

# Web Content Management System based on XML Native Database

Mihaela Sokic

*Croatian Telecom, Jurisiceva 13, Zagreb, HR-10000, CROATIA*

*mia.sokic@ht.hr*

Viktor Matic and Alen Bazant

*University of Zagreb*

*Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, HR-10000, CROATIA*

*viktor.matic@fer.hr, alen.bazant@fer.hr*

**Abstract.** *In this paper we are identifying problems in the complex Web application implementations. In more details we are addressing problem of content management and specific components of such system. We are also presenting our implementation of Web Content Management System (WCMS), which is based on relational database and XML technologies. It also uses some rather new concepts and technologies like XForms and Native XML database.*

**Keywords.** Content Management System, Native XML database, XML.

## 1. Introduction

As Internet evolves Web sites are becoming more and more complex. They are offering variety of informations and services, organized in different ways. What is common to all of them is tendency to serve information in the most flexible and accessible way to the end user.

There are two aspects of this problem: the user's side and the site administrator's. Information, which will be presented through the Web site to the user, has to be prepared, organized and populated behind the scenes. Even more, generally speaking, data storage organization and the way of presenting it to the end user do not have to be the same. Even if we focus only on content and possibilities to organize it and present it, we can foresee complexity of this problem. There is obvious

need to divide the whole problem in smaller parts and then to analyze their interaction and purpose one by one.

A Web site conceptually can be divided in the following four basic parts, like in [4]:

- Content;
- Structure;
- Functionality and
- Presentation.

Content takes care of information that will be presented to the users. Structure provides structural access to information and services offered on the Web site. Site structure should not be mistaken for information structure in content part because they can have completely different structure. Functionality means processing the content that site offers. And finally, presentation defines in which way information will be clearly and attractive presented to the user.

We can gain more benefits if we approach the problem of site's inner organization in described manner. The main benefit is possibility to change one part without affecting the other. As practical example of this approach we can imagine complete site graphical redesign while keeping existing data.

In this paper we will be focused only on the content part of site. We will try to implement our Web Content Management System (WCMS) as solution for the administration of content part of Web site. Also, we

will propose one of possible solutions to implement WCMS using XML technologies and Java programming language in conjunction with modern database approaches.

## 2. Proposed solution

After we have defined the needs we will propose our solution. As we have already seen the Web site could be analyzed and implemented through the parts presented in introduction. Content itself can be also further decomposed to the smaller functional parts. We will try to identify the functions which should be covered by the WCMS. The main functions are:

- Data structure;
- User interface (UI);
- Storing data;
- Search and retrieval.

Data structure functions are mechanism for data structuring and organizing. It is important that administrator can easily alter data structure. Data manipulation is done using UI, and UI have to be conformant to the defined data structure. UI should be automatically adjusted to changes in data structure. Information entered through UI should be validated against adequate structure and then, if conformant, it should be stored. At end we need mechanism for searching through the content and for data retrieval.

We are proposing the solution in which structure is described in a hybrid way, with XML Schema [6] and relational schema. HTML UI is generated automatically, based on XML Schema, using eXtensible Stylesheet Language (XSLT) [6]. Entered data is stored in XML [6] format inside Native XML database. Data retrieval, based on search mechanizam, is conducted using standard SQL extended with XPath syntax [6]. We will call this complex system Web Content Management System (WCMS).

## 3. Implementation

Today, most application data and Web content is stored either in a relational database or the file system or a combination of both. However, when implementing flexible data structures none of these two technologies or combination is not adequate.

XML has arrived as a key technology for the next stage of evolution of the Internet. Integrating XML as a key technology into WCMS systems is not a new solution, however using XML only to separate presentation from content is not always the most efficient solution. During the process of generation of Web pages, data are retrieved from data storage, transformed into XML format, further processed or combined and finally transformed into HTML. However, as the XML technology progresses and new technologies emerge (especially Native XML databases) it becomes obvious that the data can also be stored in XML format. Comparing to the relational database, we are no longer constrained on current database schema and yet we are able to maintain the consistency of data more efficiently than having data in file system.

Native XML database has an XML document as its fundamental unit of logical storage, just as a relational database has a row in a table [5]. Since XML documents are used in process of generating Web content there are good reasons to use Native XML databases for storing these documents.

Supposing we already have Web content and application data stored in a relational database. If we want to extend our WCMS functionalities with flexible content structures related to existing content, the Native XML database would not be the most efficient solution. Transformation of existing relational data into XML structures and adjustments of existing application functionality in process of retrieving and storing data is far to problematical at the moment and thus inefficient. The solution comes in a form of combination of relational and Native XML database [3]. This gives us ability to store and manage both fixed and flexible data structures.

### 3.1. Data Storage

Analyzing the content, our conclusion is that every content structure has some attributes and relations that are common. It is efficient to keep these data stored in relational database because of all the advantages the relational databases bring us, like retrieval speed, indexes, data constraints, etc.

Relational structure that would satisfy our common structure is shown in Fig. 1.

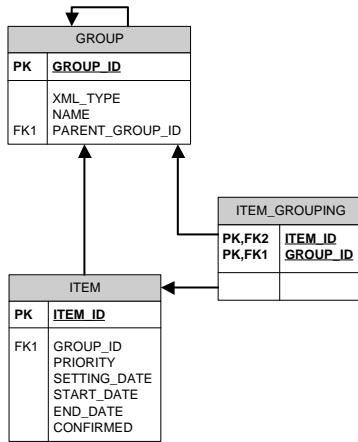


Figure 1: Information grouping model

Since hierarchical organized structure of groups is not flexible enough, we store alternative grouping definitions in `ITEM_GROUPING` table. We have also introduced a restriction where all the items belonging to the same group have to be of the same content type. As it can be seen on Fig. 1 groups are hierarchically organized. Every branch in the group can contain data of only one type. This makes data retrieval and presentation much easier. Every item stored in `ITEM` table has an XML document, where the rest of the attributes and its values are stored. These XML documents are inside an XML table that can be represented in relational database notation as shown in Fig. 2.

The XML table has following properties:

- The name of the table;
- Relation to the `ITEM` in relational table;

ITEM	
PK	ITEM_ID
FK1	GROUP_ID PRIORITY SETTING_DATE START_DATE END_DATE CONFIRMED

ITEM_VARCONTENT	
PK,FK1	ITEM_ID
	VARCONTENT XSD_NAME

Figure 2: XML storage table

- Storage for XML documents;
- The Name of the XML Schema document the content is validated to.

The other approach to data modelling would be having attribute-value pairs inside a relational table. In that case all the validation of the attributes would be done on the application level. The advantage of introducing the XML table versus having attribute-value pairs is that the validation of variable content is done on database level and the definition of variable content can be changed without the affecting application logic.

An example of XML Schema document that validates variable content structures is shown in Fig. 3.

In order to be able to define the presentation characteristics in UI for inserting and updating item instances, we added attributes that are part of a namespace noted with prefix "ed" in our documents. These attributes are defined for the elementary types "ID", "Name", "CoverType", "Description", etc. (Fig. 3), and later applied to the elements. We can also override these default settings when defining an element, e.g. "bookId"). The final result will be that all elements defined as "Name" type will be by default displayed as an "Entry" element, while the default settings for the element "bookId" are overridden, and it will be displayed as non-updatable "Display" element.

There is an analogy between XML Schema document that defines the structure of the con-

```

<xs:schema>
  <xs:element name="VarContent" >
    <xs:complexType>
      <xs:choice>
        <xs:element name="Book" type="BookType" />
        <xs:element name="Magazine" type="MagazineType" />
        <xs:element name="ComicBook" type="ComicBookType" />
        <!--and so on-->
      </xs:choice>
    </xs:complexType>
  ...
  <xs:complexType name="BookType" >
    <xs:sequence>
      <xs:element name="bookID" type="ID" ed:xmlform="Display" />
      <xs:element name="bookName" type="Name" />
      <xs:element name="cover" type="CoverType" />
      <xs:element name="description" type="Description" />
    </xs:sequence>
  ...
  <xs:simpleType name="ID" ed:xmlform="Entry" >
    <xs:restriction base="xs:integer" >
      <xs:totalDigits value="12" />
      <xs:fractionDigits value="0" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Name" ed:xmlform="Entry" >
    <xs:restriction base="xs:string" >
      <xs:minLength value="0" />
      <xs:maxLength value="4000" />
    </xs:restriction>
  </xs:simpleType>
  ...
</xs:schema>

```

Figure 3: A part of XML Schema for storing variable content structures

tent of XML data like the relational schema in relational databases. Using attributes we defined, we are referring to the abstract presentation objects and relating them to the content. By adding new elements inside the choice list (Fig. 3), we are actually adding support for new content structures.

Described XML Schema document and all other documents included in the process are stored inside the database and available to the designers and content administrators through WebDAV protocol [1]. Using this protocol content administrators are able to use software like XML Spy to change documents directly in the database. In that way the content administrators define the content structures by themselves.

Since the WCMS is unaware of the content structures before the actual design and loading of content starts, the further challenge is to provide a flexible UI. Since our UI is Web based, WCMS should be able to dynamically generate HTML forms for creating, updating and deleting content instances. The most challenging part for implementation was content structure interpretation and dynamic

transformation to HTML forms. The complete process of dynamic generation of user interface and retrieving/storing the content is presented in Fig. 4.

### 3.2. Presentation

The process can be segmented into following phases:

- Generation of XMLINSTANCE document;
- Generation of XMLFORM document from the content structure description;
- Transformation of XMLFORM document into HTML form;
- Generation of XMLINSTANCE document from the user input and storing it into database.

The first phase is generation of XMLINSTANCE document. This document is an instance of content. In this document the actual data presented on the Web are located. Its structure can be validated against XML Schema document described in Section 3.1. In the Fig. 4 this document is named DATA DESCRIPTION document. The information about the type of content stored in XML is located under group information. The example of XMLINSTANCE document that is an instance of "BookType" structure is shown in Fig. 5

```

<VarContent>
  <Book>
    <bookID>22</bookID>
    <bookName>Applying Use Cases</bookName>
    <cover>Soft</cover>
    <description>A practical guide to UML</description>
  </Book>
</VarContent>

```

Figure 5: An example of "BookType XML Instance"

If we are updating the existing content, the document is retrieved from the database using XPath extensions in SQL. XPath standard [6] is used while it provides the access abstraction on top of the storage model. XPath traverses nested XML elements by specifying

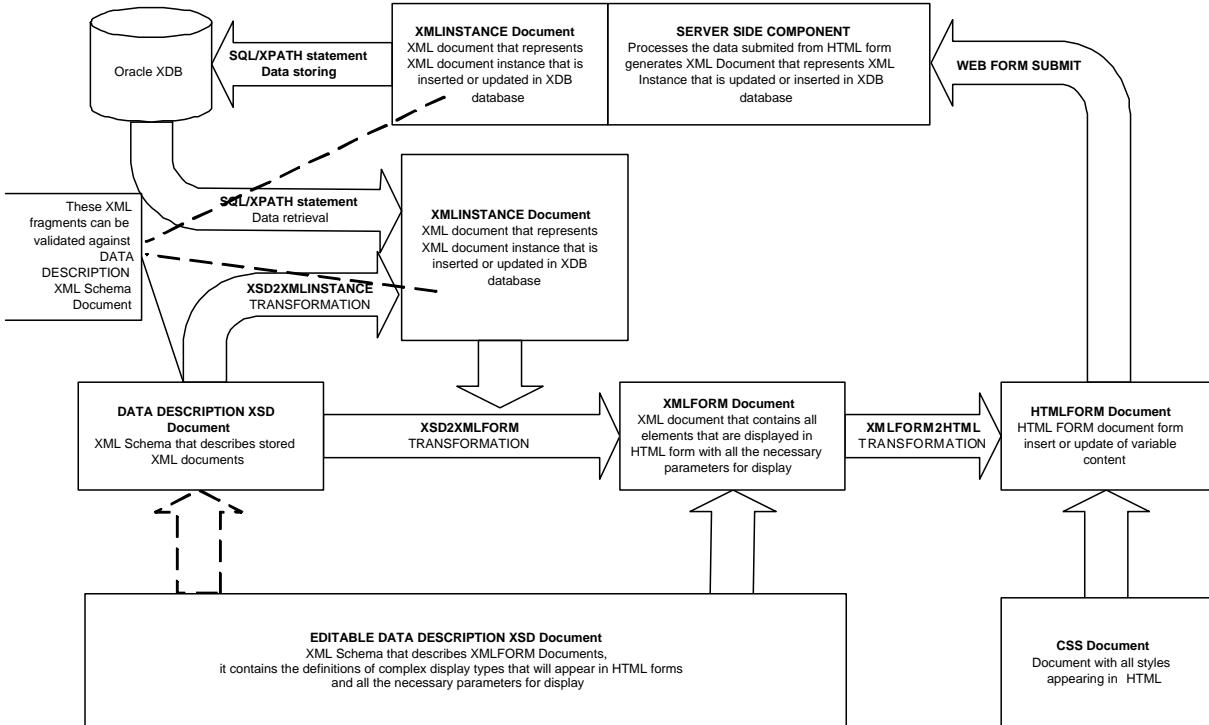


Figure 4: Transformation path

the elements to navigate through with a slash-separated list of element and attribute names. This conjunction of SQL and XPath gives us a method for accessing both data from the relational and XML tables using single statement.

If we are creating a new instance of content then we use XML Schema definition and XSL transformation to create an XML document that has exactly the same structure as the one previously shown, only without any element value inside it.

The next phase uses XMLINSTANCE document as a part of XMLFORM document. XMLFORM document is XML document that combines content structure with the presentation characteristics of each element. The structure of XMLFORM document is defined using XML Schema document and in the Fig. 4 is named EDITABLE DATA DESCRIPTION document. An example of this document and the idea can be found in [2].

Elements appearing in the "Form" element represent common form inputs types appearing in HTML forms or parts of HTML syntax. If we compare them to the part of

XML Schema document that describes content structure for "BookType", we can notice that each element of the complex structure is transformed into one element inside the "Form" element.

Using XMLINSTANCE document inside XMLFORM document we define the structure of XML document that is inserted or updated and we are passing the data during and update action. An example of complete XMLFORM document is shown in Fig. 6

```
<XMLForm>
  <XMLInstance>
    <VarContent>
      <Book>
        <bookID>22</bookID>
        <bookName>Applying Use Cases</bookName>
        <cover>Soft</cover>
        <description>A practical guide to UML</description>
      </Book>
    </VarContent>
  </XMLInstance>
  <Form>
    <Display key="bookID" label="ID"/>
    <Entry key="bookName" label="Name"/>
    <RadioGroup key="cover" label="Cover type">
      <Option values="S" valueTitle="Soft"/>
      <Option value="H" valueTitle="Hard"/>
    </RadioGroup>
    <LargeEntry key="description" label="Description"/>
  </Form>
</XMLForm>
```

Figure 6: An example of XMLFORM document

The next phase is the generation of HTML form. It is done also using XSLT. During the process of generation of HTML form each element inside the "Form" element will be transformed to create an element inside a HTML form. Elements inside an "XMLInstance" element will be used to populate the HTML form with data. For example, transformation of previously shown XMLFORM would result in HTML form that would look like in the Fig. 7.

HTML form Book

ID 22

Name

Cover type  Soft  Hard

Description  
A practical guide to UML

Change Book

Figure 7: Final HTML form

This concept is very simplified idea found in XForms standard [6]. XForms is W3C specification of Web forms that can be used with a wide variety of platforms. It has separate sections that describe how the form looks, and what the form does. Our implementation is similar to the section of XForms that describes how the form looks. XForms have ability to suspend and resume the completion of a form. This is done in XForms Processor that is part of Web client software. Currently, that is the main disadvantage of XForms technology and the reason why we did not use XForms for presentation purposes.

The final phase begins after content administrator fills in a form and submits the content to the server. Server component (Fig. 4) creates XMLINSTANCE document for the given type and populates the values of the XML elements with user data. If all required data are filled in, the XMLINSTANCE is stored inside the database.

#### 4. Conclusions and Further Work

In this paper we presented a part of implementation of Web Content Management System (WCMS). Our implementation use established technologies like relational database and XML technologies as basis, and introduces some rather new concepts and technologies like XForms and Native XML Database. We have decided to build hybrid model combining XML Native database with relational database. The next step would be storing the entire content into an XML database. We are also planning to fully implement XForms standard, instead of modified version described in this paper.

#### References

- [1] IETF. RFC:2518 - HTTP Extensions for Distributed Authoring – WEBDAV. University of California, Irvine; January 2002. <http://www.ics.uci.edu/~ejw/authoring/protocol/rfc2518.txt>
- [2] Ogbuji C. Editing XML Data Using XUpdate and HTML Forms. O'Reilly xml.com; June 2002. <http://www.xml.com/pub/a/2002/06/12/xupdate.html>
- [3] Oracle Corporation. Oracle XML DB. Oracle Official Web Site; January 2002. [http://otn.oracle.com/tech/xml/xmldb/pdf/xmldb\\_92twp.pdf](http://otn.oracle.com/tech/xml/xmldb/pdf/xmldb_92twp.pdf)
- [4] Simic H. Patterns of Web Site Structure in UriGraph. To be published on The 7th International Conference on Telecommunications, ConTEL; 2003, Zagreb, Croatia.
- [5] XML:DB Initiative. XML:DB Initiative for XML Databases. XML:DB Initiative Official Web Site; 2003. <http://www.xmldb.org>
- [6] W3C. W3C Technical Reports and Publications. W3C Official Web Site; 2003. <http://www.w3.org/TR/>