Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 2005 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

December 2005

XML-Based Heterogeneous Database Integration For Data Warehouse Creation

Frank Tseng
National Kaohsiung First University of Science and Technology

XML-Based Heterogeneous Database Integration For Data Warehouse Creation*

Frank S.C. Tseng[1]
Dept. of Information Management
National Kaohsiung First University of Science and Technology
1, University Road, YenChao, Kaohsiung County, Taiwan, 824 R.O.C.
imfrank@ccms.nkfust.edu.tw

Abstract

In the past decade, research works in heterogeneous database integration have established a good and solid framework to alleviate this task. However, there are still works need to be accomplished to bring these achievements to be easily implemented and integrated to Internet applications. In this paper, by employing the metadata of participate sites, we propose using XML, together with XSLT, as a general platform to achieve this task. We first define the formal definitions for the problems of semantic conflicts among heterogeneous databases and present their solutions. Then, some illustrative examples are presented to show that, by requesting local sites to transform the data into XML format and prepare the corresponding XSLT files on the global site, various kinds of schema integration problems can be unified and integrated into a global view seamlessly. The proposed methodology is not only suitable for heterogeneous database integration, but is also suitable for data warehouse creation and World-Wide Web presentation.

Keywords: Data Warehouse, Heterogeneous Database Integration, Metadata, XML, and XSLT.

1. Introduction

Database management systems (DBMS) pervade and proliferate tremendously throughout industry in the past decades. However, due to the storage capacity and cost, most of the prior database applications are mainly tailored to serve the information needs of people who handle day-to-day or short-term operations, such as inventory or purchasing.

Thanks to the ever-increasing capability and decreasing price of storage devices, together with the speed promotion of Internet technologies, it is now feasible to bring historical data on-line to serve corporate decision-makers to access all the organization's data, wherever it is located. The challenge for organizations now is the need to turn their archives of data into an integrated source of knowledge, such that a consolidated view of the organization's data can be presented for decision-making.

Contemporary business environments are more competitive and dynamic than ever. Since Inmon (1993) proposed the concept of *data warehouse*, which describes a subject-oriented, integrated, time-variant, and non-volatile collection of data in support of decision-making process, database vendors have rushed to implement the functionalities for constructing data

* This research was partially supported by the National Science Council, Republic of China, under contract no. NSC 93-2416-H-327-007.
[1] To whom all correspondence should be sent. Tel: +886-7-6011000 Ext. 4113, Fax: +886-7-7659541

warehouses. With the new enterprise-wide decision support architecture now emerging, data warehousing is gaining in popularity as organizations realize the benefits of being able to perform multi-dimensional analyses of cumulated historical business data to help contemporary administrative decision-making (Inmon and Kelley 1994; Kimball 1996; Srivastava and Chen 1999). That makes on-line analytical processing (OLAP) emerge as enabling technology for decision support systems. The successful implementation of a data warehouse can reveal previously untapped or unavailable information, which will increase productivity of corporate decision-makers.

However, since a data warehouse creation needs to integrate various enterprise-wide corporate data into a single repository, from which users can query via various dimensions and produce analysis reports. There are problems may arise in building a data warehouse with pre-existing data, since it has various types of heterogeneity. That makes it a common consensus that the ETL process (i.e., extraction, transformation, and loading) of data from various sources is indispensable before constructing a data warehouse. Therefore, the general conclusion is that the task has proven to be labor-intensive, error-prone, time-consuming and generally frustrating, leading a number of data warehousing projects to be abandoned mid-way through development. However, Trisolini *et al.* (1999) and Srivastava and Chen (1999) have pointed out that the situation is not as tough as it appears. In fact, the heterogeneity problems that are being encountered in data warehouse establishment are very similar to those encountered in heterogeneous database integration, which have been well studied in the past decade (*ACM Computing Survey* 1990; Batini *et al*. 1986; Breitbart *et al*. 1986; Breitbart 1990; Castano *et al.* 2001; *IEEE Computer* 1991; Hsiao 1992ab). Those works accomplished in dealing with heterogeneous schema integration have established a good framework to alleviate this task.

In this paper, we review the general problems of heterogeneous database schema integration and intend to identify the common issues in data integration and data warehouse creation. By employing the metadata of participate sites, we propose a framework for integrating heterogeneous database for data warehouse creation through XML technologies. We first define the formal definitions for the problems of semantic conflicts among heterogeneous databases and present their solutions. Then, some illustrative examples are presented to show that, by requesting local sites to transform the data into XML formats and prepare the corresponding XSLT files (http://www.w3c.org/TR/xslt) on the global site, various kinds of schema integration problems can be unified and integrated into a global view to construct loosely coupled data warehouse systems.

We outline the general process of employing XML technology to integrate heterogeneous databases for data warehouse creations in Figure 1. In this process, data stored in different databases are respectively transformed into XML format and formatted to adhere to the global schema by the XSLT transformation module. Then, the data will be further integrated and stored as an integrated database for data warehouse creation, which can be further used for on-line analytical processing for various business models. The proposed methodology is not only suitable for heterogeneous database integration, but also suitable for data warehouse creation and web presentation.

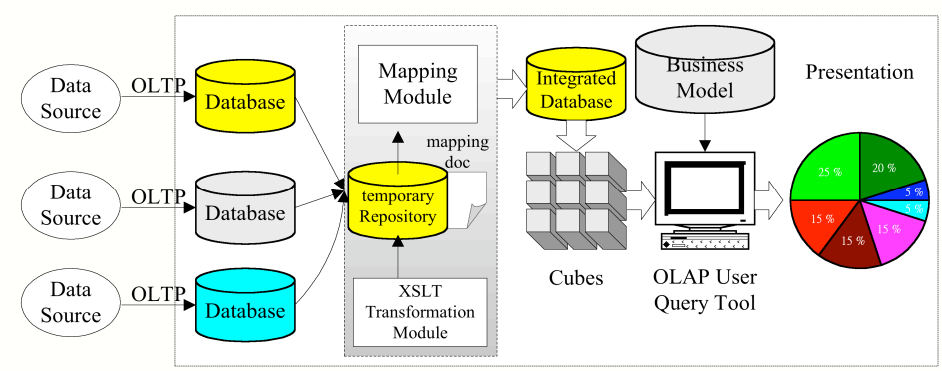

Figure 1: Loosely-Coupled Heterogeneous Database Integration by XML Technology.

2. Related Works

To integrate heterogeneous schemata and derive data in a *heterogeneous database environment*, prior works can be classified into the following general approaches.

1. One is to provide a global schema for the independent databases by integrating their schemas. Dayal and Hwang (1984) and Motro (1987) adopted this approach based on functional model, while Breitbart *et al.* (1986) and Deen *et al.* (1987) were based on relational model. Based on this approach, we have also proposed a probabilistic model to integrate heterogeneous database systems in (Tseng *et al.* 1993). Another variation of this approach does not require the creation of a global schema. On the other hand, for each application, the database administrator creates a schema describing only data that the application may access in the local databases. This type of system is also called a *federated database system* (Heimbigner and McLeod 1985; Hsiao 1992ab). For a comprehensive survey of methodologies developed for schema integration, readers are referred to (Batini 1986; Bukhres and Elmagarmid 1996; Elmargamid and Pu 1990; Kim and Seo 1991; *IEEE Computer* 1991).
2. The other approach is by providing users a *multi-database query language* (Grant *et al.* 1993). Users refer to the schemas and pose their queries against these schemas using the multi-database query language. Litwin *et al.* (1987ab) and Czejdo *et al.* (1987) fell into this category. A multi-database query language provides basic language constructs that allow users to issue queries across multiple database systems with heterogeneities. Grant *et al.* (1993) has proposed a theoretical foundation for such languages by extending the relational algebra and calculus to a multi-relational algebra and calculus. In the following, we only use the term *heterogeneous database systems*.

The schema integration process may present a large number of problems caused by various aspects of semantic discrepancy due to the design autonomy of each participant. Prior works of schema integration usually focus on resolving the incompatibility problems that may exist in different databases for semantically related data (Breitbart 1990).

In (Kim and Seo 1991), schematic and data heterogeneity in heterogeneous database systems are systematically classified. Reddy *et al.* (1994) also proposes a classification scheme for various kinds of semantic incompatibilities and data inconsistencies and presents a methodology covers both schema integration and database integration. Moreover, Lee *et al.* (1995) establishes a similar classification and proposes a way of optimizing multi-database queries, which takes advantage of the conflicts of schemas in searching for the execution plan with the least execution cost.

In this paper, we adopt the classification scheme proposed by Lee *et al.* (1995), which

categorizes the types of conflicts as follows.

1. *Value-to-value conflicts.* These conflicts occur when databases use different representations for the same data. This type of conflicts can be further distinguished into the following types.
   (a) *Data representation conflicts.* These conflicts occur when semantically related data items are represented in different data types.
   (b) *Data scaling conflicts.* These conflicts occur when semantically related data items are represented in different databases using different units of measure.
   (c) *Inconsistent data.* These conflicts occur when semantically related attributes for the same entity have different definite data values in different databases. Agarwal *et al.* (1995) present a good work on addressing the problem of dealing with data inconsistencies while integrating data sets derived from multiple autonomous relational databases.
2. *Value-to-attribute conflicts.* These conflicts occur when the same information is expressed as attribute values in one database and as an attribute name in another database.
3. *Value-to-table conflicts.* These conflicts occur when the attribute values in one database are expressed as table names in another database.
4. *Attribute-to-attribute conflicts.* This occurs when semantically related data items are named differently or semantically unrelated data items are named equivalently. The former case is also called *synonyms* and the latter case *homonyms* (Reedy *et al.* 1994). Some classification schemes call both cases *naming conflicts*.
5. *Attribute-to-table conflicts.* These conflicts occur if an attribute name of a table in a database is represented as a table name in another database.
6. *Table-to-table conflicts.* These conflicts occur when information of a set of semantically equivalent tables are represented in a different number of tables in another databases. When integrating relations with such conflicts into a global relation, null values are usually generated. Such phenomenon is also called *missing data*.

We have further classified these types of conflict into the following two categories in (Tseng *et al.* 1998).

1. *Conflicts of similar schema structures.* This category includes *value-to-value conflicts*, *attribute-to-attribute conflicts*, and *table-to-table conflicts*.
2. *Conflicts of different schema structures*. This category includes *value-to-attribute conflicts*, *value-to-table conflicts*, and *attribute-to-table conflicts*.

For tables with conflicts of similar schema structures, an *outerjoin* operation (Date 1983) is usually employed to integrate the tables into a unified one. DeMichiel (1989) has shown that some imprecise data, called *partial values*, which was proposed by Grant (1979), may be derived due to scaling conflicts. We have also established some related result regarding the incompatibility problems and partial values. We developed some efficient algorithms to evaluate relational operations over partial values (Tseng *et al.* 1993a, 1996). Besides, we generalized partial values into *probabilistic partial values* and proposed a general methodology to integrate relations with conflicts of similar schema structures (Tseng *et al*. 1993b). Some properties that can be employed to refine partial values into more informative ones or even definite values were also being studied (Tseng *et al*. 1993c).

However, for the integration of tables with conflicts of different schema structures, most

of the studies did not take into account the value-to-attribute conflicts, value-to-table conflicts, or attribute-to-table conflicts. Krishnamurthy *et al.* (1991) advocates some language features of a multi-database query language should be added to cover these types of conflicts. In (Tseng *et al*. 1998), we have proposed an approach to integrating tables with different schema conflicts.

Lee *et al*. (2002) established a framework to integrate heterogeneous information via XML Schema and employ XQuery (Chamberlin 2002; http://www.w3.org/XML/Query) for resolving conflicts in the integration processes. Under their framework, when integrating heterogeneous information, we shall first decide a common data model, which may be semi-structured or object-oriented. Then, after transforming heterogeneous information into XML documents, there may be conflicts among the source XML documents. Therefore, they defined some possible conflicts and use XML Schema as the target for integrating XML documents via XQuerySD (XQuery For Schema Definition) (Lee *et al*. 2002) based on XQuery.

In this paper, we propose another XML-based approach to integrating tables with aforementioned schema conflicts. Our presentation is organized as follows. In Section 3, we propose a general architecture of our approach, formally define the problems of schema conflict, and present some examples to illustrate the problems. Section 4 devotes to our approach to resolving the schema conflict problems by XML technologies. Finally, we draw a conclusion and outline some future works in Section 5.

3. Basic Concepts and Definitions

3.1 The Proposed Architecture

In the following, we will focus on the inner module surrounded by dotted lines in Figure 1 to illustrate the proposed XML-based framework. This module is further detailed in Figure 2. In this illustration, the data warehouse creation module can be regarded as a user of the heterogeneous database system. When a user pose a global query on the integrated system, the global site decomposes the global query into sub-queries to request each participant to return the data in XML format. This can be easily achieved in contemporary commercial database products (e.g., MS SQL Server 2000, IBM DB2 Extender, and Oracle *i*FS). Besides, the DBA in each site should prepare some XSLT format files, together with some necessary template files, in advance to transform local data into the global schema format. Finally, the transformed data are integrated into a consolidated view with the global query applied on it to return global data to the user.

Our integration process utilizes the semantic knowledge of all participate local schemas. We call these semantic knowledge *metadata*, which should be prepared or discovered before integration.

3.2 Metadata

For a participate relation, the metadata should consist of the following components:

1. *The domain of each attribute*: We use $Dom(A) = D$ to denote the domain of attribute A is D.
2. *The semantic description of each attribute*: This is used to ensure local autonomous systems to agree on the meaning of their exchanged data are not a trivial task. These semantic descriptions often depend on context information, the database origin, the applications, and so on. A very good work on formulating semantic information to support heterogeneous database integration has been established in (Sciore *et al.* 1994). By regarding context information as the metadata, the basic approach is based on the

concept of a *semantic value*, which is defined to be a piece of data together with its associated context. To convert a semantic value from one context to another, *conversions functions* will be employed.

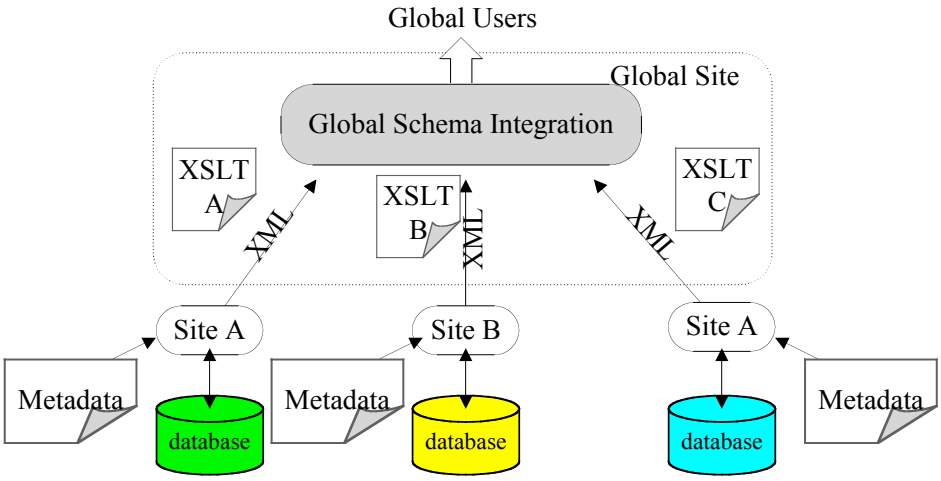

Figure 2: The Heterogeneous Database Integration Architecture.

To describe a semantic description of an attribute, we employ the following notation adopted from (Siegel and Madnick 1991):

$$Des(A) = \{S_1, S_2, \ldots, S_n\}$$

to denote the necessary descriptions for schema integration and to supply the semantics of attribute A, where S_1 is called the *primary description*, which will be denoted $Des^*(A) = S_1$ in the following. It represents the 'actual' meaning of the attribute values. When the attribute name is clearly enough for self-explanatory (i.e., there is no need of a primary description), then it can be replaced by an asterisk (*). The other S_i's are called the *auxiliary descriptions*, which will be denoted $Des'(A) = Des(A) - Des^*(A) = \{S_2, \ldots, S_n\}$ in the following. They will be used to supply auxiliary information of attribute A. If there is no need of any primary description and auxiliary description for an attribute A, then $Des(A) = \varnothing$, which can be regarded as an empty set or *null*.

To illustrate, for example, if the actual meaning of an attribute A is 'amount of money', then there may be an auxiliary description representing 'unit of currency', and $Des(A)$ = {'amount of money', 'unit of currency'}.

3.3 Formal Definitions for Schema Heterogeneities

In this section, we formally define the types of schema heterogeneity based on the classification scheme proposed by Lee *et al.* (1995) and distinguish them into two groups according to our prior work (Tseng *et al.* 1998), namely *conflicts of similar schema structure* and *conflicts of different schema structures*. We will discuss the heterogeneous database integration process based on these two categories.

We first define the following notations:

1. $Sch(R)$ is used to represent the *attribute set* of a relation R.
2. For two attributes A and B, we say A and B are semantically equivalent, denoted $A \Leftrightarrow B$, if and only if $(A = B) \vee (A = Des^*(B) \vee (B = Des^*(A)) \vee (Des^*(A) = Des^*(B))$, where "=" means the items at the left-hand-side and right-hand-side are the same but not homonyms.

3.3.1 The Conflicts of Similar Schema Structures

In this following, we present an example to illustrate the conflicts of similar schema structures, namely the *value-to-value*, *attribute-to-attribute*, and *table-to-table* conflicts. These conflicts are not exclusive, they may occur in any pair of relations at the same time. Such heterogeneity occurs frequently in two distinct pre-existing databases, when different databases are designed by different designers or by different considerations.

Example 1: Consider the scenario in Figure 3. There are two databases containing semantically related information about books but in different formats. Sites ***X*** and ***Y*** contain tables named **Books** and **Booklist**, respectively. Note that there exist some semantic discrepancies between these sites. We list them as follows.

1. There are two attribute-to-attribute conflicts: The attributes **Books**.*bno* and **Books**.*title* in Site ***X*** are respectively named **Booklist**.*bid* and **Booklist**.*bookname* in Site ***Y***.
2. There are three value-to-value conflicts: The **Books**.*price* stores the list price of every book, but the **Booklist**.*price* stores the discounted price (with 20% off). Besides, The **Booklist**.*location* stores more detailed data than **Books**.*location*. Finally, **Books**.*publisher* stores publisher full name but the data in **Booklist**.*publisher* are in a concise format.
3. There is a table-to-table conflict: The **Books**.*pages* is missing in the relation **Booklist**. Besides, the **Booklist**.*year* is also missing in the relation **Books**.

Books

bno	*title*	*author*	*price*	*publisher*	*location*	*pages*
1	Database Theory	Frank	200	IRWIN Company	IN	730
2	Algorithms	Jesse	250	SYBEX Company	CA	620

(Site ***X***)

Booklist

bid	*bookname*	*author*	*price*	*publisher*	*location*	*year*
1	Database Theory	Frank	160	IRWIN	Bloomington, IN	1980
2	Algorithms	Jesse	200	SYBEX	San Jose, CA	1983

(Site ***Y***)

Figure 3: Two Sites with Conflicts of Similar Schema Structures.

3.3.2 The Conflicts of Different Schema Structures

For the conflicts of different schema structure, we also use an example for illustration. Such conflicts of different schema structures include the *value-to-attribute*, *value-to-table*, and *attribute-to-table* conflicts. These conflicts are not exclusive, they may occur in any pair of relations at the same time.

Example 2: Consider the situation described in Figure 4. There are three databases containing the same information for global stock markets but in different schema structure formats. Site ***A*** contains a self-explanatory table named **Stock_Markets**. For the relation **Stock_Index** in Site ***B***, the attribute values of Tokyo, Taipei, and Bangkok in a record represent their closed index numbers of a date, respectively. In Site ***C***, all daily-closed index in different markets are stored in different tables with the market names as the table names, respectively. Although the three databases contain semantically equivalent data with respect to dates, markets, and the index numbers, they appear to be in conflicting schema structures. That is, there are *value-to-attribute* between Site ***A*** and Site ***B***, *value-to-table* between Site ***A*** and Site ***C***, and *attribute-to-table conflicts* between Site ***B*** and Site ***C***.

Stock_Markets

date	*stock*	*index*
9/5/2001	Tokyo	24600
9/5/2001	Taipei	4120
9/5/2001	Bangkok	780
9/6/2001	Tokyo	26065
9/6/2001	Taipei	4321
9/6/2001	Bangkok	803

(Site ***A***)

Stock_Index

date	*Tokyo*	*Taipei*	*Bangkok*
9/5/2001	24600	4120	780
9/6/2001	26065	4321	803

(Site ***B***)

Tokyo

date	*index*
9/5/2001	24600
9/6/2001	26065

Taipei

date	*index*
9/5/2001	4120
9/6/2001	4321

Bangkok

date	*index*
9/5/2001	780
9/6/2001	803

(Site ***C***)

Figure 4: The Problems of Conflicts of Different Schema Structures.

4. Our Approach to Heterogeneous Database Integration

In the following, we will present our XML-based solutions to integrate heterogeneous databases with semantic conflicts according to the two categories discussed in Section 3.2.

4.1 Resolving Conflicts of Similar Schema Structures

We have pointed out that the conflicts of similar schema structures may occur in any pair of relations at the same time. Therefore, to resolve these conflicts between any pair of relations, namely R and S with primary keys K_R and K_S respectively, at different sites the following procedures should be conducted.

1. Resolve value-to-value conflicts:
   (a) Identify all pairs of value-to-value conflicts between $Sch(R)$ and $Sch(S)$, namely x_i and y_i, and define the corresponding mapping functions f_i: $Dom(x_i) \rightarrow Dom(y_i)$ to translate the values between $Dom(x_i)$ and $Dom(y_i)$.
   (b) If the function f_i is one-to-one and onto, then the transformation is easy to accomplish and we can arbitrarily choose x_i or y_i as the target attribute in the global relation. Note that if we choose x_i as the target attribute, then we should use the inverse function of f, i.e. f^{-1}: $Dom(y) \rightarrow Dom(x)$, as the mapping function.
   (c) Otherwise, if the function f_i is many-to-one, then the transformation is easy if we choose y_i as the target attribute in the global relation. If we wish to choose x_i as the target attribute in the global relation, then the concept of *partial value* (Grant 1979) should be employed to capture all the values derived from the inverse function of f[2]. This makes the process of schema integration more complicated; we think such case is beyond the scope of this paper. Gentle readers are referred to (DeMichiel 1989; Tseng *et al.* 1993b).

[2] This is not a valid definition of function from the viewpoint of mathematics. However, we use the notation here just for describing the implementation details.

2. Resolve attribute-to-attribute conflicts: Identify all pairs of attribute-to-attribute conflicts between $Sch(R)$ and $Sch(S)$, namely x_i and y_i, and respectively define g_i: $Sch(R) \rightarrow Sch(S)$ as the mapping function defined for converting the conflicting attribute names x_i into another attribute names y_i and choose y_i as the target attribute in the global relation.
3. Resolve table-to-table conflicts: Directly perform an *outerjoin* operation (Date 1983) to integrate R and S to obtain the global relation.

These resolving procedures can be directly implemented in SQL commands. In the following, we will elaborate to resolve the conflicts found in Example 1 by MS SQL Server 2000 with the XML extension in Transact-SQL, which has already been employed and proven being effective to build loosely coupled systems over the Internet by Rys (2001).

Example 3: Suppose we wish to integrate both tables in Example 1 into a global relation named **BookData**(*bno*, *title*, *author*, *price*, *publisher*, *location*, *pages*, *year*). Suppose we wish to show **BookData**.*price* by the original list price, **BookData**.*publisher* by the full names, and **BookData**.*location* by the concise names. Then, we may use two sets of XSLT and template files to transform both tables into **BookData**, respectively. We use Figure 5(a) and Figure 5(b) to show the XSLT and template files for Site ***X***, respectively. For Site ***Y***, the XSLT and template files are illustrated in Figure 6(a) and Figure 6(b), respectively. After the transformations, **Books** and **Booklist** can be respectively rendered as Figure 7 depicts. Finally, both tables can be outer-joined into **BookData** as Figure 8 illustrates.

Books.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match = '*'>
    <xsl:apply-templates />
    </xsl:template>
    <xsl:template match = 'Books'>
        <TR><TD><xsl:value-of select = '@bno' /></TD>
                <TD><B><xsl:value-of select = '@title' /></B></TD>
                <TD><B><xsl:value-of select = '@author' /></B></TD>
                <TD><B><xsl:value-of select = '@price' /></B></TD>
                <TD><B><xsl:value-of select = '@publisher' /></B></TD>
                <TD><B><xsl:value-of select = '@location' /></B></TD>
                <TD><B><xsl:value-of select = '@pages' /></B></TD>
        </TR>
    </xsl:template>
    <xsl:template match = '/'>
      <HTML>
        <HEAD>
            <STYLE>th { background-color: #CCCCCC }</STYLE>
        </HEAD>
        <BODY>
          <TABLE border='1' style='width:600;'>
            <TR><TH colspan='7'>BookData</TH></TR>
        <TR><TH>bno</TH><TH>title</TH><TH>author</TH><TH>price</TH>
        <TH>publisher</TH><TH>location</TH><TH>pages</TH></TR>
            <xsl:apply-templates select = 'ROOT' />
          </TABLE>
        </BODY>
      </HTML>
    </xsl:template>
</xsl:stylesheet>
```

Figure 5(a): The XSLT for Site ***X***

Books.xml

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql" sql:xsl="Books.xsl">
    <sql:query>
      SELECT    bno, title, author, price, publisher, location, pages FROM Books
      FOR XML AUTO
    </sql:query>
</ROOT>
```

Figure 5(b): The Template file for Site ***X***.

Booklist.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
        <xsl:template match = '*'>
            <xsl:apply-templates />
        </xsl:template>
        <xsl:template match = 'Booklist'>
            <TR>
              <TD><xsl:value-of select = '@bid' /></TD>
              <TD><B><xsl:value-of select = '@bookname' /></B></TD>
              <TD><B><xsl:value-of select = '@author' /></B></TD>
              <TD><B><xsl:value-of select = '@price' /></B></TD>
              <TD><B><xsl:value-of select = '@publisher' /></B></TD>
              <TD><B><xsl:value-of select = '@location' /></B></TD>
              <TD><B><xsl:value-of select = '@year' /></B></TD>
            </TR>
        </xsl:template>
        <xsl:template match = '/'>
          <HTML>
            <HEAD>
              <STYLE>th { background-color: #CCCCCC }</STYLE>
            </HEAD>
            <BODY>
              <TABLE border='1' style='width:600;'>
                <TR><TH colspan='7'>BookData</TH></TR>
                <TR><TH>bno</TH><TH>title</TH><TH>author</TH><TH>price</TH>
                <TH>publisher</TH><TH>location</TH><TH>year</TH></TR>
                <xsl:apply-templates select = 'ROOT' />
              </TABLE>
            </BODY>
          </HTML>
        </xsl:template>
</xsl:stylesheet>
```

Figure 6(a): The XSLT for Site ***Y***.

Booklist.xml

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql" sql:xsl="Booklist.xsl">
    <sql:query>
      SELECT bid, bookname, author, price/0.8 as price,
          rtrim(publisher) + ' Company' as publisher, right(rtrim(location), 2) as location, year
      FROM Booklist
      FOR XML AUTO
    </sql:query>
</ROOT>
```

Figure 6(b): The Template file for Site ***Y***.

Books

bno	*title*	*author*	*price*	*publisher*	*location*	*pages*
1	Database Theory	Frank	200	IRWIN Company	IN	730
2	Algorithms	Jesse	250	SYBEX Company	CA	620

(Site ***X***)

Booklist

bid	*bookname*	*author*	*price*	*publisher*	*location*	*year*
1	Database Theory	Frank	200	IRWIN Company	IN	1980
2	Algorithms	Jesse	250	SYBEX Company	CA	1983

(Site ***Y***)

Figure 7: Two Sites with Conflicts of Similar Schema Structures.

BookData

bno	*title*	*author*	*price*	*publisher*	*location*	*pages*	*Year*
1	Database Theory	Frank	200	IRWIN Company	IN	730	1980
2	Algorithms	Jesse	250	SYBEX Company	CA	620	1983

Figure 8: The Integrated Relation **BookData** in the Global Site.

4.2 Resolving Conflicts of Different Schema Structures

In such cases, the *domain* of an attribute of a table in Site ***A*** may be represented as a subset of the *attribute set* of another relation in Site ***B***, or in turn be *a set of table names* in

Site ***C***. For data warehouse creations, we claim that the global schema structure should be chosen to conform the schema structure of Site ***A***. This is because the others hide the necessary data in the attribute names or table names, which cannot be retrieved when creating a data cube.

To create data warehouses under such circumstances, we only have to resolve the value-to-attribute and value-to-table conflicts. This is because we choose the schema of Site ***A*** as the global schema, and the attribute-to-table conflict is inherently resolved by resolving the other two conflicts. To resolve the value-to-attribute and value-to-table conflicts, the following procedures should be conducted.

1. Resolve value-to-attribute conflicts: If R and S with primary keys K_R and K_S respectively have value-to-attribute conflict, then
   (a) Directly use SQL command to retrieve all the data from R.
   (b) For each $x \in Sch(R)$ such that $Dom(x) \cap Sch(S) = \{a_1, a_2, \ldots, a_n\} \neq \emptyset$ and $(\forall y \in Dom(x) \cap Sch(S))(\exists z \in Sch(R) - \{x\})(y \Leftrightarrow z)$, directly retrieve all the data from S by the following SQL command:

      SELECT K_S, 'a_1' as x, a_1 as z FROM S UNION
      SELECT K_S, 'a_2' as x, a_2 as z FROM S UNION
      …
      SELECT K_S, 'a_i' as x, a_i as z FROM S UNION
      …
      SELECT K_S, 'a_n' as x, a_n as z FROM S

2. Resolve value-to-table conflicts: If relation R with primary keys K_R and a set of relations $S = \{S_1, S_2, \ldots, S_i, \ldots, S_k\}$ with primary keys $\{K_{S_1}, K_{S_2}, \ldots, K_{S_i}, \ldots, K_{S_k}\}$ respectively, have value-to-table conflict, such that $(\exists x \in Sch(R))(S_i \in Dom(x) \wedge (\exists r \in Sch(R) - \{x\})(\exists s \in Sch(S_i))(r \Leftrightarrow s))$ then
   (c) Directly use SQL command to retrieve all the data from R.
   (d) For each $S_i \in S$, directly retrieve and union all the data from S_i by the following SQL command:

      SELECT K_{S_1}, 'S_1' as r, s FROM S_1 UNION
      SELECT K_{S_2}, 'S_2' as r, s FROM S_2 UNION
      …
      SELECT K_{S_i}, 'S_i' as r, s FROM S_i UNION
      …
      SELECT K_{S_k}, 'S_k' as r, s FROM S_k

The resolving procedures can be derived analogously by Transact-SQL with the XML extension in Transact-SQL. Due to the space limitations, we leave this part to the gentle readers.

5. Summary and Future Directions

We have elaborated a heterogeneous databases integration scheme for data warehouse creations through XML technologies. The mapping from heterogeneous database schemas to XML documents can be prepared according to the proposed procedures. The whole process can be smoothly implemented in a seamless manner.

The contribution of our work can be summarized as follows.

1. *Simplicity and Flexibility*—We proposed a simple and symmetric mapping scheme between database schemas and XML documents to build loosely-coupled data warehousing systems over the Internet. The mapping is shown to be effective and any contemporary DBMS product supporting XML capabilities can be adopted to implement the whole process.
2. *Semantics and Efficiency*—By employing the XSLT, each local site only has to prepare their metadata and the XSLT files according to the local database semantics and the global schema, respectively.

The integration of heterogeneous databases can be regarded as a *vertical* integration of pre-existing databases. From the other point of view, we have to work toward the *horizontal* integration of heterogeneous databases, which corresponds to inter-organizational workflow streamline processes. In the next step, we intend to enhance the proposed approach to manipulate XML documents over Web workflow applications in a more complete and subtle way.

Besides, since our work does not take the data types into account. It just addresses the data transformation framework. To make the integration of heterogeneous databases more flexible, several problems remain to be further studied. For example, how to employ XML schema (http://www.w3.org/XML/Schema; Roy and Ramanujan 2001) to enrich the integration result needs to be further investigated.

References：

ACM Computing Surveys, A Special Issue on Heterogeneous Database, (22:3), Sep. 1990.

Agarwal, S., A.M. Keller, G. Wiederhold, and K. Saraswat, "Flexible Relation: An Approach for Integrating Data from Multiple, Possibly Inconsistent Databases," *Proc. IEEE Int'l Conf. Data Eng.* 1995, pp. 495-504.

Batini, C., M. Lenzerini, and S.B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys* (18:4), 1986, pp. 323-364.

Breitbart, Y., P.L. Olson, and G.R. Thompson, "Database Integration in a Distributed Heterogeneous Database System," *Proc. IEEE International Conference on Data Engineering*, 1986, pp. 301-310.

Breitbart, Y., "Multidatabase Interoperability," *SIGMOD Record* (19:3), 1990, pp. 53-60.

Bukhres, O.A. and A.K. Elmagarmid (Eds.), *Object-Oriented Multidatabase Systems: Solutions for Advanced Applications* (Prentice-Hall, 1996).

Castano, S., V. De Antonellis, and S. De Capitani di Vimercai, "Global Viewing of Heterogeneous Data Sources," *IEEE Trans. Knowledge & Data Engineering* (13:2), 2001, pp. 277-297.

Chamberlin, Don, "XQuery: An XML query language," *IBM Systems Journal* (41:4), 2002, pp. 597-615.

Czejdo, B., M. Rusinkiewicz, and D.W. Embley, "An Approach to Schema Integration and Query Formulation in Federated Database Systems," *Proc. IEEE Int'l Conf. on Data Engineering*, 1987, pp. 477-484.

Date C.J., "The Outer Join," *Proc. International Conference on Databases—ICOD'83*, Churchill College, Cambridge, 1983, pp. 76-106.

Dayal, U. and H.Y. Hwang, "View Definition and Generalization for Database Integration in a Multi-database System," *IEEE Trans. on Software Engineering* (10:6), 1984, pp. 628-644.

Deen, S.M., R.R. Amin, and M.C. Taylor, "Data Integration in Distributed Databases," *IEEE Trans. on software Engineering* (13:7), 1987, pp. 860-864.

DeMichiel, L.G., "Resolving Database Incompatibility: An Approach to Performing

Relational Operations over Mismatched Domains," *IEEE Trans. Knowledge and Data Engineering* (1:4), 1989, pp. 485-493.

Elmargamid, A. and C. Pu (Eds.), *ACM Computing Surveys*, (Special Issue on Heterogeneous Databases) (22:3), 1990.

Grant, J., "Partial Values in a Tabular Database Model," *Information Processing Letters* (9:2), 1979, pp. 97-99.

Grant, J., W. Litwin, N. Roussooulos, and T. Sellis, "Query Languages for Relational Multidatabases," *The International Journal on Very Large Data Bases—The VLDB Journal* (2:2), 1993, pp. 153-171.

Heimbigner, D., D. McLeod, "A Federated Database Architecture for Information Management," *ACM Transactions on Office Information Systems* (3:3), 1985, pp. 253-278.

Hsiao, D., "Tutorial on Federated Databases and Systems (Part 1)," *The International Journal on Very Large Data Bases—The VLDB Journal* (1:1), 1992, pp. 127-179.

Hsiao, D., "Tutorial on Federated Databases and Systems (Part 2)," *The International Journal on Very Large Data Bases—The VLDB Journal* (1:2), 1992, pp. 285-322.

IEEE Computer, A Special Issue on Heterogeneous Distributed Database Systems, (24:12), Dec., 1991.

Inmon, W.H., *Building the Data Warehouse*, New York, NY: John Wiley and Sons, 1993.

Inmon, W.H. and C. Kelley, "The 12 Rules of Data Warehouse for a Client/Server World," *Data Management Review* (4:5), 1994, pp. 6-16.

Kim, W., and J. Seo, "Classifying Schematic and Data Heterogeneity in Multidatabase Systems," *IEEE Computer* (24:12), 1991, pp. 12-18.

Kimball, R., *The Data Warehouse Toolkit*, John Wiley and Sons, Inc., 1996.

Krishnamurthy, R., W. Litwin, and W. Kent, "Language Features for Interoperability of Databases with Schematic Discrepancies," *Proc. ACM SIGMOD—International Conference on Management of Data*, 1991, pp. 40-49.

Lee, C., C.-J. Chen, and H. Lu, "An Aspect of Query Optimization in Multidatabase Systems," *ACM SIGMOD Record* (24:3), 1995, pp. 28-33.

Lee, Kyong-Ha, *et al*., "Conflict classification and resolution in heterogeneous information integration based on XML schema," *Proc. of IEEE TENCON'02*, 2002.

Litwin, W. and A. Abdellatif, "An Overview of the Multi-Database Manipulation Language MDSL," *Proc. of the IEEE* (75:5), 1987, pp. 621-632.

Litwin, W., A. Abdellatif, B. Nicolas, Ph. Vigier, and A. Zeronnal, "MSQL: A Multidatabase manipulation language," *Information Sciences: An International Journal* (49:1), 1987, pp. 59-101.

Motro, A., Superviews: Virtual Integration of Multiple Databases, *IEEE Trans. on Software Engineering* (13:7), 1987, pp. 785-798.

Rys, M., "Bringing the Internet to your database: using SQL Server 2000 and XML to Build Loosely-Coupled Systems," *Proc. IEEE International Conf. on Data Engineering*, 2001, pp. 465-472.

Reddy, M.P., B.E. Prasad, P.G. Reddy, and A. Gupta, "A Methodology for Integration of Heterogeneous Databases," *IEEE Transactions on Knowledge and Data Engineering* (6:6), 1994, pp. 920-933.

Roy, J. and A. Ramanujan, "XML schema language: taking XML to the next level," *IEEE IT Professional* (3:2), Mar/Apr 2001, pp. 37-40.

Sciore, E., M. Siegel, and A. Rosenthal, "Using Semantic Values to facilitate Interoperability among Heterogeneous Information Systems," *ACM Trans. Database Systems* (19:2), June, 1994, pp. 254-290.

Siegel, M. and S.E. Madnick, "A Metadata Approach to Resolving Semantic Conflicts," *Proc. 17th International Conference on Very Large Data Bases (VLDB)*, 1991, pp. 133-145.

Srivastava, J. and P.Y. Chen, "Warehouse Creation—A Potential Roadblock to Data Warehousing," *IEEE Trans. Knowledge & Data Eng.* (11:1), 1999, pp. 118-126.

Trisolini, S. M., M. Lenzerini and D. Nardi, "Data Integration and Warehousing in Telecom," *Proc. the 1999 ACM SIGMOD Int'l Conf. on Management of Data*, 1999, pp.538-539.

Tseng, F.S.C., A.L.P. Chen, and W.P. Yang, "Searching a Minimal Semantically-Equivalent Subset of a Set of Partial Values," *The International Journal on Very Large Data Bases—the VLDB Journal* (2:4), (1993) pp.489-512.

Tseng, F.S.C., A.L.P. Chen and W.P. Yang, "Answering Heterogeneous Database Queries with Degrees of Uncertainty," *Distributed and Parallel Databases—An International Journal* (1:1), 1993, pp.281-302.

Tseng, F.S.C., A.L.P. Chen and W.P. Yang, "Refining Imprecise Data by Integrity Constraints," *Data and Knowledge Engineering* (11:3), 1993, pp. 299-316.

Tseng, F.S.C., A.L.P. Chen, and W.P. Yang, "Implementing the Division Operation on a Database Containing Uncertain Data," *Journal of Information Science and Engineering* (12:1), 1996, pp. 51-78.

Tseng, F.S.C., J.J. Chiang and W.P. Yang, "Integration of Relations with Conflicting Schema Structures in Heterogeneous Database Systems," *Data & Knowledge Engineering* (27:2), 1998, pp. 231-248.