Novi Sad J. Math.
Vol. 31, No. 1, 2001, 151-158

A DATA MODEL FOR INDEXING AND SEARCHING XML DOCUMENTS

Andrej Kovčić[1], Branko Milosavljević[2]

Abstract. One of the methodologies in the development of information systems is based on structured documents. The Standard Generalized Markup Language (SGML) was defined for the purpose of document structure description. Extensible Markup Language (XML) was defined as a subset of SGML. The goal of the new XML technology is electronic document exchange on World Wide Web. Since then, many well-known software companies have developed systems to process XML documents. The development of text servers for XML documents follows these systems. These technologies support building information systems based on the XML documents. This paper presents a new approach to specification and implementation of text server for indexing and searching XML documents. Specification is given in UML, and implementation in Java programming language.

AMS Mathematics Subject Classification (1991): 68P20

Key words and phrases: XML, text server, UML, Java.

1. Introduction

XML (*Extensible Markup Language*) [1, 2] is defined as a subset of the SGML (*Standard Generalized Markup Language*) [3] general markup language. XML specification describes the class of textual objects called XML documents and partially describes the behavior of the programs for their processing. XML documents represent structured textual documents with structure *elements* organized as tree structure. Elements are uniquely determined by their name. There exist a certain number of elements:

- Elements without content;
- Elements containing text of arbitrary length;
- Elements owning some subelements;
- Elements containing text and owning subelements;

[1]University of Novi Sad, Trg D. Obradovića 3, 21000 Novi Sad, Yugoslavia
[2]University of Novi Sad, Faculty of Technical Sciences, Trg D. Obradovića 6, 21000 Novi Sad, Yugoslavia

- Elements whose content can be text or any subelement.

Each element has exactly one assigned parent element, excluding the one particular element representing the root of the document tree. The attributes can also be assigned to the element. Each attribute is determined by the name and has the content.

A well formed XML document is the document strictly respecting the sintax of the XML language. XML specification also defines the concept of the *document type definition* (DTD). DTD is used to specify the structure of the XML document. For each XML document declared to agree with certain DTD specification, one can check if this document really agrees with this specification. The documents whose structure corresponds to the given DTD are called *valid* documents. Each valid document is also a well formed document.

The software module handling an XML document and its elements is called XML parser. One option supported by modern parsers is the document validity checking. DOM [4] and SAX [5] are specifications providing one with the interface to the rest of the software system.

Eminent DBMS vendors are offering the text server modules as part of the DBMS. These servers support document indexing and search for a wide number of formats like HTML, Word, Excel, PDF, XML, etc. One of these servers is the Oracle8i interMedia Text. All tables of the text server are of textual type. Counting advantages of the text servers, one has to note some basic disadvantages:

- Indexing and searching of other data types (INTEGER, DATE, etc.);

- Indexing and searching by attribute value.

In [6], a data model of the XML server is described. Some aspects of the physical data handling due to improvement of the server performance are determined. The paper also suggests the way multimedia data should bi integrated with XML. This paper describes modeling and implementation of the system for generating XML servers database based on *document type definition.*

2. Document type definition

For each DTD sent to the server, the corresponding subscheme is generated for storing the XML documents representing instances of the given DTD as well as additional data (indexes) to be used in the search process. To be processed by XML server, the data describing indexing mode and data types of the elements should extend the standard DTD. Additional information is given within special comments <!--**%TYPE%** -->. The software modules not recognizing the syntax of this comment will simply ignore it. Example of a DTD extended by this comment is given in Figure 1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Document type definition -->

<!ELEMENT A (B, E, F)>
<!ELEMENT B (#PCDATA | C | D) > <!-- %TYPE% TEXT -->
<!ATTLIST B
Att1 CDATA #REQUIRED
Att2 CDATA #IMPLIED>

<!ELEMENT C (#PCDATA)> <!--%TYPE% TEXT -->
<!ATTLIST C
Att1 (x | y | z) "x">

<!ELEMENT D (#PCDATA)>
<!ELEMENT E (#PCDATA)> <!--%TYPE% INTEGER-->
<!ELEMENT F (#PCDATA)> <!--%TYPE% DATE-->
```

Figure 1. An example of the DTD containig additional data for the XML server

Comments within this DTD specify the following information:

- Element A is the root element; it consists of tags B, E and F; there is no special comment for its type, so it is not to be inexed;
- Element B is a mixed tag, i.e. along with text it can contain tags C and D;
- Element C is a textual tag; for this element the index will be generated;
- Element D is textual, but the corresponding index will not be generated (its type does not follow the declaration);
- Element E is an integer; the corresponding index will be generated
- Element F is a date; the corresponding index will be generated.

One should notice that the contents of the element D is the part of the element's B index, since the element D is the subtag of the element B. On the other hand, the element D is indexed separately.

3. Class diagram

The concept of the software system handling XML documents presumes some relational database management system as its background. The programming langauage *Java* is used for implementation. The XML parser implementing standard SAX interface handles the documents. The system handles valid documents. Class digaram of the system is shown in Figure 2.

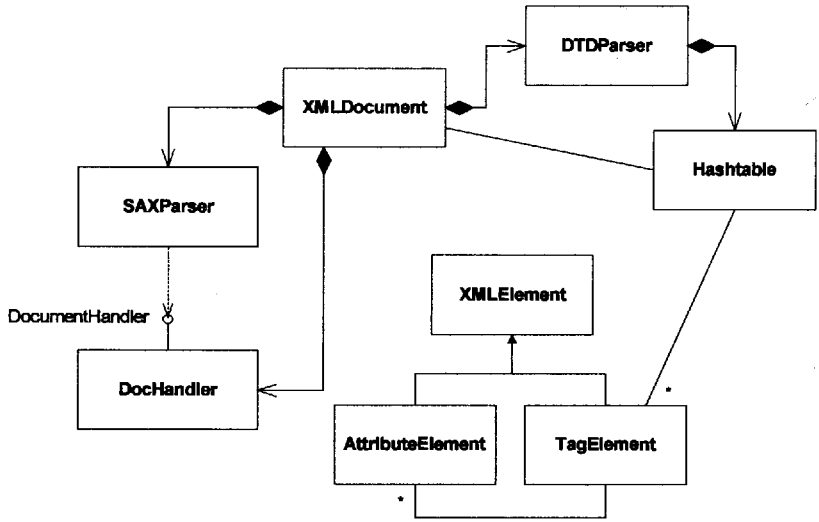

Figure 2. UML class digram of the XML server

The class XMLDocument is the interface of the XML server to the user. Instances of this class represent all the documents - instances of the single DTD. This class has the method specifying DTD and operations over documents (adding, deleting, modifying and search).

The class DTDParser is aimed to pars DTD and generate internal information describing the document structure. Internal information is presented by the class Hashtable which implements hash table used to store the data about the elements of the particular DTD by which indexing is carried out.

Index elements are represented by the class TagElement. Their attributes are represented by the class AttributeElement. The common parent class is the abstract class XMLElement.

Insertion and modification operations request document parsing. Parsing is done by the class SAXParser implementing the standard SAX interface. SAX standard assumes the class implementing DocumentHandler interface aimed to process particular parsed elements.

4. Relational database scheme

Implementation of the class diagram shown in Figure 2 is done using the Java and Oracle database management system. The appropriate subscheme within the database is generated on the basis of DTD's extended by special comments carrying indexing information. The basic table within this database subscheme is aimed to store complete XML documents. Its columns are ID

(internal numerical document identificator) and DOC (document content). The name of this table is the same as the name of the DTD's root element.

For each indexing element two corresponding index tables are defined. The first one, aimed to index elements content, contains three columns: ID (primary relation key), DOC_ID (identification of the document owning the index) and CONTENT (index content). The type of the CONTENT column depends on the type declared in special comment within the DTD. The name of this table is generated by concatenating the name of the DTD's rooot element and the name of the element assigned to the table.

The second table, aimed to index by attribute value, contains four columns: DOC_ID (document identificator), ELEM_ID (identificator of the element owning the attribute), NAME (attribute name), and VALUE (attribute value). The type of the NAME and VALUE column is always textual. The name of this table is generated by concatenating the name of the previous table and the string ATTR.

In the tables used to index the element content an additional index is generated by the column CONTENT; the database management server uses this index for searching. Similarly, within the tables for attribute content indexing, an additional index is generated over the column pair (NAME,VALUE).

Figure 3 shows the SQL script in Oracle8i system generated for the DTD given in Figure 1.

```
CREATE TABLE A (
  ID INTEGER NOT NULL,
  DOC LONG);
CREATE TABLE A_B (
  ID INTEGER NOT NULL,
  DOC_ID INTEGER NOT NULL,
  CONTENT VARCHAR2(255) NOT NULL);
CREATE TABLE A_B_ATTR (
  DOC_ID INTEGER NOT NULL,
  ELEM_ID INTEGER NOT NULL,
  NAME VARCHAR2(255) NOT NULL,
  VALUE VARCHAR2(255) NOT NULL);
CREATE TABLE A_C (
  ID INTEGER NOT NULL,
  DOC_ID INTEGER NOT NULL,
  CONTENT VARCHAR2(255) NOT NULL);
CREATE TABLE A_C_ATTR (
  DOC_ID INTEGER NOT NULL,
  ELEM_ID INTEGER NOT NULL,
  NAME VARCHAR2(255) NOT NULL,
  VALUE VARCHAR2(255) NOT NULL);
CREATE TABLE A_E (
```

```
  ID INTEGER NOT NULL,
  DOC_ID INTEGER NOT NULL,
  CONTENT INTEGER NOT NULL);
CREATE TABLE A_E_ATTR (
  DOC_ID INTEGER NOT NULL,
  ELEM_ID INTEGER NOT NULL,
  NAME VARCHAR2(255) NOT NULL,
  VALUE VARCHAR2(255) NOT NULL);
CREATE TABLE A_F (
  ID INTEGER NOT NULL,
  DOC_ID INTEGER NOT NULL,
  CONTENT DATE NOT NULL);
CREATE TABLE A_F_ATTR (
  DOC_ID INTEGER NOT NULL,
  ELEM_ID INTEGER NOT NULL,
  NAME VARCHAR2(255) NOT NULL,
  VALUE VARCHAR2(255) NOT NULL);
```

Figure 3. An example of the SQL script generating database subscheme

5. XML documents handling operations

The XML server offers four types of opertions handling XML documents stored in database: insertion, deletion, modification and search. All operations are requested from the XMLDocument class instance. Execution of any of these operations preassumes the XMLDocument class object is initialiezed to handle documents corresponding to the appropriate DTD; initialization is done by calling the method public void setDTD(String dtd). Call to this method instances the class DTDParser object which analyzes the obtained DTD and generates hash table containing the data about elements and attributes to be indexed.

Inserting the document in the XML server database is done by calling the method public int insert(String doc). Call to this method instances single object of each of SAXParser and DocHandler classes which are used to input document parsing and parsed tokens processing, respectively. The content of the complete document and parsed contents of the elements to be used in serach are stored in the corresponding database tables. The result of the execution of this method is the identifier value of the new document.

Documents are deleted from the XML server database by calling the method delete(int id). Deleting the document from the database actually means erasure of the corresponding rows from the tables belonging to the given subscheme.

Documents are modified in the XML server database by calling the method public void update(int id, String doc). Document modification is reduced to

modification of the corresponding row in the table containing complete documents and, followed by removing and adding the indexes.

Document search is done by using the following XMLDocument class methods:

public void query(String tag, int operator, Object value)
public void queryAND(String tag, int operator, Object value)
public void queryOR(String tag, int operator, Object value)
public void queryNOT(String tag, int operator, Object value)
public int[] queryResult()

The method query represents an elementary query to the server: searching by single tag within the document complying with the given DTD. The operation used is represented by one of the standard operators offered by SQL language (=, <, >, >=, <=, !=, LIKE). Such an elementary query can simply be converted to the corresponding SQL query.

The methods queryAND, queryOR i queryNOT (with the parameters having the same meaning as for method query) are aimed to generate complex queries. A complex query consists of the elementary queries connected with the logical operators AND, OR and NOT. The complex queries can be converted to SQL queries by converting their elementary queries and using the SQL operators UNION, INTERSECT and MINUS.

For example, the following complex query:

```
xmlDoc.query("lastName", XMLDocument.EQUALS, "Smith");
xmlDoc.queryAND("firstName", XMLDocument.EQUALS,
"John");
```

is converted to the SQL query:

```
SELECT a.id FROM student a, student_lastName b
  WHERE a.id=b.doc_id AND b.content='Smith'
INTERSECT
SELECT a.id FROM student a, student_firstName b
  WHERE a.id=b.doc_id AND b.content='John';
```

6. Conclusion

The XML server presented in this paper is a solution to storing, indexing and searching XML documents. It provides the possibilities of inserting, deleting, indexing, and searching XML documents. It preassumes the valid XML documents having a DTD extended by information for generating the server's database scheme, as well as for indexing and searching the corresponding document elements.

Appart from the commercially available XML servers, this server supports:

- Indexing and searching by different data types;
- Indexing and searching by attribute values.

References

[1] *Extensible Markup Language (XML) 1.0*, W3C Recommendation, WWW Consortium, http://www.w3.org/XML/

[2] *XML Developer's Guide, Microsoft Corporation*, http://www.msdn.microsoft.com/workshop/index/default.html

[3] Martin, B.,*SGML: An Author's Guide to Standard Generalized Markup Language*, Addison-Wesley, New York 1995.

[4] *Document Object Model (DOM)*, W3C Recommendation, WWW Consortium, http://www.w3.org/DOM/

[5] Megginson D.,*Simple API for XML*, http://www.megginson.com/SAX/

[6] Milosavljević, B., Surla, D.,*One architecture of a text server for XML documents*, Proceedings of The XXVI Yugoslav Symposium on Operations Research, p.149-152 , Belgrade, 1999 (in Serbian)