CHAPTER 1

INTRODUCTION

1.1 Introduction

As the amount of data available on the Internet grows rapidly, more and more of the data becomes semi structured and hierarchical. The data has no absolute schema fixed in advance, and its structure is irregular or incomplete. Semi structured data arises when the source does not impose a rigid structure, and when the data is obtained by combining several heterogeneous data sources during the process of data integration. Semi structured data is often called schemaless or self-describing, because there is no separate description of its type or structure and its data (Ganguly, Sarkar, 2012).

The Extensible Markup Language (XML), as a format for semi structured data, has become a standard for the representation and exchange of data over the Internet. Using XML as a data representation standard, it is possible to represent not only the data itself, but also its semantics. XML was designed specifically to describe content, rather than presentation. It is a textual representation of data (Seligman and Rosenthal, 2001).

The popularity and wide-spread use of XML among a diverse set of organizations has engendered a rethinking of the storage and retrieval practices for data. Most early XML storage practices relied on mappings and transformations between XML data trees and relational database tuples within a conventional Relational Database Management Systems (RDBMS).
As a consequence, a single business operation might require numerous translations of the data from XML to relational table formats and vice versa at a significant cost in speed, reliability and efficiency of representation (Nicola and John, 2003; SC Haw, 2007).

XML documents in relational systems would be simply mapped into existing relational database structures like “Large Objects” (LOBs), with XML being stored intact as plain unparsed text. Another approach used by RDBMs has been to create appropriate relational schemas and “shred” XML documents into many tables. The limitations of these two approaches are now well known. While inserting and extracting full XML documents is relatively fast in the case of LOB storage, this approach can be relatively slow during query processing and fragment extraction due to the need for XML parsing at query execution time (Nicola, and Linden, 2005; Zhang, and et al, 2009). While the “shredded” approach can provide reasonable performance given a good mapping, mapping an XML schema to an equivalent relational schema is usually a complicated job. Also, the nested and repeating elements in XML documents can quite easily result in an unmanageable number of tables. Furthermore, it is usually very difficult after insertion to change the relational schema due to XML schema changes. Thus, the flexibility of XML is essentially lost with this approach (Shao, 2010; Fiebig et al, 2002).

XML data is a challenge for relational databases. Hence the need to develop appropriate storage systems to store XML data and have the ability to speed the access of relevant and accurate information (Oracle 2012) Therefore, Native XML database becomes more and more popular and gained popularity as a flexible storage format. In Native XML database, XML data stored in the internal solid data model which retains the structure of the XML data. When storing XML data in a database, many questions arise:

(i) How to store the data in the databases?
(ii) How to express queries and updates?
(iii) How can XML data be indexed to speed up frequent queries?
(iv) How to detect if a modifying operation affects an established index and must therefore be updated to keep it consistent?
(v) How can indexes that are best for a given application determined automatically?
Thus, the querying process in XML data has become a challenge (Qtaish and Ahmad, 2014), and there has been a strong demand for improved query languages for processing XML documents. XML data is hierarchical structure (tree structure), and document data represented in XML comprise a sequence of possibly nested tags which can be expressed by a tree structure, and querying and transforming XML data from one format into another will be a frequent task.

This thesis focuses on the path expressions in native XML databases. The main focus will be enhancing path expressions as expressive for database management systems (Native XML database and XML Enabled database); path expressions specify special regular expressions on trees. In addition, path expressions as ubiquitous in many XML query languages (e.g., XPath (Berglund and Boag), XQuery (Boag et al) and XSLT (Clark).

1.2 Background

XML stands for extensible Markup Language, which is originated from Standard Generalized Markup Language (SGML, ISO 8879). It is a markup language designed to be relatively human-legible. Hypertext Markup Language (HTML) is another predominant markup language for web page design. It is also originated from SGML prior to the inception of XML. However, XML is not a replacement for HTML, as they are designed for different objectives, XML is designed to describe data and to focus on what data is. HTML is designed to display data and to focus on how data looks. In other words, HTML is about displaying information, while XML is about describing information.

Consider the bibliography information shown in Figure 1.1, and the corresponding HTML and XML representations in Figure 1.2. We can observe some differences between HTML and XML. In HTML, the tags and document structures are fixed. Only the tags which are defined in the HTML standard (e.g., (hi), (p)) can be used. In XML, however, users can define their own tags (e.g., (book), (title)) and document structure. As a result, it is more flexible and powerful. XML schema languages, e.g., Document Type Definition (DTD) and XML schema, are proposed to provide a high level abstraction of XML documents in terms of constraints on both structure and content.