# AN IMPLEMENTATION OF STORAGE AND RETRIEVAL METHODS FOR XML DOCUMENTS IN RELATIONAL DBMSs

by

Aişe Zülal SU

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering



August 2003
Istanbul, Turkey

## APPROVAL PAGE

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assist. Prof. Atakan Kurt
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist Prof. Atakan Kurt
Supervisor

Examining Committee Members

Assist. Prof. Atakan Kurt (supervisor)

Assoc. Prof. Haluk Gümüşkaya

Assist. Prof. Metin Artıklar

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

Assist. Prof. Nurullah Arslan
Director

Date

# AN IMPLEMENTATION OF STORAGE AND RETRIEVAL METHODS FOR XML DOCUMENTS IN RELATIONAL DBMSs

Aişe Zülal SU

M. S. Thesis - Computer Engineering
August 2003

Supervisor: Assist. Prof. Atakan Kurt

## ABSTRACT

In this thesis, our aim is to implement a middleware for storing and retrieving XML documents in Relational Database Management Systems. While storing XML documents in RDBMS, several methods can be used. We chose to implement fixed schema methods. These methods store any well-formed XML documents in fixed schema tables and do not require extending data models or SQL in order to support storage and querying of XML documents. So the implementation can be run with small changes for most RDBMSs. In our implementation, we used MySQL 4.0.13-nt as the database system and PhpMyAdmin 2.4.0 program as the web interface. We extended the PhpMyAdmin with a set of PHP classes. The implementation gives users two alternative storage methods (XRel and Edge).We also compared these methods for the performance of insertion and retrieval operations.

**Keywords:** XML, Documents, Storage Methods, XML-Enabled Database Systems

# XML DÖKÜMANLARINI İLİŞKİSEL VERİTABANI YÖNETİM SİSTEMLERİNDE SAKLAMA VE GERİ ELDE ETME METODLARININ BİR UYGULAMASI

Aişe Zülal SU

Yüksek Lisans Tezi – Bilgisayar Mühendisliği
Ağustos 2003

Tez Yöneticisi: Yrd.Doç.Dr. Atakan Kurt

## ÖZ

Bu tezi yaparken amacımız XML dökümanlarını verimli bir şekilde ilişkisel veritabanlarında saklayabileceğimiz bir uygulama geliştirmektir. XML dökümanlarını ilişkisel veritabanlarında saklamak için bir çok yöntem geliştirilmiştir. Biz bu uygulamamızda XML dökümanlarının yapısına bakmadan sadece iyi oluşumlu olan tüm XML dökümanlarını saklayabileceğimiz bir yöntemi benimsedik.Ayrıca bu yöntem kullanacağımız veritabanın varolan olanaklarıyla işlemini hallettiğinden, yapacağımız uygulama birçok veritabanı yönetim sistemlerine adapte edilebilir.Uygulamamızda veritabanı olarak MySQL 4.0.13-nt ve web arayüzü olarakta PhpMyAdmin seçtik. Yazdığımız PHP sınıflarını PhpMyAdmin programına gömerek, MySql'i XML dökümanlarını işleyebilecek hale getirdik. Bu uygulama da kullanıcıya iki farklı yöntem ile (XRel ve Edge) XML dökümanlarını saklama imkanı verdik. Sonuç olarak bu yöntemlerin performanslarını karşılaştırdık.

**Anahtar Kelimeler:** XML,XML Dökümanlarını saklama metodları , XML ve VTYS

# ACKNOWLEDGEMENT

First of all, I would like to thank my thesis supervisor Assistant Prof. Atakan KURT his motivation and will power kept me doing all the work. His academic activities will be references in my future studies.

I would like to thank Assistant Prof. Veli Hakkoymaz for his kind help.

I would like to thank friends of my department for time support and patience.

Finally, I would like to thank my husband and parents for their love, patience and encouragement.

# TABLE OF CONTENTS

# LIST OF TABLES

**TABLE**

# LIST OF FIGURES

**FIGURE**

# CHAPTER 1

# INTRODUCTION

## 1.1 Problem Definition

XML –eXtensible Markup Language (Bray et al., 2000) is emerging as a new major standard for representing data on the World Wide Web. Despite the excitement XML gives us, a important question comes to our minds is how to manage large amounts of data stored in XML efficiently.

To solve this problem, there are various approaches developed for effective, automatic conversion of XML documents into relational databases (Deutsch et al.,1999)(Florescu and Kossman,1999)(Jiang and Lu, 2001)(Shanmugasundaram et al.,1999)(Shanmugasundaram et al.,2001)(Tatarinov et al.,2002). Generally, we can classify all studies on XML storage methods into two categories as in (Goldfarb and Prescod, 2001). The first one is *integral storage*, which normally keeps the unparsed document intact and stores it as a BLOB in a database or as a file in the file system. With this approach, access to individual elements is realized by parsing and therefore it is inefficient. This approach is also rather efficient for assembling documents, but requires specific index structures for retrieval. The second one is *dispersed storage*; that is, the documents are parsed and their data values, and other components are stored in tables. When XML documents are stored via dispersed storage, a problem occurs about schema design. The problem of storage model design becomes a database schema design problem. In (YoshiKawa and Amagasa, 2001), the authors categorize such database schemas into two categories: *dynamic schema mapping approach* and *fixed schema mapping approach*. In the former, the design of database schema is based on the understanding of DTD (Document Type Descriptor) that describes the structure of XML documents. In the latter, a fixed database schema is used to store any XML documents

without the assistance of DTD. As indicated in (YoshiKawa and Amagasa, 2001) (Jiang et al., 2000), the main advantages of the fixed schema mapping approach are;

a. it is capable of supporting any sophisticated XML documents that are considered either as static (the DTDs are not changed) or dynamic (the DTDs vary from time to time),

b. it is capable of supporting well-formed and non-DTD XML documents,

c. it does not require extending the power of database models, in order to support XML documents. Therefore, it is possible to store large XML documents in database management systems.

However, database vendors such as IBM, Oracle, Microsoft and Sybase have developed tools to assist in converting XML documents into relational tables. The common features of all vendors are to use dynamic schema mapping approach for storing XML documents or to keep unparsed document intact and store it as a BLOB.

Finally, we decided to develop an implementation that takes advantage of the fixed schema mapping approach. We adapted PhpMyAdmin 2.4.0 web base interface for storing and retrieving XML documents to MySQL open source database. With our implementation, users can store XML documents regardless of structure of documents like Oracle XML SQL Utility Model or IBM DB2 XML Extender. Also this implementation does not require extending SQL. The application gives users two alternatives (Xrel and Edge) for storing and retrieving XML document. Then these alternatives were compared with storage and retrieval time of XML documents. Finally, the results were discussed.

Our implementation can help both the users who want to manage XML documents in Relational Database with fixed schema methods and the researchers who study XML technologies.

## 1.2 Related Work

Database vendors such have developed tools to help in converting XML documents into relational tables. In the following, the storing and retrieving methods that are developed by Microsoft, IBM and Oracle database vendors, will be explained.

More information about databases with extensions for transferring data between XML documents and themselves can be found at (Dayen, 2001).

### 1.2.1 Access 2002

Access 2002 transfers data to/from XML documents using a table-based mapping. Individual data values must be in child elements (attributes are ignored) and table/column names must match element names. In addition, Access can create an XML Schema document describing exported data. Further information about XML support in Access 2002 can be found at (Frank, 2001).

### 1.2.2 IBM DB2 XML Extender

As indicated in (Dayen, 2001) IBM's XML Extender provides two access and storage methods for using DB2 as an XML repository:

- **XML Column**: stores and retrieves entire XML document as DB2 column data.
- **XML Collection**: decomposes XML documents into a collection of relational tables, or composes XML documents from a collection of relational tables.

The mapping between the structure of the XML document and the database tables is defined by means of a Data Access Definition (DAD) file. DAD refers to a processed document DTD, thus providing a bridge between an XML documents, its DTD, and mapping rules onto database tables. XML document composition and decomposition are handled by stored procedures: stored procedure dxxGenXML() extracts XML documents from a database; stored procedure dxxShredXML() stores XML documents in a database. Both of them need DAD file as input parameter. Further information about XML support in DB2 can be found at (DB2 XML Extender, 2002).

### 1.2.3 Oracle 9i XDB

Oracle 9i XDB has all advantages of relational database technology and XML technology at the same time.It supports both XML-enabled and native storage of XML

data. A number of operators have been added to SQL to help view XML data as relational data and vice versa.

XMLType data type has an important role in XML DB. This is a predefined object type that can store an XML document and provide lots for features for managing XML storage. Like any object type, XMLType can be used as the data type of a column in a table or view. XMLType data can be stored in either of two ways: with object-relational storage or as a CLOB.

When using object-relational storage, users define the mapping with an XML Schema. This states the name and data type (which can be an object type) of the SQL structure used to store a given element or attribute. Users can create their own mappings or use a default mapping generated by XDB. (Bourret, 2003)

In the following example, xml document stored with using XMLType data type.

```
<!ELEMENT FXTRADE (CURRENCY1, CURRENCY2, AMOUNT, SETTLEMENT,
ACCOUNT)>
<!ELEMENT ACCOUNT (BANKCODE,BANKACCT)>
<!ELEMENT BANKCODE  (#PCDATA)>
<!ELEMENT BANKACCT  (#PCDATA)>
<!ELEMENT CURRENCY1  (#PCDATA)>
<!ELEMENT CURRENCY2  (#PCDATA)>
<!ELEMENT AMOUNT    (#PCDATA)>
<!ELEMENT SETTLEMENT (#PCDATA)>
```

(a) DTD of Bank.xml document

```
CREATE TABLE FXTRADE
{

 CURRENCY1  CHAR(3),
 CURRENCY2  CHAR(3),
 AMOUNT     NUMERIC (18,2)
 SETTLEMENT DATE,
 ACCOUNT  sys.XMLType

}
```

(b) The FXTRADE Table

Figure 1.1 Storing XML documents in Oracle 9i (Dayen, 2001)

Further information about XML support in Oracle 9i can be found in (Oracle XML DB, 2002).

### 1.2.4 SQL Server 2000

Microsoft SQL Server 2000 extends SQL language for storing and retrieving XML documents. Retrieving extends a SELECT-clause by use of the FOR XML construct. Storing extends with a row set function OPENXML. Before XML document stores in tables, "field-element" association should be defined using XPath notation.

In (Dayen, 2001), author compared vendor with each other like Table 1-1

Table 1.1 XML Product Comparisons

| Vendor | Mapping rules | Single table / Multiple tables | Means of transformation | Symmetrical extraction / storing |
|---|---|---|---|---|
| **Oracle** | Implicitly; by constructing object-relational data model | Multiple | Designated Java classes | Symmetrical, if XML document and object-relational model match |
| **IBM** | Data Access Definition file | Multiple | Designated stored procedures | Symmetrical |
| **Microsoft** | SQL extension; row set function | Multiple for extraction; Single for storing | By using SQL construct FOR XML and row set OPENXML | Asymmetrical |

The common feature of these vendors is to use either dynamic schema mapping approach or BLOB methods for storing XML documents.

In this study, we want to implement fixed schema mapping approach that does not require a new special mapping rules while storing or retrieving an XML document in or from RDBMS.

## 1.3 Outline of Thesis

The next chapter introduces basics of XML. The Chapter 3 gives details of the fixed schema mapping approach. Chapter 4 describes our implementation. Experimental results are shown in Chapter 5. Conclusions are given in Chapter 6.

# CHAPTER 2

# BACKGROUND

This chapter gives overview of XML, XML components and XML technology.

## 2.1 What is XML?

The World Wide Web (WWW) and The HyperText Markup Language (HTML) have brought us to a new information age. The extraordinary growth of the Web gives us opportunities to easily distribute documents to anywhere and anybody in the world. While the Web documents become larger and more complex, the structure and data checking needed for large-scale commercial publishing and data exchange (Bosak, 2001).

To solve this problem, the World Wide Web Consortium (W3C) proposed XML – the eXtensible Markup Language – as a standard for the dispersion and exchange of information from all types of data sources and applications.(Bray et al.,2000) (Widow,1999) The basic idea underlying XML is very simple: tags on data elements identify the meaning of data, rather than for example, specifying how the data should be formatted (as in HTML), and relationships between data elements are provided via simple nested and references. Web servers and applications encoding their data in XML can quickly make their information available in a simple and usable format and such information provide can work together much easier than before. Because information content is separated from information rendering (XSL), and data in XML is self-describing. It is easy to provide multiple views of the same data, and the difficulty of extracting useful data from HTML Web pages is greatly reduced. As and example, consider the HTML fragment shown in Figure 2.1

```
<UL>
 <LI>
        <P>M. Rousset,
            <A href="http://www-rocq.inria.fr/verso/LEVEL1/kick-off/sld034.htm">
                        Semantic Data Integration in Xyleme
            </A>, Presentation at INRIA, September 1999.
        </P>
 <LI>
        <P>R. Goldman and J. Widom,
            <A href="http://www-db.stanford.edu/pub/papers/interact.ps">
                        Interactive Query and Search in Semistructured Databases
            </A>, Technical Report, Stanford University, 1998.
        </P>
</UL>
```

Figure 2.1 An HTML Fragment (DBLP, 1999)

One way of encoding the same information of Figure 2.1 in XML is shown in Figure 2.2. Although document in XML formatted is containing more words, it provides the information in a far more convenient and usable format from the data management perspective. Furthermore, the XML document can be transformed and rendered easier than document formatted with HTML. (Brayetal, 2000)

```
<Publication URL=" http://www-rocq.inria.fr/verso/LEVEL1/kickoff/sld034.htm "
Authors=" M. Rousset ">
        <Title> Semantic Data Integration in Xyleme </Title>
        <Journal> Presentation at INRIA </Journal>
        <Valume></Valume>
        <Number></Number>
        <Pages></Pages>
        <Date>
                <Month>September </Month>
                <Year> 1999 </Year>
        </Date>
</Publication>

<Publication URL=" http://www-db.stanford.edu/pub/papers/interact.ps "
Authors=" R. Goldman and J. Widom ">
        <Title> Interactive Query and Search in Semistructured Databases </Title>
        <Journal> Technical Report, Stanford University </Journal>
        <Valume/>
        <Number/>
        <Pages/>
        <Date>
                <Month>September </Month>
                <Year> 1998 </Year>
        </Date>
</Publication>
```

Figure 2.2 XML Encoding of the HTML fragment in Figure 2.1 (DBLP, 1999)

## 2.2 Structure of XML Documents

XML documents are consisted of markup and content. There are six kinds of markup that can occur in an XML document: elements, entity references, comments, processing instructions, marked sections, and document type declarations. The following sections introduce each of these markup concepts.

### Elements

Elements are the most common form of markup. Delimited by angle brackets, most elements identify the nature of the content they surround. Following example shows us an element of XML document in Figure 2.2

<Title> Interactive Query and Search in Semistructured Databases </Title>

It is introduced by a start tag. In this example, start tag is: "<Title>". And it is terminated by an end element "</Title>". An element may be empty as in "<Number></Number>" and this can be abbreviated to: "<Number/>"

### Attributes

Attributes are name-value pairs that occur inside start-tags after the element name. Following example has a Publication element with the attribute URL having the value "http://www-db.stanford.edu/pub/papers/interact.ps". In XML, all attribute values must be quoted.

<Publication URL="http://www.db.stanford.edu/pub/interact.ps" Authors=" R. Goldman and J. Widom ">

### Entity References

In order to introduce markup into a document, some characters have been reserved to identify the start of markup. The left angle bracket, < , for instance, identifies the beginning of an element start- or end-tag. In order to insert these characters into your document as content, there must be an alternative way to represent them. In XML, entities are used to represent these special characters. Entities are also

used to refer to often repeated or varying text and to include the content of external files.

Every entity must have a unique name. In order to use an entity, you simply reference it by name. Entity references begin with the ampersand and end with a semicolon. For example, the lt entity inserts a literal < into a document. So the string <element> can be represented in an XML document as < element>.

**Comments**

Comments begin with <!-- and end with -->. Comments can contain any data except the literal string . You can place comments between markups anywhere in your document.Comments are not part of the textual content of an XML document. An XML processor is not required to pass them along to an application.

**Processing Instructions**

Processing Instructions are information for the application. PI's allow documents to contain instructions for applications. They are not really of interest to the XML parser. Instead, the instructions are passed to the application using the parser, because the purpose of processing instructions is to represent special instructions for the application. Like comments, they are not textually part of the XML document.

All processing instructions, including the XML declaration, begin with <? and end with ?>. Following the initial <?, you will find the name of the processing instruction. The PI begins with the PITarget used to identify the application to which the instruction is directed as shown in example:

```
<?name pidata?>
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

**CDATA Sections**

In a document, a CDATA section instructs the parser to ignore most markup characters.Consider a source code listing in an XML document. It might contain characters that the XML parser would ordinarily recognize as markup (< and &, for example). In order to prevent this, a CDATA section can be used.

```
<![CDATA[*p = &q;b = (i <= 3);]]>
```

Between the start of the section, <![CDATA[ and the end of the section, ]]>, all character data is passed directly to the application, without interpretation. Elements, entity references, comments, and processing instructions are all unrecognized and the characters that comprise them are passed literally to the application.The only string that cannot occur in a CDATA section is ]]>.

## 2.3 XML Validation – Document Type Definition

There are some specific rules for XML document to be properly written or, in other words to be well formed. If an XML document is not well-formed, it must be reported as being in error. The document in Figure 2.3 is a well formed XML document. Followings are the rules for an XML document to be *well formed* (Fan and Libkin, 2001):

- The document must be beginning with an XML declaration.
- An XML document may only have one unique root element.
- All no empty elements must have an opening and closing tag.
- And empty element is terminated with a forward slash preceding the "less-than" bracket of the tag.
- Nesting of tags is allowed, but tags must not overlap.
- All attributes values must be quoted.
- XML is a case-sensitive language.

If we want to make real use of XML, we should decribe unique guidelines. These guidelines specify the elements of our XML document, sequence of those elements, and what contents those elements contain. This is done using a DTD – Document Type Definition. When XML document follows the basic XML rules for well formed and rules of its specified DTD, it is said to be a *valid XML document*. In Figure 2.4 a valid XML document is given with its DTD preceding.

As you can see in Figure 2.4, XML elements are declared with a DTD. A DTD can appear in the same file as the XML document or in a separate file.

A DTD is enclosed in <!DOCTYPE name [DTD declaration]>. Name is the outermost enclosing tag names' and DTD declaration is the text of the rules of the DTD. DTD starts with the outermost element, also called the root element, which is "adress_book" in our example given in Figure 2.4. Let us look DTD rules with our example.

```
<?xml version="1.0"?>

<address_book>
 <card no="1">
        <name>
                <surname>Brown</surname> <given>Paul</given>  <other>Micheal</other>
        </name>
        <title>Prof.Dr</title>
        <address>
                <street>20th Floor, 300 Lakeside</street>
                <city>OAKLAND</city><state>CA</state><zip>98520</zip>
        </address>
        <contact>
                <phone>510 628 3993</phone>
        </contact>
 </card>

 <card no="2">
        <name>
                <surname>Buneman</surname><given>Peter</given>
        </name>
        <title>Instructor</title>
        <address>
                <street>23th Floor, 456 Leiben</street>
                <city>New York</city><state>TX</state><zip>5188</zip>
        </address>
        <contact>
                <phone>4 852 96 0000</phone>
                <phone>4 852 96 0102</phone>
        </contact>
 </card>
</address_book>
```

Figure 2.3 Well Formed XML Document

```
<?xml version="1.0"?>
<!DOCTYPE address_book [
<!ELEMENT address_book(card*)>
<!ELEMENT card (name,title, address,contact)>
<!ATTLIST card no CDATA #REQUIRED>
<!ELEMENT name  (surname, given, other?)>
<!ELEMENT surname #PCDATA>
<!ELEMENT given #PCDATA>
<!ELEMENT other  #PCDATA>

<!ELEMENT title  #PCDATA>
<!ELEMENT address (street,city, state,zip)>

<!ELEMENT street #PCDATA>
<!ELEMENT city #PCDATA>
<!ELEMENT state  #PCDATA>
<!ELEMENT zip  #PCDATA>

<!ELEMENT contact  (phone*)>
<!ELEMENT phone #PCDATA>
]>

<address_book>
 <card no="1">
        <name>
                <surname>Brown</surname><given>Paul</given><other>Micheal</other>
        </name>
        <title>Prof.Dr</title>
        <address>
                <street>20th Floor, 300 Lakeside</street>
                <city>OAKLAND</city><state>CA</state><zip>98520</zip>
        </address>
        <contact><phone>510 628 3993</phone></contact>
</card>
<card no="2">
        <name>
                <surname>Buneman</surname> <given>Peter</given>
        </name>
        <title>Instructor</title>
        <address>
                <street>23th Floor, 456 Leiben</street>
                <city>New York</city><state>USA</state> <zip>5188</zip>
        </address>
        <contact><phone>4 852 96 0000</phone><phone>4 852 96 0102</phone></contact>
</card>
</address_book>
```

Figure 2.4 Valid XML File with internal DTD

In DTD, an element declaration has the following syntax:

```
<!ELEMENT element-name category>
<!ELEMENT element-name (element-content)>
```

Element can be declared with different content type as #EMPTY, only character data (#PCDATA) or with children. You can see two different types elements declaration in our example.

```
<!ELEMENT name  (surname, given, other?)>
<!ELEMENT address book(card*)>
<!ELEMENT street #PCDATA>
```

The "?","*" or "+" regular expression are refer to occurrence of the same element. "*" refers to zero or more times, "+" refers to one or more time and "?" mark refers to zero or one time can be occurrence.

In a DTD, Attributes are declared with an ATTLIST declaration. An attribute declaration has the following syntax:

```
<!ATTLIST element-name attribute-name  attribute-type default-value>
```

Following example gives attribute declaration in DTD :

```
<!ATTLIST card no CDATA #REQUIRED>
```

Following example gives attribute declaration in XML document:

```
<card no="1">
```

The #REQUIRED keyword is used whether you don't have an option for a default value, but still want to force the attribute to be present.

The detail information about XML DTD can be found at (Bray et al., 2000)

## 2.4 W3C XML Technologies

This section explains sub-standards with coming XML standard. These standards describe various aspects of document representation and reproduction.

### XML Schema

XML Schema describes the structure of an XML document just like a DTD. It defines elements, attributes, child elements, order of child elements, number of child elements, whether an element is empty or can include text, data types and values for elements and attributes. So that XML Schemas are richer and more useful than DTDs. Also XML Schemas are written in XML language. So we can think XML Schema file as XML documents. Like XML documents, it is parsed or transformed with XSLT and more.

Further information about syntax of XMLSchema can be found at (Biron and Malhotra, 2001)

### XPath

XPath is a language for addressing parts of an XML document. XPath was designed to be used by both XSLT and XPointer. Without XPath knowledge you will not be able to create XSLT documents. Following examples give you an opinion about how Xpath language is.

/address_book: This XPath expression selects the ROOT element address_book.

/address_book/card [title="Instructor"]: This XPath expression selects all the cards elements which title's equals Instructor and parent element is address_book.

Further information about XPath language can be found at (Clark and DeRose, 1999).

### XSL - eXtensible Stylesheet Language

XSL is a language for expressing style sheets. It consists of three parts:

- XSLT (a language for transforming XML documents)
- XPath (a language for defining parts of an XML document)

- XSL Formatting Objects (a vocabulary for formatting XML documents)

If it is explained much more clearly, XSL is a language that can transform XML into XHTML, filter and sort XML data and define parts of an XML document, or format XML data based on the data value, like displaying negative numbers in red or output XML data to different devices, like screen, paper or voice. Further information about XSL can be found at (Clark, 1999).

**XLink-XPointer-Xbase**

The XML Linking Language (XLink), allows elements to be inserted into XML documents in order to create links between XML resources.

The XML Pointer Language (XPointer), supports addressing into the internal structures of XML documents, such as elements, attributes, and content

XML Base is a standard for defining a default reference to external XML resources (similar to <base> in HTML).Further information about these standarts can be found at (DeRose et al., 2001)

**XQuery**

The XML Query Language supports query facilities to extract data from XML documents. The mission of the XQuery is providing interaction between the web world and the database world. Ultimately, collections of XML files will be accessed like databases.

An XQuery language is a just an expression, together with some optional function and other definitions. Also it uses *path expressions* from XPath.So that XQuery can be viewed as a generalization of XPath. For this reason the XPath specification is also being revised by the XQuery committee.

Further information and tutorials on Xquery language can be found at (Chamberlin et al., 2001).

**XML DOM**

The XML Document Object Model (DOM) is a programming interface for XML documents. It defines the way an XML document can be accessed and manipulated.The XML DOM is designed to be used with any programming language and any operating system.With the XML DOM, a programmer can create an XML document, navigate its structure, and add, modify, or delete its elements.

Further information about these standards can be found at (Apparao et al., 1998)

# CHAPTER 3

# METHODS FOR STORING XML DOCUMENTS IN RDBMS

## 3.1 An XML Data Model

XML documents have been developed in four different proposed thought W3C: the Infoset model (Cowan and Tobin, 2001), the Xpath data model (Clark and DeRose, 1999), the DOM model (Le Hors et al., 2000) (Apparao et al., 1998), and the XQuery 1.0 and Xpath 2.0 Data Model (Femandez and March, 2001). These models describe document structure, which is often used to encode enterprise data.

In our study, we employ the data model of XPath (Clark and DeRose, 1999) to represent XML documents and introduce the XPath data model according to our example of XML document shown at Figure 1.4. The full specifications of the data model can be found in (Clark and DeRose, 1999).

In the XPath data model, XML documents are modeled as as an ordered and directed labeled tree. There are seven types of nodes. In this thesis, we consider only the following four types of nodes for the sake of simplicity: root, element, text and attribute. The root node is a virtual node pointing to the root element of an XML document. Elements in an XML document are represented as an element node with an expanded-name (the element-type name specified in the tag). Element nodes can have n ($n \geq 0$) other elements or text as its children. A text node or an element can only have one parent. Text nodes are string-valued leaf-nodes. Text nodes do not have any child nodes. An element node can have a set of attribute nodes. An attribute node has an attribute-name and an attribute-value. Attribute nodes do not have any children. It is interesting to know that, despite the fact that an element node is the parent of a set of attribute nodes associated with; the attribute nodes are not the children of the element node by definition.

Figure 3.1 shows an XML data graph such that a left-to-right and depth-first traversal describes the order of the XML content within our document in Figure 2.4.

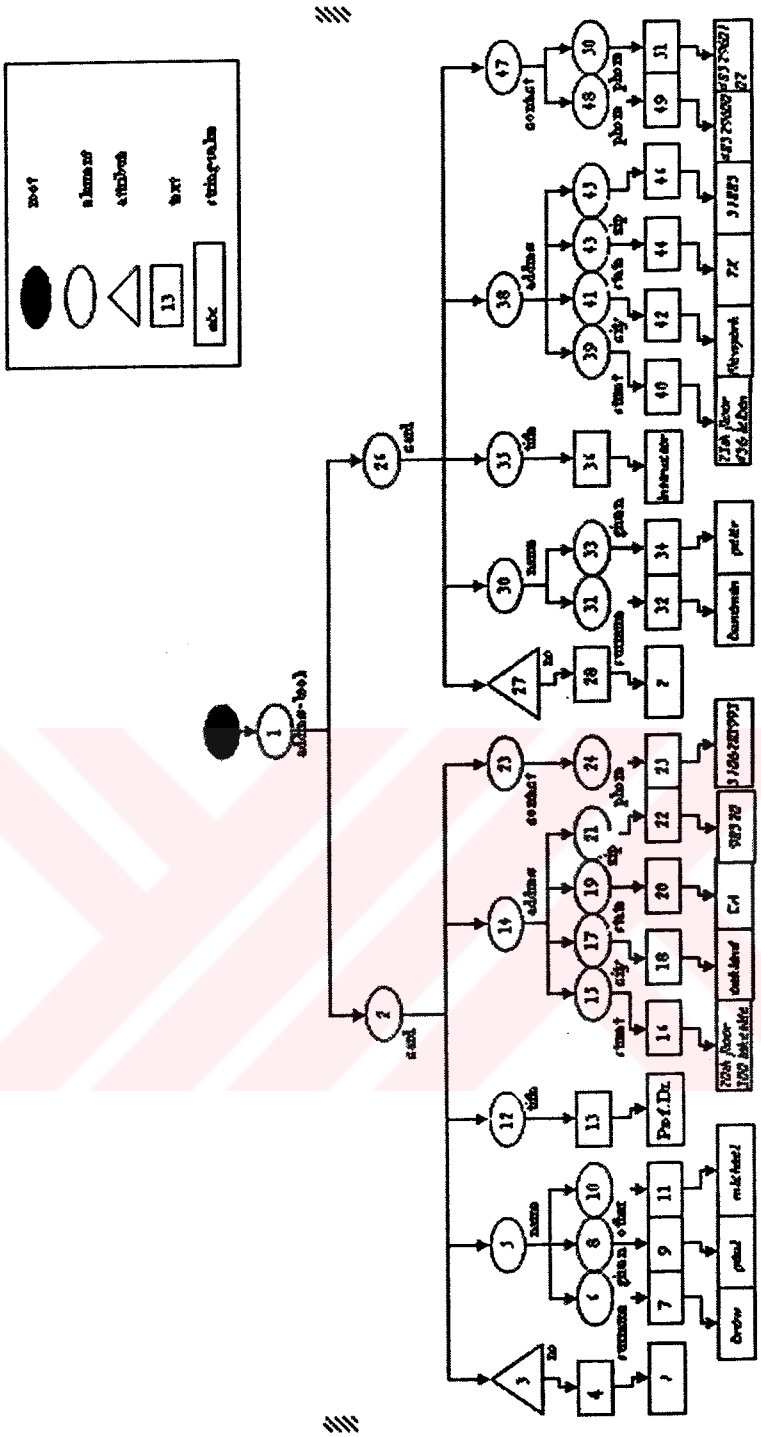

Figure 3.1 XML Tree

## 3.2 Three Existing Methods

### 3.2.1 Florescu and Kossmans' Method (Edge)

The Edge method stores all edges of the graph that represents an XML document in a single table named Edge (Florescu and Kossman, 1999).

Edge (Source, Ordinal, Target, Name, Flag, Value)

Table 3.1 Edge Table for the Data Graph in Figure 1.4

| Source | Ordinal | Target | Name | Flag | Value |
|---|---|---|---|---|---|
| 0 | 1 | 1 | address_book | ref | - |
| 1 | 1 | 2 | card | ref | - |
| 2 | 1 | 3 | @no | val | "1" |
| 2 | 2 | 5 | name | ref | - |
| 5 | 1 | 6 | surname | val | "Brown" |
| 5 | 2 | 8 | given | val | "Paul" |
| 5 | 3 | 10 | other | val | "Michael" |
| 2 | 3 | 12 | title | val | "Prof.Dr." |
| 2 | 4 | 14 | address | ref | - |
| 14 | 1 | 15 | street | val | "20th floor 300 Lakeside" |
| 14 | 2 | 17 | city | val | "oakland" |
| 14 | 3 | 19 | state | val | "CA" |
| 14 | 4 | 21 | zip | val | "98520" |
| 2 | 5 | 23 | contact | ref | - |
| 23 | 1 | 24 | phone | val | "510 628 39 93" |
| 1 | 2 | 26 | card | ref | - |
| 26 | 1 | 27 | @no | val | "2" |
| 26 | 1 | 30 | name | ref | - |
| 30 | 1 | 31 | surname | val | "Buneman" |
| 30 | 2 | 33 | given | val | "Peter" |
| 26 | 2 | 35 | title | val | "Instructor" |
| 26 | 3 | 38 | address | ref | - |
| 38 | 1 | 39 | street | val | "23th floor 456 Leiben" |
| 38 | 2 | 41 | city | val | "Newyork" |
| 38 | 3 | 43 | state | val | "TX" |
| 38 | 4 | 45 | zip | val | "51885" |
| 26 | 4 | 47 | contact | ref | - |
| 47 | 1 | 49 | phone | val | "485 29 600" |
| 47 | 2 | 51 | phone | val | "485 296 01 02" |

An edge is specified by two node identifiers, namely Source and Target. The Name attribute stores the element or attribute name. The Ordinal attribute records the ordinal of the edge among its siblings. A Flag value indicates whether the target node is an inter-object reference (ref) or points to a value (val). Table 3.1 shows the Edge table for the XML data graph given in Figure 3.1. For example, here, the tuple, (5, 2, 8, given, val, "Paul"), describes the element that has the name *given* and *value* "Paul". The parent node of this element is 5 and. Its ordinal is 2 (the second outgoing edge from the element node 5).

This approach is quite simple. It keeps parent-child edges only. But a large number of joins is needed during reconstruction.

### 3.2.2 Schmidts' Method (Monet)

As a variation of the Edge approach, Monet (Schmit et al., 2000) stores XML data graphs in multiple tables. In other words, Monet partitions the Edge table according to all possible label-paths. For each unique path, Monet creates a table. If unique path ends with text node, the table attributes are text node id and value of text. If unique path ends with element node, the table attributes are source node id, target node id and ordinal.

This approach is distinguished by two features. Firstly, the decomposition method is independent of the presence of DTDs, but rather explores the structure of the document availed after the decomposition. Secondly, it reduces the volume of the data irrelevant to a query that has to be processed during querying. Data relevant for a given query can be accessed directly in the form of a separate table avoiding large and expensive scans over irrelevant data. For the XML data graph in Figure 2.4, the corresponding Monet table is shown in Table 3.1

With Monet approach, Table 3.1 is partitioned into 34 tables. Although this is seen as a disadvantage, a user query that dedicated to a single path is efficiently processed. However, for complex queries with multiple paths, it needs to conduct joins as explained in (Jiang et al., 2001). Besides, Monet cannot handle regular path expressions involving "*" operator well due to the nature of the Monet schema. It is also difficult to support XML applications which change its structure by adding or removing paths. Therefore, the benefits of this approach are limited.

Table 3.2 Monet tables for the Data Graph in Figure 2.1

address_book/card

| Source | Target | Ordinal |
|---|---|---|
| 1 | 2 | 1 |
| 1 | 26 | 2 |

address_book/card/title

| Source | Target | Ordinal |
|---|---|---|
| 2 | 12 | 1 |
| 26 | 35 | 1 |

address_book/card/address

| Source | Target | Ordinal |
|---|---|---|
| 2 | 14 | 1 |
| 26 | 38 | 2 |

address_book/card/@no

| Source | Target | Ordinal |
|---|---|---|
| 2 | 3 | 1 |
| 26 | 27 | 1 |

address_book/card/contact

| Source | Target | Ordinal |
|---|---|---|
| 2 | 23 | 1 |
| 26 | 47 | 2 |

address_book/card/name

| Source | Target | Ordinal |
|---|---|---|
| 2 | 5 | 1 |
| 26 | 30 | 2 |

address_book/card/name/given

| Source | Target | Ordinal |
|---|---|---|
| 5 | 8 | 1 |
| 30 | 33 | 1 |

address_book/card/name/surname

| Source | Target | Ordinal |
|---|---|---|
| 5 | 6 | 1 |
| 30 | 31 | 1 |

address_book/card/name/other

| Source | Target | Ordinal |
|---|---|---|
| 5 | 10 | 1 |

address_book/card/name/surname/text

| Source | Target | Ordinal |
|---|---|---|
| 6 | 7 | 1 |
| 31 | 32 | 1 |

address_book/card/name/given/text

| Source | Target | Ordinal |
|---|---|---|
| 8 | 9 | 1 |
| 33 | 34 | 1 |

address_book/card/name/other/text

| Source | Target | Ordinal |
|---|---|---|
| 10 | 11 | 1 |

address_book/card/title/text

| Source | Target | Ordinal |
|---|---|---|
| 12 | 13 | 1 |
| 35 | 36 | 1 |

address_book/card/address/street

| Source | Target | Ordinal |
|---|---|---|
| 14 | 15 | 1 |
| 38 | 39 | 1 |

address_book/card/address/city

| Source | Target | Ordinal |
|---|---|---|
| 14 | 17 | 1 |
| 38 | 41 | 1 |

address_book/card/address/state

| Source | Target | Ordinal |
|---|---|---|
| 14 | 19 | 1 |
| 38 | 43 | 1 |

address_book/card/address/zip

| Source | Target | Ordinal |
|---|---|---|
| 14 | 21 | 1 |
| 38 | 43 | 1 |

address_book/card/address/street/text

| Source | Target | Ordinal |
|---|---|---|
| 15 | 16 | 1 |
| 39 | 40 | 1 |

address_book/card/address/city/text

| Source | Target | Ordinal |
|---|---|---|
| 17 | 18 | 1 |
| 41 | 42 | 1 |

address_book/card/address/state/text

| Source | Target | Ordinal |
|---|---|---|
| 19 | 20 | 1 |
| 43 | 44 | 1 |

address_book/card/address/zip/text

| Source | Target | Ordinal |
|---|---|---|
| 21 | 22 | 1 |
| 43 | 44 | 1 |

address_book/card/contact/phone

| Source | Target | Ordinal |
|---|---|---|
| 23 | 24 | 1 |
| 47 | 48 | 1 |
| 47 | 50 | 2 |

address_book/card/contact/phone/text

| Source | Target | Ordinal |
|---|---|---|
| 24 | 25 | 1 |
| 48 | 49 | 1 |
| 50 | 51 | 2 |

address_book/card/@no/text/data

| Target | Value |
|---|---|
| 3 | 1 |
| 27 | 2 |

address_book/card/name/surname/text/data

| Target | Value |
|---|---|
| 7 | Brown |
| 32 | Buneman |

address_book/card/name/given/text/data

| Target | Value |
|---|---|
| 9 | Pul |
| 34 | Peter |

address_book/card/name/other/text/data

| Target | Value |
|---|---|
| 11 | Michael |

address_book/card/address/street/text/data

| Target | Value |
|---|---|
| 16 | 20th floor 300 Lakeside |
| 40 | 23th floor 456 Leiben |

address_book/card/address/city/text/data

| Target | Value |
|---|---|
| 18 | Oakland |
| 42 | Newyork |

address_book/card/address/state/text/data

| Target | Value |
|---|---|
| 20 | CA |
| 44 | TX |

address_book/card/address/zip/text/data

| Target | Value |
|---|---|
| 22 | 98520 |
| 44 | 51885 |

address_book/card/contact/phone/text/data

| Target | Value |
|---|---|
| 25 | 510 6283993 |
| 49 | 4852960000 |
| 51 | 4852660102 |

address_book/card/title/text/data

| Target | Value |
|---|---|
| 13 | Prof.Dr |
| 36 | Instructor |

### 3.2.3 YoshiKawa and Amagasas' Method (XRel)

This approach stores XML document in four tables; "Element Table" which holds only document structure , Text table which holds only text data, "Attribute Table" and "Path Table" which holds unique path in XML document.

Element ( DocID, PathID, Start, End, Index)
Attribute (DocID, PathID, Start, End,Value)
Text (DocID, PathID, Start, End,Value)
Path (PathID,Pathexp)

Unlike Monet, this approach uses a fix number of tables. Again unlike Edge approach, a node is described with start and end positions. The region (or the pair of start and end positions) implies a containment relationship. For example, a node, $n_i$, is reachable from another node $n_j$, if the region of $n_i$ is included in the region of $n_j$. The containment relationship is not for parent-child relationship, but rather ancestor-descendant relationship.

In our study, we modified some attributes of tables in Xrel approach. Most of XPath queries are based on parent-child relationship. With this Xrel schema, if we want to find parent of its nodes, we have to parse pathexp column in Path table. Instead of this, we added a ParentID column to Element, Attribute and Text tables to find parent node easily. In addition, we put NodeID and last descendant node id (EndDescID) columns instead of start and end columns in Element table.

A new Xrel schema can be defined as follows:

Table 3.3 Tables of XRel

| PathID | PathExp |
|---|---|
| 1 | /address_book |
| 2 | /address_book/card |
| 3 | /address_book/card/@no |
| 4 | /address_book/card/name |
| 5 | /address_book/card/name/surname |
| 6 | /address_book/card/name/given |
| 7 | /address_book/card/name/other |
| 8 | /address_book/card/title |
| 9 | /address_book/card/address |
| 10 | /address_book/card/address/street |
| 11 | /address_book/card/address/city |
| 12 | /address_book/card/address/state |
| 13 | /address_book/card/address/zip |
| 14 | /address_book/card/contact |
| 15 | /address_book/card/contact/phone |

(a) Path Table

| NodeId | PathID | ParentID | Value |
|---|---|---|---|
| 7 | 5 | 6 | Brown |
| 9 | 6 | 8 | Paul |
| 11 | 7 | 10 | Michael |
| 13 | 8 | 12 | Prof.Dr. |
| 16 | 9 | 15 | 20th floor 300 Lakeside |
| 18 | 10 | 17 | Oakland |
| 20 | 11 | 19 | CA |
| 22 | 12 | 21 | 98520 |
| 25 | 15 | 24 | 5106283993 |
| 32 | 5 | 31 | Buneman |
| 34 | 6 | 33 | Peter |
| 36 | 8 | 35 | Instrucotor |
| 40 | 9 | 39 | 23th floor 456 Leiben |
| 42 | 10 | 41 | NewYork |
| 44 | 11 | 43 | TX |
| 46 | 12 | 45 | 51885 |
| 49 | 15 | 48 | 4852960000 |
| 51 | 15 | 50 | 4852960102 |

(b) Text Table

| NodeId | EndDescID | PathID | ParentID | Ordinal |
|---|---|---|---|---|
| 1 | 51 | 1 | 0 | 1 |
| 2 | 25 | 2 | 1 | 1 |
| 5 | 11 | 4 | 2 | 1 |
| 6 | 7 | 5 | 5 | 1 |
| 8 | 9 | 6 | 5 | 1 |
| 10 | 11 | 7 | 5 | 1 |
| 12 | 13 | 8 | 2 | 1 |
| 14 | 22 | 9 | 2 | 1 |
| 15 | 16 | 10 | 14 | 1 |
| 17 | 18 | 11 | 14 | 1 |
| 19 | 20 | 12 | 14 | 1 |
| 21 | 22 | 13 | 14 | 1 |
| 23 | 25 | 14 | 2 | 1 |
| 24 | 25 | 15 | 23 | 1 |
| 26 | 51 | 2 | 1 | 2 |
| 30 | 34 | 4 | 26 | 1 |
| 31 | 32 | 5 | 30 | 1 |
| 33 | 34 | 6 | 30 | 1 |
| 35 | 36 | 8 | 26 | 1 |
| 38 | 46 | 9 | 26 | 1 |
| 39 | 40 | 10 | 38 | 1 |
| 41 | 42 | 11 | 38 | 1 |
| 43 | 44 | 12 | 38 | 1 |
| 45 | 46 | 13 | 38 | 1 |
| 47 | 51 | 14 | 26 | 1 |
| 48 | 49 | 15 | 47 | 1 |
| 50 | 51 | 15 | 47 | 2 |

(c) Element Table

| NodeId | PathID | ParentID | Value |
|---|---|---|---|
| 3 | 3 | 2 | 1 |
| 27 | 3 | 26 | 2 |

(d) Attribute Table

# CHAPTER 4

## IMPLEMENTATION : Extending RDBMS MySQL for XML

In our study, we adopted a RDBMS for storing and retrieving XML documents according to fixed-schema mapping approaches. To achieve this, we should choose a database and then implement methods. We chose MySQL database which is a free and open source but relatively simple database system (MySQL AB, 2003). Like any database systems, the data in a database is organized into tables. And each table is organized into rows and columns. MySQL users manage data with *insert*, *delete*, *update* and *drop* operations. Also MySQL administrators manage user accounts, maintain log files or repair database tables...etc. With this implementation, we wish to add new operations to MySQL for XML documents. As a result, MySQL will become repository for both XML documents and relational data.

Our implementation adds "collection" support to the MySQL database. A collection is a set of XML documents stored in fixed schema of tables. In reality, MySQL database model does not change, but from the user point of view, inside a database there is not only tables but also collections. In addition, the users can not modify or access to the tables of collection directly. Users know the existing collection names and types only. And they can create, drop or browse collections. Also users can insert, browse or delete XML documents into collections.

To realize this new model, we have two ways'. One is to extend SQL language in MySQL. The other one is to create independent classes and embedding these classes to PhpMyAdmin program which is a web based interface between MySQL and users. The PhpMyAdmin program provides all database operations with user-friendly web interface. We apply second solution to our implementation. As explained in Chapter 3, there are three different methods that we can use in fixed schema mapping approach.

Edge and XRel methods can apply in our implementation. They use a fixed number of tables. But in Monet model, number of tables is not fixed. So we did not use this model in our implementation.

We can summarize our new model as follows;

- A database has two types of sub-structures; tables and collections.
- Relational data is stored in tables and XML documents are stored in collections
- Collections are created using two approaches (XRel and Edge).
- Multiple XML document can be stored in a collection.

We applied our model to MySQL 4.0.13-nt database and we used PhpMyAdmin 2.4.0 program as web interface. We extended PhpMyAdmin program with PHP classes. Seven classes are written. These are "XMLCollection", "XRel", "Edge", "XRelParse", "EdgeParse", "XRelXMLFile" and "EdgeXMLFile".

**XMLCollection Class**: XMLCollection class is embedded PHPMyAdmin program. The other classes are called from "XMLCollection" class. So that, we can say XMLCollection class is our main class. This class has three constructors and seven member functions. This class gives user the following options;

- Create collection using one of the methods (XRel and Monet)
- Insert XML document into collection
- Delete XML document from collection
- Retrieve XML document from collection
- Browse collections
- Drop collections

**XRel and Edge Class**: These classes have two constructors. If a new collection is wanted to create, these classes creates it according to collection type.If existing collection is wanted to call from database, XRel or Edge classes retrieve collection object from database.

**XRelParse and EdgeParse Class**: XRelParse and EdgeParse classes upload the XML document and then parse it. The document's element, attribute and text values are stored in tables of collection.

**XRelXMLFile and EdgeXMLFile Class**: These classes reconstruct xml documents from tables of collections.

These classes are given in Figure 4.1, 4.2, 4.3 and 4.4

```
class XMLCollection {

        var $db;    // database name
        var $name;  // collection name
        var $type;  // collection type (Edge,XRel,Monet)
        var $owner; // collection owner
        var $datetime;  // collection creation date and time
        var $tables;// tables of collection

        function XMLCollection($db){}

        function XMLCollection($db,  $name){}

        function XMLCollection($db, $name, $type, $date, $owner){}

        function create_collection($name,$type){}

        function insert_xmlfile($xmlfile){}

        function delete_xmlfile($xmlfile){}

        function browse_collection(){}

        function browse_xmlfile($xmlfile, $col_obj){}

        function drop_collection(){}

        function browse_collections(){}

}
```

Figure 4.1 XMLCollection Class

```
class XRel {

        var $name;          // collection name
        var $ElementT;      // T
        var $TextT;         // a
        var $AttributeT;    // b
        var $PathT;         // l
        var $DocumentT;     // e
                            // s

        function XRel($name) {

          // For existing collection, XRel tables name is taken from
database          }

        function XRel($name,$collectionID){

          // For new collection, XRel tables are created.
        }
}
```

Figure 4.2 XRel Class

```
class Edge {

        var $name;          // collection name
        var $EdgeT;         // Egde Tables
        var $ DocumentT;


        function Edge($name) {

          // For existing collection, Edge tables name is taken from database
        }

        function Edge($name,$collectionID){

          // For new collection, Edge tables are created.
        }
}
```

Figure 4.3 Edge Class

```
class XRelParse {

        var $file; // xml file name
        var $col; // collection name
        var $elements=0; // number of elements in $file variable
        var $texts=0; // number of text section in $file variable
        var $text_size=0; // size of text in $file variable
        var $attributes=0; number of attribute in $file variable
        var $size=0; // xml file size

        function XMLParse($myfile, $col_obj){}

        function startElement($parser, $name, $attrs){}

        function endElement($parser, $name){}

        function characterData($parser, $data){}
}
```

Figure 4.4 XRelParse Class

In the following part, XMLCollection class members are explained step by step.

**function XMLCollection**($db)

This is one of the constructor functions. This constructor selects database whose value comes with $db parameter. This constructor is used for browsing all collections in database.

**function XMLCollection($db, $name)**

This function takes two parameters. First one is database name, the other one is collection name. This constructor is used for creating a new collection. User chooses one of the collection types and write collection name on the form then click "Go" button. Collection type can be XRel or Edge. Each type forms special tables which are explained in chapter 3. The tables of collection take their name as [collection name] _[table name]. Also each collection is registered in _xml_collection table. This table holds all collection names and creation dates of that collection in the database. In the following, figures of our implementation are displayed. Figure 4.5 shows all databases in Mysql. The difference from original version is all databases are described with tables (t) and collection (c). For example the database of "deneme" has got 7 tables and 1 collection. When we choose one of the databases, tables and collection lists come to the

main frame. In Figure 4.6 a user wants to create a new collection called "student". He must choose one of the collection types and then press "Go" button. If processing is successful, the message and collection on the left side are shown as in Figure 4.7.

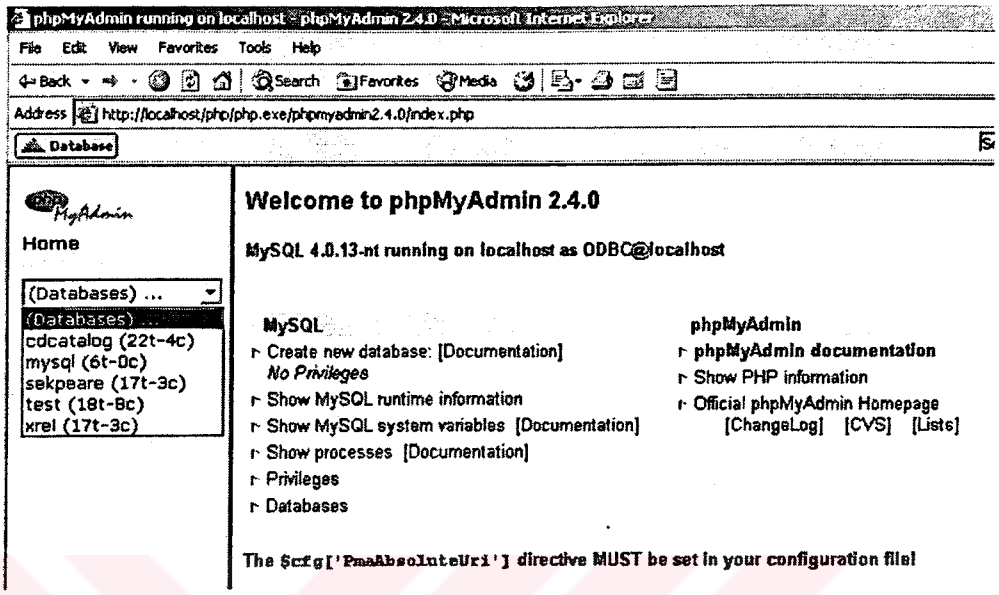


Figure 4.5 Index page of PhpMyAdmin program

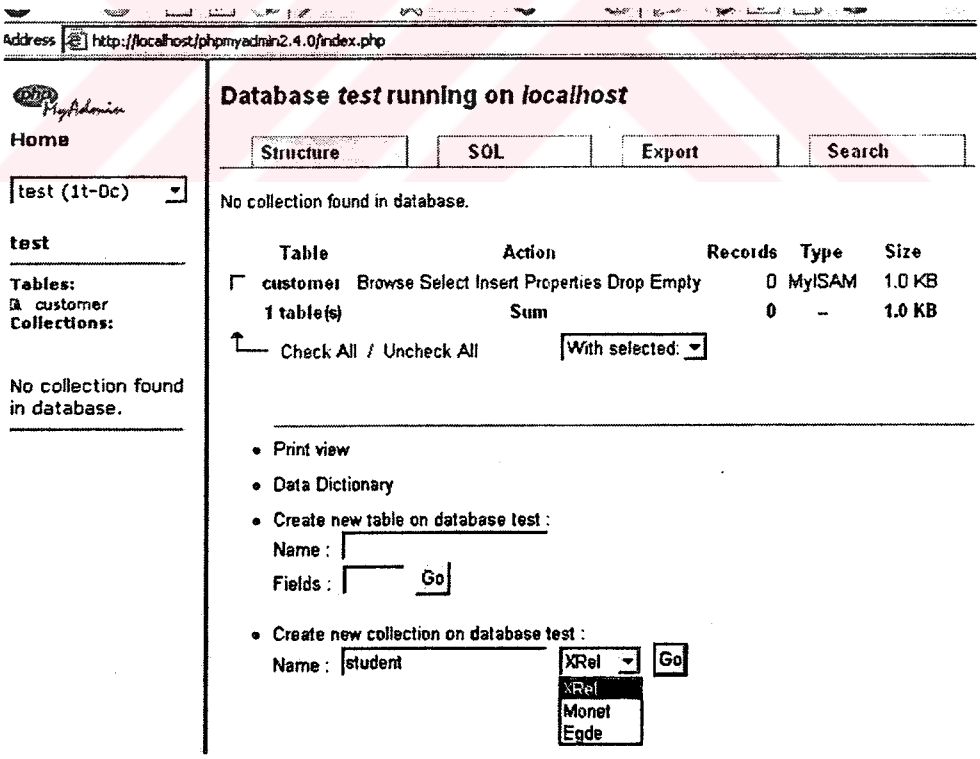


Figure 4.6 Creating collection in PhpMyAdmin

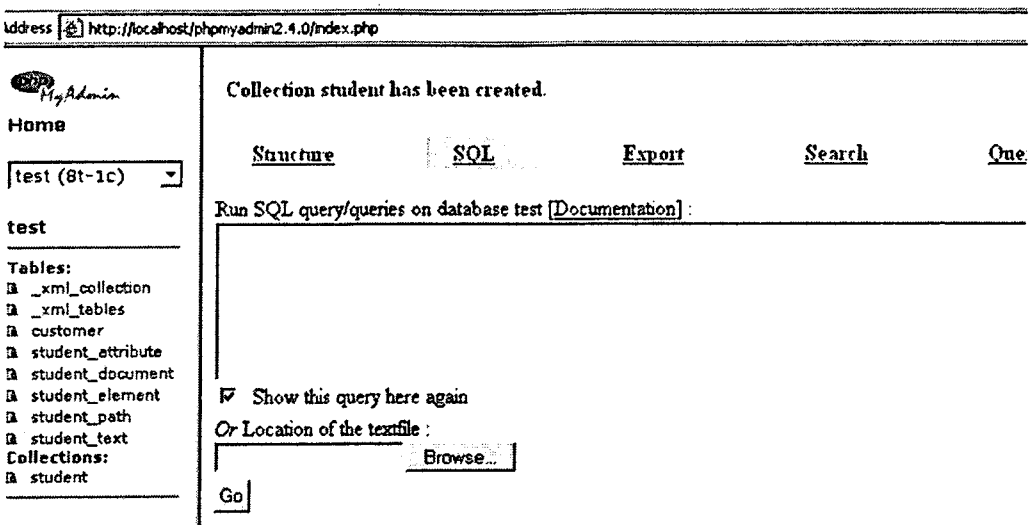


Figure 4.7 Student collection in test database

**function XMLCollection**($db, $name, $type, $date, $owner)

This constructor is used for holding existing collection in our class. When user wants to browse or drop collection or browse, delete, insert XML document in/from existing collection in database, XMLCollection class create with this constructor.

**function browse_collections()**

It returns all collection names, types, creation dates, number of xml documents and links of XML actions in the database as it can be seen at Figure 4.8

**function insert_xmlfile**($xmlfile, $col_obj)

This function create an XRel or Edge object according to collection type.This object parses XML document and stores elements, attributes and data one by one in the collection tables. If collection type is "XRel", function calls constructor of XRelParse class and if collection type is "Edge", function calls constructor of EdgeParse class. The message is displayed when the storing finished as in Figure 4.9.

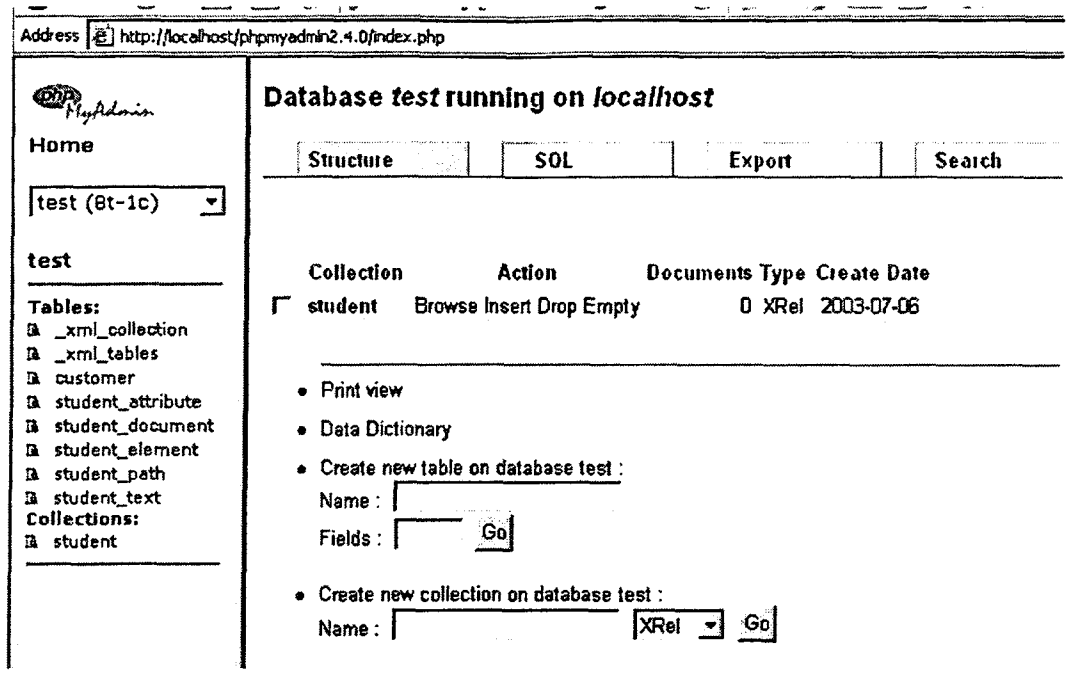


Figure 4.8 Browsing collections into test database

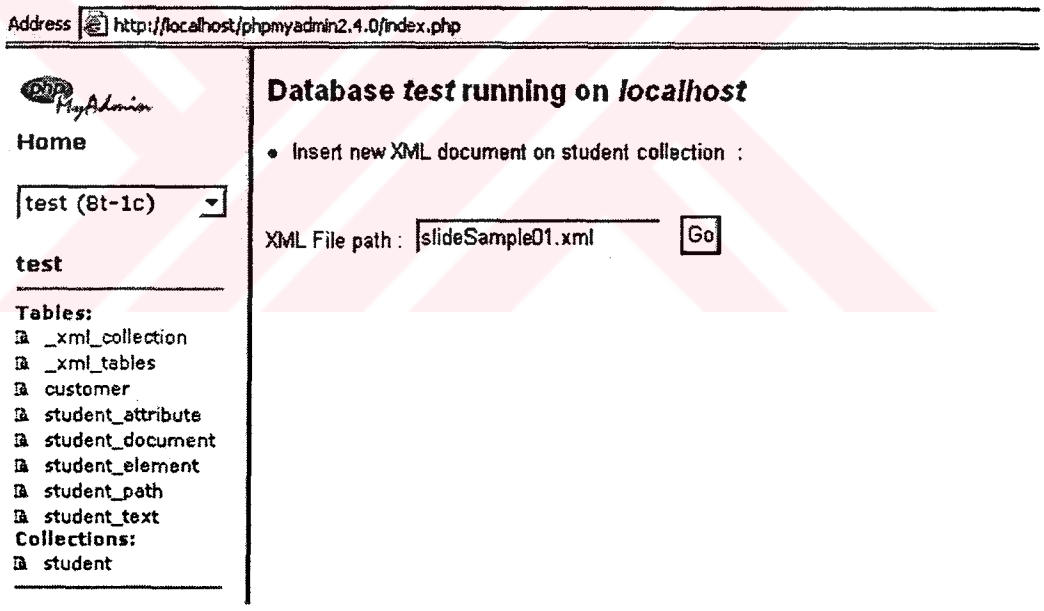


Figure 4.9 Insertion new XML file into student collection

**function browse_collection()**

This function returns storing xml files in the collection. It gives specific knowledge about each xml file in the collection as in Figure 4.10

**function delete_xmlfile**($xmlfile)

This function deletes xml file from the current collection. The xml document name value comes with input parameter.

**function browse_xmlfile**($xmlfile,$col_obj)

This function calls "XRelXMLFile" or "EdgeXMLFile" classes' constructor. These classes compose data from collection to new XML document. At the end of processing, a new XML document occurs in the server folder. User can download file to his computer

**function drop_collection**()

This function drops the current collection from database. All tables and xml files belonging to the collection are deleted.

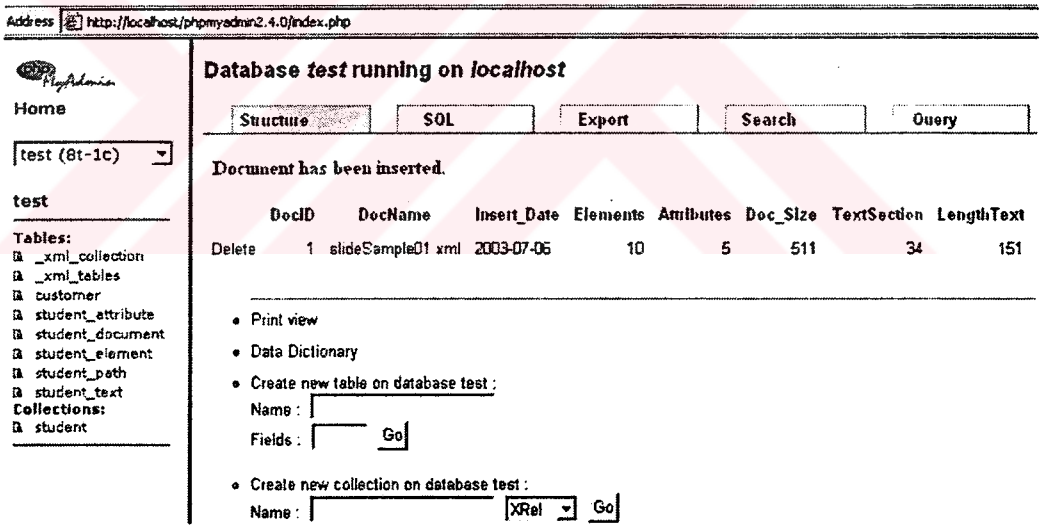


Figure 4.10 Browsing student collection

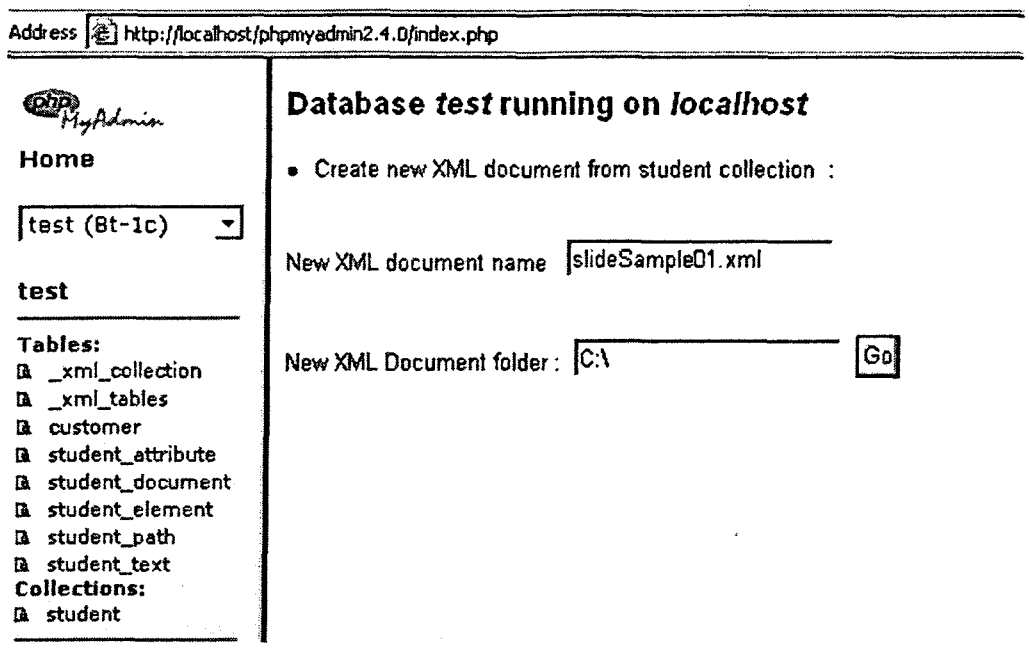


Figure 4.11 Downloading XML file page

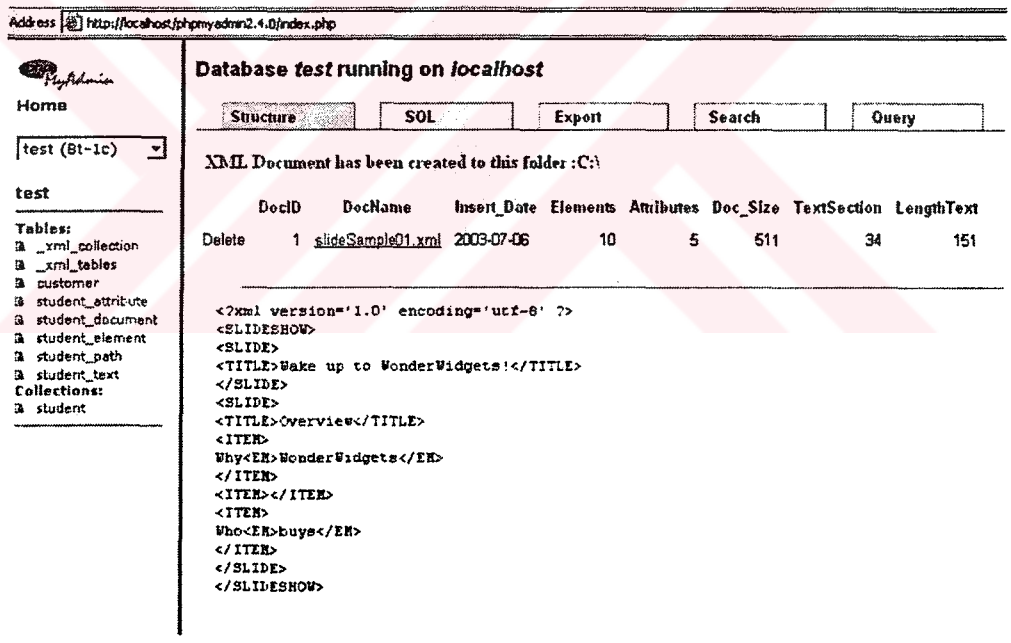


Figure 4.12 Browsing XML file

# CHAPTER 5

# EXPERIMENTAL RESULT

This chapter gives experimental results of storage and reconstruction time for XRel and Edge methods. All the experiments were conducted on a Pentium IV 350 MHz PC with 1 GB RAM 30GB hard disk. The RDBMS used was MYSQL 4.0.13. We conducted our experiments using two kinds of XML document which are *data-centric documents* and *text centric documents*.Data-centric documents are used for data transport.(Borruet,2002) Their structure are regular.Also order of sibling elements is generally not important. Text- centric documents structure less regular. Length of PCDATA nodes can be very long.The order in which sibling elements is always important.

In our study, we used Catalog.xml (XBench, 2003) as an example of data-centric documents and Shakespeare's plays (Bosak, 1999) and Dictionary.xml (XBench, 2003) documents as examples of text-centric documents. You can see DTD of test documents in Appendix. Table 5.1 summarizes the characteristics of the test documents.

Table 5.1 The characteristics of the Test documents

| | Shakes | Catalog | Dictionary |
|---|---|---|---|
| Document size | 9,75 MB | 10 MB | 9,83 MB |
| Depth | 7 | 8 | 8 |
| # of Unique Path | 45 | 67 | 27 |
| # of Element node | 231477 | 239527 | 270604 |
| # of Attribute node | 0 | 14999 | 730 |
| # of Text node | 191610 | 147376 | 231671 |
| Ratio text/document (MB) | 0,60 | 0,42 | 0,78 |
| Ratio element node/total nodes | 0,55 | 0,60 | 0,54 |

These documents are stored in MySQL database using two approaches. Table 5.3 shows database size when the documents were stored.

Table 5.2 Table sizes in MySQL

| *XRel tables* | **Catalog** | **Dictionary** | **Shakespeare** |
|---|---|---|---|
| **Element** | 17,9 MB | 17.9 MB | 16.7 MB |
| **Attribute** | 878.4 KB | 44.0 KB | 1,0 KB |
| **Text** | 13,0 MB | 21.9 MB | 15.9 MB |
| **Path** | 7.3 KB | 3.2 KB | 4.7 KB |
| *Total* | **31,7MB** | **39,7MB** | **32,6 MB** |

| *Edge table* | **Catalog** | **Dictionary** | **Shakespeare** |
|---|---|---|---|
| **Edge** | 21,1 MB | 23,7 MB | 22,2 MB |

## 5.1 Insertion Results

We have two kinds of dataset. Firstly, we represented insertion time of each document without comparing seperate others. In Figure 5.1 and 5.2 show us that Edge method finished insertion processing earlier than XRel. The reason could be that it has only one table for inserting data in Edge. The other reason might be XRel designs for Xpath queries, so that its table structure is much more complex than Edge. For example, XRel element table has additional columns as pathid, end_descandent_id. Furthermore, while XML document is being parsed, Each row in XRel tables could not be taken all columns value that described the node. For example, end_descandent_id column value of one node comes when parser reach its last desendant node.So that XRel method does more update processing than Edge while insertion operation is running.

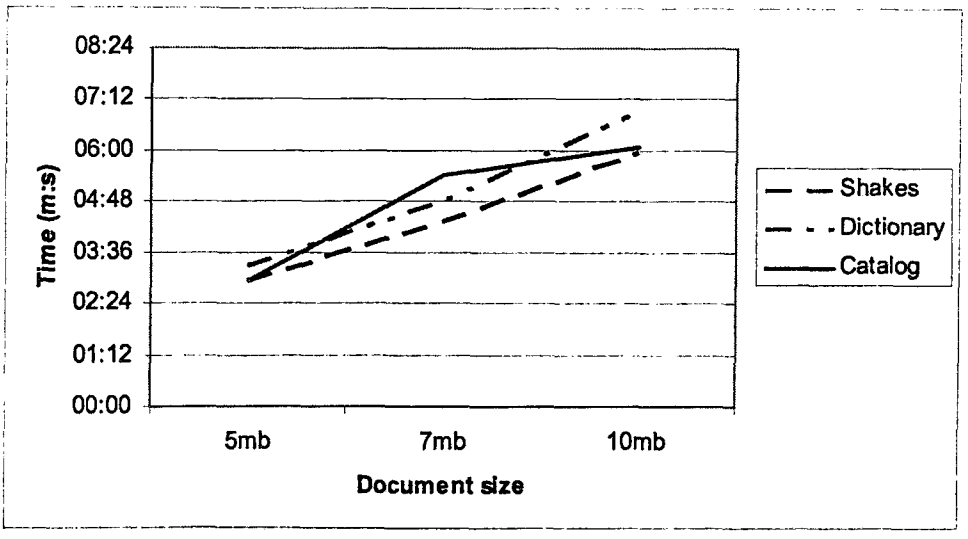

Figure 5.1 Insertion time of Documents using XREL

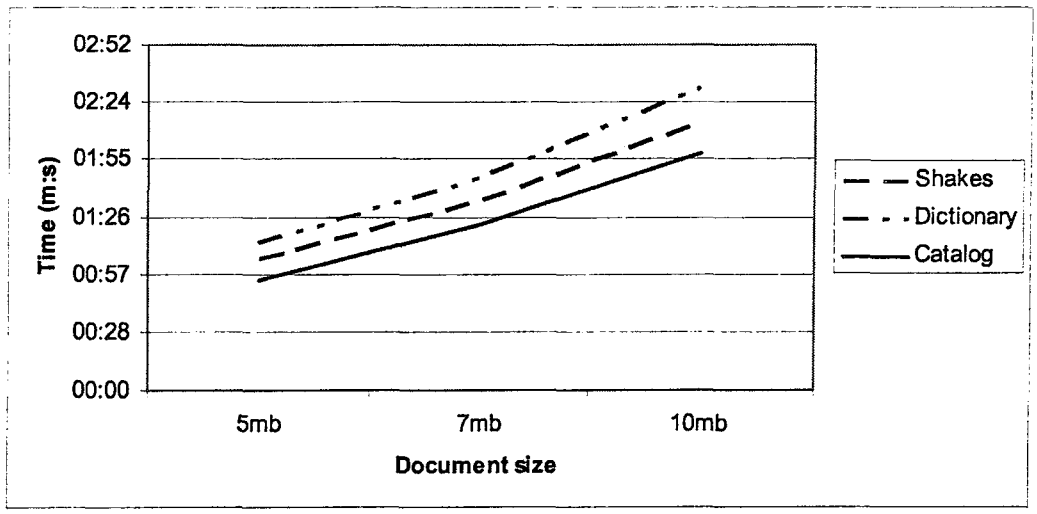


Figure 5.2 Insertion time of documents using Edge

In Figure 5.3 shows that the number of inner nodes in documents is an important criterion for insertion. We compared Shakespeare, Catalog and Dictionary documents in 10 MB size.As you can see in Table 5.1, Shakespeare document has the smallest number of element nodes and Dictionary has the highest number of element nodes. As a result if the documents size are equal, we can say that the insertion process of document which has less inner nodes finishes ealier than the other document.

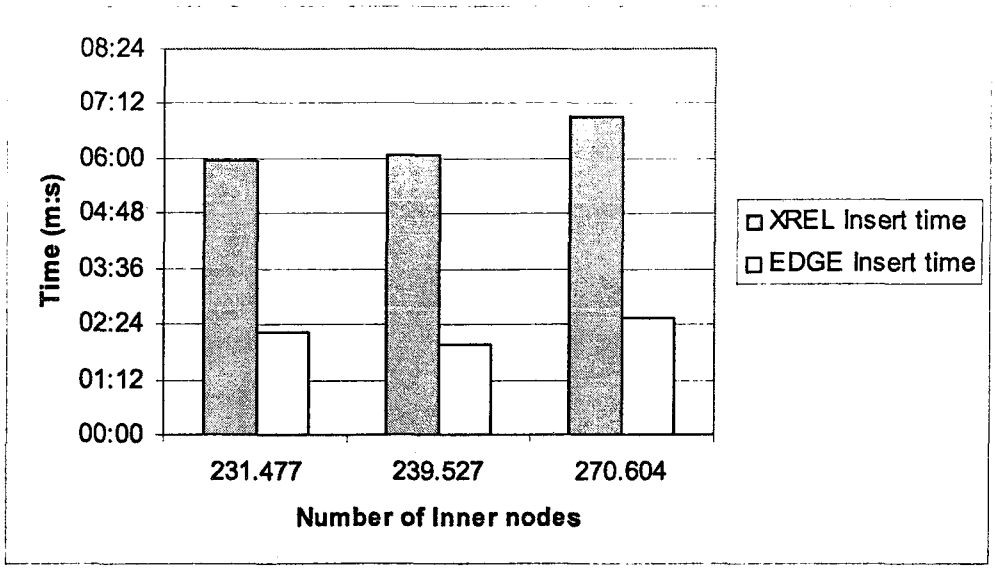


Figure 5.3 Insertion time according to inner node numbers

## 5.2 Reconstruction Results

As it is shown in Figure 5.4 and 5.5, the reconstruction time of two approaches is very close. But results are affected by the number of inner nodes as in the insertion processing. When we look at Figure 5.4 and 5.5 generally, reconstruction time is quite small than the average of insertion time. Because we did not use any kind of join statement in program. Also index on NodeID is useful for faster processing.

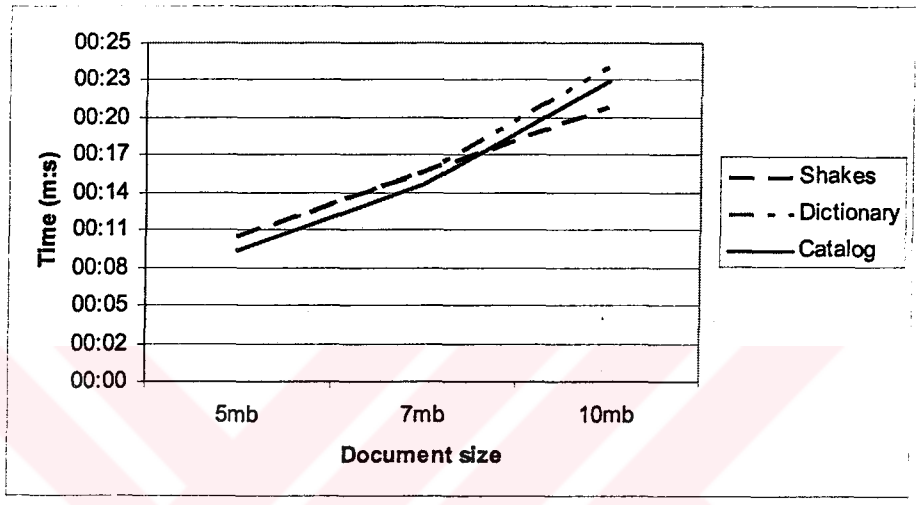


Figure 5.4 Reconstruction time of each document using XREL

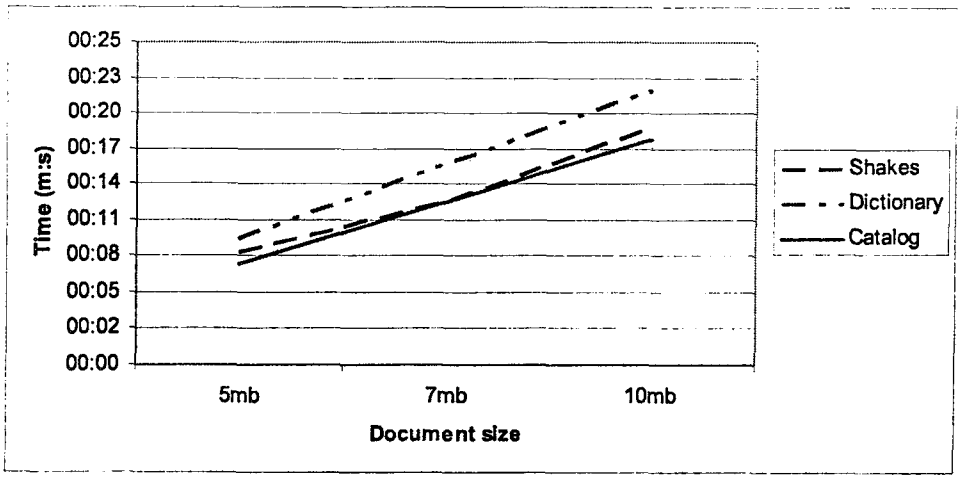


Figure 5.5 Reconstruction time of each document using Edge

# CHAPTER 6

# CONCLUSION

In this thesis, we studied an alternative implementation to store XML documents in a relational database. While storing XML documents in RDBMS, several approaches can be used. In our implementation we used fixed schema mapping approach. This approach stores well-formed XML documents into fixed schema database and does not require extending database models or SQL syntax in order to store XML documents. With this implementation from user point of view, MySql RDBMS now supports storage & retrieval of XML documents. This utility adds "collection" support to the MySQL database. A collection is a set of XML documents stored in fixed schema of tables. As a result, MySql database supports both tables and collections for data and document processing.

In this implementation, MySQL 4.0.13-nt as database and PhpMyAdmin 2.4.0 program as web interface are used. PhpMyAdmin are extended with PHP classes. These classes give user the following options;

- Create collection using one of the methods (XRel and Monet)
- Insert XML document into collection
- Delete XML document from collection
- Retrieve XML document from collection
- Browse collections
- Drop collections

Our implementation can help both the users who want to manage XML data with fixed schema methods and the researchers who study XML technologies.

This implementation is a first step for storing XML documents into RDBMS. As our future work, we can add XML query processing to this implementation. This query processing should retrieve XML document according to XPath queries. It can automatically translate XPath queries to standart SQL. Another future work can be PHP classes that we used in our implementation, support to others database systems. We can divide these classes into two parts. First part supports database connection the other part supports storing and retrieving XML documents.

# APPENDIX A : DTD OF CATALOG DOCUMENT

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT FAX_number (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT attributes (ISBN, number_of_pages, type_of_book, size_of_book)>
<!ELEMENT author (name, date_of_birth, biography, contact_information)>
<!ELEMENT authors (author+)>
<!ELEMENT biography (#PCDATA)>
<!ELEMENT catalog (item+)>
<!ATTLIST catalog
        xmlns:xsi CDATA #REQUIRED
        xsi:noNamespaceSchemaLocation CDATA #REQUIRED
>
<!ELEMENT contact_information (mailing_address, FAX_number?, phone_number, email_address?, web_site?)>
<!ELEMENT cost (#PCDATA)>
<!ATTLIST cost currency CDATA #REQUIRED>
<!ELEMENT country (name, exchange_rate, currency)>
<!ELEMENT currency (#PCDATA)>
<!ELEMENT data (#PCDATA)>
<!ELEMENT date_of_birth (#PCDATA)>
<!ELEMENT date_of_release (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT email_address (#PCDATA)>
<!ELEMENT web_site (#PCDATA)>
<!ELEMENT exchange_rate (#PCDATA)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT height (#PCDATA)>
<!ATTLIST height    unit CDATA #REQUIRED >
<!ELEMENT image (data)>
<!ELEMENT item (title, authors, date_of_release, publisher, subject, description, related_items, media, pricing, attributes)>
<!ATTLIST item  id ID #REQUIRED >
<!ELEMENT item_id (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT length (#PCDATA)>
<!ATTLIST length  unit CDATA #REQUIRED>
<!ELEMENT mailing_address (street_information, name_of_city, name_of_state, zip_code, name_of_country?, country?)>
<!ELEMENT media (thumbnail, image)>
<!ELEMENT middle_name (#PCDATA)>
<!ELEMENT name (#PCDATA | first_name | middle_name | last_name)*>
<!ELEMENT name_of_city (#PCDATA)>
<!ELEMENT name_of_country (#PCDATA)>
<!ELEMENT name_of_state (#PCDATA)>
<!ELEMENT number_of_pages (#PCDATA)>
<!ELEMENT phone_number (#PCDATA)>
<!ELEMENT pricing (suggested_retail_price, cost, when_is_available, quantity_in_stock)>
<!ELEMENT publisher (name, contact_information)>
<!ELEMENT quantity_in_stock (#PCDATA)>
<!ELEMENT related_item (item_id)>
<!ELEMENT related_items (related_item+)>
<!ELEMENT size_of_book (length, width, height)>
<!ELEMENT street_address (#PCDATA)>
<!ELEMENT street_information (street_address+)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT suggested_retail_price (#PCDATA)>
<!ATTLIST suggested_retail_price currency CDATA #REQUIRED >
<!ELEMENT thumbnail (data)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT type_of_book (#PCDATA)>
<!ELEMENT when_is_available (#PCDATA)>
<!ELEMENT width (#PCDATA)>
<!ATTLIST width unit CDATA #REQUIRED >
<!ELEMENT zip_code (#PCDATA)>
```

# APPENDIX B : THE DTD OF DICTIONARY DOCUMENT

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT a (#PCDATA)>
<!ELEMENT bib (#PCDATA)>
<!ELEMENT cr (#PCDATA)>
<!ELEMENT def (#PCDATA | cr)*>
<!ELEMENT dictionary (e+)>
<!ELEMENT e (hwg, vfl?, et?, ss)>
<!ATTLIST e          id ID #IMPLIED >
<!ELEMENT et (cr)+>
<!ELEMENT hw (#PCDATA)>
<!ELEMENT hwg (hw | pr? | pos?)+>
<!ELEMENT loc (#PCDATA)>
<!ELEMENT pos (#PCDATA)>
<!ELEMENT pr (#PCDATA)>
<!ELEMENT q (qd, a?, w, bib?, loc, qt)>
<!ELEMENT qd (#PCDATA)>
<!ELEMENT qp (q+)>
<!ELEMENT qt (#PCDATA | cr | i | b)*>
<!ELEMENT s (def, qp)>
<!ELEMENT ss (s+)>
<!ELEMENT vd (#PCDATA)>
<!ELEMENT vf (#PCDATA)>
<!ELEMENT vfl (vd* | vf+)+>
<!ELEMENT w (#PCDATA)>
<!ELEMENT i (#PCDATA)>
<!ELEMENT b (#PCDATA)>
```

# APPENDIX C : THE DTD OF SHASKPEARE DOCUMENT

```
<!ENTITY amp "&#38;#38;">
<!ELEMENT PLAY     (TITLE, FM, PERSONAE, SCNDESCR, PLAYSUBT, INDUCT?, PROLOGUE?, ACT+, EPILOGUE?)>
<!ELEMENT TITLE    (#PCDATA)>
<!ELEMENT FM       (P+)>
<!ELEMENT P        (#PCDATA)>
<!ELEMENT PERSONAE (TITLE, (PERSONA | PGROUP)+)>
<!ELEMENT PGROUP   (PERSONA+, GRPDESCR)>
<!ELEMENT PERSONA  (#PCDATA)>
<!ELEMENT GRPDESCR (#PCDATA)>
<!ELEMENT SCNDESCR (#PCDATA)>
<!ELEMENT PLAYSUBT (#PCDATA)>
<!ELEMENT INDUCT   (TITLE, SUBTITLE*, (SCENE+|(SPEECH|STAGEDIR|SUBHEAD)+))>
<!ELEMENT ACT      (TITLE, SUBTITLE*, PROLOGUE?, SCENE+, EPILOGUE?)>
<!ELEMENT SCENE    (TITLE, SUBTITLE*, (SPEECH | STAGEDIR | SUBHEAD)+)>
<!ELEMENT PROLOGUE (TITLE, SUBTITLE*, (STAGEDIR | SPEECH)+)>
<!ELEMENT EPILOGUE (TITLE, SUBTITLE*, (STAGEDIR | SPEECH)+)>
<!ELEMENT SPEECH   (SPEAKER+, (LINE | STAGEDIR | SUBHEAD)+)>
<!ELEMENT SPEAKER  (#PCDATA)>
<!ELEMENT LINE     (#PCDATA | STAGEDIR)*>
<!ELEMENT STAGEDIR (#PCDATA)>
<!ELEMENT SUBTITLE (#PCDATA)>
<!ELEMENT SUBHEAD  (#PCDATA)>
```

# APPENDIX D : PHP CLASSES CODE

## XMLCollection.inc

```
<?php
require("XRelParse.inc");
require("EdgeParse.inc");
require("XRel.inc");
require("Edge.inc");
require("EdgeXMLFile.inc");
require("XRelXMLFile1.inc");

class XMLCollection
{
        var $db;    // database name
        var $name;  // collection name
        var $type;  // collection type (Edge,XRel)
        var $owner; // collection owner
        var $datetime;  // collection creation date and time
        var $tables;// tables of collection

        function XMLCollection(){

                $argCount=func_num_args();
                $funcname = 'XMLCollection'.$argCount;
                $arg_list =func_get_args();
                if ($argCount==1){

                        $arg1=func_get_arg(0);
                        $this->$funcname($arg1);
                }
                else if ($argCount==2){
                        $arg1=func_get_arg(0);
                        $arg2=func_get_arg(1);
                        $this->$funcname($arg1,$arg2);
                }
                else if ($argCount==5) {

                        $arg1=func_get_arg(0);
                        $arg2=func_get_arg(1);
                        $arg3=func_get_arg(2);
                        $arg4=func_get_arg(3);
                        $arg5=func_get_arg(4);
                        $this->$funcname($arg1,$arg2,$arg3,$arg4,$arg5);
                }


        }


        function XMLCollection1($db){

                $this->db=$db;
                mysql_select_db($db);           }

        function XMLCollection2($db,$name){

                mysql_select_db($db);
                $sql="SELECT type,create_date,owner FROM _xml_collection WHERE Name='".$name."'";
                $result=mysql_query($sql);
                $myrow=mysql_fetch_array($result);

                $this->db=$db;
                $this->name=$name;
                $this->type=$myrow[0];
                $this->datetime=$myrow[1];
                $this->owner=$myrow[2];

                if ($this->type=="XRel")
                        $this->tables=new XRel($this->name);
                else if($this->type=="Edge")
                        $this->tables=new Edge($this->name);
```

```
        else if($this->type=="Monet")
                $this->tables=new Monet($this->name);


}

function XMLCollection5($db,$name,$type,$date,$owner){


        $this->db=$db;
        $this->name=$name;
        $this->type=$type;
        $this->datetime=$date;
        $this->owner=$owner;

        // create xml_collection and xml_tables tables for storing collection and its' tables

        mysql_select_db($this->db);

        $sql = 'CREATE TABLE IF NOT EXISTS _xml_Collection ( Name VARCHAR( 50 ) NOT NULL , Type
        VARCHAR( 20 ) NOT NULL , Create_Date VARCHAR( 30 ) NOT NULL ,Owner VARCHAR(30), Numb
        BIGINT NOT NULL
        , PRIMARY KEY ( Name ) ) ';

        $result1    = mysql_query($sql) or PMA_mysqlDie('', '', '', $err_url);

        $sql = 'CREATE TABLE IF NOT EXISTS _xml_Tables ( Noo BIGINT NOT NULL AUTO_INCREMENT,
        Table_Name VARCHAR( 50 ) NOT NULL , Col_No BIGINT NOT NULL , PRIMARY KEY ( Noo ) ) ';

        $result1    = mysql_query($sql) or PMA_mysqlDie('', '', '', $err_url);

        $sql = 'SELECT max( numb )AS buyuk FROM _xml_collection';
        $result = mysql_query($sql)or PMA_mysqlDie('', '', '', $err_url);
        $myrow=mysql_fetch_array($result);

        $collectionID=$myrow[0];
        $collectionID++;

        $sql="INSERT INTO _xml_Collection ( Name . Type , Create_Date ,Owner, Numb ) VALUES
        ('".$this->name."','".$this->type."','".$this->datetime."','".$this->owner.
        "','".$collectionID." )";

        $result1    = mysql_query($sql) or PMA_mysqlDie('', '', '', $err_url);

        if ($this->type=="XRel")
                $this->tables=new XRel($this->name.$collectionID);
        else if($this->type=="Edge")
                $this->tables=new Edge($this->name.$collectionID);
        else if($this->type=="Monet")
                $this->tables=new Monet($this->name.$collectionID);


}

function insert_xmlfile($xmlfile,$col_obj){

        if ($this->type=="XRel")
                $objParse=new XRelParse($xmlfile,$col_obj);
        else if($this->type=="Edge")
                $objParse=new EdgeParse($xmlfile,$col_obj);


}

function delete_xmlfile($xmlfile){

$sql="select DocID from ".$this->tables->DocumentT." where docname='$xmlfile'";
$result1=mysql_query($sql);
if ($myrow=mysql_fetch_array($result1)){

        $docID=$myrow[0];
        $sql="select table_name from _xml_collection, _xml_tables  where _xml_collection.name='".$this->name."'
        and _xml_collection.numb=_xml_tables.col_no ";

        //$sql="select table_name from _xml_tables where col_no=$docID";
```

```
                $result1=mysql_query($sql);

                while($myrow1=mysql_fetch_array($result1))
                {

                  $sqldelete="DELETE FROM ".$myrow1[0]." WHERE    DocID=$docID";
                 $result=mysql_query($sqldelete);

                }

        }

        }


        function browse_collection(){

          $common_url_query = PMA_generate_common_url($this->db);
          $table_row='';
          $sql="select * from ".$this->tables->DocumentT;
          $result=mysql_query($sql);
          if ($result) {

                    $table_header="<table border='0' cellpadding='5'><tr><td colspan='1'></td>";
        while($field = mysql_fetch_field($result))
                          {
                           $table_header.="<td  bgcolor='#DDDDDD' align='center'>
                           <font color='#0000FF'><b>$field->name</b></font></td>";
                          }

           $table_header.="</tr>" ;

                    while($myrow = mysql_fetch_array($result))
                   {
                   $delete_link='<a  href="db_details_structure.php?'. $common_url_query .'&coll=7 &collname='.$this-
                  >name.'&file='.$myrow[1].'">Delete</a>' ;
                  $browse_link='<a  href="db_details_structure.php?'. $common_url_query .'&coll=9 &collname='.$this-
                  >name.'&file='.$myrow[1].'">'.$myrow[1].'</a>' ;
                  $common_url_query .'&collname='.$this->name.'&file='.$myrow[1].'">'.$myrow[1].'</a>' ;


                   $table_row.="<tr>
                   <td valign='top' bgcolor='#DDDDDD'><font color='#0000FF'>
                   $delete_link
                   </td>
                   <td align='right' valign='top' bgcolor='#DDDDDD'>$myrow[0]</td>
                   <td valign='top' bgcolor='#DDDDDD'>
                   $browse_link
                   </td>
                   <td valign='top' bgcolor='#DDDDDD'>$myrow[2]</td>
                   <td align='right' valign='top' bgcolor='#DDDDDD'>$myrow[3]</td>
                   <td align='right' valign='top' bgcolor='#DDDDDD'>$myrow[4]</td>
                   <td align='right' valign='top' bgcolor='#DDDDDD'>$myrow[5]</td>
                   <td align='right' valign='top' bgcolor='#DDDDDD'>$myrow[6]</td>
                   <td align='right' valign='top' bgcolor='#DDDDDD'>$myrow[7]</td>
                   </tr>";

      }

           $table_display=$table_header.$table_row."</table>" ;
        }

         return $table_display;


        }

        function browse_xmlfile($xmlfile,$col_obj){

                  if ($this->type=="XRel")
                            $objXMLFile=new XRelXMLFile($xmlfile,$col_obj);
                  else if($this->type=="Edge")
                            $objXMLFile=new EdgeXMLFile($xmlfile,$col_obj);


}
```

```
function drop_collection()      {

        $sql="select Name,Numb from _xml_collection where name='$this->name'";
        $result1=mysql_query($sql);
        $sqldrop="DROP TABLE ";
        $flag=true;
        if ($myrow=mysql_fetch_array($result1)){

        $sql="select table_name from _xml_tables where col_no=$myrow[1]";
        $result=mysql_query($sql);
        while($myrow1=mysql_fetch_array($result)){
        if ($flag){
          $sqldrop=$sqldrop.$myrow1[0];
          $flag=false;
         }
        else
          $sqldrop=$sqldrop.", ".$myrow1[0];
        }

        $result=mysql_query($sqldrop);

        $sqldelete="DELETE FROM _xml_tables WHERE col_no=$myrow[1]";
        $result=mysql_query($sqldelete);
        $sqldelete="DELETE FROM _xml_collection WHERE name='$this->name'";
        $result=mysql_query($sqldelete);
        }

}
function browse_collections(){

  $table_display='';
  $table_row='';
  $sql='SELECT * FROM _xml_collection';

  $result=mysql_query($sql);

  if ($result) {

        $table_header="<table border='0'>
        <tr>
        <td bgcolor='#F5F5F5'></td>
        <th> Collection </th>
        <th colspan='4'>Action</th>
        <th>Documents</th>
        <th>Type</th>
        <th>Create Date</th>
        </tr>";


        $common_url_query = PMA_generate_common_url($this->db);

        while($myrow = mysql_fetch_array($result))
                {
                $sql_document="select count(*) from ".$myrow[0]."_document";
                $result_document=mysql_query($sql_document);
                if ($myrow_document=mysql_fetch_array($result_document))

                $document_count=$myrow_document[0];
                $browse_link='<a  href="db_details_structure.php?'. $common_url_query .'&coll=4
                &collname='.$myrow[0].'">Browse</a>' ;
                $insert_link='<a  href="collection_insert.php?'. $common_url_query
                .'&collname='.$myrow[0].'">Insert</a>' ;
                $drop_link='<a  href="db_details_structure.php?'. $common_url_query .'&coll=5
                &collname='.$myrow[0].'">Drop</a>' ;
                $delete_link='<a  href="db_details_structure.php?'. $common_url_query .'&coll=6
                &collname='.$myrow[0].'">Empty</a>' ;

                $table_row.="<tr>
                <td align='center' bgcolor='#DDDDDD'>
                <input type='checkbox' name='selected_tbl[]' value='$myrow[0]' id='checkbox_tbl_1' />
                </td>
                <td bgcolor='#DDDDDD' nowrap='nowrap'>
                 <b><label for='checkbox_tbl_1'>$myrow[0]</label> </b> 
                </td>
```

```
                                <td bgcolor='#DDDDDD'>
                                $browse_link
                                </td>
                                <td bgcolor='#DDDDDD'>
                                $insert_link
                                </td>
                                <td bgcolor='#DDDDDD'>
                                $drop_link
                                </td>
                                <td bgcolor='#DDDDDD'>
                                $delete_link
                                </td>
                                <td align='right' bgcolor='#DDDDDD'><a
                                href='app.php?browse=true'>$document_count</a></td>
                                <td bgcolor='#DDDDDD' nowrap='nowrap'> $myrow[1] </td>
                                <td bgcolor='#DDDDDD' nowrap='nowrap'> $myrow[2] </td>
                                </tr>";
        }

                $table_display=$table_header.$table_row."</table>" ;
                        }

                return $table_display;


                }

 }

?>
```

## XRel.inc

```
<?php
class XRel
{
        var $name;
        var $ElementT;
        var $TextT;
        var $AttributeT;
        var $PathT;
        var $DocumentT;

        function XRel(){
        $argCount=func_num_args();
        $funcname = 'XRel'.$argCount;
        $arg_list =func_get_args();
        if ($argCount==1){

        $arg1=func_get_arg(0);
        $this->$funcname($arg1);
        }
        else if ($argCount==2){
        $arg1=func_get_arg(0);
        $arg2=func_get_arg(1);
        $this->$funcname($arg1,$arg2);
        }
        }

        function XRel1($name){

                $this->name=$name;
                $this->ElementT=$name.'_element';
                $this->TextT=$name.'_text';
                $this->AttributeT=$name.'_attribute';
                $this->PathT=$name.'_path';
                $this->DocumentT=$name.'_document';
        }

        function XRel2($name,$collectionID){

                $this->name=$name;
                $this->ElementT=$name.'_element';
                $this->TextT=$name.'_text';
```

```
$this->AttributeT=$name.'_attribute';
$this->PathT=$name.'_path';
$this->DocumentT=$name.'_document';

// create xml collection tables
$sql_document ='CREATE TABLE IF NOT EXISTS '.$this->DocumentT.'( DocID INT NOT NULL ,
DocName VARCHAR( 100 )
NOT NULL ,Insert_Date VARCHAR( 10 ) NOT NULL, Elements BIGINT , Attributes BIGINT, Doc_Size
BIGINT, TextSection BIGINT, LengthText BIGINT, PRIMARY KEY ( DocID ) ) ';

$sql_element = 'CREATE TABLE IF NOT EXISTS '.$this->ElementT.'( DocID INT NOT NULL , NodeID
BIGINT NOT NULL
 , EndDesID BIGINT NOT NULL , ParentID BIGINT NOT NULL , PathID INT NOT NULL , elname
 VARCHAR(50), PRIMARY KEY ( DocID , NodeID ) ) ';

$sql_attribute = 'CREATE TABLE IF NOT EXISTS '.$this->AttributeT.'( DocID INT NOT NULL , NodeID
BIGINT NOT
NULL , ParentID BIGINT NOT NULL, PathID INT NOT NULL , Value VARCHAR( 70 ),Name
VARCHAR( 70 ) , PRIMARY KEY ( DocID , NodeID ) ) ';

$sql_text= 'CREATE TABLE IF NOT EXISTS '.$this->TextT.' ( DocID INT NOT NULL , NodeID BIGINT
NOT NULL
,ParentID BIGINT NOT NULL, PathID INT NOT NULL , Value VARCHAR( 100 ) , PRIMARY KEY (
DocID ,
 NodeID )) ';

$sql_path='CREATE TABLE IF NOT EXISTS '.$this->PathT.'( DocID INT NOT NULL ,PathID INT NOT
NULL ,
PathExp VARCHAR( 255 ) NOT NULL , PRIMARY KEY ( DocID ,PathID ) ) ';

// Executes the query
$result1   = mysql_query($sql_document) or PMA_mysqlDie('', '', '', $err_url);
$result1   = mysql_query($sql_element) or PMA_mysqlDie('', '', '', $err_url);
$result1   = mysql_query($sql_attribute) or PMA_mysqlDie('', '', '', $err_url);
$result1   = mysql_query($sql_text) or PMA_mysqlDie('', '', '', $err_url);
$result1   = mysql_query($sql_path) or PMA_mysqlDie('', '', '', $err_url);

/*$sql="SELECT numb FROM _xml_collection WHERE name='".$name."'";
$result=mysql_query($sql);
$myrow=mysql_fetch_array($result); */

$tmpID=$collectionID;

$sql="INSERT INTO _xml_Tables ( Table_Name , Col_No ) VALUES ('" . $this->DocumentT .
 "',".$tmpID. ")";
$result1   = mysql_query($sql);

$sql="INSERT INTO _xml_Tables ( Table_Name , Col_No ) VALUES ( '". $this->ElementT
 ."',".$tmpID . ")";
$result1   = mysql_query($sql) or PMA_mysqlDie('', '', '', $err_url):

$sql="INSERT INTO _xml_Tables ( Table_Name , Col_No ) VALUES ( '". $this->AttributeT .
"',".$tmpID . ")";
$result1   = mysql_query($sql)or PMA_mysqlDie('', '', '', $err_url) ;

$sql="INSERT INTO _xml_Tables ( Table_Name , Col_No ) VALUES ('". $this->TextT .
"',".$tmpID . ")";
$result1   = mysql_query($sql) or PMA_mysqlDie('', '', '', $err_url);

$sql="INSERT INTO _xml_Tables ( Table_Name , Col_No ) VALUES ('". $this->PathT .
 "',".$tmpID . ")";
$result1   = mysql_query($sql) or PMA_mysqlDie('', '', '', $err_url);


    }

}

?>
```

## Edge.inc

```
<?php
```

```
class Edge
{
        var $EdgeT;
        var $DocumentT;

        function Edge(){
          $argCount=func_num_args();
          $funcname = "Edge".$argCount;
          $argArr =func_get_args();

        if ($argCount==1){

          $arg1=func_get_arg(0);
          $this->$funcname($arg1);
        }
        else if ($argCount==2){
          $arg1=func_get_arg(0);
          $arg2=func_get_arg(1);
          $this->$funcname($arg1,$arg2);
        }

        }
        function Edge1($name){

                $this->EdgeT=$name.'_edge';
                $this->DocumentT=$name.'_document';
                }

        function Edge2($name,$collectionID){

                $this->EdgeT=$name.'_edge';
                $this->DocumentT=$name.'_document';

                // create xml collection tables
                $sql_document ='CREATE TABLE '.$this->DocumentT.'( DocID INT NOT NULL , DocName VARCHAR(
                100 )
                NOT NULL ,Insert_Date VARCHAR(30) NOT NULL, Elements BIGINT , Attributes BIGINT, Doc_Size
                BIGINT, TextSection BIGINT, LengthText BIGINT, PRIMARY KEY ( DocID ) ) ';

                $sql_edge = 'CREATE TABLE '.$this->EdgeT.'( DocID INT NOT NULL , NodeID BIGINT NOT NULL
                 , ParentID BIGINT NOT NULL , Ordinal INT NOT NULL , Name VARCHAR(50),Flg
                 VARCHAR(3)NOT NULL,
                 TextVal VARCHAR(100)NOT NULL, PRIMARY KEY ( DocID , NodeID ) ) ';


                $result1   = mysql_query($sql_document) or PMA_mysqlDie('', '', '', $err_url);
                $result1   = mysql_query($sql_edge) or PMA_mysqlDie('', '', '', $err_url);

                $sql="SELECT numb FROM _xml_collection WHERE name='".$name."'";
                $result=mysql_query($sql);
                $myrow=mysql_fetch_array($result);

                $tmpID=$collectionID;

                $sql="INSERT INTO _xml_Tables ( Table_Name , Col_No ) VALUES ('" . $this->DocumentT .
                "','".$tmpID. ")";
                $result1   = mysql_query($sql);

                $sql="INSERT INTO _xml_Tables ( Table_Name , Col_No ) VALUES ('" . $this->EdgeT .
                "','".$tmpID. ")";
                $result1   = mysql_query($sql);


        }

}

?>
```

## XrelParse.inc

```
<?php
```

```
class XRelParse {

        var $tmpdocID=0;
        var $file;
        var $col_obj;
        var $elements=0;
        var $parentid=0;
        var $pathid=0;
        var $sayac=-1;
        var $pathint=array();
        var $pathexp=" ";
        var $texts=0;
        var $text_size=0;
        var $attributes=0;
        var $size=0;

function XRelParse($myfile,$col_obj)
        {
        $start=date('d/m/y-H:i:s');
        echo $start;
         $this->col_obj=$col_obj;
         $table_doc=$this->col_obj->tables->DocumentT;


          // 1.Create xml_parser and open $file
         $this->parser = xml_parser_create();
         xml_set_object($this->parser,$this);
         xml_set_element_handler($this->parser, "startElement", "endElement");
         xml_parser_set_option($this->parser, XML_OPTION_CASE_FOLDING, true);
         xml_set_character_data_handler($this->parser,"characterData");

        if (!($fp = fopen($myfile, "r")))
        {
        die("could not open XML input");
        }
         // 2. If opened test with document table record

         $query = 'SELECT count(*) FROM '.$table_doc.' WHERE docname ="'.$myfile.'"';

         $result = mysql_query($query) or Die("Hataaaa");
         $myrow = mysql_fetch_array($result);
         $exist = $myrow[0];

        if($exist)
        {
                die("The collection $table_doc already has a document with name=$myfile");
        }
        else
        {
                $sql = 'SELECT max( docID )AS buyuk FROM '.$table_doc;
                $result = mysql_query($sql)or Die("Error" );

                $myrow=mysql_fetch_array($result);
                $this->tmpdocID=$myrow[0];
                $this->tmpdocID++;
                        }

        while ($data = chop(fread($fp, 10240)))
                {
                  $this->size+=strlen($data);

                if (!xml_parse($this->parser, $data, feof($fp)))
                        {
                          die(sprintf("XML error: %s at line %d",
                       xml_error_string(xml_get_error_code($this->parser)),
                    xml_get_current_line_number($this->parser)));
                    }//end if
        }//end while
        $datetime=date('d/m/y-H:m:s');
        $sql="insert into ".$table_doc."
        (docID,docname,insert_date,Elements,Attributes , Doc_Size, TextSection , LengthText )
         values(".$this->tmpdocID.",'".$myfile."','".$datetime."',".$this->elements.",".
         $this->attributes.",".$this->size.",".$this->texts
 .",".$this->text_size.")";
```

```
        $result = mysql_query($sql) or Die("Error");

        //$sql="delete from ".$collection." where value=''"
        //$result=mysql_query($sql) or Die("Error");

        xml_parser_free($this->parser);
        $end=date('d/m/y-H:i:s');
                echo " finish time:".$end;
}

function startElement($parser, $name, $attrs) {

 $table_element=$this->col_obj->tables->ElementT;
 $table_attribute=$this->col_obj->tables->AttributeT;
 $table_path=$this->col_obj->tables->PathT;

 $this->attributes+=count($attrs);
 $elname=$name;
 $this->elements++;

        if ($this->pathexp==" ")
         $this->pathexp="/".$elname;
 else
        $this->pathexp=$this->pathexp."/".$elname;


        $this->pathint[++$this->sayac]=$this->elements;

        $sql="select pathid from ".$table_path." where pathexp='".$this->pathexp."' and
            DocID=$this->tmpdocID";

        $result=mysql_query($sql);

 if(!$myrow=mysql_fetch_array($result))
        {
        $sql="SELECT MAX(pathid) FROM ".$table_path." WHERE DocID=$this->tmpdocID";
        $result=mysql_query($sql);

        if(!$myrow1=mysql_fetch_array($result))
                $this->pathid=1;
        else
                $this->pathid=$myrow1[0]+1;

                $sql="insert into ".$table_path." values($this->tmpdocID,$this->pathid
                        ,'$this->pathexp' )";

                $result=mysql_query($sql);
        }
        else
                $this->pathid=$myrow[0];


          // adding record to element table

                $sql="insert into ".$table_element." (docID,nodeID,parentid,pathid,elname)
                 values($this->tmpdocID,$this->elements,$this->parentid,$this->pathid,'$elname')";


                $result=mysql_query($sql);

                $this->parentid=$this->pathint[$this->sayac];
        // attribute insertion
        while (list ($key,$val)=each ($attrs))          {

                $this->elements++;
                $att_path=$this->pathexp."/@".$key;
                $sql="select pathid from ".$table_path." where pathexp='".$att_path."' And    DocID=$this->tmpdocID";

                $result=mysql_query($sql);

        if(!$myrow=mysql_fetch_array($result))
                {
                $sql="SELECT MAX(pathid) FROM ".$table_path." WHERE DocID=$this->tmpdocID";
                $result=mysql_query($sql);

                if(!$myrow1=mysql_fetch_array($result))
```

```
                    $this->pathid=1;
                else
                    $this->pathid=$myrow1[0]+1;

                    $sql="insert into ".$table_path." values($this->tmpdocID,$this->pathid  ,'$att_path' )";

                    $result=mysql_query($sql);
                }
                else
                    $this->pathid=$myrow[0];


                $sql="insert into ".$table_attribute." (docID,nodeID,parentid,pathid.value,name)
                 values($this->tmpdocID,$this->elements,$this->parentid,$this->pathid,'$val','$key')";


                $result=mysql_query($sql);


        }

}

        function endElement($parser, $name)
        {
         $table_element=$this->col_obj->tables->ElementT;
         $table_path=$this->col_obj->tables->PathT;

          $isaret=strrpos($this->pathexp,"/");
          $this->pathexp=substr($this->pathexp,0,$isaret);

         $sql="select pathid from ".$table_path." where pathexp='".$this->pathexp."' and
         DocID=$this->tmpdocID";
         $result=mysql_query($sql);
         if($myrow1=mysql_fetch_array($result))
             $this->pathid=$myrow1[0];

           $sql="update ".$table_element." set EndDesID=".$this->elements." where
            nodeID=".$this->pathint[$this->sayac]." and docID=$this->tmpdocID";


         $result=mysql_query($sql);
            $this->sayac-=1;

         if ($this->sayac >= 0)
        $this->parentid=$this->pathint[$this->sayac];
         $flagStart=false;
}

  function characterData($parser, $data)
  {
        $table_text=$this->col_obj->tables->TextT;
        $text=trim($data);
        if (strlen($text)>0){

                $this->texts++;
        $this->text_size+=strlen($data);
                $this->elements++;

                $sql="insert into ".$table_text." (docID,nodeID,parentid,pathid,value)
                        values($this->tmpdocID,$this->elements,$this->parentid,$this->pathid,'$data')";
                $result=mysql_query($sql);
        }


}

}

?>
```

## EdgeParse.inc

```
<?php
class EdgeParse {

        var $tmpdocID=0;
        var $file;
        var $col_obj;
        var $elements=0;
        var $parentid=0;
        var $sayac=-1;
        var $pathint=array();
        var $texts=0;
        var $text_size=0;
        var $attributes=0;
        var $size=0;

function EdgeParse($myfile,$col_obj)
        {
        $start=date('H:i:s');
        echo "start time:".$start;

        $this->col_obj=$col_obj;
        $table_doc=$this->col_obj->tables->DocumentT;

         // 1.Create xml_parser and open $file
        $this->parser = xml_parser_create();
        xml_set_object($this->parser,$this);
        xml_set_element_handler($this->parser, "startElement", "endElement");
        xml_parser_set_option($this->parser, XML_OPTION_CASE_FOLDING, true);
        xml_set_character_data_handler($this->parser,"characterData");

                if (!($fp = fopen($myfile, "r")))
                {
                die("could not open XML input");
                }
        // 2. If opened test with document table record

        $query = 'SELECT count(*) FROM '.$table_doc.' WHERE docname ="'.$myfile.'"';

        $result = mysql_query($query) or Die("Hataaaa");
        $myrow = mysql_fetch_array($result);
        $exist = $myrow[0];

                if($exist)
                {
                        die("The collection $table_doc already has a document with name=$myfile");
                }
                else
                {
                        $sql = 'SELECT max( docID )AS buyuk FROM '.$table_doc;
                        $result = mysql_query($sql)or Die("Error" );

                        $myrow=mysql_fetch_array($result);

                        $this->tmpdocID=$myrow[0];
                        $this->tmpdocID++;
                        echo "docID:".$this->tmpdocID."<br>";
                }

                while ($data = chop(fread($fp, 10240)))
                {
                        $this->size+=strlen($data);

                if (!xml_parse($this->parser, $data, feof($fp)))
                        {
                        die(sprintf("XML error: %s at line %d",
                      xml_error_string(xml_get_error_code($this->parser)),
                      xml_get_current_line_number($this->parser)));
                }//end if
        }//end while

        $datetime=date('d/m/y-H:m:s');
         $sql="insert into ".$table_doc."
         (docID,docname,insert_date,Elements,Attributes , Doc_Size, TextSection , LengthText ) values(".$this-
        >tmpdocID.",'".$myfile."','".$datetime."',".$this->elements."
```

```
            $this->attributes.",".$this->size.",".$this->texts
            ."," .$this->text_size.")";

            $result = mysql_query($sql) or Die("Error");

            xml_parser_free($this->parser);
            $finish=date('H:i:s');
            echo "finish time:".$finish;
}

function startElement($parser, $name, $attrs) {

 $table_element=$this->col_obj->tables->EdgeT;

 $this->attributes+=count($attrs);
 $elname=$name;
 $this->elements++;


        $this->pathint[++$this->sayac]=$this->elements;


          // adding record to element table

       $sql="insert into ".$table_element." (docID,nodeID,parentid,name,flg)
      values($this->tmpdocID,$this->elements,$this->parentid,'$elname','ref')";


        $result=mysql_query($sql);

        $this->parentid=$this->pathint[$this->sayac];
        // attribute insertion
        while (list ($key,$val)=each ($attrs))          {

                $this->elements++;
                $sql="insert into ".$table_element." (docID,nodeID,parentid,name,flg,textval)
                 values($this->tmpdocID,$this->elements,$this->parentid,'$key','att','$val')";

                $result=mysql_query($sql);


        }
}

        function endElement($parser, $name)
        {
         $this->sayac-=1;

        if ($this->sayac >= 0)
           $this->parentid=$this->pathint[$this->sayac];

}

 function characterData($parser, $data)
 {
        $table_element=$this->col_obj->tables->EdgeT;
        $text=trim($data);
        $text=ereg_replace("","  ",$text);

                if (strlen($text)>0){

                $this->texts++;
                $this->text_size+=strlen($data);
                $deger="val";

                $sql="update ".$table_element." set Flg='$deger' , TextVal='$text' where  nodeID=".$this->pathint[$this-
                >sayac]. " and docID=$this->tmpdocID";

                $result=mysql_query($sql);
        }


}
```

```
}

?>
```

## XRelXMLFile.inc

```
<?php
class XRelXMLFile
{
        var $name;
        var $coll_obj;

        function XRelXMLFile($xmlfile,$col_obj){

                $this->coll_obj=$col_obj;

                $DocumentT=$this->coll_obj->tables->DocumentT;
                $ElementT=$this->coll_obj->tables->ElementT;
                $AttributeT=$this->coll_obj->tables->AttributeT;
                $TextT=$this->coll_obj->tables->TextT;
                $start=date('d/m/y-H:i:s');
                echo $start;
                $folder="c:\\xmldocuments\\new";
                $elm_array=array();
                $name_array=array();


                $sql="SELECT DocID FROM ".$DocumentT." WHERE docname='$xmlfile'";

                $result = mysql_query($sql);
                $myrow = mysql_fetch_array($result);
                $exist = $myrow[0];

                if(!$exist) {
                   $message="Document could not find in the collection";
                    exit;
                }
                else{
                $tmpDocID=$myrow[0];
                $xml_file="<?xml version='1.0' encoding='utf-8' ?>";
                $fp=fopen("$folder\\$xmlfile","w");
                fwrite($fp,$xml_file);
                fclose($fp);
                $fp=fopen("$folder\\$xmlfile","a");


                $sql_attribute="SELECT ParentID,Value,Name FROM ".$AttributeT." WHERE DocID="
                .$tmpDocID." ORDER BY NodeID";

                $result_att = mysql_query($sql_attribute) ;
                $myrow_att=mysql_fetch_array($result_att);

                if ($myrow_att){
                        $att_parentid=$myrow_att[0];
                        $att_name=$myrow_att[2];
                        $att_val=$myrow_att[1];
                        echo "$att_parentid,$att_name,$att_val";
                        }
                else
                        $att_parentid=0;


                $sql_text="SELECT ParentID,Value FROM ".$TextT." WHERE DocID="
                .$tmpDocID." ORDER BY NodeID";

                $result_text = mysql_query($sql_text) ;
                $myrow_text=mysql_fetch_array($result_text);

                if ($myrow_text){

                        $text_parentid=$myrow_text[0];
                        $text_val=$myrow_text[1];
```

```
            echo "$text_parentid,$text_val";
            }
else
            $text_parentid=0;

$sql_element="SELECT NodeID,ParentID,elname FROM ".$ElementT." WHERE DocID="
.$tmpDocID." ORDER BY NodeID";
$result_element=mysql_query($sql_element);

$i=-1;

While ($myrow_element=mysql_fetch_array($result_element)){

            $el_nodeid=$myrow_element[0];
            $el_parentid=$myrow_element[1];
            $el_name=$myrow_element[2];

            if ($i>=0){
                        while ($el_parentid<>$elm_array[$i]){


                  if ($text_parentid<>0){
                        while ($text_parentid==$elm_array[$i])
                  {
                        $data=trim($text_val);
                        fwrite($fp,$data);
                        $myrow_text=mysql_fetch_array($result_text);
                        if ($myrow_text){
                        $text_parentid=$myrow_text[0];
                        $text_val=$myrow_text[1];
                        }
                        else
                        $text_parentid=0;
                  }
                  }//end if

                              $elname=$name_array[$i];
                              fwrite($fp,"</$elname>\n");

                              array_pop($elm_array);
                              array_pop($name_array);
                              $i--;
                        }
            }


            $attflag=false;
            $attval=" ";
            $elementval=" ";

            if ($att_parentid<>0){
                  while ($att_parentid==$el_nodeid)
                  {
                        $attflag=true;
                        $attval.=$att_name.'="'.$att_val.'" ';
                        $myrow_att=mysql_fetch_array($result_att);
                        if ($myrow_att){
                        $att_parentid=$myrow_att[0];
                        $att_val=$myrow_att[1];
                        $att_name=$myrow_att[2];
                        }
                        else
                        $att_parentid=0;
                  }
                  }

            if ($attflag){
                        $attval=rtrim($attval);
                        $elementval="<$el_name$attval>\n";}
            else
                        $elementval="<$el_name>\n";

            fwrite($fp,$elementval);
```

```
                    if ($text_parentid<>0){
                            while ($text_parentid==$el_nodeid)
                            {
                                    $data=trim($text_val);
                                    fwrite($fp,$data);
                                    $myrow_text=mysql_fetch_array($result_text);
                                    if ($myrow_text){
                                    $text_parentid=$myrow_text[0];
                                    $text_val=$myrow_text[1];
                                    }
                                    else
                                    $text_parentid=0;
                            }
                            }




                    array_push($elm_array,$el_nodeid);
                    array_push($name_array,$el_name);
                    $i++;

                            }//end while
            }

            while ($i>=0){

                    if ($text_parentid<>0){
                            while ($text_parentid==$elm_array[$i])
                            {
                                    $data=trim($text_val);
                                    fwrite($fp,$data);
                                    $myrow_text=mysql_fetch_array($result_text);
                                    if ($myrow_text){
                                    $text_parentid=$myrow_text[0];
                                    $text_val=$myrow_text[1];
                                    }
                                    else
                                    $text_parentid=0;
                            }
                            }
                    $elname=$name_array[$i];

                    fwrite($fp,"</$elname>\n");
                    array_pop($elm_array);
                    array_pop($name_array);
                    $i--;

            }

            fclose($fp);
            $end=date('d/m/y-H:i:s');
            echo " finish time:".$end;

    }//end file



}
}
?>
```

## EdgeXMLFile.inc

```
<?php
class EdgeXMLFile
{
        var $col_obj;
```

```
var $myfile;

function EdgeXMLFile($xmlfile,$col_obj){

  $this->col_obj=$col_obj;
  $DocumentT=$this->col_obj->tables->DocumentT;
  $ElementT=$this->col_obj->tables->EdgeT;

  $start=date('d/m/y-H:i:s');
  echo "start time:".$start;

  $folder="c:\\xmldocuments\\new";
  $fp=fopen("$folder\\$xmlfile","w");

  $elm_array=array();
  $name_array=array();
  $xml_file="<?xml version='1.0' encoding='utf-8' ?>";
  fwrite($fp,$xml_file);
  fclose($fp);
  $fp=fopen("$folder\\$xmlfile","a");

 $sql="SELECT DocID FROM ".$DocumentT." WHERE    docname='$xmlfile'";

$result = mysql_query($sql);
$myrow = mysql_fetch_array($result);
$exist = $myrow[0];

if(!$exist) {
        $message="Document could not find in the collection";
        exit;
        }
else{
        $tmpDocID=$myrow[0];


        $sql_edge="SELECT nodeID,parentID,Name,Flg,TextVal FROM ".$ElementT."
        WHERE  docID=$tmpDocID and flg<>'att' ORDER BY nodeID";
        $result_edge = mysql_query($sql_edge) ;

        $sql_att="SELECT parentID,Name,TextVal FROM ".$ElementT."
        WHERE  docID=$tmpDocID and flg='att' ORDER BY nodeID";
        $result_att=mysql_query($sql_att) ;
        $myrow_att=mysql_fetch_array($result_att);
        if ($myrow_att){
                $att_id=$myrow_att[0];
                $att_name=$myrow_att[1];
                $att_val=$myrow_att[2];}
        else
                $att_id=0;

        $i=-1;


        While ($myrow_edge=mysql_fetch_array($result_edge))
        {
           $nodeid=$myrow_edge[0];
                $parentid=$myrow_edge[1];
                $name=$myrow_edge[2];
                $flag=$myrow_edge[3];
                $text=$myrow_edge[4];



                if ($i>=0){
                                while ($parentid<>$elm_array[$i]){

                                        $elname=$name_array[$i];
                                        fwrite($fp,"</$elname>\n");

                                        array_pop($elm_array);
                                        array_pop($name_array);
                                        $i--;
                                }
                }
                $attval=" ";
                $elementval="";
```

```
                $attflag=false;

                if ($att_id<>0){
                while ($att_id==$nodeid)
                {
                        $attflag=true;
                        $attval.=$att_name.'="'.$att_val.'" ';
                        $myrow_att=mysql_fetch_array($result_att);
                        if ($myrow_att){
                        $att_id=$myrow_att[0];
                        $att_name=$myrow_att[1];
                        $att_val=$myrow_att[2];
                        }
                        else
                        $att_id=0;
                }
                }


                if ($attflag)
                        $elementval="<$name$attval>";
                else
                        $elementval="<$name>";

                if ($flag=="val")
                        $elementval.="$text\n";

                fwrite($fp,$elementval);


            array_push($elm_array,$nodeid);
                array_push($name_array,$name);
                $i++;


        }//end while

        while ($i>=0){
                $elname=$name_array[$i];

                fwrite($fp,"</$elname>\n");
                array_pop($elm_array);
                array_pop($name_array);
                $i--;

        }
        fclose($fp);
        $end=date('d/m/y-H:i:s');
        echo " finish time:".$end;

}//end file


}
}

?>
```

# REFERENCES


Apparao, V., Byrne, S., Champion, M., Isaacs, S., Jacobs, I., Le Hors, A., Nicol, G., Robie, J., Sutor, R., Wilson, C., and Wood, L. *Document Object Model (DOM) Level 1 Specification Version 1.0,* W3C Recommendation 1 October,1998, http://www.w3.orgiTRl998/REC-DOM-Level-l-19981001/.

Biron, B. V., Malhotra, A., *XML Schema Part 2: Datatypes*, W3C Recommendation 02 May 2001, http://www.w3.orglTRl2001lREC-xml-schema-2-20010502/.

Bourret, R., *XML and Databases*, 2002, http://www.rpbourret.com/xml/XMLandDatabases.htm

Bourret, R., *XML Database Products*, 2003, http://www.rpbourret.com/xml/XMLDatabaseProds.htm

Bosak, J., *The Plays of Shakespeare*, 1999, http://www.ibiblio.org/bosak/

Bosak, J., *XML Java Future of the WEB*, 2001, http://www.ibiblio.org/bosak/xml/why/xmlappls.htm

Bray, T., Paoli, J., Sperberg-McQueen, C. M., and Maler, E. *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation 6 October 2000, http://www.w3.orglTR2OOOlREC-xml-200010061.

Chamberlin, D., Clark, J., Florescu, D., Robie, J., Simeon, J., and Stefanescu, M. *XQuery 1.0: An XML Query Language*, W3C Working Draft, 07 June 2001, http://www.w3.org/TRl2001/WD-xquery-200106071.

Clark, J., *XSL Transformations (XSLT) Version 1 .0*, W3C Recommendation, 16 November 1999, http://www.w3.org/TRl999/REC-xslt-19991116.

Clark, J., and DeRose, S., *XML Path Language (XPath) Version 1.0*, W3C Recommendation 16 November 1999, http://www.w3.org/TRl999/REC-xpatb-19991116

Cowan, J., and Tobin, R., *XML Information Set*, W3C Proposed Recommendation 10 August 2001, http://www.w3.org/TR/2001/PR-xrnl-infoset-20010810.

Dayen, I., *Storing XML in Relational Databases*, 2001, http://www.xml.com/pub/a/2001/06/20/databases.html.

DB2 XML Extender, 2002, http://www.4.ibm.com/software/data/db2/extenders/xmlext.html

DBLP Bibliography, 1999, http://www.informatik.uni-trier.de/~ley/db/index.html

DeRose, S., Maler, E., and Orchard, D., XML Linking Language (XLink) Version 1 .0, W3C Recommendation, 27June 2001, http://www.w3.orglTR/2001/REC-xlink-200106271.

Deutsch, A., Fernandez, M. and Suciu, D., "Storing Semistructured Data with STORED", In Proc. of ACM SIGMOD, Philadelphia,PN,1999

Fan, W., and Libkin, L. "On XML integrity constraints in the presence of DTDs", In Proc. 20 Symp. on Principles of Database Systems, 2001, 114-125.

Femandez, M., and March, J., XQuery 1.0 and Xpath 2.0 Data Model, W3C Working Draft 7 June 2001. http://www.w3.orglTRl2001lWD-query-datamodel-200106071.

Florescu, D., and Kossmann, D., "Storing and quering xml data using an RDBMS", IEEE Data Engineering Bulletin 22, 3, 27-34, 1999

Frank C., Exploring XML and Access 2002, 2001, http://msdn.microsoft.com/library/en-us/dnacc2k2/html/odc_acxmllnk.asp

Goldfarb, C., F., Prescod, P., XML Handbook, Prentice Hall 4th Edition, 2001

Jiang, H., Lu, H., "Path Materialization Revisited:An Efficient Storage Model for XML Data",2nd Australian Institute of Computer Ethics Conference, Canberra, 2001

Le Hors, A., Le Hegaret, P., Wood, L., Nicol, G., Robie, J., Champion, M., and Byrne, S. Document Object Model (DOM) Level 2 Core Specification Version 1.0, W3C

Recommendation 13 November, 2000. http://www.w3.org/TR/2000/ REC-DOM-Level-2-Core-20001113.

MySQL AB, 2003, http://mysql.com/benchmark.html

Nolan, C., *SQL Server 2002: New Features Provide Unmatched Ease of Use and Scalability to Admins and Users,* 2002, http://msdn.microsoft.com/library/periodic/period00/sql2000.htm (see "XML Support" section),

Oracle XML DB, 2002, http://otn.oracle.com/tech/xml/xmldb/pdf/xmldb92tfovw.pdf

Schmit, A., Kersten, M. L., Windhouwer, M., and Wass, F. "Efficient relational storage and retrieval of XML documents.", In WebDB (Informal Proceedings), pages 47-52, 2000

Shanmugasundaram, J., Zhang, C., Tufte, K., He, G., DeWitt and D., Naughton, J., "Relational Databases for Querying XML Documents: Limitations and Opportunities.", Proceeding of the 25th VLDB Conference, Edinburgh, Scotland, 1999

Shanmugasundaram, J., Tatarinov, I., Shekita, E., Kiernan, J., Viglas, E. and Naughton, J., "A General Technique for Querying XML Documents using a Relational Database System.", SIGMOD, 2001

Tatarinov, I., And Iglas, S.,D.,V., "Storing and Quering Ordered XML Using a Relational Database System", ACM SIGMOD,Wisconsin, USA, 2002

Widom, J., "Data management for XML", Research Directions bulletin of IEEE Computer Society on Data Engineering Vol: 22, 1999

XBench, 2003, A Family of Benchmarks for XML DBMSs, http://db.uwaterloo.ca/~ddbms/projects/xbench/

YoshiKawa, M. And Amagasa, T., "XRel: A Path -based approach to storage and retrieval of XML documents using relational databases", ACM Transaction on Internet Technology, 2001.