XML 在階層式資料庫資料交換之應用

An Application of XML on Hierarchical Database for Data Exchange

Jeang-Kuo Chen

Department of Information Management
Chaoyang University of Technology,
Wufeng, Taichung Country, Taiwan,
R.O.C.

Email: jkchen@mail.cyut.edu.tw

Min-Jane Liu

Department of Information Management
Chaoyang University of Technology,
Wufeng, Taichung Country, Taiwan,
R.O.C.

Email: s9054618@mail.cyut.edu.tw

摘要

XML 已被廣泛應用在許多領域，尤其是在資料庫的應用上，諸如企業或組織之間的資料交換。許多資料交換的應用是集中在關聯式資料庫，但是至今仍然有一些階層式資料庫被使用在商業上。在本篇文章中，我們提出一個改進的階層式資料庫的資料交換模型[3]。在這一個新的模型中，被交換資料的資料型態在傳遞的過程中並不會遺失。一個資料交換的情節用來展示如何使用此一資料交換模型，在兩個公司之間處理資料交換的過程。

ABSTRACT

XML has been widely applied to many areas especially for database applications, such as data exchange between enterprises or organizations. Many applications for data exchange focus on relational databases. However, hierarchical databases are still being used in some businesses. In this paper, we propose an improved version of the data exchange model [3] for hierarchical databases. With this new model, original data type would not lose in the process of data transmission. A scenario is illustrated to demonstrate the process of data exchange between two companies.

關鍵詞：XML, XML 綱要，階層式資料庫，DBD，資料交換

Keyword: XML, XML Schema, Hierarchical Database, DBD, Data Exchange

1. Introduction

Data exchange is gradually important in business. Typical media are HTML, SGML, XML, and PDF, while XML is the most popular one [2], [5], [9]. XML is applied widely in many applications, such as content publishing, data integration, etc [1], [5], [8], [9]. Many researches focused on data exchange using XML among relational databases [2], [4],

[7]. Today, relational databases are dominant in database field. However, there are still some alternatives can be chosen such as network database, object-oriented database, and hierarchical database (HDB). Among these alternatives, HDBs are still being used in some organizations or enterprises. Therefore, it is necessary to develop technologies for data exchange using XML among HDBs.

Proposed by W3C in 1998, the eXtensible Markup Language (XML) [11] has become a popular media for data exchange [3], [5], [9]. A data exchange model [3] was proposed to support data exchange between two corporations and the advantages of data exchanging between XML and HDBs have been discussed in [3]. Although data can be delivered by XML documents in this model, original data type of attributes in an XML document cannot be interpreted directly because all data are presented in character strings in an XML document. This situation may make unpredictable errors when interpreting the contents of an XML document. An improvement must be required to solve this problem.

The Document Type Definition (DTD) or XML Schema [12], [13], [14], proposed by W3C, can be used to structure XML documents, but the XML Schema provides more data types than DTD does. The corresponding DTD and XML Schema of the XML document in Figure 1 are shown in Figure 2a and 2b, respectively. In Figure 2a, the keyword “CDATA” means character data. If this DTD is used to validate the XML document in Figure 1, the values of attribute “stdID” are interpreted as character strings “101” and “102.” An error may occur if the original data are integer 101 and 102. On the contrary, the XML Schema in Figure 2b can correctly identify the two integers in Figure 1 because it uses the keyword “integer” to interpret the values of the attribute “stdID.” Therefore, the XML Schema is the best choice to associate with an XML document if XML contains non-character data. Detailed discussions about XML Schemas will be introduced in section 2.4.

An HDB is composed of several segments. Each segment contains one or more fields. Like an XML, an HDB can have an associated file called Database Description Block (DBD) [6] to describe the structure of the HDB, the schemas of segments, and the data type of fields. In this paper, two modules are proposed to improve the data exchange model in [3]. One module, called DBDToXSchema, can convert a DBD file into an XML Schema file. The other module, called XSchemaToDBD, can recover a DBD file from an XML Schema file. By this improved data exchange model, HDB data of one enterprise can be well transmitted to the other enterprise without worrying the missing of original data type.

The paper is organized as follows. In Section 2, XML, XML Schemas, hierarchical database, and DBD of HDB are introduced briefly. In Section 3, an improved data exchange model is proposed for data transmission between XML document and HDB. Section 4 illustrates a scenario to demonstrate how the model works. Finally, Section 5 makes a conclusion

2. Previous work

2.1. Hierarchical Databases and the Schema Data Description Language

Based on the hierarchical data model, an HDB is a tree-like data structure. Some commercial HDBs have been used for many years. The most famous one is IBM's IMS [6]. Interested readers are referred to [3] for a more detailed discussion on this subject. One of the most difficult areas of database management is the design of information structure [10]. To describe and manipulate data, the Database Task Group (DBTG) of CODASYL specified the Schema Data Description Language (DDL) for describing the data structure of database. The DDL of IMS is named as Database Description Block (DBD) [6]. The structure of a DBD is described as followed. DBD name is first defined; then, segment followed by field are defined sequentially. The DBD name is the same as the HDB name. Defined by the keyword "SEGM," each segment contains the segment name (the "NAME"), the total byte number of the fields that constitute the segment (the "BYTES"), and the parent segment name of the segment (the "PARENT"). Defined by the keyword "FIELD," each field has three items: the field name (the "NAME"), the data type (the "TYPE"), and the data length (the "BYTES").

A DBD example is illustrated in Figure 3. The DBD is declared as "coursedb" containing the root segment "department." The segment "student," defined below the keyword "SEGM," has three fields that occupy 30 bytes and its parent is segment "department." The three fields of the segment "student" are "stdID," "firstname," and "lastname" where the length of field "stdID" is 10 bytes and the data type of field "stdID" is zoned-decimal.

2.2. XML documents and XML Schemas

The XML [11], proposed by W3C, is an abbreviation of eXtensible Markup Language derived from SGML. As a descriptive language, XML can represent both data and structure. We refer interested readers to [3] which address the issue in more depth than possible in this paper. The XML Schema [12], [13], [14] is published by W3C and become a recommendation in May 2001. An XML Schema is used to describe and constrain the contents of XML documents. An XML document is valid if it conforms to the constraint rules of a particular XML Schema. The XML Schema is an auxiliary file of robust-typed schema definition. A range of primitive data types such as string, decimal, and integer are supported in an XML Schema [14]. With an XML Schema, any ambiguity can be excluded when an XML document is interpreted. The syntax of XML Schema is similar to that of XML.

The XML Schema in Figure 2b is the associated file of the XML document in Figure 1. An XML Schema starts with the keyword "schema." The attribute name is declared by the keyword "attribute" in an XML Schema. The data type of attribute value is specified by the keyword "restriction" and, if necessary, plus another keyword "length." The Document Type Definition (DTD) can also be used to describe the structure of an XML document, but DTD does not supply rich data types for various data as XML Schema does. That is why XML Schema, rather than DTD, is selected to be

associated with an XML document in our approach.

3. The Data Exchange Model

Data exchange between two enterprises or organizations is a process described as follows. First, the specified data is transformed into a medium in the *source unit* (enterprise or organization). Then the medium is transmitted via network from the *source unit* to the *destination unit*. Finally, the medium is transformed into the original data in the *destination unit*. The architecture of the data exchange model is shown in Figure 4. In this model, the specified data is extracted from the HDB and its related DBD of *Source Unit*. The module *HtoX* is a transformer used to convert the exchanged data into the medium, an XML document. The related DBD is transformed into an XML Schema by *DBDToXSchema* module. When an XML document and XML Schema are created in S*ource Unit*, they will be transmitted to *Destination Unit* via network. When *Destination Unit* receives the XML document and the associated XML Schema, they are converted into the original data by the modules *XSchemaToDBD* and *XtoH,* respectively, and then saved to the HDB of *Destination Unit*. The modules *HtoX* and *XtoH* were proposed in [3]. The *DBDToXSchema* and *XSchemaToDBD* are described in sequence.

3.1. DBDToXSchema

The algorithm of *DBDToXSchema*, used to transform a DBD to an XML Schema, is listed in the following steps:

S1: insert schema tag information to the XML Schema file

S2: read the DBD name in the DBD file and write it as the root element to the XML Schema file

S3: read each segment's information in the DBD file and write it to the XML Schema file, and declare it as *complexType*

S4: read all fields of the segment in Step S3 and write them to the XML Schema file as attributes

S5: declare a new *simpleType* in the XML Schema file for each data type in the field of step S4 in DBD file

S6: repeat S3 to S5 until all segments in the DBD file are processed

The DBD name is used as the root element name in an XML Schema. Every segment in a DBD is mapped as an element into an XML Schema; the enclosed fields of the segment are mapped as attributes. For each field, a *simpleType* declaration is used to define its data type and length in step S5. The hierarchical structure of elements in the XML Schema is the same as that of the ordered sequence of segments in the DBD.

3.2. XSchemaToDBD

After receiving an XML Schema file from the source unit, the destination unit uses the *XSchemaToDBD* module to transform the XML Schema file into a DBD. The algorithm of *XSchemaToDBD* is listed in the following steps:

S1: read the root element in the XML Schema file and write it as the DBD name to the DBD file

S2: read each sub-element in the XML Schema file and write it as a segment to the DBD

file

S3: if a sub-element has a parent element then set the value of parent item to the name of its parent segment in the DBD file, otherwise set the value of parent item to 0

S4: read all attributes of sub-elements in the XML Schema file and write them as fields to the DBD file

S5: compute the total byte numbers of fields and fill the value into the “BYTES” item in the segment of the DBD file

S6: repeat S2 to S5 until the end of the XML Schema file

The root element in an XML Schema is mapped as DBD name into the transformed DBD file. Every element in the XML Schema is converted as a segment into the DBD file. The information of the parent-child relationships between elements is unavailable directly. The *XSchemaToDBD* model needs to scan all elements in an XML Schema to identify the hierarchical structure. Step S5 is used to calculate the total length of a segment after reading the information of all attributes of an element.

4. A Data Exchange Scenario

An illustration is given to introduce the application of the data exchange model between two banks. In this scenario, assume that a bank branch A wants to transfer customers’ account information to branch B. Both A and B use HDB to store their data and agree to exchange data by XML. Initially, there may have a customer’s accounting information, as shown in Figure 5, in the HDB of A. Inputting the HBD and DBD,a simple segment definition is shown in Figure 6 to *DBDToXSchema* and *HtoX* modules, respectively, can produce the XML document, as shown in figure 7 and the corresponding XML Schema of the figure 6 is shown in figure 8. Then the XML Schema and XML document are transmitted to B. Applying the modules *XSchemaToDBD* and *XToH*, B can transform the received the XML Schema and XML document into the original DBD and HDB. Likewise, B can transmit HDB data such as the credit data to A via XML. Both data and structure in HDB can be delivered between each other; hence, the two enterprises can use those algorithms to exchange data automatically without human engagement.

5. Conclusion

In this paper, we propose an improved version of the data exchange model [3] by adding two modules to perform transmission between a DBD and an XML Schema. Utilizing the two algorithms, *DBDToXSchema* and *XSchemaToDBD*, the computer can transform a DBD into an XML Schema and vice versa. Finally, a scenario is illustrated to demonstrate how this model works. The motivation of this paper is to improve the interpretative function of the old model for identifying various data types of original HDB data. The improved model not only can exchange data between HDBs and XML documents, but also DBD and XML Schema. Therefore, two enterprises now can exchange non-character data such as product price, employee salary, order quantity, etc. This promotes the utility of the data exchange model [3].

REFERENCE

[1] L. Aversano, G. Canfora, A. De Lucia, and P. Gallucci, “Integrating document and workflow management tools using XML and web technologies: a case study,” *Proceedings of Sixth European Conference on Software Maintenance and Reengineering*, pp. 24-33, 2002.

[2] E. Bertino and B. Catania, “Integrating XML and databases,” *IEEE Internet Computing*, Vol. 5, Issue: 4, 2001.

[3] J. K. Chen and M. J. Liu, "A Model for Data Exchange between XML document and Hierarchical Databases," *proceedings of the 2002 International Computer Symposium*, Dec. 18-21, Hualien, Taiwan, ROC, Vol. 5, Session 8, E8-1, 2002

[4] J. Fong, F. Pang, and C. Bloor, “Converting relational database into XML document,” *Proceedings of 12th International workshop on Database and Expert Systems Applications*, 2001.

[5] B. Hofreiter, C. Huemerm, and W. Klas, “ebXML: status, research issues, and obstacles,” *Proceedings of the 12th International Workshop, Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems*, pp. 7-16, 2002.

[6] IBM, *IMS Primer, http://www.redbooks.ibm.com*

[7] J. S. Kim, W. Y. Lee, and K. H. Lee, “The cost model for XML documents in relational database systems,” *ACS/IEEE International Conference, Computer Systems and Applications*, pp. 185-187, 2001.

[8] A. Renner, ”XML data and object databases: the perfect couple?,” *Proceedings of the 17th International Conference on Data Engineering*, pp. 143-148, 2001.

[9] M. Sundaram and S. S. Y. Shim, “Infrastructure for B2B exchanges with RosettaNet,” *The third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, pp. 110-119, 2001.

[10] R. W. Taylor and R. L. Frank,“CODASYL Data-Base Management System,” ACM Computing Surveys, Vol. 8, Issue: 1, pp. 67-103, 1976

[11] W3C, Extensible Markup Language (XML) 1.0 (second Edition) W3C Recommendation, 6 October 2000, *http://www.w3.org/TR/2000/REC-xml-20001006*, 2000.

[12] W3C, XML Schema Part 0: Primer W3C Recommendation, 2 May 2001, *http://www.w3.org/TR/2001/REC-xmlschema-0-20010502*, 2001.

[13] W3C, XML Schema Part 1: Structures W3C Recommendation, 2 May 2001, *http://www.w3.org/TR/2001/REC-xmlschema-1-20010502*, 2001.

[14] W3C, XML Schema Part 2: Datatypes W3C Recommendation, 2 May 2001, *http://www.w3.org/TR/2001/REC-xmlschema-2-20010502*, 2001.

```
<department>
   <student stdID=”101” firstname=”John”
lastname=”Smith”></student>
   <student stdID=”102” firstname=”Mary”
lastname=”Lin”></student>
</department>
```

Figure 1. An XML document example

```
<!ELEMENT department(student)>
<!ELEMENT student EMPTY>
<!ATTLIST student stdID CDATA
                 firstname CDATA
                 lastname CDATA>
```

Figure 2a. A DTD Example

```
<schema>
   <element name="department">
      <complexType>
         <element name="student">
            <complexType>
               <attribute name="stdID">
                  <simpleType>
                   <restriction base="integer"/>
                  </simpleType>
               </attribute>
               <attribute name="firstname">
                  <simpleType>
                     <restriction base="string">
                        <length value="10"/>
                     </restriction>
                  </simpleType>
               </attribute>
               <attribute name="lastname">
                  <simpleType>
                     <restriction base="string">
                        <length value="10"/>
                     </restriction>
                  </simpleType>
               </attribute>
            </complexType>
         </element>
      </complexType>
   </element>
</schema>
```

Figure 2b. An XML Schema example

```
DBD
   NAME=coursedb
SEGM
NAME=department,BYTES=20,PARENT=0
FIELD
   NAME=departID,BYTES=10,TYPE=Z
FIELD
   NAME=departname,BYTES=10,TYPE=C
SEGM
NAME=course,BYTES=30,PARENT=department
FIELD
   NAME=courseID,BYTES=10,TYPE=Z
FIELD
   NAME=coursename,BYTES=20,TYPE=C
SEGM
NAME=student,BYTES=30,PARENT=department
FIELD
   NAME=stdID,BYTES=10,TYPE=Z
FIELD
   NAME=firstname,BYTES=10,TYPE=C
FIELD
   NAME=lastname,BYTES=10,TYPE=C
SEGM
NAME=teacher,BYTE=40,PARENT=department
FIELD
   NAME=teacherID,BYTES=10,TYPE=Z
FIELD
   NAME=teachername,BYTES=30,TYPE=C
```

Figure 3. A DBD example

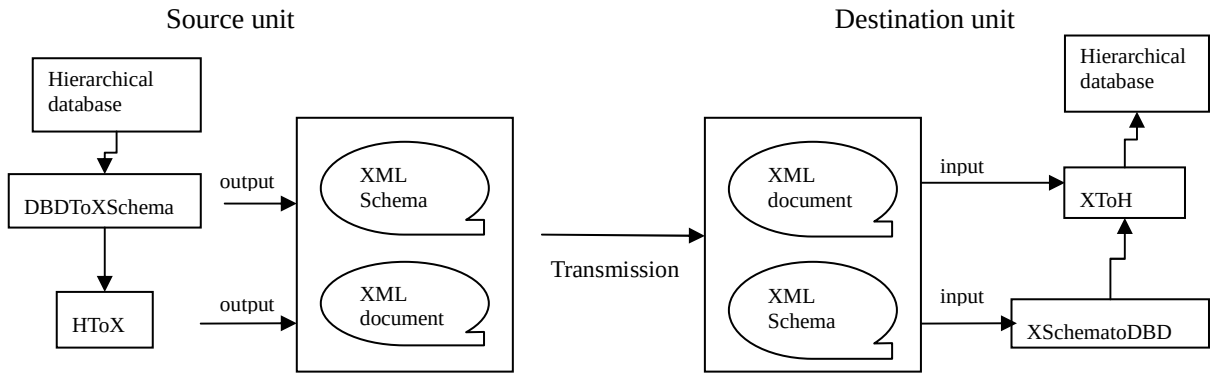

Figure 4. The data exchange model

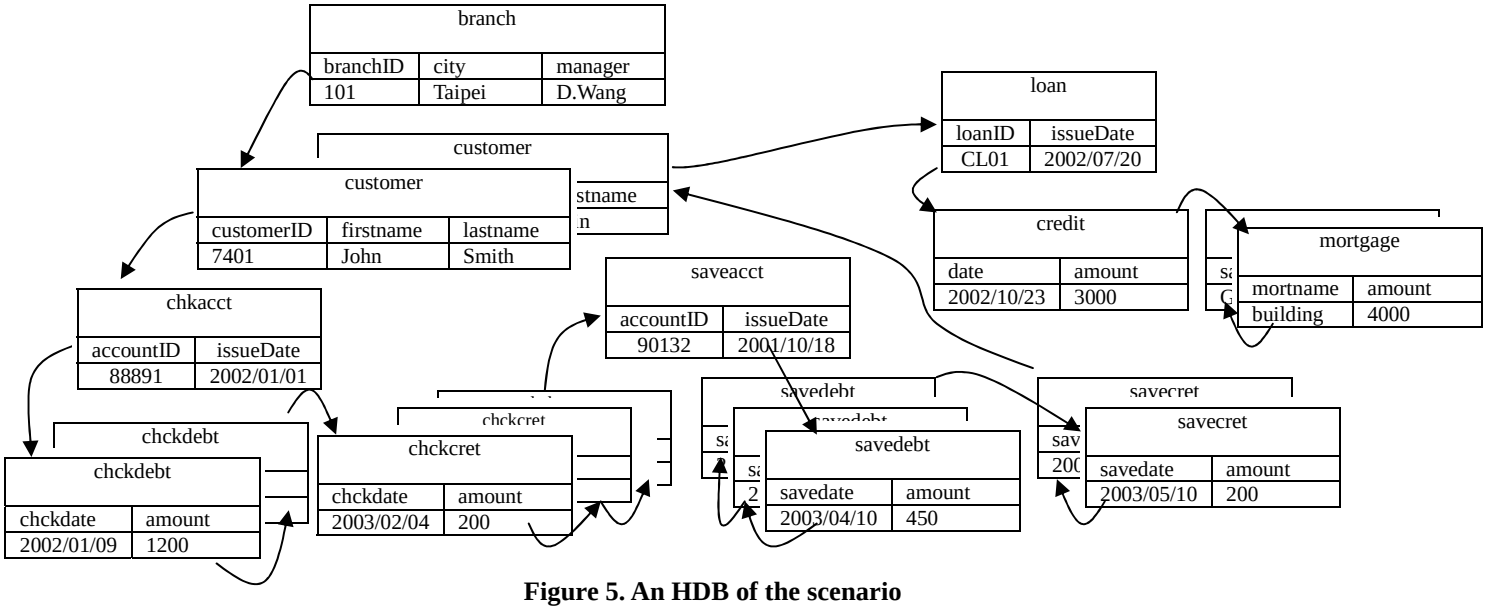

Figure 5. An HDB of the scenario

```
SEGM
   NAME=savedebt, BYTES=20, PARENT=saveacct
FIELD
   NAME=savedate, BYTES=10, TYPE=C
FIELD
   NAME=amount, BYTES=10, TYPE=F
```

Figure 6. A segment definition in the DBD of the senario

```
<element name="savedebt">
<complexType>
<attribute name="savedate"    use="required">
<simpleType name="domain_savedate">
<restriction base="string">
<length value="10"/>
</restriction>
</simpleType>
</attribute>
<attribute name="amount"    use="required">
<simpleType name="domain_amount">
<restriction base="float">
</restriction>
</simpleType>
</attribute>
</complexType>
</element>
```

Figure 8. A corresponding XML Schema of the Figure 6

```
<customerAccountInformation>
<branch branchID=”101” city=”Taipei” manager=”D. Wang”>
<customer customerID=”7401” firstname=”John”
lastname=”Smith”>
<chkacct accounted=”88891” issueDate=”2002/01/01”>
<chckdebt chckdate=”2002/01/09” amount=”1200”></chckdebt>
<chckdebt chckdate=”2002/02/20” amount=”3400”></chckdebt>
<chckcret chckdate=”2003/02/04” amount=”200”></chckcret>
<chckcret chckdate=”2003/04/01” amount=”130”></chckcret>
<chckcret chckdate=”2003/05/15” amount=”390”></chckcret>
<chkacct>
<saveacct accounted=”90132” issuedate=”2001/10/18”>
<savedebt savedate=”2003/04/01” amount=”450”></savedebt>
<savedebt savedate=”2003/05/07” amount=”630”></savedebt>
<savedebt savedate=”2003/06/01” amount=”400”></savedebt>
<savecret savedate=”2003/05/01” amount=”200”></savecret>
<savecret savedate=”2003/05/11” amount=”110”></savecret>
</saveacct>
</customer>
<customer customerID=”28891” firstname=”Mary”
lastname=”Lin”>
<loan loanID=”CL01” issuedate=”2002/07/20”>
<credit date=”2002/10/23” amount=”3000”></credit>
<mortgage mortname=”building” amount=”4000”></mortgage>
<mortgage mortname=”car” amount=”2200”></mortgage>
</loan>
</customer>
</branch>
</customerAccountInformation>
```

Figure 7. An XML document of the scenario