A Database Approach for Modeling and Querying XML Documents

Panseop Shin*, JeongEun Kim*, Jaeho Lee**, Haechull Lim*

*Dept. of computer Engineering, Hong Ik Univerity
**Dept. of Computer Education, Inchon National University of Education
Tel: +82-2-333-5319, Fax: +82-2-320-1105
E-mail : {psshin, jekim, lim}@cs.hongik.ac.kr, jhlee@mail.inue.ac.kr

Abstract : In recent years, XML applic .tions are being developed in diverse area. Especially, development of XML document repository system associated with database is carrying out widely. The previous researches of XML repository system have several defects which are update and retrieval limitations for the XML document, design limitation for a formal retrieval algorithm and data redundancy. In order to solve the above problems, in this paper, we suggest relational database schemes for overcoming limitations of updating, retrieving, and rebuilding document. And suggest query translation strategy using two-phase translation that consists of pattern analyzing phase and SQL generating phase.

1. Introduction

The major role of HTML(Hyper Text Markup Language) is an information publishing and transmission in the Internet. HTML provides a easy way to create documents used in the WWW(World Wide Web) service, but HTML has limitations of describing document structure and retrieving documents efficiently. Hence, W3C(WWW Consortium) defined XML(extensible Markup Language) that combines convenience of HTML and extensibility of SGML. XML has powerful ability that defining structured document, creating versatile representation and enhancing link facilities. Thus, XML applications are being developed in diverse area. Especially, development of XML document repository system associated with database is carrying out .widely.

The previous researches of XML repository system[1,2] have several defects as follows. The first, because of the weakness of modeling power, conventional XML repository system could not efficiently updating the XML document. The second, it has retrieval limitation for elements located randomly in DTD(Document Type Definition)-tree because the system stores information of DTD-tree based on sequential search algorithm like DFS(Depth First Search). The third, it increases the number of table to store elements and the complexity to design a formal retrieval algorithm because the tables depend on DTD structure. Finally, it has rebuilding limitation of original document for aggregated results of retrieval and data redundancy. As the results of the above defects, the system inefficiency and limitation of database utilization arise.

In order to solve the above problems, we suggest relational database schema and query translation strategy. The suggested schema, the first output of our research, stores XML contents independent to DTD structure using DTD path information for overcoming limitation of updating, retrieving, and rebuilding document. The proposed query translation strategy, the second output of our research, converts user query of XML-QL to SQL according to query pattern. The proposed strategy fully supports above two queries type and uses two-phase translation that consists of pattern analyzing phase and SQL generating phase.

2. Related Research

2.1 XML Repository System

The previous researches of XML repository systems are classified into three topics. The first is research of modeling that translates XML documents to relational or object-oriented data model. The second is design of retrieval structure that supports retrieval of XML documents efficiently. The final is development of query language for retrieving XML documents. Especially, research of modeling and query languages is very important for the performance of XML repository system.

2.2 XML Query Language

There are many XML query languages that could be used to query XML documents. The XML query language, XML-QL[3,4], XQL[6], Lorel[7], XSL[8], XML-GL[5] and XML-QL, provides a textual syntax for writing queries that construct new XML documents. The XML-QL[3.4], one of the XML query language, is declarative and can express ordered and unordered views on XML document. Moreover, XML-QL uses a nested XML-like structure and have a powerful restructuring mechanism. XML-GL[5] is a graphical query language for XML documents. Namely, the WHERE clause and the CONSTRUCT clause are specified with visual user interface. Expressive power of XML-GL is similar to that of XML-QL but the former is sophisticated because of using a figure(such like rectangle) to query. LOREL[7] is Lore(Lightweight Object Rep ository)'s query language and extended OQL. Lore is a general-purpose semi-structured data management system. XSL[8] is the style-sheet language proposed by the W3C and consists of a collection of template rules. But, expressive power of XSL is limited. XQL[6] is an extension to the XSL pattern syntax. So, its restructuring expressive power is restricted. From above, expressive power of XML-QL is better than others. Thus, we choose XML-QL for querying XML document in our XML repository system. And, we propose query translation method that XML-QL to SQL.

2.3 Example of XML document

```
<?XML version="1 0" ENCODING="KSC5601">
<!- This is a sample email data file -->
<!DOCTYPE Email SYSTEM  "email.dtd">
<Email>
  <Recipient>Ingo.Macherius@tu-clausthal.de</Recipient>
  <Sender>hb@ix.heise.de</Sender>
  <Date>Mon, 21 Apr 1997 09:27:55 +0200</Date>
  <Subject>XML Literature</Subject>

  <Textbody>
    <p lang="de"> hello.Mr
      <Name>Behme</Name>.</p>
    <p> Please read <Name>Jon Bosak</Name>'s
       introductory text </p>
    <p> SGML, Java and the Future of the Web</p>
    <p> Bestwishes,</p>
    <p>
      <Name>Ingo Macherius</Name>
    </p>
  </Textbody>
</Email>
```

```
<!ELEMENT Email (Recipient. Sender. Date. Subject, Textbody)>
<!ELEMENT Recipient (#PCDATA)>
<!ELEMENT Sender (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Subject (#PCDATA)>
<!ELEMENT Textbody (p)+>
<!ELEMENT p (#PCDATA|NAME)*>
<!ELEMENT NAME (#PCDATA)>
<!ATTLIST p lang (de|en) "en">
```

Figure 1. Sample XML document and DTD

XML is a hierarchical data format for interchange of electronic data. The XML document consists of nested element structure and each element is the form of attributes or sub-elements. <Figure 1> shows the XML document that contains information about a Email and the DTD documents describing the structure of XML documents. Root element, Email, has five sub-elements such as Recipient, Sender, Date, Subject and Textbody. Textbody has sub-element 'p' that must be repeated more than once. The attribute 'lang' of element 'p' has default value 'en'.

3. Design of XML repository system

In this paper, we design a XML repository system. The system has two important modules. The one is mapping module that converts XML documents including DTD to relational schema. The other is querying module that translates XML-QL to SQL. <Figure 2> is the framework of the suggested XML repository system.

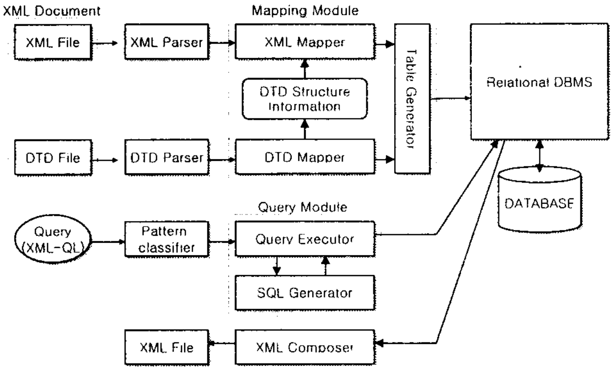


Figure 2. XML repository system

3.1 Relational schema for storing XML document

The schema consists of 6 tables : DTD_Table, DTD_Structure_Table, Attribute_Table, XML_Table, Element_-Instance_Table and Attribute_Instance_Table. All of DTD associated with XML is stored into the DTD_Table and DTD structure information extracted from DTD document is stored into the DTD_structure_Table with path information. Attribute in DTD is stored into the Attribute_Table. The XML_Table involves DTD id of XML document. Contents of XML document are inserted into the Element_Instance_Table and the Attribute_Instance_Table. The DTD_Structure_Table involves path information of DTD structure classified by each element.

In this paper, we suggest mapping method that converts XML document and DTD structure into relational data model individually. DTD structure is stored into ①②③ table of <figure 3>, and the contents of XML document are stored into ④⑤⑥ table. Detail of relational schema is presented in <Figure 3>.

① DTD_Table

DTD_id	DTD_name	DTD_content
DTD id	DTD File Name	DTD File Contents

② DTD_Structure_Table

DTD_struct_id	DTD_id	path	level
DTD Structure id	DTD id of DTD_Table	DTD Path info	Level of Tree

③ Attribute_Table

Attr_id	DTD_id	element	Att_name	default
Attribute id	DTD id of DTD_Table	Element name of Attribute involved	Attribute Name	Default value

④ XML_Table

XML_id	DTD_XML_id	XML_name	version	encoding	add_info
XML id	DTD id of DTD_Table	XML File Name	Version info.	Encoding Info.	Additional Info.

⑤ Element_Instance_Table

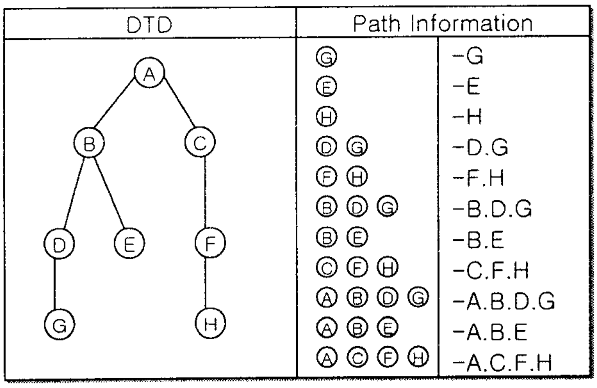
Element_id	DTD_struct_id	XML_id	Instance_id	contents
Element id	DTD struct id of DTD_Structure_Table	XML id	Instance id	Element Contents

⑥ Attribute_Instance_Table

Attr_inst_id	Attr_id	XML_id	Instance_id	value
Attr. instance id	Attr. id of Attribute_Table	XML id	Instance_id of Element_Instance_Table	Attr. contents

Figure 3. relational schema for storing XML

The path field in the DTD_Structure_Table involves path information of DTD structure classified by each element. So, the update processing of XML document is very simple since additional modification of stored data (e.g : offset, link field...) is not necessary. Also, when a user requests a element that locates anywhere or has sub elements, the proposed system supports efficient retrieval and access mechanism using path information. <Figure 4> describes the overview of the proposed path information.



DTD	Path Information	
	Ⓖ	-G
	Ⓔ	-E
	Ⓗ	-H
	Ⓓ Ⓖ	-D.G
	Ⓕ Ⓗ	-F.H
	Ⓑ Ⓓ Ⓖ	-B.D.G
	Ⓑ Ⓔ	-B.E
	Ⓒ Ⓕ Ⓗ	-C.F.H
	Ⓐ Ⓑ Ⓓ Ⓖ	-A.B.D.G
	Ⓐ Ⓑ Ⓔ	-A.B.E
	Ⓐ Ⓒ Ⓕ Ⓗ	-A.C.F.H

Figure 4. Path information

3.2 Example of translation

<Figure 5> is an example of translating XML document presented in <Figure 1> into relational schema by using proposed method.

DTD document's id, document name(e.g. "Email.dtd") and DTD document's contents are stored into the DTD_Table. The path information of "Email.dtd" is stored in the DTD_Structure_Table and attribute name "lang" and default value "1" are stored in the Attribute_Table.

The value of element in XML is inserted into the Element_Instance_Table with DTD_Structure_id of DTD_Structure_Table. "de", the value of attribute "lang" involved in element "p", is inserted into the Attribute_-Instance_Table.

DTD_Table

DTD_id	DTD_name	DTD_content
1	Email	...

DTD_Structure_Table

DTD_struct_id	DTD_id	path	level
1	1	Recipient	1
2	1	Sender	1
3	1	Date	1
4	1	Subject	1
5	1	p	2
6	1	Name	3
7	1	p.Name	2
8	1	Textbody.p	1
...	...	...	...
15	1	Email.Textbody.p.Name	0

Attribute_Table

Attr_id	DTD_id	element	Attr_name	default
1	1	p	lang	"en"

XML_Table

XML_id	DTD_XML_id	XML_name	version	encoding	add_info
1	1	Email	1.0	KSC5601	

Element_Instance_Table

Element_id	DTD_struct_id	XML_id	Instance_id	contents
1	1	1		"Ingo.Macherius @tu-clausthal.de"
2	2	1		"hb@ix.heise.de"
3	3	1		"Mon, 21 Apr 1997 09:27:55 +0200"
4	4	1		"XML literature"
5	5	1	1	"hello. Mr"
6	6	1	1	"Behme"
7	5	1	1	","
8	5	1	2	"Please read"
...	...	...	...	...
13	6	1	5	"Ingo Macherius"

Attribute_Instance_Table

Attr_inst_id	Attr_id	XML_id	Instance_id	value
1	1	1	1	"de"

Figure 5. Example of Mapping

3.3 Converting XML-QL to SQL

XML-QL[3,4] has five operations such as selection, extraction, reduction, restructuring and combination. In this paper, we consider the selection, extraction and combination part of the XML-QL, and suggest querying module that converts XML-QL to SQL according to the query patterns.

The query patterns of XML-QL is categorized into five forms by analyzing "Where" clause. The first, retrieval target is element. The second, retrieval target is element involved its attributes. The third, query retrieves element tag using "tag variable". The forth, query joins elements with its value. Finally, operation uses "Regular-path Expressions".

The translation steps of XML-QL to SQL are shown in <Figure 6>. The first query pattern uses step ①~④. In the second query patterns, step ⑤~⑦ are added.

```
   Select XML_ia, DTD_XML_id
①  From XML_Table
   Where XML_name = "XML document name";

   Select DTD_struct_id [, level]
②  From DTD_Structure_Table
   Where DTD_id = extracted DTD_XML_id in 1
        and path = 'element';

   Select Instance_id
③  From Element_Instance_Table
   Where DTD_struct_id = extracted DTD_struct_id in 2
        and XML_id = extracted XML_id in 1
        and contents = "contents of element in where clause";

   Select DTD_struct_id
④  From Element_instance_Table
   Where DTD_struct_id = extracted DTD_struct_id in 2
        and XML_id = extractea XML_id in 1
        and path = "contents of element in where clause";

   Select Attr_io
⑤  From Attribute_Table
   Where element = "element" and Attr_name= "attribute name"
        and DTD_id = extracted DTD_XML_id in 1 ;

   Select Instance_io
⑥  From Attribute_Instance_Table
   Where Attr_id = extracted Attr_id in 5 and XML_id = extracted XML_id in 1
        and value = "contents of attribute in where clause";

   Select value
⑦  From Attribute_Instance_Table
   Where Attr_id = extracted Attr_id in 5
        and Instance_id = extracted Instance_id in 3 ;
```

Figure 6. Query translation steps

In case of using "tag variable", only step ④ is added to the step ①~②. In case of joining elements by value, step of creating view table is added.

3.4 Example of "XML-QL to SQL"

The example of translating XML-QL <Who is recipient that the sender is "hb@ix.heise.ed" in "Email.xml"> into SQL is shown in <Figure 7>.

```
XML-QL :
 where <email>
              <recipient> $r </recipient>
              <sender> hb@ix.heise.de </sender>
       </email> IN "www.a.b.c./email.xml"
 construct $r
```

```
SQL :
 1 Select XML_id, DTD_XML_id
   From XML_Table
   Where XML_name = "email.xml";
 2 Select DTD_Struct_id
   From DTD_Structure_Table
   Where DTD_id = extracted DTD_XML_id in 1 and path = "recipient";
 3 Select DTD_Struct_id
   From DTD_Structure_Table
   Where DTD_id = extracted DTD_XML_id in 1 and path = "sender";
 4 Select Instance_id
   From Element_Instance_Table
   Where XML_id = extracted XML_id in 1
         and DTD_Struct_id = extracted DTD_Struct_id in 3
         and contents = "hd@ix.heise.de";
 5 Select contents
   From Element_Instance_Table
   Where DTD_Struct_id = extracted DTD_Struct_id in 2
         and Instance_id = extracted Instance_id in 4 ;
```

Figure 7. the example of translation

4. Conclusion

The mapping methodology of XML to relational database is very important factors in the XML repository systems. But, previous researches have limitations of updating XML document and direct accessing to elements dispersed in DTD tree. Also, conventional mapping method increases the number of table and the design complexity to design a formal retrieval algorithm. In addition, conventional mapping schema has limitation of rebuilding original full document with aggregated retrieval results. Thus, we suggest relational schema that can be stored with XML document separated from DTD structure and DTD path information for overcoming the limitation of update, retrieval, rebuilding document. And we propose "XML-QL to SQL" translation method and design query module based on it.

ACKNOWLEDGEMENTS

This research is supported by KOSEF (Korea Science and Engineering Foundation) under grant no. 98-0102-0901-3.

References

[1]Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang. David DeWitt, Jeffrey Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities" VLDB, pp.302-314 , 1999.

[2]Daniela Florescu, Donald Kossmann, "Storing and querying XML data using an RDBMS", IEEE Data Engineering Bulletin 22(3), pp.27-34, 1999

[3]Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, Dan Suciu, "A Query Language for XML" WWW8/Computer Networks 31(11-16), pp.1155-1169, 1999.

[4]Alin Deutsch, Mary F. Fernandez, Daniela Florescu, Alon Y. Levy, David Maier, Dan Suciu, "Querying XML Data" IEEE Data Engineering Bulletin 22(3), pp.10-18, 1999

[5]S.Ceri, S. Comai, E. Damiani, P.Fraternali, S. Paraboschi, "XML-GL:a graphical Language for querying and restructuring XML documents" WWW8/Computer Networks, 1999

[6]XML Query Language(XQL) http://www.w3.org/TranS/QL/QL98/pp/xql.html

[7]S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Winener, "The Lorel Query Language for Semistructured Data", International Journal on Digital Libraries, 1(1), pp.68-88, 1997

[8]Extensible Stylesheet Language(XSL) http://www.w3.org/Style/XSL/