A Web-Based Metadata Schema Repository

YEN-CHUN LIN, HSIANG-AN WANG, CHIEN-CHUNG HUANG, WEI CHEN
Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology
43 Keelung Road, Sec. 4, Taipei 106
TAIWAN
yclin747@gmail.com, {d9215004, m9215052}@mail.ntust.edu.tw, arc1108@yahoo.com.tw
http://faculty.csie.ntust.edu.tw/~yclin/yclin.htm

Abstract: - The metadata schema of a digital archive describes the structure and attributes of metadata. Analysis and definition of metadata schema for a new digital archive must be carefully carried out and determined at the first stage of development. To ease the task, we used an Extensible Markup Language (XML) structure to represent the metadata schema, and then designed and implemented a metadata schema repository to store metadata schemas as XML documents in a native XML database. The metadata schema repository supports storage, creation, search, and access management of metadata schemas. A user can access the repository through a Web browser. With this repository, projects and organizations can share their metadata schemas over the Internet. Since the metadata schema must be displayed on a Web browser, we also present the method of translating the XML representation of metadata schema into the HyperText Markup Language document.

Key-Words: - Digital archive, Extensible Markup Language, HyperText Markup Language, Metadata schema repository, Native XML database, Web-based

1 Introduction
Digital archives are archives in the digital form. Through them we store and manage valuable digitized resources of, for example, cultural heritage and artifacts. The digital form makes archives easily preserved permanently and widely accessible over the Internet.

Analysis of metadata plays an important role in constructing digital archives. Metadata is “data which describes attributes of a resource” [1], and is required “to find, access, use, and manage information resources” [2]. The result of metadata analysis is crucial because it has an impact on the content construction process, the accuracy of query results, the feasibility of data exchange with other systems, and the feasibility of the construction of value-added applications. Therefore, the definition of metadata schema, which describes the structure and attributes of metadata, for a new digital archive should be carefully determined at an early stage.

It is not an easy task to analyze and define a metadata schema. Creating a metadata schema requires a great amount of time and effort [3]. It usually takes four months to two years to create one [4]. Metadata analyzers and content providers should collaborate to decide the metadata schema. They usually need to consult related projects and international metadata standards. Software tools for assisting the management, analysis, comparison, and creation of metadata schemas are desirable.

ULIS Open Metadata Registry [5] and CORES [6] have been presented to manage metadata schemas. The former can only take care of the Dublin Core metadata schema [7]. The latter can handle many metadata schemas, but not metadata schemas with a hierarchical structure. Another drawback of CORES is that it cannot be used to create or edit schemas over the Internet. These two tools use the Resource Description Framework Schema language [8] to describe metadata schemas.

In this paper, we present a Web-based metadata schema repository for storing and managing various metadata schemas. The repository is accessible through Web browsers, such as Firefox and Microsoft Internet Explorer, over the Internet, and is useful for retrieving and creating metadata schemas. Users can query existing metadata schemas stored in the repository and create new metadata schemas. We store metadata schemas in Extensible Markup Language (XML) documents [9], which are suited to represent hierarchical metadata schema structures. Additionally, the structure of metadata schemas can be easily modified. Therefore, the repository can help reduce the time and effort in creating suitable metadata schemas and ultimately help develop user-friendly digital archive systems efficiently.

Table 1. Part of the Construction table.

Element			Alias			Data Type	Size
Title	Main		品名	主要名稱		Varchar	50
	Other	Type		其他	類型	Varchar	20
		Name			名稱	Varchar	50
		Remarks			備註	Varchar	50

Table 2. Part of the Attribute table

Element			Required	Repeatability			Attribute	Provider
Title	Main		＊					Person
	Other	Type			◎		Menu	Person
		Name						Person
		Remarks						Person

In Section 2, we introduce a metadata schema representation with three tables. Section 3 describes our XML version of the metadata schema introduced in Section 2. In Section 4, we present the architecture of our metadata schema repository. Section 5 explains the main functions of the metadata schema repository by showing some useful operations. Section 6 introduces the translation method from our XML documents into HyperText Markup Language (HTML) ones for displaying metadata schemas on a Web browser. Section 7 concludes this paper.

2 Metadata Schema Tables

A metadata schema can be described with three tables [10]; this representation is often used in Taiwan. The specification of the Han Dynasty Woodslip Metadata Schema is used for illustration in the following [11]. Table 1 shows part of the Construction table. The Element column is used for naming metadata elements. In this example, the Element has a hierarchical structure. Its root, Title, has two subelements named Main and Other, and Other has three subelements named Type, Name, and Remarks. The Alias column is for the Chinese version of the Element column. The Data Type column describes the data types of atomic elements, which are at the lowest level of the hierarchy of the Element. Here, Varchar means characters of variable length. The Size column describes the maximum size of atomic elements in number of characters.

Table 2 shows part of the Attribute table. The Element column has the same meaning as that in the Construction table. In the Required column, the symbol "＊" indicates that the atomic element to the left, Main, must be provided. In the Repeatability column, the symbol "◎" specifies that the corresponding element, Other, can have more than one occurrence. In the Attribute column, the Menu indicates that the value of the corresponding atomic element should be selected from a predefined pull-down menu displayed on the monitor. The Provider column describes that the value of an atomic element of the Element is to be filled in by a human or generated automatically by the digital archive system.

Table 3 shows part of the DC-correspondence table. It gives the mapping between the metadata schema and the Dublin Core standard. For example, each of the four atomic elements corresponds to Title defined by the Dublin Core.

3 Metadata Schema in XML

As a markup language, XML has self-defined tags, and thus is flexible and extensible. It has the merit of easy change of document structure and content, and

Table 3. Part of the DC-correspondence table

Element			Dublin Core Element
Title	Main		Title
	Other	Type	Title
		Name	Title
		Remarks	Title

is well suited for describing data with a complex structure [9]. Since a metadata schema usually is complex, has many items, and is subject to change, XML is suited to representation of metadata schemas.

Therefore, instead of using Tables 1-3, we use the XML document shown in Fig. 1 to include essentially the same information. Line 2 shows that the Project element has a Name attribute with value Woodslip. This information is beyond the scope of the three tables and represents the project name with which the metadata schema associates. Lines 3-6, for example, show metadata schema elements Title and Main; XML attributes are used to represent metadata schema element attributes: Alias, Data Type, and Size of Main, as well as other related information.

In order to store and manage XML documents efficiently and to reduce the complexity of software development, an appropriate database management system is required. Two main categories of database systems are usually used to store XML documents. One is XML-enabled relational database systems, and the other is native XML database systems [12]. The former must map XML schemas into relational database schemas. Before being saved in the database, XML documents are transformed based on a mapping rule. A drawback of this approach is the loss of comments and the order of elements. Moreover, if the structure of XML documents is modified, the mapping must be redefined, which is usually time-consuming. Even worse, it is generally difficult, if not impossible, to retrieve and recover from the relational database the original XML document with a complex structure.

In contrast, the native XML database has several benefits. It is intended for storing XML documents; an XML document is a basic unit of storage. With such a database, no schema mapping is needed. Hence, we may access XML documents with their original structures and contents intact. Using an XML document as a basic storage unit enables the flexibility of changing metadata schema structure. Therefore, we use a native XML database to store XML documents.

```
<?xml version="1.0" encoding="Big5"?>
<Project Name="Woodslip" Creator="admin" Public="Y">
<Title Alias="品名“ Required=“ ” Repeatability=“ “ >
   <Main Alias=“主要名稱” Data_Type=“Varchar” Size=“50”
    Required=“＊”  Repeatability=“ “Attribute=“ ”
    Provider=“塡表者” DC-Element=“Title”>
   </Main>
   <Other Alias="其他" Required=“ ” Repeatability="◎">
    <Type Alias=“類型” Data_Type=“Varchar” Size=“20”
     Required=“ ” Repeatability=“ “ Attribute=“Select_List”
     Provider=“塡表者”   DC-Element=“Title“>
    </Type>
    <Name Alias=“名稱” Data_Type=“Varchar” Size=“50”
     Required=“ ”   Repeatability=“ “ Attribute=“ ”
     Provider=”塡表者” DC-Element=“Title”>
    </Name>
    <Remarks Alias=“備註” Data_Type=“Varchar”
     Size=“50” Required=“ ”   Repeatability=“ “ Attribute=“ ”
     Provider=”塡表者” DC-Element=“Title”>
    </Remarks>
   </Other>
  </Title>
</Project>
```

Fig. 1. Metadata schema of Tables 1-3 and related information described in XML.

4 Architecture of the Metadata Schema Repository

Fig. 2 shows the architecture of our metadata schema repository. It consists of three tiers: Web browser, Web server, and native XML database system. Web browsers serve as the user interface, and no plug-ins or other application programs are required at the client side.

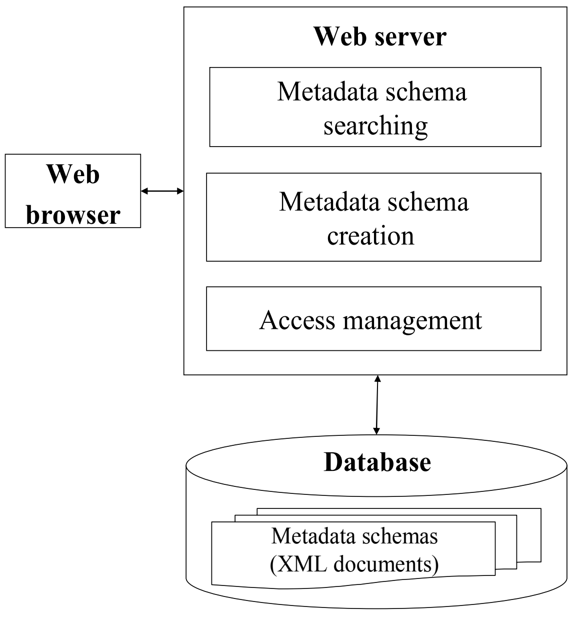


Fig. 2. Architecture of metadata schema repository.

The Web server acts as an intermediary between the Web browser and database. The Web server performs the main functions of the metadata schema repository. Apache Tomcat is used to build the Web server by performing Java servlets and JavaServer Pages (JSP) [13]. The repository is developed with Java SDK, JSP, and Java API for XML Processing (JAXP) [14]. JAXP is a Java XML application programming interface, which provides the capability of validating and parsing XML documents.

The native XML database system is based on the open-source eXist [15]. The system supports XQuery to query XML documents directly [16]. The database stores existing, well-known metadata schemas, including standards such as Dublin Core and Categories for the Description of Works of Art [17], as well as newly created metadata schemas.

5 Functions and Operations of Repository

The metadata schema repository consists of the following three main functions.

(1) Metadata schema retrieval.
It retrieves metadata schemas that match a keyword such as an element name. Queries are transformed into the XQuery statements to retrieve metadata schemas from the database. The schemas in XML are translated into HTML and displayed on the Web browser.

(2) Metadata schema creation.
Users can create new metadata schemas from scratch or by modifying existing ones in the repository. Created schemas can be saved to the repository or to personal computers.

(3) Metadata schema access management.
Access to metadata schemas is classified. Unauthorized users are not allowed to retrieve, modify, or delete any data.

The user interface of the metadata schema repository is shown in Fig. 3. It consists of three areas. On the north-west corner is the Function Area, which displays the functions available to users: Browse Metadata Schema, Search Metadata Schema, and Create Metadata Schema. Users can click to use them.

The Operation Area, located to the right of the Function Area, enables users to provide input data after choosing a function. The Schema Area, occupying most of browser window, is for displaying a metadata schema, which is retrieved from the repository or is being created.

We now consider the functions provided by the Function Area. To view a schema, users can click Browse Metadata Schema to scan through all schemas in the repository. Alternatively, they can click Search Metadata Schema and enter a keyword to search for matched metadata schemas. Users can also click Create Metadata Schema to build a new metadata schema from scratch.

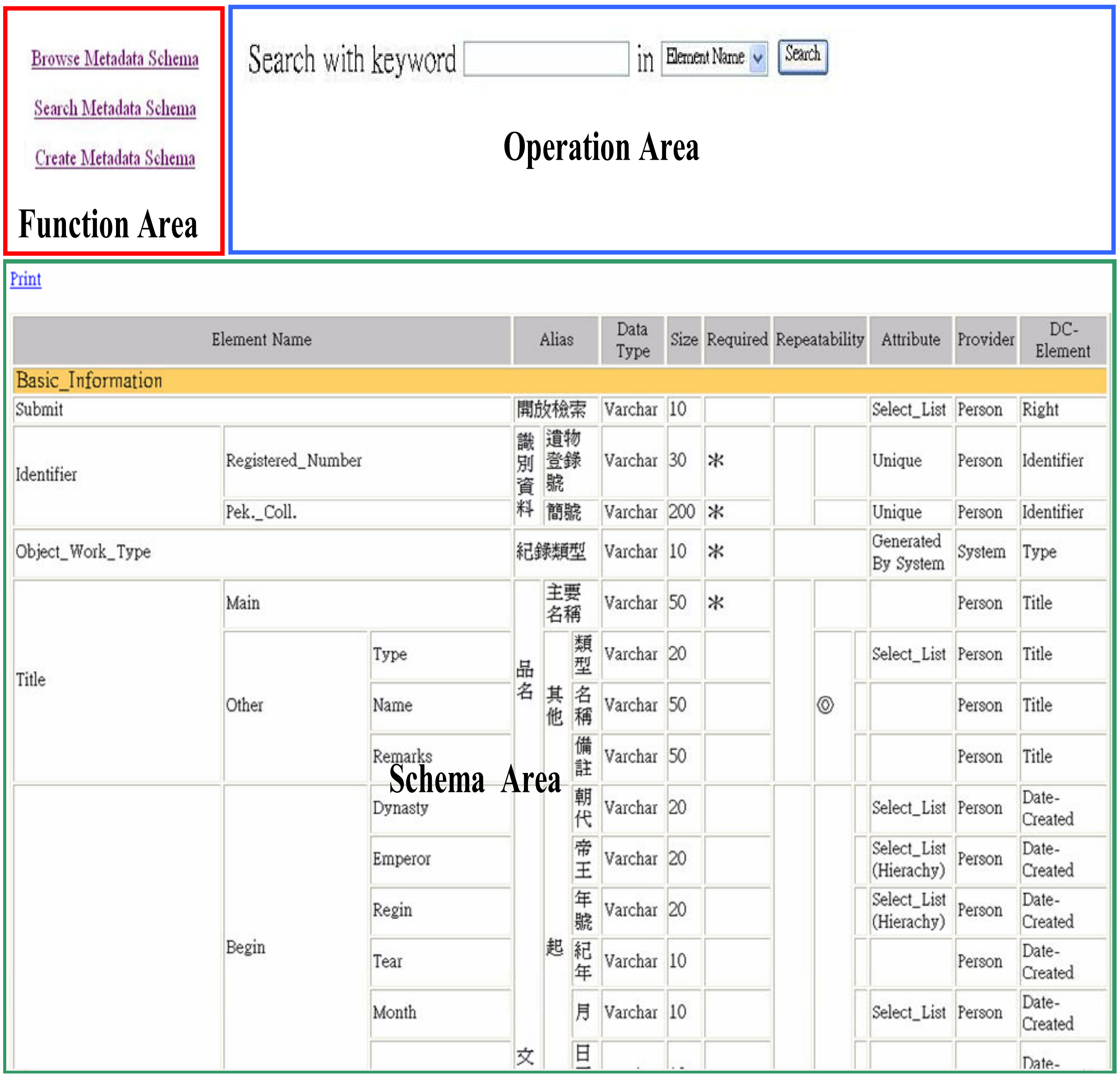
The Operation Area enables the user to enter a keyword and select Project Name, Element Name, Alias, Data Type, Size, or DC-Element field name from a pop-up menu to retrieve all the metadata schemas that have a field name matching the keyword. For example, suppose a user enters the keyword Title and chooses Element Name from the pop-up menu, and this request is sent to the Web server. The Web server then translates the request into the XQuery language to fetch the corresponding XML documents from the database. The search result displayed in the Schema Area, for example, is shown in Fig. 4.

The metadata schema creation process is shown in Fig. 5. After logging into the repository, users can choose either to create from scratch or to modify an existing schema. To create from scratch, users must first enter the project name and the element name to create a root element. Alternatively, to create by using an existing metadata schema, users must first select a metadata schema. After that, users can choose an element to add a subelement, modify contents, or delete an element. When the modification is completed, the schema can be saved with a new name into the repository.

To modify an existing metadata schema, the user has to click Create Metadata Schema in the Function Area shown in Fig. 3, and then clicks Modify Metadata Schema that appears in the Operation Area. After that, all schema names are shown in the Operation Area, and the user chooses a metadata schema to use. The contents of the metadata schema are then shown in the Schema Area. The user can click a hyperlinked element below Element Name to modify. A "★" is prefixed to the clicked element. For example, the Submit element in Fig. 6 has been selected.

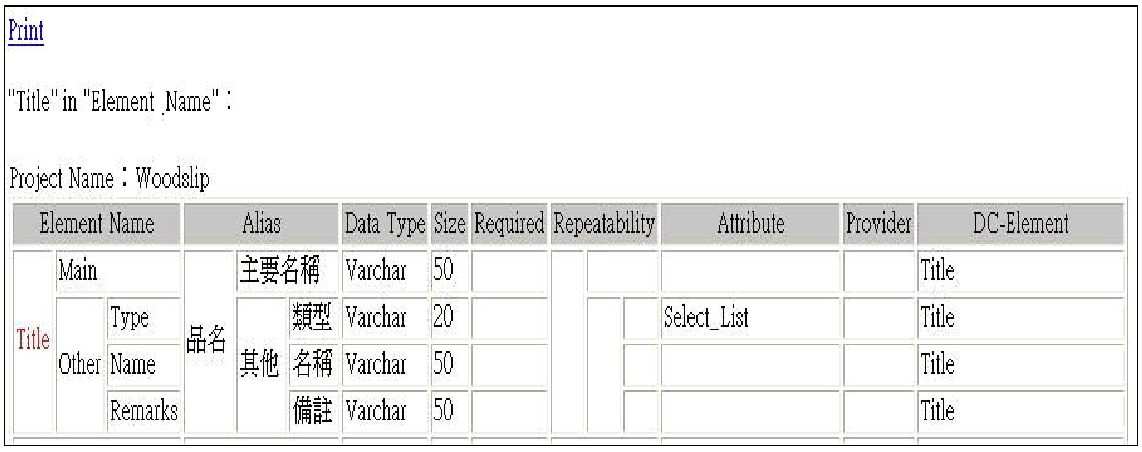
Three buttons of Add, Modify, and Delete below the selected element are used to add a new child-element to this element, modify contents of the element, or delete the element. Their operations are described in the following.

(1) Add a child-element.
The user first clicks the Add button to display a form (Fig. 7) in the operation



Element Name			Alias			Data Type	Size	Required	Repeatability	Attribute	Provider	DC-Element
Basic_Information												
Submit			開放檢索			Varchar	10			Select_List	Person	Right
Identifier	Registered_Number		識別資料	遺物登錄號		Varchar	30	*		Unique	Person	Identifier
	Pek._Coll.			簡號		Varchar	200	*		Unique	Person	Identifier
Object_Work_Type			紀錄類型			Varchar	10	*		Generated By System	System	Type
Title	Main		品名	主要名稱		Varchar	50	*			Person	Title
	Other	Type		其他	類型	Varchar	20		◎	Select_List	Person	Title
		Name			名稱	Varchar	50				Person	Title
		Remarks			備註	Varchar	50				Person	Title
	Begin	Dynasty	文	起	朝代	Varchar	20			Select_List	Person	Date-Created
		Emperor			帝王	Varchar	20			Select_List (Hierachy)	Person	Date-Created
		Regin			年號	Varchar	20			Select_List (Hierachy)	Person	Date-Created
		Tear			紀年	Varchar	10				Person	Date-Created
		Month			月	Varchar	10			Select_List	Person	Date-Created
					日							Date-

Fig. 3. User interface of the metadata schema repository.



Print

"Title" in "Element_Name"：

Project Name：Woodslip

Element Name			Alias			Data Type	Size	Required	Repeatability	Attribute	Provider	DC-Element
Title	Main		品名	主要名稱		Varchar	50					Title
	Other	Type		其他	類型	Varchar	20			Select_List		Title
		Name			名稱	Varchar	50					Title
		Remarks			備註	Varchar	50					Title

Fig. 4. Result of searching for metadata schemas with Title in Element Name.

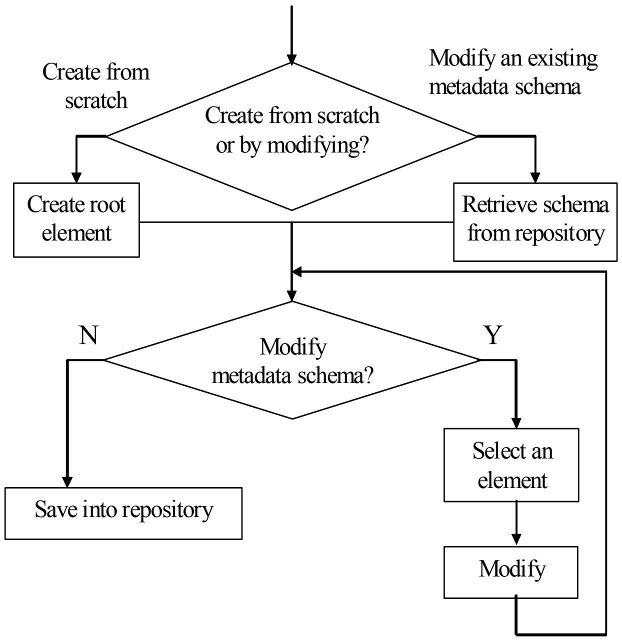


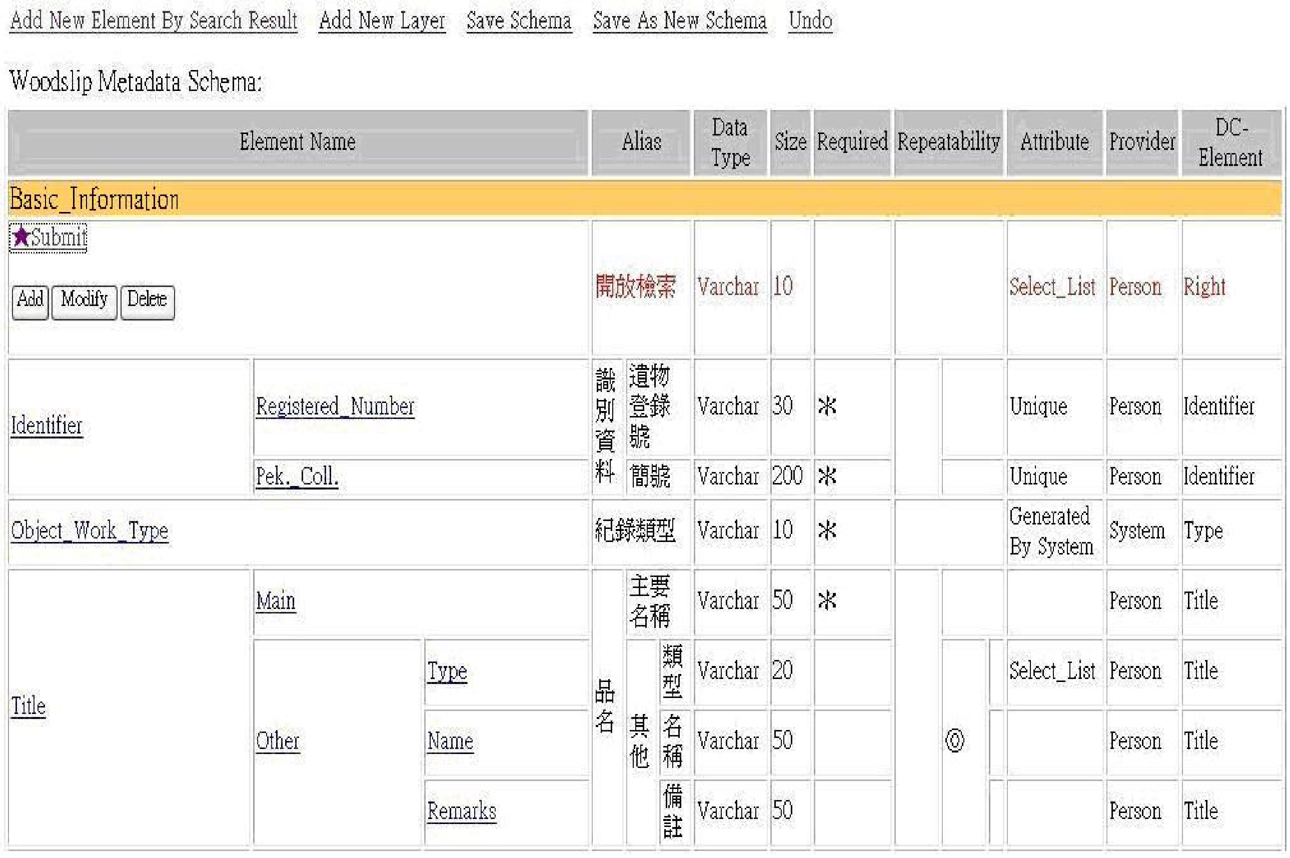
Fig. 5. Metadata schema creation process.

area. After entering the element name and filling in the form, the user presses the Submit button to generate a child-element.

(2) Modify an element.
The user clicks the Modify button in Fig. 6 to show the contents of an element (Fig. 8) in the operation area for modification. Clicking the Modify button in Fig. 8 will save the changes.

(3) Delete an element.
Clicking the Delete button deletes the element.

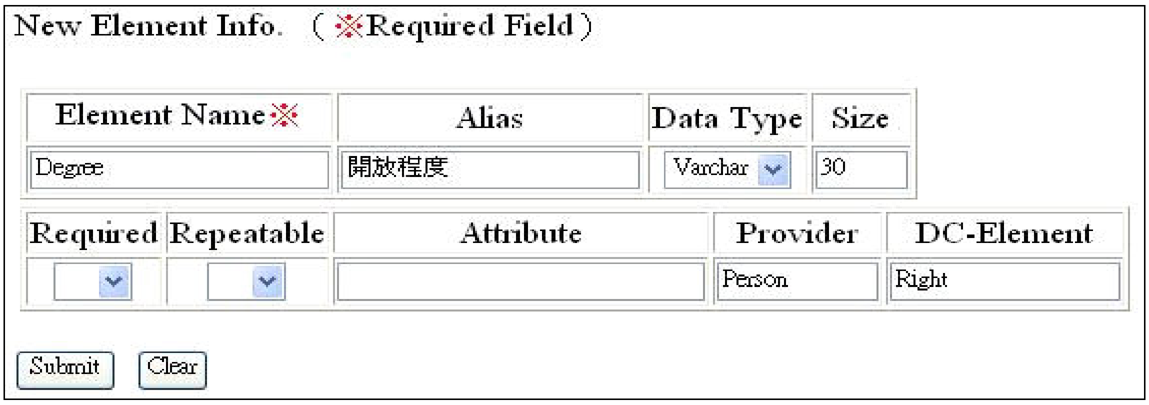
In addition to the functions mentioned above, the user can search for a specific element and add the element to be a new element of the new metadata schema. For example, after selecting the Submit element, the user can click Add New Element By Search Result (Fig. 6), and then search for metadata schemas with Citation element name. Fig. 9 shows the result of searching for the Citation element name. The element, as well as each of its children, has an Add button beside it. The user can click any Add button to add an element to become a new child-element of element Submit.



Woodslip Metadata Schema:

Element Name			Alias			Data Type	Size	Required	Repeatability	Attribute	Provider	DC-Element
Basic_Information												
★Submit Add Modify Delete			開放檢索			Varchar	10			Select_List	Person	Right
Identifier	Registered_Number		識別資料	遺物登錄號		Varchar	30	※		Unique	Person	Identifier
	Pek._Coll.			簡號		Varchar	200	※		Unique	Person	Identifier
Object_Work_Type			紀錄類型			Varchar	10	※		Generated By System	System	Type
Title	Main		品名	主要名稱		Varchar	50	※			Person	Title
	Other	Type		其他	類型	Varchar	20		◎	Select_List	Person	Title
		Name			名稱	Varchar	50				Person	Title
		Remarks			備註	Varchar	50				Person	Title

Fig. 6. The Submit element selected for update.

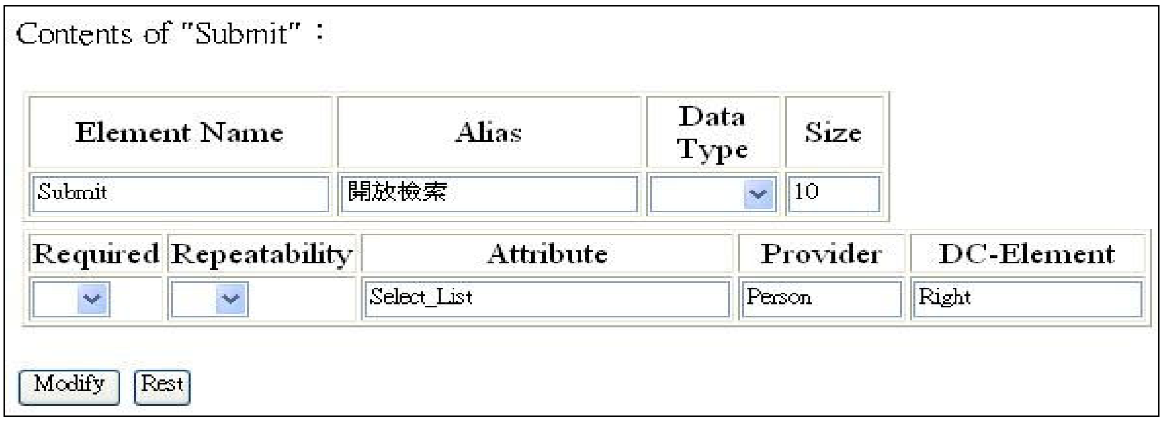

New Element Info. （※Required Field）

Element Name※	Alias	Data Type	Size
Degree	開放程度	Varchar	30

Required	Repeatable	Attribute	Provider	DC-Element
			Person	Right

Submit Clear

Fig. 7. A form for adding a new child-element.

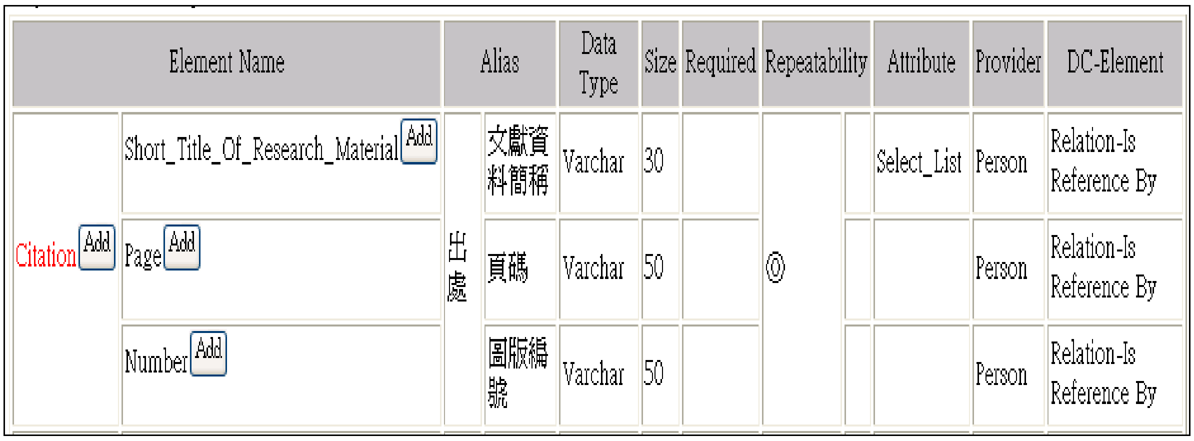

Contents of "Submit" :

Element Name	Alias	Data Type	Size
Submit	開放檢索		10

Required	Repeatability	Attribute	Provider	DC-Element
		Select_List	Person	Right

Modify Rest

Fig. 8. A form for modifying an existing element.

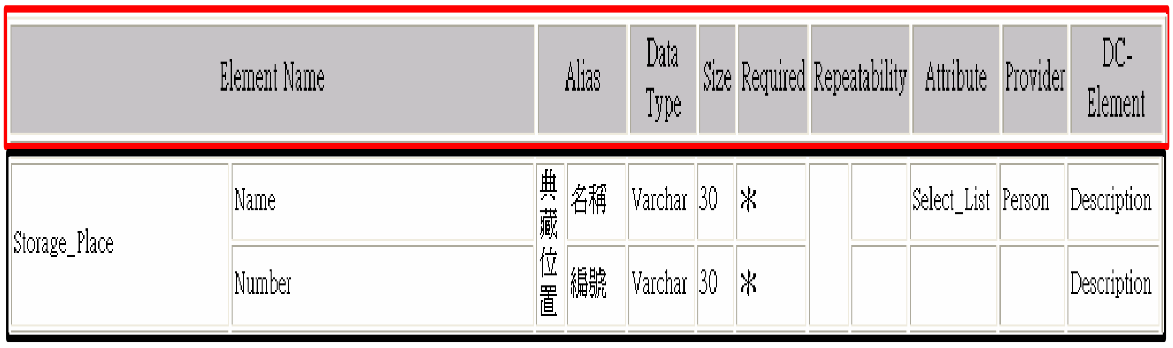


Element Name		Alias		Data Type	Size	Required	Repeatability	Attribute	Provider	DC-Element
Citation Add	Short_Title_Of_Research_Material Add	出處	文獻資料簡稱	Varchar	30			Select_List	Person	Relation-Is Reference By
	Page Add		頁碼	Varchar	50		◎		Person	Relation-Is Reference By
	Number Add		圖版編號	Varchar	50				Person	Relation-Is Reference By

Fig. 9. Result of searching for the Citation element name.

```
<?xml version="1.0" encoding="Big5"?>
<Storage_Place Alias="典藏位置" Repeatability="" >
  <Name Alias="名稱" Data_Type="Varchar" Size="30" Required="＊" Repeatability=""
    Attribute="Select_List" Provider="Person" DC-Element="Description" >  </Name>
  <Number Alias="編號" Data_Type="Varchar" Size="30" Required="＊" Repeatability=""
    Attribute="Select_List" Provider="Person" DC-Element="Description" >  </Number>
</Storage_Place>
```

Fig. 10. An XML document.



Element Name		Alias		Data Type	Size	Required	Repeatability		Attribute	Provider	DC-Element
Storage_Place	Name	典藏位置	名稱	Varchar	30	＊			Select_List	Person	Description
	Number		編號	Varchar	30	＊					Description

Fig. 11. A table displayed on the browser based on the XML document shown in Fig. 10.

6 Translating XML into HTML

XML documents are not suited for the human to read. To view them with a Web browser, we need to translate the XML data into HTML. Taking the XML document in Fig. 10 as an example, we want to display it as the table shown in Fig. 11 on the browser. Fig. 11 is in fact generated by the browser from the HTML data shown in Fig. 12, which is translated from the XML document shown in Fig. 10. The upper and lower parts of Fig. 11 correspond to the upper and lower parts of Fig. 12, respectively.

To produce the upper part of Fig. 12, we need to set the value for the three colSpan attributes in the <td> tags for the three columns named Element Name, Alias, and Repeatability. This is because the elements in these columns are hierarchical. The <td> tags for the other columns Data Type, Size, Required, Attribute, Provider, and DC-Element do not have the colSpan attribute. We found that the value of colSpan equals the number of layers in the XML document. As Fig. 10 shows, the root element, which is Storage_Place, is at layer 1. When an element has at least a child element, the number of layers grows by 1. Clearly, from Fig. 10 we can see the number of layers of this XML document is 2. Therefore, each of the three values of the colSpan attributes in Fig. 12 has been set to 2.

To produce the lower part of Fig. 12, we need to set the value for the rowSpan attribute in the <td> tags for three elements, such as Storage_Place. This is because the three elements are hierarchical. We noticed that the value of rowSpan equals the number of child-elements of the element. As Fig. 10 shows, the element Storage_Place has two child-elements Name and Number. Therefore, as shown in Fig. 12, the value of rowSpan in <td> tag of Storage_Place, for example, is 2.

The program that translates the XML document into HTML is written with the Document Object Model (DOM) [18] package from JAXP. DOM is a

```
<table border="1">
  <tr>
    <td bgColor="#c0c0c0" colSpan="2">Element Name</td>
    <td bgColor="#c0c0c0" colSpan="2">Alias</td>
    <td bgColor="#c0c0c0">Data Type</td>
    <td bgColor="#c0c0c0">Size</td>
    <td bgColor="#c0c0c0">Required</td>
    <td bgColor="#c0c0c0" colSpan="2">Repeatability</td>
    <td bgColor="#c0c0c0">Attribute</td>
    <td bgColor="#c0c0c0">Provider</td>
    <td bgColor="#c0c0c0">DC-Element</td>
  </tr>
  <tr>
    <td  rowSpan="2">Storage_Place</td>
    <td>Name</td>
    <td  rowSpan="2">典藏位置</td>
    <td>名稱</td>
    <td>Varchar</td>
    <td>30</td>
    <td>＊</td>
    <td  rowSpan="2">   </td>
    <td>  </td>
    <td>Select_List</td>
    <td>Person</td>
    <td>Description</td>
  </tr>
  <tr>
    <td>Number</td>
    <td>編號</td>
    <td>Varchar</td>
    <td>30</td>
    <td>＊</td>
    <td>  </td>
    <td>  </td>
    <td>  </td>
    <td>Description</td>
  </tr>
</table>
```

Fig. 12. HTML data translated from the XML document shown in Fig. 10 to display the table shown in Fig. 11.

language-neutral interface that enables a program to dynamically access and update XML documents. The translator needs to count the number of layers and the number of child-elements of each element and then finds out the name of each element to create the HTML file.

7 Conclusions

For easing the creation of metadata schemas of digital archives, we have presented an XML structure to represent the metadata schema, and designed and implemented a metadata schema repository to store metadata schemas as XML documents in a native XML database. The metadata schema repository supports analysis, creation, storage, search, and access management of metadata schemas. The repository can contribute to sharing and reusing of metadata schemas over the Internet, and serve as a centralized portal for reusing and creating metadata schemas. A user can access it through any Web browser. Ultimately, the repository is useful for fast construction of digital archive systems.

Since the metadata schema must be displayed on a Web browser, we need to translate the XML representation of metadata schema into the HTML version. For this, we have also presented the key to such a translation.

Some further research directions are as follows.

(1) Intelligent mechanism to help metadata schema construction.
When the need for more metadata schemas increases, it is desirable to have an intelligent mechanism to automatically analyze the characteristics of each element and the relations between elements of metadata schemas. Thereby, suggestions for metadata schemas can be offered. Users should find these suggestions useful in constructing metadata schemas. The good tool should be continuously improved based on feedbacks from users.

(2) Performance evaluation of the metadata schema repository.
When a repository contains many metadata schemas, we should find out the performance bottleneck, if any. We can test whether the metadata schema repository can efficiently support many concurrent user operations.

(3) XML-based digital archive systems.
At present, most digital archive systems use a relational database system to store data. Thus, data structures and programs must be modified when the metadata schema changes. This would require a large amount of time and effort. If we store metadata in XML documents, we can increase the flexibility of data structures to lower the cost of maintenance.

Acknowledgment
This research was supported in part by the National Science Council of Taiwan under contract NSC 94-2422-H-011-001.

References:

[1] L. Dempsey, ROADS to desire: Some UK and other European metadata and resource discovery projects, D-Lib, Vol. 2, 1996, http://www.dlib.org/dlib/july96/07dempsey.html.

[2] R. Wendler, LDI update: Metadata in the library, Harvard University Library Notes, 1999, pp. 4-5, http://hul.harvard.edu/publications/hul_notes_pdfs/HULN_1286.pdf.

[3] L.M. Chan, M.L. Zeng, Metadata interoperability and standardization — A study of methodology, Part I: Achieving interoperability at the schema level, D-Lib, Vol. 12, 2006, http://www.dlib.org/dlib/june06/06contents.html.

[4] Y.-N. Chen, S.-J. Chen, S.C. Lin, A metadata lifecycle model for digital libraries: Methodology and application for an evidence-based approach to library research, in *Proceedings of 69th IFLA General Conference and Council*, Berlin, Germany, 2003, http://www.ifla.org/IV/ifla69/papers/141e-Chen_Chen_Lin.pdf.

[5] M. Nagamori, T. Baker, T. Sakaguchi, S. Sugimoto, K. Tabata, A multilingual metadata schema registry based on RDF schema, in *Proceedings of International Conference on Dublin Core and Metadata Applications*, Tokyo, Japan, 2001, pp. 209-212.

[6] R. Heery, P. Johnston, C. Fülöp, A. Micsik, Metadata schema registries in the partially Semantic Web: the CORES experience, in *Proceedings of Dublin Core Conference*, Seattle, WA, 2003, pp. 11-18.

[7] Dublin Core Metadata Initiative, http://dublincore.org/.

[8] W3C, Resource Description Framework (RDF), http://www.w3.org/RDF/.

[9] E.R. Harold, W.S. Means, *XML in a Nutshell*, 3rd ed., O'Reilly, Sebastopol, CA, 2004.

[10] Metadata Architecture and Application Team, Guideline for Metadata System Analysis and Specification, http://www.sinica.edu.tw/~metadata/news/md_spec-diy_v0-6.pdf.

[11] Metadata Architecture and Application Team, Han Dynasty Wooden Slips Metadata Requirement Specification, http://www.sinica.edu.tw/~metadata/project/filebox/stone-HangJan/stone_HangJan_spec_v1-0.pdf.

[12] R. Bourret, Native XML databases in the real world, in *Proceedings of XML 2005 Conference & Exhibition*, Atlanta, GA, 2005, http://www.idealliance.org/proceedings/xml05/ship/58/Native_XML_Databases.PDF.

[13] J. Brittain, I.F. Darwin, *Tomcat: The Definitive Guide*, O'Reilly, Sebastopol, CA, 2003.

[14] Java API for XML Processing (JAXP), http://java.sun.com/xml/jaxp/index.jsp .

[15] A.B. Chaudhri, A. Rashid, R. Zicari, *XML Data Management: Native XML and XML-Enabled Database Systems*, Addison-Wesley, Boston, MA, 2003.

[16] W3C, XQuery 1.0: An XML query language, http://www.w3.org/TR/xquery/.

[17] M. Baca, P. Harpring, Categories for the Description of Works of Art (CDWA), http://www.getty.edu/research/conducting_research/standards/cdwa/.

[18] W3C, Document Object Model (DOM), http://www.w3.org/DOM/.