TUP-Springer **Project**

Jiaheng Lu

XML数据查询和检索技术

An Introduction to XML Query Processing and Keyword Search

清华大学出版社

Springer

Jiaheng Lu

XML数据查询和检索技术

An Introduction to XML Query Processing and Keyword Search

清华大学出版社
北 京

Springer

内 容 简 介

本书系统、全面地阐述了关于 XML 数据检索和查询方面最新的研究成果，包括 XML 数据的编码、索引、模式匹配，结果估计，关键词检索，查询改写等方面内容。本书主要强调内容的先进性，将作者最新的 XML 数据查询和检索的最新成果涵盖进来，其中某些内容还未公开发表。本书主要面向系统开发人员，研究生和科研工作者，也可供其他读者学习使用，对专业人士可作为最新的技术参考资料。

图书在版编目 (CIP) 数据

XML 数据查询和检索技术 = An introduction to XML query processing and keyword search: 英文 / 陆嘉恒主编. -- 北京：清华大学出版社, 2013

ISBN 978-7-302-28563-2

Ⅰ. ①X…　Ⅱ. ①陆…　Ⅲ. ①智能检索系统－可扩充－语言－程序设计－英文　Ⅳ. ①G354.4-39

中国版本图书馆 CIP 数据核字(2013)第 301916 号

责任编辑： 汪汉友
封面设计： 何凤霞
责任校对： 白　蕾
责任印制： 宋　林

出版发行： 清华大学出版社
　　网　　址：http://www.tup.com.cn, http://www.wqbook.com
　　地　　址：北京清华大学学研大厦 A 座　　　邮　　编：100084
　　社 总 机：010-62770175　　　邮　　购：010-62786544
　　投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
　　质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 装 者： 三河市春园印刷有限公司
经　　销： 全国新华书店
开　　本： 153mm × 235mm　　**印张：** 19　　**字数：** 329 千字
版　　次： 2013 年 12 月第 1 版　**印次：** 2013 年 12 月第 1 次印刷
印　　数： 1 ~ 1000
定　　价： 99.00 元

产品编号：035424-01

To my wife Chun, my daughter Anqi and my parents with all my love.

Preface

XML is short for eXtensible Markup Language, whose purpose is to aid information systems in sharing structured data, especially via the Internet, to encode documents, and to serialize data. The properties that XML inherited make it the widely popular standard of representing and exchanging data.

When working with those XML data, there are (loosely speaking) three different functions that need to be performed: adding information to the repository, searching and retrieving information from the repository, and updating information from the repository. We focus on all three parts of functions. There are two decades since the beginning of XML, and this field has expanded tremendously and is still expanding. Thus, no book on XML can be comprehensive now—certainly this one is not. We just present which we could present clearly.

What Is the Uniqueness of This Book?

This book aims to provide an understanding of principles and techniques on XML query processing and XML keyword search. For this purpose, progress has been made in the following aspects:

Firstly, we give a brief introduction of XML, including the emergence of XML database, XML data model, and searching and querying XML data. In order to facilitate query process over XML data that conforms to an ordered tree-structure data model efficiently, a number of labeling schemes for XML data have been proposed.

Secondly, we proposed several XML path indexes. Over the past decade, XML has become a commonly used format for storing and exchanging data in a wide variety of systems. Due to this widespread use, the problem of effectively and efficiently managing XML collections has attracted significant attention. Without a structural summary and an efficient index, query processing can be quite inefficient due to an exhaustive traversal on XML data. To overcome the inefficiency, several path indexes have been proposed, such as prefix scheme, extended Dewey ID, and CDBS.

Thirdly, answering twig queries efficiently is important in XML tree pattern processing. In order to perform efficient processing, we introduce two kinds of join algorithms, both of which play significant roles. Also, solutions about how to speed up query processing and how to reduce the intermediate results to save spaces are present.

Fourthly, we show a set of holistic algorithms to efficiently process the extended XML tree patterns. Previous algorithms focus on XML tree pattern queries with only P-C and A-D relationships. Little work has been done on extended XML tree pattern queries which contain wildcards, negation function, and order restriction, all of which are frequently used in XML query languages. The holistic algorithm will make it more completed.

Fifthly, we study XML keyword search semantics algorithms and ranking strategy. We present XML keyword search semantics such as SLCA, VLCA, and MLCEA, which is useful and meaningful for keyword search. Based on some of the semantics, we present XML keyword search algorithms such as DIL Query Processing Algorithm. In addition, we introduce the XML keyword search ranking strategy; we propose an IR-style approach which basically utilizes the statistics of underlying XML data to address these challenges.

Sixthly, we introduce the problem of XML keyword query refinement and offer a novel content-aware XML keyword query refinement framework. We also introduce LCRA, which provides a concise interface where user can explicitly specify their search concern—publications (default) or authors.

Lastly, we present several future works, such as graphical XML data processing, complex XML pattern matching, and MapReduce-based XML query processing.

Jiaheng Lu

Acknowledgement

I would like to express my gratitude to Prof. Tok Wang Ling in National University of Singapore, for his support, advice, patience, and encouragement. He is my PhD advisor and has taught me innumerable lessons and insights on the workings of academic research in general.

My thanks also go to Prof. Mong-Li Lee, Prof. Chee Yong Chan, and Prof. Anthony K H. Tung in National University of Singapore, who provided valuable feedback and suggestions to my idea.

I shall thank my colleagues Prof. Shan Wang, Prof. Xiaoyong Du, Prof. Xiaofeng Meng, Prof. Hong Chen, Prof. Cuiping Li, and Prof. Xuan Zhou in Renmin University of China, who give me tremendous supports and advices on my research on XML data processing.

My thanks also go to my friends Ting Chen, Yabing Chen, Qi He, Changqing Li, Huanzhang Liu, Wei Ni, Cong Sun, Tian Yu, Zhifeng Bao, and Huayu Wu in National University of Singapore. They have contributed to many interesting and good spirited discussions related to this research. They also provided tremendous mental support to me when I got frustrated at times.

My thanks also go to my students Chunbin Lin, Caiyun Yao, Junwei Pan, Haiyong Wang, Si Chen, Siming Yang, and Xiaozhen Huo in Renmin University of China. My thinking on XML was shaped by a long process of exciting and inspiring interactions with my students. I am immensely grateful to all of them.

I would also like to thank Dr. Jirong Wen, Microsoft Research Asia; Prof. Rui Zhang, Melbourne University; Prof. Jianzhong Li, Harbin Institute of Technology; Prof. Bin Cui, Peking University; Prof. Jianhua Feng and Prof. Guoliang Li, Tsinghua University; and Mr. Hanyou Wang and Ms. Hui Xue, Tsinghua University Press, for their recommendation and valuable suggestions.

Last, but not least, I would like to thank my wife Chun Pu for her understanding and love during the past few years. Her support and encouragement was in the end what made this book possible. My parents and parents-in-law receive my deepest gratitude and love for their dedication and the many years of support during my studies.

My research is partially supported by National 863 project (2009AA01Z133); National Science Foundation, China (61170011); Beijing National Science Foundation (109004); and Research Funds of Renmin University of China (No: 11XNJ003).

Jiaheng Lu

Contents

Chapter 1
Introduction

Abstract When working with those XML data, there are three different functions that need to be performed: adding information to the repository, searching and retrieving information from the repository, and updating information from the repository. A good XML database must handle those functions well. In this chapter, we will introduce solutions for XML database, including flat files, relational database, object relational database, and other storage management system.

Keywords Relational database • Object relational database

1.1 XML Data Model

An XML document always starts with a prolog markup. The minimal prolog contains a declaration that identifies the document as an XML document. XML identifies data using tags, which are identifiers enclosed in angle brackets. Collectively, the tags are known as "markup." The most commonly used markup in XML data is element. Element identifies the content it surrounds. For example, Fig. 1.1 shows a simple example XML document. This document starts with a prolog markup that identifies the document as an XML document that conforms to version 1.0 of the XML specification and uses the 8-bit Unicode character encoding scheme (Line 1). The root element (Line 2–14) of the document follows the declaration, which is named as bib element. Generally, each XML document has a single root element. Next, there is an element book (Line 3–13) which describes the information (including author, title, and chapter) of a book. In Line 9, the element text contains both a subelement keyword and character data *XML stands for* . . . .

```
<?xml version = "1.0" encoding = “UTF–8”?>
<bib>
 <book>
  <author>Suciu</author>
  <author>Chen</author>
  <title> Advanced Database System </title>
  <chapter><title>XML</title>
   <section><title>XML specification</title>
    <text><keyword>markup</keyword> XML stands for...
    </text>
   </section>
  </chapter>
 </book>
</bib>
```

Fig. 1.1 Example XML document

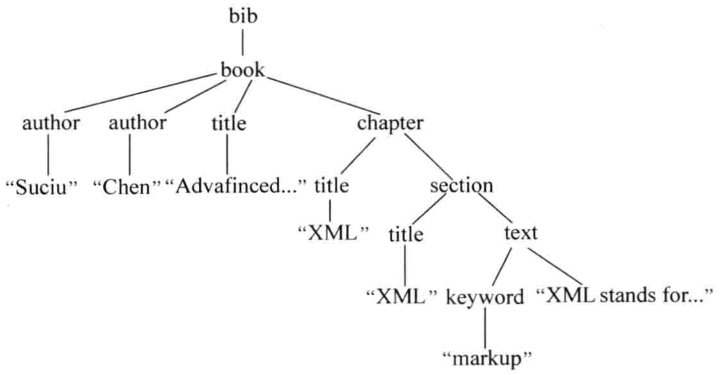

Fig. 1.2 Example XML tree model

Although XML documents can have rather complex internal structures, they can generally be modeled as trees,[1] where tree nodes represent document elements, attributes, and character data and edges represent the element–subelement (or parent–child) relationship. We call such a tree representation of an XML document as an XML tree. Figure 1.2 shows a tree that models the XML document in Fig. 1.1.

XML has grown from a markup language for special purpose documents to a standard for the interchange of heterogenous data over the Web, a common language for distributed computation, and a universal data format to provide users with different views of data. All of these increase the volume of data encoded in XML, consequently increasing the need for database management support for XML documents. An essential concern is how to store and query potentially huge amounts of XML data efficiently [AJP+02, AQM+97, JAC+02, LLHC05, MW99, ZND+01].

[1]For the purpose of this book, when we model XML document as trees, we consider IDREF attributes as not reference links but subelements.

1.2 Emergence of XML Database

XML has penetrated virtually all areas of Internet-related application programming and become the frequently used data exchange framework in the application areas [Abi97, CFI+00, DFS99]. When working with those XML data, there are (loosely speaking) three different functions that need to be performed: adding information to the repository, searching and retrieving information from the repository, and updating information from the repository. A good XML database must handle those functions well. Many solutions for XML database have been proposed, including flat files, relational database [FTS00, Mal99, SSK+01, STZ+99, TVB+02, ZND+01], object relational database [ML02, SYU99], and other storage management system, such as Natix [FM01], TIMBER [JJ06, JLS+04, PJ05, YJR03], and Lore [MAG97]. We briefly discuss these solutions as follows.

1.2.1 Flat File Storage

The simplest type of storage is flat file storage, that is, the main entity is a complete document; internal structure does not play a role. These models may be implemented either on the top of real file systems, such as the file systems available on UNIX, or inside databases where documents are stored as binary large objects (BLOBs). The operation store can be supported very efficiently at low cost, while other operations, such as search, which require access to the internal structure of documents may become prohibitively expensive. Flat file storage is not most appropriate when search is frequent, and the level of granularity required by this storage is the entire document, not the element or character data within the document.

1.2.2 Relational and Object Relational Storage

XML data can be stored in existing relational database. They can benefit from already existing relation database features such as indexing, transaction, and query optimizers. However, due to XML data that is a semistructured data, converting this data model into relation data is necessary. There are mainly two converting methods: generic [FK99] and schema-driven [STZ+99]. Generic method does not make use of schemas but instead defines a generic target schema that captures any XML document.

Schema-driven depends on a given XML schema and defines a set of rules for mapping it to a relational schema. Since the inherent significant difference between rational data model and nested structures of semistructured data, both converting methods need a lot of expensive join operations for query processing.

Mo and Ling [ML02] proposed to use object relational database to store and query XML data. Their method is based on ORA-SS (Object-Relationship-Attribute model for Semistructured Data) data model [DWLL01], which not only reflects the nested structure of semistructured data but also distinguishes between object classes and relationship types and between attributes of objects classes and attributes of relationship types. Compared to the strategies that convert XML to relational database, their methods reduce the redundancy in storage and the costly join operations.

1.2.3 Native Storage of XML Data

Native XML engines are systems that are specially designed for managing XML data [MLLA03]. Compared to the relational database storage of XML data, native XML database does not need the expensive operations to convert XML data to fit in the relational table. The storage and query processing techniques adopted by native XML database are usually more efficient than that based on flat file and relational and object relational storage. In the following, we introduce three native XML storage approaches.

The first approach is to model XML documents using the Document Object Model (DOM) [Abi97]. Internally, each node in a DOM tree has four pointers and two sibling pointers. The filiation pointers include the first child, the last child, the parent, and the root pointers. The sibling pointers point to the previous and the next sibling nodes. The nodes in a DOM tree are serialized into disk pages according to depth-first order (filiation clustering) or breadth-first order (sibling clustering). Lore [MAG97, MW99] and XBase [LWY+02] are two instances of such a storage approach.

The second approach is TIMBER project [JA02], at the University of Michigan, aiming to develop a genuine native XML database engine, designed from scratch. It uses TAX, a bulk algebra for manipulating sets of trees. For the implementation of its Storage Manager module, it uses Shore, a back-end storage system capable for disk storage management, indexing support, buffering, and concurrency control. With TIMBER, it is possible to create indexes on the document's attribute contents or on the element contents. The indexes on attributes are allowed for both text and numeric content. In addition, another kind of index support is the tag index, that, given the name of an element, it returns all the elements of the same name.

Finally, Natix [FM01] is proposed by Kanne and Moerkotte at the University of Mannheim, Germany. It is an efficient and native repository designed from scratch tailored to the requirement of storing and processing XML data. There are three features in Natix system: (1) subtrees of the original XML document are stored together in a single (physical) record; (2) the inner structure of subtrees is retained; and (3) to satisfy special application requirements, the clustering requirements of subtrees are specifiable through a split matrix. Unlike other XML DBMS which provide fully developed functionalities to manage data, Natix is only a repository. It is built from scratch and has no query language, no much work done on indexing and query processing, and no use of DTDs or XML schema.

1.3 XML Query Language and Processing

To retrieve such tree-structured data, a few XML query languages have been proposed in the literature. Examples are Lorel [AQM+97], XML-QL [DFF98], XML-GL [CCD+99], Quilt [CRF00], XPath [BBC04], and XQuery [BCF03]. Of all the existing XML query languages, XQuery is being standardized as the major XML query language. XQuery is derived from the Quilt query language, which in turn borrowed features from several other languages such as XPath. The main building block of XQuery consists of path expressions, which addresses part of XML documents for retrieval, both by value search and structure search in their elements. For example, the following path expression */bib/book[author= 'Suciu']/title* asks for the title of the book written by "Suciu." In Fig. 1.1, this query returns the title *Advanced Database System*.

1.4 XML Keyword Search

The extreme success of web search engines makes keyword Search the most popular search model for ordinary users. As XML is becoming a standard in data representation, it is desirable to support keyword search in XML database. It is a user-friendly way to query XML databases since it allows users to pose queries without the knowledge of complex query languages and the database schema.

Most previous efforts in this area focus on keyword proximity search in XML based on either tree data model or graph (or digraph) data model. Tree data model for XML is generally simple and efficient for keyword proximity search. However, it cannot capture connections such as ID references in XML databases. In contrast, techniques based on graph (or digraph) can capture those connections, but the algorithms based on the graph model are very expensive in many cases. In this book, we will show interconnected object trees model for keyword search to achieve the efficiency of tree model and meanwhile to capture the connections such as ID references in XML by fully exploiting the property and schema information of XML databases. In particular, we will propose ICA (Interested Common Ancestor) semantics to find all predefined interested objects that contain all query keywords. We will also introduce novel IRA (Interested Related Ancestors) semantics to capture the conceptual connections between interested objects and include more objects that only contain some query keywords. Then a novel ranking metric, RelevanceRank, is studied to dynamically assign higher ranks to objects that are