999).

[4] S. Amer-Yahia, F. Du, and J. Freire, “A comprehensive solution to the XML-to-relational mapping problem,” Proc. 6th Annual ACM International Workshop on Web Information and Data Management, pp.31-38 (2004).

[5] I. Varlamis and M. Vazirgiannis, “Bridging XML-schema and relational databases: a system for generating and manipulating relational databases using valid XML documents,” Proc. 2001 ACM Symposium on Document engineering, pp.105-114 (2001).

[6] P. Bohannon, J. Freire, P. Roy, and J. Simeon, “From XML Schema to Relations: A Cost-Based Approach to XML Storage,” Proc. 18th International Conference on Data Engineering, pp. 64-75 (2002).

[7] G. Kappel, E. Kapsammer, and W. Retschitzegger, “Integrating XML and Relational Database Systems,” World Wide Web: Internet and Web Information Systems, 7, pp. 343-384 (2004).

[8] R. Li, Z. Lu, W. Xiao, B. Li, and W. Wu, “Schema Mapping for Interoperability in XML-Based Multidatabase Systems,” Proc. 14th International Workshop on Database and Expert Systems Applications, (2003).

[9] Y. Wada, Y. Watanabe, J. Sawamoto, and T. Katoh,” Database Virtualization Technology in Ubiquitous Computing,” Proc. 6th Innovations in Information Technology (Innovations’09), pp.170-174 (2009-12).

[10] Y. Wada, Y. Watanabe, K. Syoubu, J. Sawamoto, and T. Katoh, ”Virtualization Technology for Ubiquitous Databases,” Proc. 4th Workshop on Engineering Complex Distributed Systems (ECDS 2010) (2010-02)(to be appeared)