Data Integration for XML based on Semantic Knowledge

Kamsuriah Ahmad[a], Ali Mamat[b], Hamidah Ibrahim[c] and Shahrul Azman Mohd Noah[d]

[a,d]Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia,
Tel:03- 89216730 Fax :03- 89296184,
E-mails: {kam, samn}@ftsm.ukm.my

[b,c]Fakulti Sains Komputer dan Teknologi Maklumat, Universiti Putra Malaysia,
Tel: 03-89466557 Fax:03-89466576
E-mails: {ali, hamidah} @fsktm.upm.edu.my

ABSTRACT

Reconciling of knowledge from multiple heterogeneous data sources has been a major focus of database research for more than a decade. As a standard for exchanging business data on the WWW, XML should provide the ability of expressing data and semantics among them. Since most of application data are stored in relational databases due to its popularity and rich development experiences over it. Therefore, how to provide a proper mapping approach from relational model to XML model becomes the major research problem in the field of current information exchanging, sharing and integration.. The model needs to be integrated and at the same time maintain the semantic knowledge among the data. The aim of this paper is to provide an overview for XML based data integration on semantic knowledge. At the end of the paper, we review some methodologies from existing literature.

Keywords:

Relational model, XML model, heterogeneous data source, data integration, semantic knowledge

1.0 INTRODUCTION

Heterogeneity of database arises from storing types of data, representing data differently (even using the same data model) and using different technologies to manipulate information. With the widespread acceptance of the Web as the primary vehicle for data interchange, the need to integrate a wide variety of formats and data on the Web has renewed interest in semistructured data integration and has contributed to the emergence of XML (E**X**tensible **M**arkup **L**anguage) as a standard format. Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data. Recently, XML has emerged as the de facto standard for data formats on the web. The use of XML as the common format for representing, exchanging, storing, and accessing data poses many new challenges to database systems. Since the majority of everyday data is still stored and maintained in relational database systems, we expect that the needs to convert data formats between XML and relational models will grows substantially. To this end, several schema conversion algorithms have been proposed (Deutsch et al,1998, Florescu et al 1999; Shanmugasundram et al,1999, Carey et al,2000). Although they work well for the given applications, the XML-to-Relational or Relational-to-XML conversion algorithms only capture the structure of the original schema and largely ignore the hidden semantic constraints. This paper focuses on the development of a data integration algorithm based on XML DTD as a global schema and at the same time maintain the semantic knowledge among the data. The semantic knowledge in this context means the study of relationships among the data at the semantic level. In this paper, we discuss the semantic knowledge that needs to be captured during transformation to ensure a correct relational schema. Compared to existing methods (Xia Yang et al, 2003), our works is only focus on the mapping from XML to relational. The rest of the paper will organize as follows; the second section will explain the difference between XML data model and relational data model, the third section will discuss an overview of the integration system, at the end of paper we make some conclusion and discuss further activities in our project.

2.0 RDB/XML BASICS

Relational databases are famous for their searching capability using SQL and for their impressive querying performance using indexes. They store data efficiently and with no redundancy because each unit of information is saved in only one place (normalization). Their reliability and scalability is unequeled and thy can be accessed by a very large number of concurrent users. IN addition, they have strong management and security features with locking and caching mechanisms. While XML is still inefficient as a data storage and access mechanism, XML strengths lie in different areas: It is text-based, humanreadable, platform-independent, and an open standard. In fact, it has becomne the lingua franca of all IT systems. Its self-describing nature allows it to represent structured data without any aditional information. Its structure is "inherent" to the XML document, making its transmission, validation, and presentation more intuitive than any other data standard. Looking at mapping XML to a relational database, two different data structures need to be considered.

2.1 Relational Database Structure

A relational database is best described as a database schema that is defined by the different entities (tables) created to meet the business requirements. It is represented by a set of flat tables linked together by one-to-one, one-to-many, or many-to-many relationships. Each table is made of a fixed collection of columns (also called fields) corresponding to attributes of the data model. An indefinite number of rows (or records) occurs within each table. A relational database supports many data types. Each table is characterized by a unique primary key, which other tables can refer to when using a foreign key that holds the same value. Those foreign keys serve, in turn, as primary keys for the same tables. Designing relational tables' relationships to make the database answer all the necessary business questions and normalize all the data can become quite elaborate because each piece of non-key data needs to be present in only one place (normalization).

2.2 XML Data Structure

In contrast, XML data is best represented by a tree structure, where relationships between elements are expressed mostly by containment. Each node of the tree is expressed by an XML element, which can hold one or more attributes as well as other elements. The degree of complexity of this tree is represented by a DTD or an XML schema. XML's strength may also be its weakness. It can be a versatile format with a loose structure that can represent any data element almost anywhere. By contrast, it can also appear as a rigid hierarchical structure in other applications. However, XML is less natural in representing relational data because it is difficult to describe the set of relationships that can exist between relational tables in an XML document. The same constraints that make RDBMS so efficient are not expressed easily in XML. In addition, the verbosity and formating looseness of XML are quite opposite to the performance strategies of RDBMS. Therefore, mapping an XML document to a relational database is complex in most cases. It also requires some expertise in both RDBMS and XML data design and a clear method to map each element to the XML schema to the database schema.

3.0 Relational Versus Hierarchical Representation Of The Data

As XML has developed over the past few years, its role has expanded beyond its original domain as semantic-preserving mark up language for online documents, and it is now also the de facto formats for interchanging data between heterogeneous systems. XML is a markup language much like HTML but it was not meant to replace it. In future Web development will use XML to describe the data, while HTML will be used to format and display the same data.

The basic component of an XML document is the element, that is, a piece of text bounded by matching tags such as <book> and </book>. For example, consider the following piece of XML data:

```
<book>
        <title>XML Databases</title>
        <year>2003</year>
        <author>C.J.Date</author>
</book>
```

Figure 1 : An example of XML Document

The structure of an XML document can be described by a Document Type Definition (DTD). A DTD provides a list of elements and attributes contained in a document and the relationships between them. The following DTD specification describes the structure of the previous XML example :

```
<!ELEMENT book(title, year,author) >
<!ELEMENT title          (#PCDATA) >
<!ELEMENT year           (#PCDATA) >
<!ELEMENT  author        (#PCDATA) >
```

Figure 2 : An example of XML DTD

Now, let us consider the following DTD that models conference proceedings :

<!ELEMENT book(title, year, author,paper)>
<!ELEMENT paper(paperid, title,abstract)

Suppose the combination of title and year uniquely identifies the conf. Using the hybrid inlining algorithm (Shanmugasundram et al, 1999) the DTD would be transformed to the following relational schema:

book (title, year, author)
paper (paperid, title, book_title, book_year, abstract)

While the relational schema correctly captures the structural aspect of the DTD, it does not enforce correct semantics. For instance, it cannot prevent a tuple t_1: paper (111,'XML ..', 'XML2RDB',2005,'….') from being inserted. However, tuple t_1 is inconsistent with the semantics of the given DTD since the DTD implies that the paper cannot exist without being associated with a BOOK and there is apparently no YEAR '2005' yet. In database terms, this kind of violation can be easily prevented by an inclusion dependency saying

"paper[book_title, book_year] $\subseteq$ book[title, year]".

The reason for this inconsistency between the DTD and the transformed relational schema is that most of the proposed conversion algorithms, so far, have largely ignored the hidden semantic knowledge of the original schema.

To understand the difference between the XML object model and relational object model, it is easy to first look at some simple examples. To start, notice that there is an obvious difference between XML document as we described above in Figure 1 with the one below in the form of row in a table:

Table name : BOOK

Title	Year	Authors

Figure 3 : An example of Table structure in Relational Database

Similarly, there is an obvious mapping between the following element type definition in Figure 2 and table schema:

```
CREATE TABLE BOOK (
    TITLE        VARCHAR(10)  NOTNULL
    YEAR         VARCHAR(10) NOT NULL
    AUTHORS   VARCHAR (10) NOT NULL
)
```

Figure 4 : An example of schema definition in Relational Database

4.0 DATA INTEGRATION ARCHITECTURE

The goal of data integration is to allow data from different sources to be used cooperatively. As pointed out (Sheth A.,1989), the key to succesful data integration is the identification of interschema relationship. The more expressive the underlying data models, the higher the chances of identifying interschema relationships and hence the easier the task of data integration. Manually integrating data from different sources, however is extremely labor intensive. Data integration will integrate data from different sources into a common view. Thru data integration systems provides a uniform query interface to a multitude of data sources, thereby freeing the user from the tedious job of accessing the individual sources, querying them, and manually combining the answers. The figure below illustrates the big picture of data integration in heterogeneous environment.

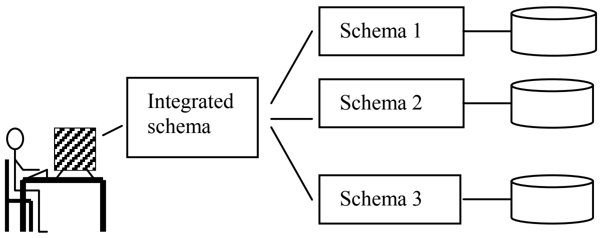

Figure 5 : An overview of the Integration System

Consider for example a data integration system that helps users find books from various databases. The system provides the uniform interface in terms of an integrated schema that captures the relevant aspects of the publication domain. The integrated schema may contain elements such as TITLE, YEAR and AUTHORS. The system maintains for each data source a source schema that describes the content of the source. An agent that attached to each data source, handle data formatting transformation and mapping between the local data model and the data model in integration system.

Given a user query formulated in the integrated schema, such as " FIND BOOKS THAT WAS PUBLISHED ON 2000", the system translate the query into queries in the source schema, executes these queries with the help of the wrappers, then combines the data returned by the sources to produce the final answers.

To translate user queries, a data integration system uses semantic mapping between the integrated schema and the local schemas of the data sources. A schema conversion algorithm is needed to convert if the local data models are different from the mediated schema. While a schema mapping algorithm is used if the local data models are the same with the mediated schema.

Consider again the above example. The data about its TITLE, YEAR, and AUTHORS are stored in a XML database, whose schema is represented as a XML data model, DTD. The DTD conforms to the standard recommended by the marketplace participants. In a more complex scenario is that the company strikes a partnership with another online merchant in a slightly different data model such as relational database (due to its popularity and rich development experiences), whose schema is represented as a model RDB. The relational data is then need to be mapped into XML DTD using CONVERT algorithm, so that it can serve the data onto their web site and to share data in a broader marketplace of online merchants. This algorithm will converts data between two different model by translating data from a relational database into XML. Suppose there are another company want to share the data but have slightly different DTDs, since nobody fully conforms to the standards (and even if they do, there are many versions of the same standard). This DTD need to match with the standard DTD at the mediated schema, this can be done through MATCH algorithm. The above scenario can be visualize in the figure below:

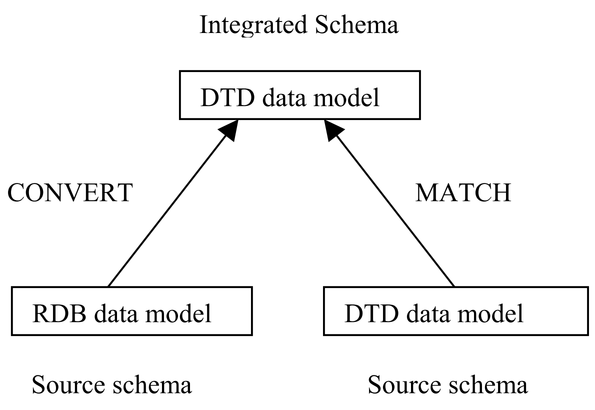

Figure 6 : Operations on schema

5.0 THE MAPPING BETWEEN MODELS

Typically, during integration process, a graph model is used as an internal representation of schemas. Hence, schema elements may either be represented by nodes or paths in the schema graphs which also impact the representation of schmea mapping. As a start, let us consider the simplest Relational-to-XML conversion algorithm, termed as FT (Flat Translation). This algorithm is to convert tables in a relational schema to elements in an XML schema. FT is a simple and effective translation algorithm since it only convert the "flat" relational model to a "flat" XML model in a one-to-one manner. There are more complex conversion problems such as one-to-many and many-to-many relationships. Figure below shows a simple match problem with two small schema in directed graph representation.

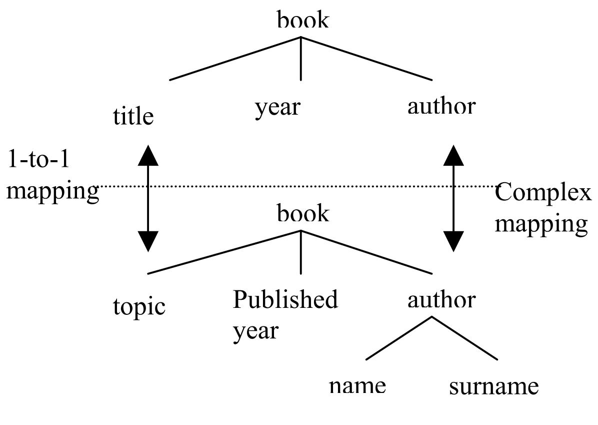

Figure 7 : Schema mapping example

Let say, the above graph represent schema M_1 from relational schema with three attributes: TITLE, YEAR, AUTHOR. While global schema M_2 which is the targeted schema in XML based with four attributes: TOPIC, PUBLISHED-YEAR, NAME, SURNAME. Schema M_1 need to transformed to M_2. From the graph it is much easier to map the attribute TITLE from one tree representation to the attribute TOPIC from another tree representation because it is straight forward one-to-one mapping. But it is much harder to map an attribute AUTHOR with composite of NAME and SURNAME from another tree representation because of complex mapping involved.

However (Doan, 2000) provides a powerful matching solution that can exploit multiple types of information. It discovers 1-to-1 semantic mappings by using machine learning techniques to match a new data source against a previously determined global schema. Given a user-supplied mapping from a data source to the global schema, the pre-processing step looks at instances from that data source to train the learner, thereby discovering characteristic instance patterns and matching rules. These patterns and rules can then be applied to match other data sources to the global schema. The system managed to reach 77% accuracy level.

6.0 CONCLUSION

Integrating existing databases is certainly not an easy task. Still, it is something that enterprises probably cannot avoid if they want to launch new applications or to reorganize the existing information system for a better profit. In this paper, we have discussed an overview of data integration. As been expressed in the paper that the key to data integration is the identification of inter schema relationships. The more semantic properties of the data captured by the underlying data model, the higher the chances of identifying inter schema relationships and hence the easier the task of data integration. As a problem in data warehouse, a web of data coming from multiple sources must be transformed to data conforming to a single target schema, to enable further data analysis. Even from conceptual level, data might present in the form of relational schemas, ontologies, and XML DTDs. In this scenario the application must establish semantic mappings among the representations. The mapping will capture relationships between models such as transformation and matchings. We also highlight some examples regarding how to map from XML to relational. But several important problems remain to be investigated. Examples are integration of complex objects and integration of integrity constraints. Through this studies we hope that will give some contribution to the database community.

7.0 REFERENCE

Dongwon L et al. (2002) *Effective Schema Conversions Methods between XML and Relational Models*. European Conference on Artificial Intelligence, France

Sun Hongwei (2002). *Constraints-Preserving Mapping Algorithm from XML-Schema to Relational Schema*. Engineering and Deployment of Cooperative

Information Systems: First International Conference, EDCIS: Springer-Verlag Heidelberg. 193-207

Doan A., Domingo P., Levy A. (2000). *Learning Source Description for Data Integration*. Proceedings of Web Data Base.55-65

Deutsch et al. (1998). *Storing Semistructured Data with STORED.* In ACM SIGMOD International Conference on Management of Data.

Florescu et al. (1999). *Storing and Querying XML Data Using an RDBMS.* IEEE Data Engineering Bulletin 22: 27-34.

Shanmugasundaram et al (1999*). Relational Databases for Querying XML Documents: Limitation and Opportunities.* In International Conference on Very Large Data Bases.

Carey et al (2000). XPERANTO*:Publishing Obejct-Relational Data as XML.* In International Workshop on the Web and Databases.

Seth A. and Gala A. (1989). Attribute relationships: An Impediment in Automating Schema Integration. In Proceedings of the NSF Workshop in Heterogeneous Databases.

Xia Yang et al (2003). Resolving Strcutural Conflicts in the Integration of XML Schemas: A Semantic Approach. In Proceedings of the Entity Relationships Conference. pg 520-533.