Workshop on Databases and Software Engineering

A Model for Data Exchange between XML Documents and Hierarchical Databases

J. K. Chen*
Department of Information Management
Chaoyang University of Technology, Wufeng, Taichung Country, Taiwan, R.O.C.
Email: jkchen@mail.cyut.edu.tw

M. J. Liu
Department of Information Management
Chaoyang University of Technology, Wufeng, Taichung Country, Taiwan, R.O.C.
Email: s9054618@mail.cyut.edu.tw

Abstract

In recent years, data exchange is popular among enterprises and organizations. The data may be structured or unstructured such as purchase orders, product catalogs, official documents and so on. XML, developed by W3C, has been extensively applied to many areas, especially for database applications. Most researches focused on data exchange between XML documents and relational databases. However, the hierarchical database is still used in some businesses. In this paper, we propose a model for enterprises or organizations to perform data exchange between XML documents and hierarchical databases. A scenario is illustrated to demonstrate the process of data exchange between two companies.

Keywords: XML, Hierarchical Database, Data Exchange

*To whom all correspondence should be addressed.

1. Introduction

Data exchange is gradually important in business. The exchange is performed with one or more media. Typical media are HTML, SGML, XML, and PDF [12], [20], [22]. Among them, XML is the most popular one [10], [11], [16], [21], [23]. The XML was introduced by W3C since 1998. An XML document contains both structure definition and data; in other word, it is self-contained. Logically, an XML document consists of elements and attributes. XML is applied widely in many applications, such as content publishing, data integration, etc [8], [17], [18], [19]. Many researches focused on data exchange between XML documents and relational databases such as [14], [17], [25]. Today, relational databases are dominant in the database field. However, there are still some alternatives can be chosen such as network databases, object-oriented databases and hierarchical databases (HDB). Among these alternatives, HDB are still used in some organizations or enterprises. Therefore, it is necessary to develop technologies for data exchange between XML and HDB.

In this paper, we propose a model for data exchange between XML and HDB. By this model, organizations or enterprises that use HDB can exchange data via XML documents. The model includes two important modules that transform data between HDB and XML. In the model, an XML document can be created by two manners. One is using the module to create an XML document. The other is directly creating an XML document without using the module. The details of the model will be described in Section 3. The advantages of the model can be described as follows.

1. Extend the utilization of HDB to approach relational databases.
2. Provide fast and robust data integration and communication to business partners.

The paper is organized as follows. In Section 2, XML and HDB are introduced briefly. In Section 3, a model is proposed for data transformation between XML documents and HDB. Section 4 illustrates a scenario to demonstrate how the model works. Finally, Section 5 makes a conclusion.

2. Previous Work

2.1. XML

The XML [1], proposed by W3C, is an abbreviation of eXtensible Markup Language derived from SGML. As a descriptive language, XML can represent both data and structure. This mechanism is suitable for data exchange via public networks. Some advantages of XML can be listed as follows.

1. It is public. An XML document is independent with any platform; it can be shared with any user or system.

2. The self-describing characteristic makes XML become a good and efficient choice for data exchange.

The specifications in [1], [2], [3], [4] define the basic structure of an XML document. An XML document is a well-formed one if it obeys the following rules.

1. There is only one root element.
2. Every start tag must have a closing tag.
3. Attribute's value must be quoted.
4. Elements must be nested properly.
5. Element's name is case sensitive.

An XML example is shown in Figure 1. The four strings: <studentenrollment>, <student>, <enrollment>, and <class> are called elements where <studetnenrollment> is the root element. The elements <student>, <enrollment>, <class> are the sub-elements of elements <studentenrollment>, <student>, <enrollment>, respectively. Each element must have a corresponding closing tag. For example, the string </student> is the closing tag of the element <student>. The string "stdid" is an attribute of the element <student> and its value is "John." Elements in an XML document have a hierarchical relation. XML documents are free formatted. Common element name sets, known as "vocabularies," are needed for semantic understand and data exchange.

```
<studentenrollment>
   <student stdid=”01” stdname=”John”>
      <enrollment id=”9101” date=”2002/01/01”>
         <class>Computer Sciense</class>
         <class>Data Structure</class>
      </enrollment>
   </student>
   <student stdid=”02” stdname=”Mary”>
      <enrollment id=”9102” date=”2002/01/01”>
         <class>Data Structure</class>
         <class>O.S.</class>
      </enrollment>
   </student>
</studentenrollment>
```

Figure 1: An XML document.

2.2. Hierarchical Database

Based on HDB, some commercial products have been used for many years. The most famous one is IBM’s IMS. Logically, an HDB is a tree-like data structure. A physical structure of HDB is a link list. In an HDB, there is a root node leveled 0. The levels of the child nodes of a node leveled *i* is *i*+1. A parent node may have one or more child nodes. In each node, several records with the same schema are grouped together and ordered from left to right. Each record in a parent node has a pointer to a record in the child node of the parent node. The depth-first search is used to traverse the nodes in an HDB. An HDB example is shown in Figure 2. In this example, “student” is the root node contains two records and has a child node “enrollment” contains two records. The child node “class” of node “enrollment” contains four records. The records in nodes “student,” “enrollment,” and “class” are ordered by the fields "stdid," "id," and "name," respectively. Some particular enterprises such as banking prefer HDB as their databases. The reasons are as follows. The data model of HDB is simple and easy to implement. HDB are suitable for handling one-to-many relationships of data. The performance of HDB is better than relational databases for some applications if these applications have fixed and predefined relationships of data.

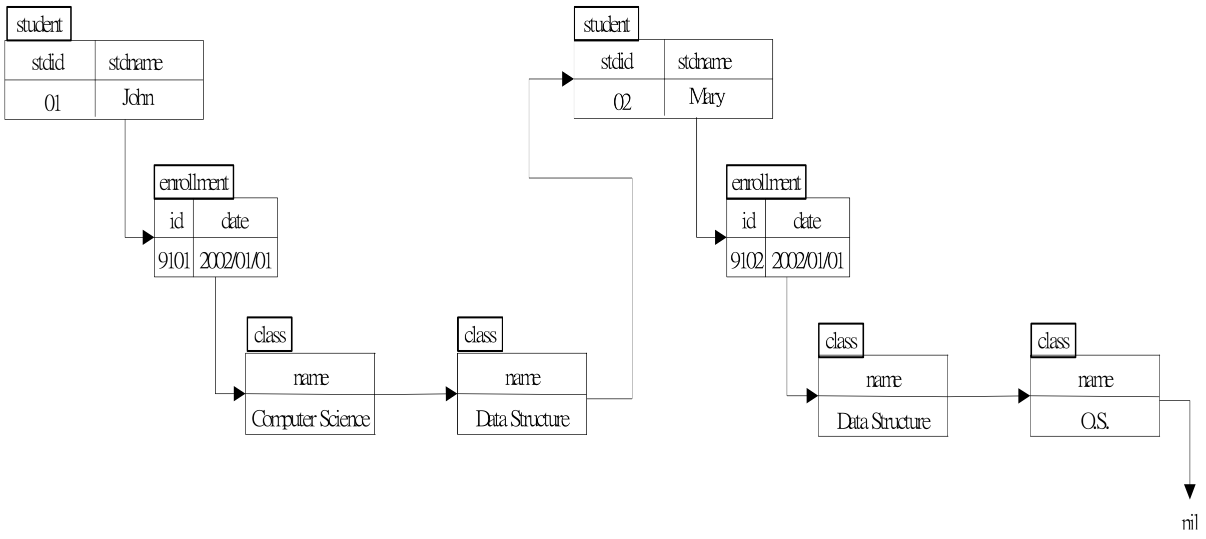


Figure 2: An HDB example.

2.3. Relationship between XML and Database

XML documents are grouped into two categories: data-centric and document-centric [7], [9]. Data-centric XML documents contain structured data extracted from database or other various data sources. Document-centric XML documents contain semantic text-based data. Some papers about the application of XML documents in databases are proposed. Some of the papers have to do with XML documents and relational databases [14], [17], [25]. The others concentrate on XML documents and object-oriented systems [6], [13].

Modern hierarchical data models, X.500 and LDAP [15], claim that the hierarchical data model will have an opportunity for a new lease on life. In X.500 or LDAP, the data structure is constructed as an *information tree*. Data is stored in an *entry* that has unique name and one or more *attributes*. However, there are no detailed descriptions about how to store XML documents to the hierarchical data model. Storing XML documents to “Native XML” databases with the original format without restructuring is another issue [24]. The paper in [24] discusses that the XML-type databases should have several functions in the point of traditional database view.

3. 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 3. In this model, the specified data is extracted from the HDB of *Source Unit*. The module *HtoX* is a transformer used to convert the exchanged data into the medium, an XML document. The XML documents can also be created manually without using *HtoX* if necessary. When an XML document is prepared, it will be transmitted to *Destination Unit*. For security, encoding/decoding techniques can be applied to the XML documents during transmission. When *Destination Unit* receives the XML document, the document is converted into its original data by the module *XtoH* and then saved to the HDB of *Destination Unit*. The modules *HtoX* and *XtoH* for data transformation are described in sequence as follows.

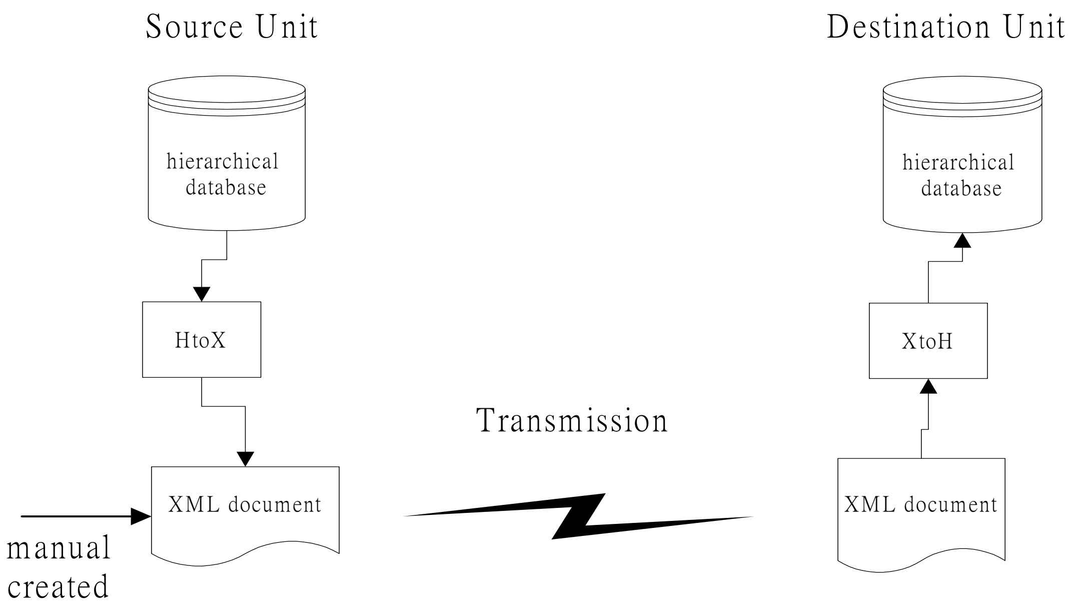


Figure 3: The architecture of the data exchange model.

3.1. The *HtoX* module

The *HtoX* module is used to transform the data in an HDB into an XML document. The record structure of HDB is defined in Figure 4. Data of a real record is stored in the fields of *f1*, *f2*, …, *fn*. The field *level* represents the level of the node to which a record belongs. The field *next* is a pointer pointing to the next record of a record.

level	*f1*	*f2*	…	*fn*	*next*

Figure 4: A record structure of HDB.

A node called *n* will be transformed into an element named *n*. A record in node *n* will be transformed into a sub-element of element *n*. Both name and value of each field in a record are transformed into the name and value of the corresponding attribute in an element. The transformation process of *HtoX* is performed with the algorithm, HDBtoXML, listed as follows.

```
Algorithm HDBtoXML(HDB_TREE H, XML_DOC X)
// Transform the data of a hierarchical database to an XML document. //
Input: H    // a pointer to a tree of a hierarchical database //
Output: X    // an XML document //
begin
    STACK S; // a stack for saving temporary data //
    initialize S;
    write start tag of H to X;
    get the names and values of fields in H and write to X;
    if H.next <> nil, then
        if H.level < H.next.level, then push(H, S);
        else if H.level = H.next.level, then write end tag of H to X;
            else if H.level > H.next.level, then
                    write end tag of H to X;
                    call closeingtag(H.level - H.next.level, S);
                endif;
    else
        write end tag of H to X;
        call closingtag(the size of S, S);
        return X;
    endif;
    HDBtoXML(H.next, X);
end HDBtoXML.
```

```
Procedure closingtag(integer n, STACK S)
Input n: the number of items in a stack S that should be popped off
      S: a stack
begin
    HDB_TREE n; // a dummy data item popped from S //
    integer i;
    for i = 1 to n, do
        n = pop(S);
        write end tag of n to X;
    end for;
end closingtag.
```

3.2. The *XtoH* module

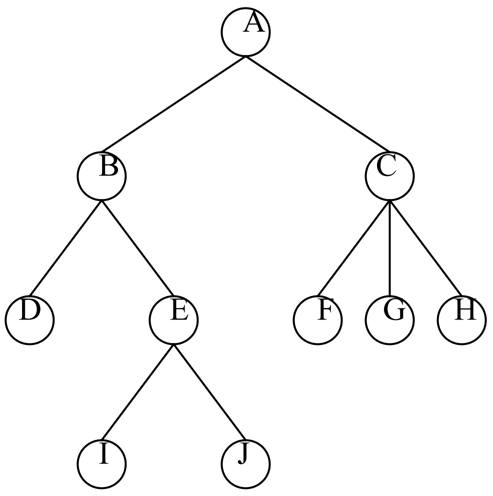
The *XtoH* module is used to convert the data in an XML document to an HDB. Each XML element will be transformed into a record except the root element that is transformed into the root node of the HDB. All records derived from the XML elements of the same level will be stored orderly in the same node whose name is the name of these XML elements. The attributes of an XML element will be transformed into the fields of the record that is derived from the same element. The transformation process of *XtoH* is performed with the algorithm XMLtoHDB that has two major phases. First, this algorithm converts the XML document into an m-way tree. Then this algorithm converts the m-way tree into an HDB tree. The details of XMLtoHDB are listed as follows.

```
Algorithm XMLtoHDB(XML_DOC X, HDB_TREE H)
// Transform an XML document to a hierarchy database. //
Input: X    // an XML document //
Output: H    // a pointer to a tree of a hierarchical database //
begin
    STACK S; // a stack for saving temporary data //
    M_WAY_TREE_NODE T, ptr; //pointers to a tree of m-way node structure//
    STRING tag_string; //a string variable for saving a tag//
    reset S to be empty;
    //First convert the XML document into an m-way tree//
    read the root start tag to tag_string from X;
    push(tag_string, S);
    create a node for T and put the data of the root element to T;
    ptr points to T;
    while X is not empty, do
        read a tag form X to tag_string;
        if tag_string is a start tag, then
            push(tag_string, S);
            create a child node for ptr;
            read the attributes and content, if exists, of the tag_string element from X and put
            them to ptr;
            ptr points to the child node;
        else //tag_string is a end tag//
            pop(S);
            ptr points to its parent node;
        end if;
    end while;
    //Then convert the m-way tree into an HDB tree//
    for each level of the m-way tree T, do
       merge and sort the data in those nodes that have the same tag name and attribute names
       into a node;
    end for;
    use the depth-first search to traverse the m-way tree T, do
       for each node, create a record for each element in this node;
       combine these records orderly to become a link list;
    end;
    combine each link list corresponding to each node in T to a big link list;
    H points to the head of the big link list;
    return H;
end XMLtoHDB.
```

An example of converting an XML document into an m-way tree is shown in Figure 5. In Figure 5(a), the XML document has a root element A containing two sub-elements B and C. Elements B and C contains two sub-elements, D and E, and three sub-elements, F, G, and H, respectively. Element E has two sub-elements I and J. The corresponding m-way tree, shown in Figure 5(b), of the XML document is created after the XML document is converted by the algorithm XMLtoHDB in *XtoH*. Each node in this m-way tree corresponds to an element or a sub-element in the XML document.

```
<A>
  <B>
    <D> </D>
    <E>
       <I> </I>
       <J> </J>
    </E>
  </B>
  <C>
    <F> </F>
    <G> </G>
    <H> </H>
  </C>
</A>
```

(a) An XML document



(b) An m-way tree

Figure 5. An example of converting an XML document to an m-way tree.

4. A Data Exchange Scenario

An illustration is given to introduce the application of the proposed data exchange model in Figure 3 between two companies. Assume that a book store A wants to place an order with a publisher B. Both A and B use HDB to store their data and agree to exchange data by XML. Initially, there may have a book order, as shown in Figure 6, in the HDB of A. The XML document, as shown in Figure 7, can be produced by inputting the HDB of order to the *HtoX* module, or, created directly without the module if necessary. Then the XML document is transmitted to B. Applying the module *XtoH*, B can transform the received XML document into the original HDB and the data exchange work is finished. Contrarily, B also can transmit HDB data such as the stationery order to A with XML.

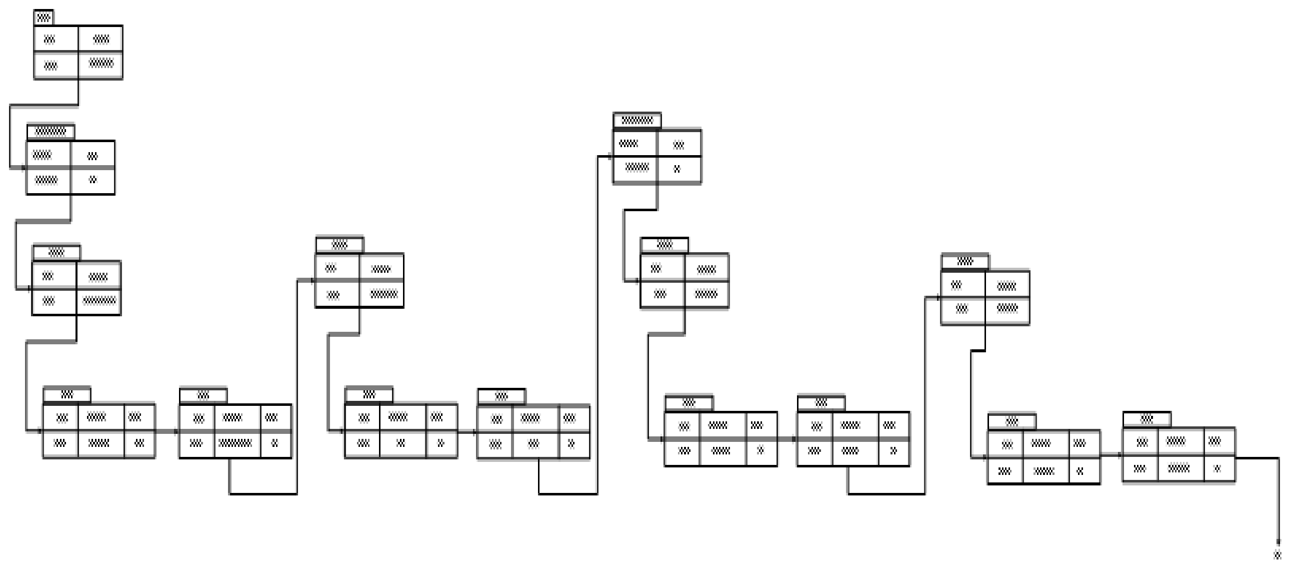


Figure 6. An HDB for the scenario.

```
<order o_id=”O001” o_date=”3/12/2002”>
   <classification c_name=”computer” dept=”IM”>
      <author a_id=”a001” a_name=”Richard Chen”>
         <book b_id=”b001” b_name=”database” price=”100”> </book>
         <book b_id=”b002” b_name=”data structure” price=”90”> </book>
      </author>
      <author a_id=”a002” a_name=”Tom Wang”>
         <book b_id=”b003” b_name=”OS” price=”80”> </book>
         <book b_id=”b004” b_name=”MIS” price=”70”> </book>
      </author>
   </classification>
   <classification c_name=”Language” dept=”IE”>
      <author a_id=”a003” a_name=”Mary Lin”>
         <book b_id=”b005” b_name=”English” price=”50”> </book>
         <book b_id=”b006” b_name=”French” price=”50”> </book>
      </author>
      <author a_id=”a004” a_name=”Jane Liu”>
         <book b_id=”b007” b_name=”Chinese” price=”40”> </book>
         <book b_id=”b008” b_name=”Japanese” price=”40”> </book>
      </author>
   </classification>
</order>
```

Figure 7. An XML document for the scenario.

5. Conclusion

In this paper, we propose a model for data exchange between XML and HDB. Associated with two algorithms, two modules are used to transform an HDB into an XML document and vice versa. Finally, a scenario is illustrated to demonstrate how the proposed model works. The motivation of this paper is to provide a technique for some enterprises or organizations that desire to exchange data with XML and their databases are the early data model: hierarchical database. The proposed model not only can exchange data of HDB, but also can exchange data of other databases such as relational databases, network databases, and object-oriented databases if the module *HtoX* or *MtoX* can be replaced by another suitable module.

Reference

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

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

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

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

[5] Understand LDAP. IBM redbook. http://www.redbooks.ibm.com

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

[7] R. Bourret, XML and databases, http://www.rpbourret.com/xml/XMLAndDatabases.html.

[8] Hofreiter, B.; Huemer, C.; Klas, W. "ebXML: status, research issues, and obstacles," *Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, 2002. RIDE-2EC 2002. Proceedings. Twelfth International Workshop,* pp. 7 -16 ,2002

[9] Bertino, E.; Catania, B. "Integrating XML and databases," *IEEE Internet Computing,* vol. 5, Issue: 4, 2001

[10] Bertino, E.; Ferrari, E. "XML and data integration," *IEEE Internet Computing ,* Vol. 5,

Issue: 6, pp. 75 -76, 2001

[11] Cleveland, F.M. “Information exchange modeling (IEM) and extensible markup language (XML) technologies,” *Power Engineering Society Winter Meeting, IEEE ,* vol.1, pp. 592 -595, 2002

[12] Kappel, G.; Rausch-Schott, S.; Reich, S.; Retschitzegger, W. **“**Hypermedia document and workflow management based on active object-oriented databases**,**” *System Sciences, Proceedings of the Thirtieth Hwaii International Conference,* vol.4, pp. 377 -386, 1997

[13] Jingyu Hou; Yanchun Zhang; Kambayashi, Y. “Object-oriented representation for XML data Cooperative,” *Database Systems for Advanced Applications, 2001. CODAS 2001. The Proceedings of the Third International Symposium,* pp. 40 –49, 2002

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

[15] H. V. Jagadish, Laks V. S. Lakshmanan, Divesh Srivastava. “Hierarchical or Relational? A Case for a Modern Hierarchical Data Model,” *Knowledge and Data Engineering Exchange, 1999. (KDEX '99) Proceedings. 1999 Workshop ,* 2000

[16] Singh, J. “XML for power market data exchange**,”** *Power Engineering Society Winter Meeting, IEEE,* vol. 2, pp. 755 -756, 2001

[17] Ji Sim Kim; Wol Young Lee; Ki Ho Lee “The cost model for XML documents in relational database systems,” *Computer Systems and Applications, ACS/IEEE International Conference,* pp. 185 -187, 2001

[18] Walczak, K.; Cellary, W**. “**X-VRML-XML based modeling of virtual reality**,”** *Applications and the Internet, 2002. (SAINT 2002). Proceedings. 2002 Symposium,* pp. 204 -211, 2002

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

[20] Spring, M.B. “Reference model for data interchange standards,” *Computer ,* vol. 29 Issue: 8 ,

pp. 87 -88, 1996

[21] Sundaram, M.; Shim, S.S.Y. "Infrastructure for B2B exchanges with RosettaNet**,"** *Advanced Issues of E-Commerce and Web-Based Information Systems, WECWIS 2001, Third International Workshop,* pp. 110 -119, 2001

[22] Summers, R.; Chelsom, J.J.L.; Nurse, D.R.; Kay, J.D.S. "Document management: an Intranet approach," *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine., 18th Annual International Conference of the IEEE ,* vol. 3, pp. 1236 -1237, 1997

[23] Wegener, *S.; Davis, D.* "XML TPS data exchange," *AUTOTESTCON Proceedings, 2001. IEEE Systems Readiness Technology Conference ,* pp. 605 -615, 2001

[24] Airi Salminen, Frank Wm. Tompa. "Requirements for XML Document Database Systems," *Proceeding of the ACM Symposium on Document Engineering,* 2001.

[25] Kudrass, Thomas. "Management of XML documents without schema in relational database systems," *Information and Software Technology,* vol.44, Issue: 4, pp. 269-275, 2002

[26] Data Design, 2nd Edition, Gio Wiederhold, McGraw-Hill, April, 1995.