# STORING AND QUERY XML DOCUMENT USING RELATION DATABASE

SITI HAJAR BINTI SULAIMAN

Report submitted in partial fulfilment of the requirements

For the award of the degree of

Bachelor of Computer Science (Software Engineering)

Faculty of Computer Systems & Software Engineering

UNIVERSITI MALAYSIA PAHANG

JUNE 2015

## TABLE OF CONTENTS

## LIST OF TABLE

## LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 BACKGROUND

XML is stands for eXtensible Markup Language. A markup language describes the structure and contain of data. The term of extensible means capable being extended and modified. Thus, XML is a markup language that can be extended and modified to match the needs of the author and data content. (Goldberg, 2007) . Xml was specially designed for data storage and transportation. XML looks a lot like HTML, complete with tags, attributes, and values. The reason to use XML is that it is easy extended and adapted. Xml can also be used to share the data between disparate system and organizations. On the other resources says, XML or eXtensible Markup Languages is a specification for storing information. It is also a specification for describing the structure of that information. XML is a Markup language (just like HTML), while the XML has no tags of its own. (Goldberg, 2007)

The structure of XML is fundamentally tree oriented. This document explores relationships found in the tree structures of XML, and derives XML Query Language requirements from this structure. We will see that supporting the tree structure of XML makes a query language very expressive, capable of combining hierarchy, sequence, position, and text in powerful ways. (Robie, 1999). Hierarchical database are the example of the oldest kinds of database. They arrange data in 'tree' structured. In fact that hierarchical structure must have the parent and child relationship.

Hierarchical database is suitable for the simple structure only. These schema which is required instances to maintain relationship among entities strictly through nested structures. However, XML document have its own ability to transfer the data into the relational structure. Similarly outlook with the other database language called Relational Database. The relational structure can maintain relationship among entities through use the keys rather than physical nesting.

A relational database (RDB) is a collective set of multiple data sets organized by tables, records and column. Relational database (RDB) establish as well-define relationship between database tables. Table communicate and share information which facilitates data search ability, organization and reporting. Relation database organize data in different way if we compare with hierarchical structure.

Each table in relation database is known as relation, which contains one or more data category columns. Each table have rows contain a unique data instance defined for a corresponding column category. One or More data record characteristics relate to one or many records to form functional dependencies. There have its own meaning with the type of relation that has classified. For example, One to One, One to many, Many to one, and many to many relationship among the table.

Relational database performs the database operation by “select”, “project”, and “join”. Select is used for retrieved the data. Project is identifies data attributes and join combines relations. On the other hand, there had a lot of advantages of use relational database. The advantages are easy to extendibility, as new data may be added without modifying existing records. Relational database also promote a flexibility performance with multiple data requirement capabilities.

Data is king! This statement is made by IT professionals because a large age of the application is data driven. In particular, developer needs solution to generate XML document using information stored in databases. XML and database are needed to integrate. The XML and database integrations are important because the XML provides a standard technique to describe data.

## 1.2 PROBLEM STATEMENT

XML (eXtensible Markup Language) is the example of the tools that carry a data in a hierarchical structure. The hierarchical database is the traditional organization of data. This concept is acceptable if handle by machine. But not design to human look. User has its own difficulties to read and store data in a hierarchical structure.

XML is bulky indeed. Metadata in XML document, which are encode as element name, attribute, comment or processing instruction can result in verbose presentation. Besides, to track the data in a hierarchical structure database will cost a lot of time. This is because the set of large data lead us to track it line by line. Moreover, the new records cannot be added to a child table until it has already been incorporated into the parent table. The hierarchical structure database still creates repetition of database system and welcoming the redundancy.

Moreover, with the hierarchical model we will easy to see the problem arise because as each "child" can only have one "parent". The ability to describe the relationship between data such as "many-to-many" or "many-to-one" are not well form if there is involve of more than one child.

The propose method approach is clear about the content in data sets. The data in tree view make a lot of confuse because sometime the redundancy occur. To avoid this situation, the methods and a way have to prove by the end of the research.

## 1.3 OBJECTIVE

These are objectives of this research:

I. To provide an visual interface for an XML database
II. To introduce technique for converting xml document into a relational structured database
III. To enable users processing any querying and searching activities on the converted relational structured database.

## 1.4 SCOPES OF STUDY

The thesis is focusing on development of search engine that have the criteria to adequacy a searching of a query. To fulfil the objective of this thesis, the scope is shown as below:

I. Present the visual interface and converter.
II. Reconstruct the simple XML file to relational database.

## 1.5 THESIS ORGANIZATION

This thesis consists of seven (5) chapters.

I. **Chapter 1** contain introduction of the project including problem statement, objective, scope and overview on every chapter

II. **Chapter 2** describes the literature Review on the querying and storing the XML document using relational database. This part will discuss about the concepts, existing systems which are related to the case study will be reviewed.

III. **Chapter 3** is system methodology. It will be discuss on the method that is used to build the system and project planning. It also provides the needs of the project.

IV. **Chapter 4** describes the project implementation and design that will use technically and practically. This chapter also elaborates in details the work flow of the research. . Supposedly, result analysis is expected to be parallel with the research objective. This chapter also covers on the research constraints.

V. **Chapter 5** is conclusion. Conclusion will summarize the research findings as a whole, and discussed for any future enhancement for the research topic or technique. References and appendices will be added to the last part of this project.

# CHAPTER 2

# LITERATURE REVIEW

## 2.0 INTRODUCTION

Chapter two is the important chapter for any thesis development. A selected literature review will be present in this chapter. Besides that the purpose of this chapter also to describe and explain the literature review carried out to be used in developing the system. The previous researches related to this project also discussed in this section and the existing system related to this project also explained and compared to highlights the differences.

## 2.1 INTRODUCTION TO XML

Before we know the deep in the XML document, better to introduce the XML, there are many opinions from the book author about the XML; table 1 will show the explanations about XML from the books.

| Author | Definition |
|---|---|
| (Goldberg, 2007) | XML is stands for eXtensible Markup Language. A markup language describes the structure and content of data. The term of extensible means capable being extended and modified. Thus, XML is a markup language that can be extended and modified to match the needs of the author and data content. |
| (Godberg, 2009) | XML or eXtensible Markup Languages is a specification for storing information. It is also a specification for describing the structure of that information. And while XML is a Markup language (just like HTML) , while the XML has no tags of its own. |

**Table 1**

XML was specifically designed for data storage and transportation. XML look a lot like HTML, complete with tags, attribute and Values. The XML document is represented in the tree structure. XML documents must contain a root of every element. In fact that tree structure must have the parent and child relationship.

## 2.2 A SHORT HISTORY OF XML.

Extensible Markup Language (XML) history begin with the development of standardised generalised Markup language (SGML) by Charles Goldfarb, along with Ed Mosher and Ray Lorie in the 1970s while working at IBM (Anderson,2004). SGMLL despite the name is not a mark-up languages in its own right, but language used to specify mark-up languages. The purpose of SGML was to create vocabularies which could be used to mark-up documents with structural tags. It was imagined at the time, that certain machine readable documents should remain machine readable for perhaps decades.

One of the most popular applications of SGML came with the development of HyperText Markup Language (HTML) by Tim Berners Lee in the late 1980s (Raggett, Lam, Alexander & Kmiec, 1998). Since its development HTML has somewhat become a victim of its own popularity, as it was rapidly adopted and extended in many ways, beyond its original vision. It remains popular today, though as a presentation technology, and is considered unsuitable as a general purpose data storage format.

When it comes to data storage and interchange, HTML is a bad fit, as it was originally intended as a presentation technology, while SGML is considered too complex for general use. XML bridges this gap by being both human and machine readable, while being flexible enough to support platform and architecture independent data interchange.

## 2.3 Overview of the XML document.

XML documents consist of three parts that is XML declaration, a DTD (Document Type Definition) and an XML instance. An XML declaration and a DTD are not mandatory for an XML document. An XML declaration specifies the version and encoding of XML being used. A DTD is a schema that contains the structure of XML instance, and corresponds to an extended context-free grammar. An Xml instance is a tagged document. We omit concrete description of and XML declaration and a DTD.

An XML document instance is a hierarchical element. The boundaries of which are either delimited by start-tags and end-tags, or empty elements, by empty-element tags. Characters of every element are in between start-tag and end-tags. Figure 1 show an example of an XML instance. (yoshikawa, Aug 1999)

```
<issue>
   <editor>
      <first>Michael</first>
      <family>Franklin</family>
   </editor>
   <articles>
      <article category="research surveys">
         <title>Comparative Analysis of Six XML Schema Languages</title>
         <authors>
            <author>
               <first>Dongwon</first>
               <family>Lee</family>
            </author>
            <author>
               <first>Wesley</first>
               <middle>W.</middle>
               <family>Chu</family>
            </author>
         </authors>
         <summary>As <keyword>XML</keyword> is emerging ... </summary>
      </article>
   </articles>
</issue>
```

**Figure 1**

A start-tag is the token that enclose an element with < and >. Hence, an end-tags is token that enclose the element type with </ and >. The element can nest properly within each other and the nesting represent a logical structure. Within start-tags, attribute name and attribute values can be specified.

XML document have two level of conformance which is valid and well-formed. A well-formed XML document follows tagging rules prescribe in XML. Hence, the XML valid is depend on the XML is well-formed or not. The XML document also valid if the document compiles with the constraint expressed in an associated DTD.

It is an XML processor that examine whether an XML document is well-formed (or valid). The XML processor is a software module, which is used to read XML documents and provide access to their content and structure. It is assumed that an XML processor doing its work on behalf of another module, called the application (Consortium, 1998)

## 2.4 Data Model for XML Document

We employ the data model XPath (Consortium, 1998) to represents XML documents. We assume that the XML documents are guaranteed to be valid or well-formed by XML documents XM processors. Here we briefly introduce the XPath data model. The full specification of data model can be found in [World Wide Web consortium 1999]

In the XPath data model, XML documents are modelled as an ordered tree. There are seven types of nodes. In this thesis, we consider only the following four types of nodes for the sake of simplicity. For each type of nodes, there is a way of determining a string- value for a node of the type. Some of node also has expanded-name.

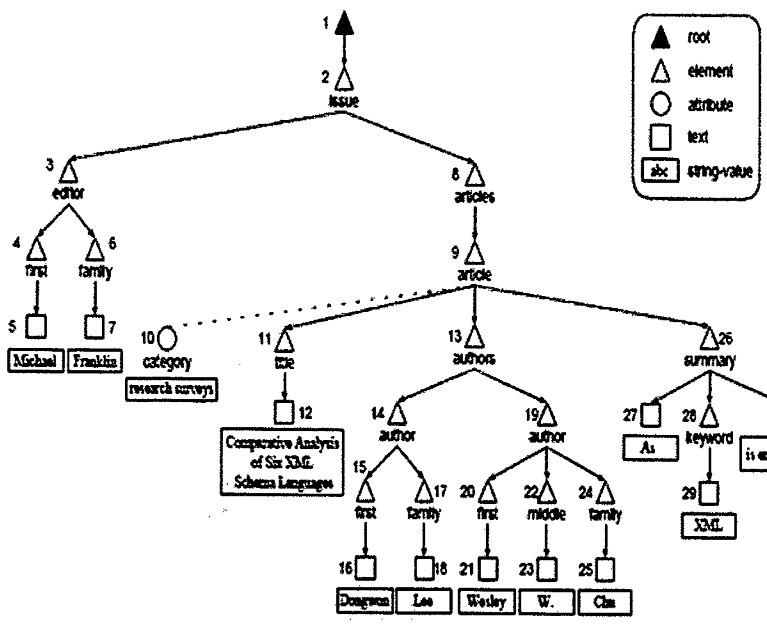

**Figure 2**

The XML document order is define among all nodes on the corresponding from the first character. Figure2 above had shown an XML tree. Reverse document order is simply the reverse of the document order.

I. Root node:

The root node is the root of tree. The element node for the document element is a child of the roots node. The string-value of the root node is the concatenation of the string-value of all text node descendant of the root node in document order.

II. Element node:

There is an element node for every element in the document. An element node has an expanded-name, which is the element type name specified in the tag. Element nodes have zero or more children. The type of each node is element or text. The string-value of an element node is the concatenation of the string-values of all text node descendant of the element node in document order.

III. Attribute node:

Each element node has an associated set of attribute nodes. Note that the element node is the parent of each attribute nodes. However, an attribute node is not child of element node. Attribute node have an attribute name and attribute value. Attribute nodes have no child nodes.

IV. Text node:

Text nodes have character data specified in the XML. To use the text node is recommended by use string-value. A text node does not have an expanded-name. Text nodes have no child nodes.

The remaining three types of nodes are namespace node, processing instruction nodes, and document nodes. This discussion on this thesis will be extended to include these three types of nodes.

## 2.5 RELATIONAL DATABASE

The dominant storage mechanism for structured enterprise data is the relational database. A relational database is a collection of data item organized as a set of formally-described table from which data can be accessed or reassembled in many different ways without having reorganized the database tables. (Codd, 1970).

On the other hand, relational database is store in a table with rows and column. Each table composed of record which is called as tuple. Each record is identified by a field or attributes that containing a unique value. Every table shares at least one field with another table in 'one to one', 'one to many', 'or 'many to many' relationships. These relationships allow the database user to access the data in almost an unlimited number of ways. These relationships also allow user to combine the tables as building blocks to create complex and very large databases. Figure shown below is the structured of relational database.

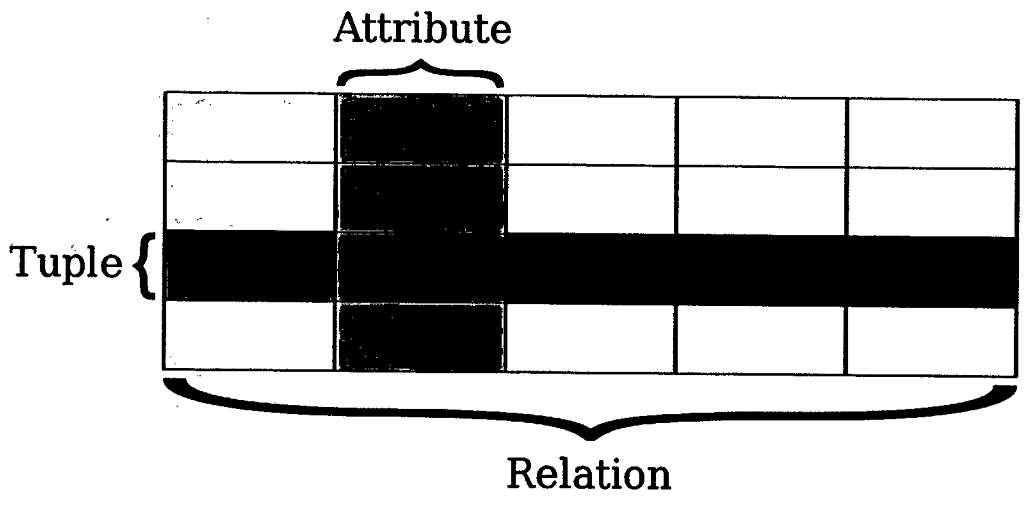

**Figure 3**

## 2.6 MAPPING RELATIONAL DATA

Getting XML data into a database is only half of the issue when using XML. As older legacy system are being upgraded for service oriented architecture, and web services, making the contained data available in XML formats is becoming more important for engineers. This requires approaches for mapping existing relational data to XML formats.

Lv & Yan (2006) approach this problem by attempting to translate relational database schemas to DTD documents. They present a method to generate DTDs from relational schemas in the presence of keys, foreign keys, and functional dependencies, which can preserve the semantics, implied by functional dependencies, keys and foreign keys of relational schemas and can convert multiple tables to XML DTDs. While this is a forward step towards full semantic conversion of relational schemas to XML DTDs, they note there is still work remaining in converting further relational semantics such as multi-valued dependencies

DTDs are the most commonly used XML schema definition documents, but as Lim, Joo, Kim & Choi (2007) note, it is a simple format, which does not have the resolution to take into account some of the finer points of relational data, such as maintaining primary and foreign key and other constraints.

DTD defines the structure of a well formed XML document using simple format expressions. These do not allow a sufficient level of detail to be used in XML to relational mapping. For example, DTD can define a list to contain zero or more, or one or more elements, though it cannot define other limits.

Lim, Joo, Kim & Choi (2007) base their XML mapping algorithm using the newer XML Schema Definition documents (XSD). This can specify a list to contain 2 to 5 elements for example, and can be used to ensure that an XML document is both well-formed and valid against this schema. XML Schema allows for finer grained control of an XML format than DTD and is better suited to enabling automated mapping of an XML Schema to SQL Data Definition Language for relational database mapping.

## 2.7 TECHNIQUE TO RECONSTRUCT THE XMLVIEW

Whenever an XML document repository is to be created over relational database system, one of the possibly many relational schema generation techniques is used automatically create the relational table. A relational table is use to storing XML documents. Inserted XML document are then shredded and “stored” as rows in these tables. In addition, a construction XML view is created over the created relational table. This way is virtually reconstruct the “stored” XML document from the shredded rows. The reconstruction of XML view is specified just like the regular XML view of relational data.

The key of observation here is that a reconstruction XML view makes it possible to treat XML document through the XML view of relational data. This turn can be efficiently handled by a query processor used for processing Further, this query processor can process queries over XML documents and XML views of existing relational data, because they are all just XML views of relational data. This makes it possible to query seamlessly over XML documents and relational data query over XML views of relational data.

By reconstruct the XML document using relational database is generally enough to support many mechanism for relational schema generation. This is because, for a given mechanism, only a program stub that does the following is required. When the stub is invoked by possibly with schema of XML documents to be stored, it does the following:

1) Generate the desired relational schema for storing XML documents

2) Produces and XML shredder object that can take in XML document and shred them into rows in a table of generated relational schema.

3) Creates a reconstruction XML view over the generated relational schema that indicates how shredded XML documents are to be (virtually) re-constructed.

However, using the proposed technique, it is sufficient to just generate a reconstruction XML view instead of full writing a full blown XML query processor.

## 2.8 SUMMARY

In conclusion, XML has proved hugely successful in the areas of document mark-up, data and meta-data sharing, enabling interoperability, and transparently transporting and storing data. Using relational database in the XML document by reconstructing the XML view is good to user understand the element.

# CHAPTER 3

# METHODOLOGY

## 3.1 INTRODUCTION

This chapter discuss about the methodology that will used for designing and implementing the construction of XML document using the relational database. The technique used in this section is by Xrecursive techniques. Methodology used in this thesis is for make sure the research will be done correctly with plan.

## 3.2 RESEARCH METHODOLOGY

In this thesis, the right methods are the most important part. The right method will lead to the success. When the methods apply is right, confident level of the running application is raised up. First of all, I read and explore the entire journal in this area. There is so much information about constructing an XML document using relational Database. From cover them all, the conclusion be made to develop the techniques approached. Figure 3.1 show the overall methodology flow in this thesis.

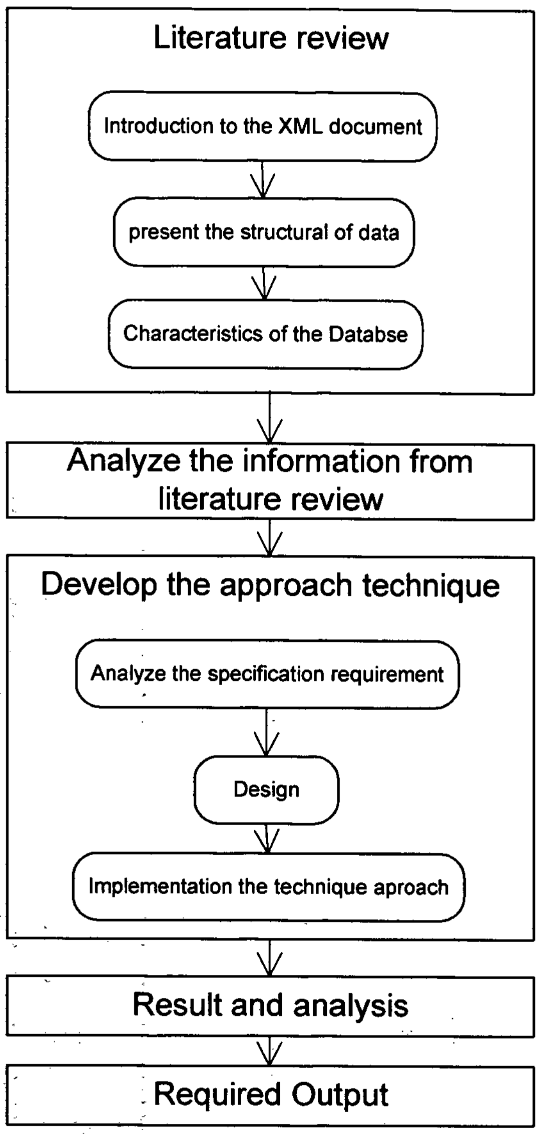
Literature review
Introduction to the XML document
present the structural of data
Characteristics of the Databse
Analyze the information from literature review
Develop the approach technique
Analyze the specification requirement
Design
Implementation the technique aproach
Result and analysis
Required Output

**Figure 3.1**

## 3.3 Flowchart.

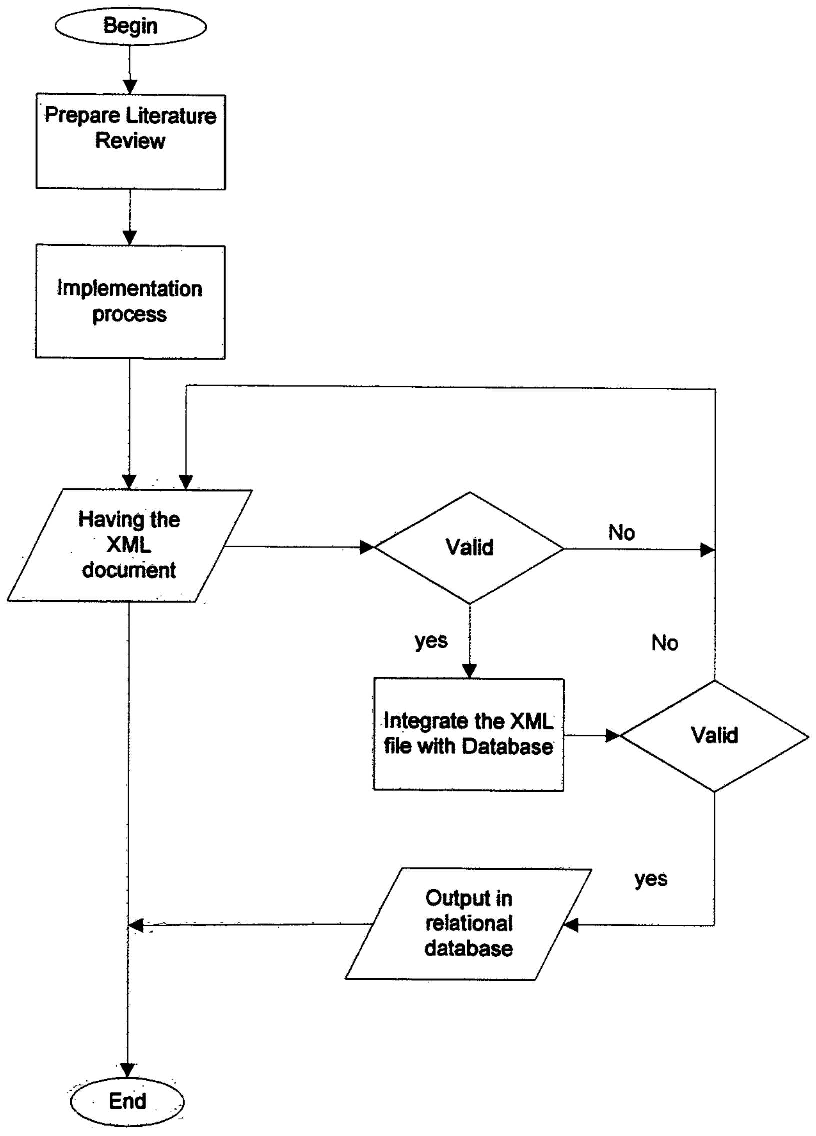
Begin
Prepare Literature Review
Implementation process
Having the XML document
Valid
No
yes
No
Integrate the XML file with Database
Valid
yes
Output in relational database
End

**Figure 3.2**

Figure 3.2 shows the flowchart of the constructing the XML document using Relational Database. To begin the implementation process the valid of XML document should be ready. When the XML document is valid, the systems will integrate the XML file with the database. At the end of the process, the XML document will change into relational database. Then, the process of the searching or querying database activity can be done.

## 3.4 Algorithm

```
1: Begin
2: let N = { } as empty set, where N represents the list of node of the XML file
1       Let V = { } as empty set, where V represents the list of the value of th
node.
2       Let String filename = null and int id = 1;

5: filename = read XML document name
6: while xml Document is not null
7: Begin
8: read element name as name
9: read the element type
10: if element type is Element or attribute
11: begin
12: read parent as pName
13: id = id + 1;
14: add name, pName, id to the N
15: End
16: else if element type is #text
17: Begin
18: Read textvalue as value
19: Id = id + 1;
20: Add name, value, id to V
21: end
22: store N into database
23: store V into database
24: End.
```

## 3.5 Summary

Project methodology describes the approach use in the study. The waterfall model is suitable for development in a step by step. With this method, the development process is carefully focused. For performance on this methodology explain the basic activities. Every phase must be followed to ensure that the process of development run smoothly and achieve its goals.

# CHAPTER 4

# RESULTS AND DISCUSSION

## 4.1 Introduction

Practically, this thesis covers the process of storing XML document using Relational database. The separation of the element is shown in this chapter. In the end, the discussion about the output result is stated clearly.

## 4.2 Analysis process

Depend on the process; the flow of the system is representing in a flowchart. Figure 4.1 shows the process flow.

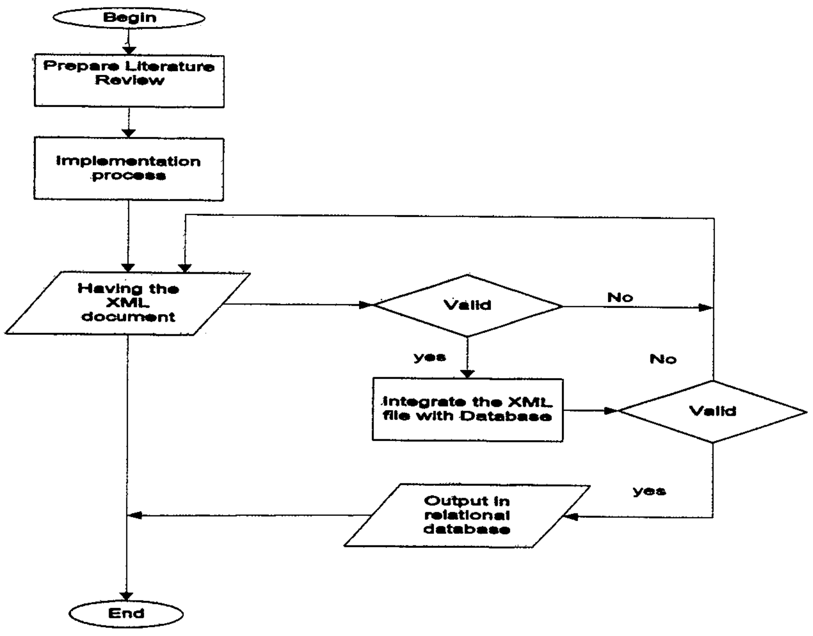
Begin
Prepare Literature Review
Implementation process
Having the XML document
Valid
No
yes
No
Integrate the XML file with Database
Valid
yes
Output in relational database
End

Figure 4.1

## 4.3 XML DOCUMENT

The data structure of XML document is hierarchical, consist of nested structures. The element is strictly marked by the beginning and ending tags, for empty elements by empty-element tags. Character data between tags are the content of the elements. It is an instance of XML document contain information about books as follows.

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
   </book>
   <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
      <description>A former architect battles corporate zombies,
      an evil sorceress, and her own childhood to become queen
      of the world.</description>
   </book>
```

**Figure 4.2**

## 4.4 Tree structure

In this section , the tree structure of XML document in figure 4.2 is represent with Xrecursive labelling as below.

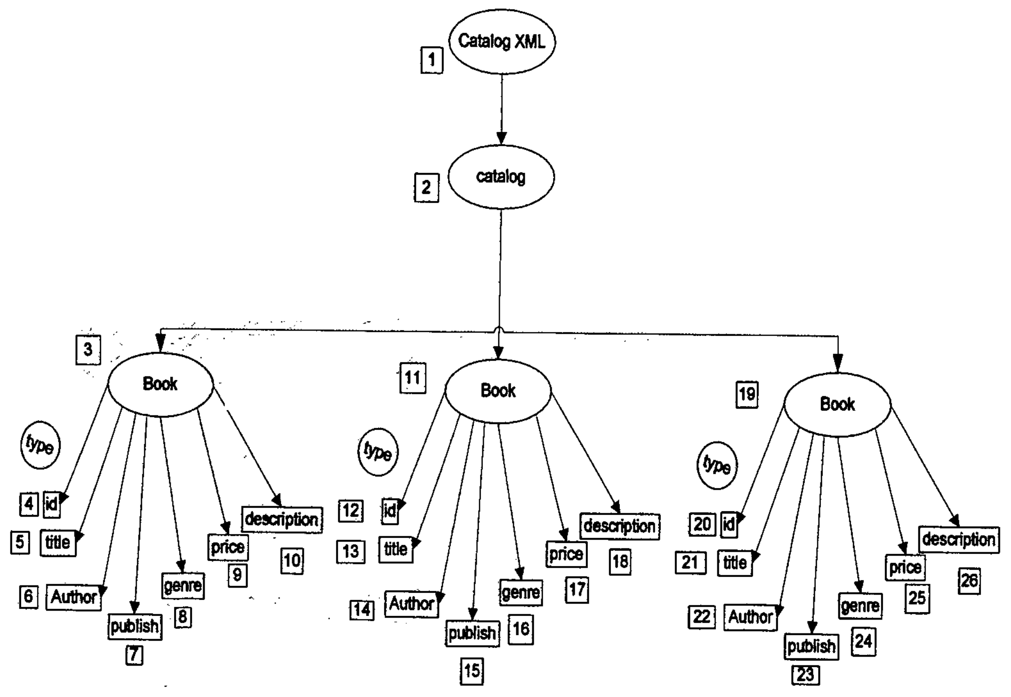

Figure 4.3

Each of every XML can describing as a XML tree. In this figure the oval are the element and the square are the attribute of the elements. A generated XML tree has been shown in a figure. Every element or attributes are identify by a signature (number).