一种新的基于索引的多版本XML文件的结构查询法

丁 峥 周 虹
(苏州科技学院电子与信息工程学院 苏州 215011)

摘 要 主要讨论如何突破版本恢复的限制直接对任意版本的XML文件进行复杂的结构查询,围绕这个主题,首先介绍了目前XML文档版本管理的一般办法,然后在多版本XML文档的编码方式的基础上提出并实现了一种新的索引机制,进而将结构化连接的查询方法引入XML版本管理的领域,改进了3个经典的结构连接算法,这些算法均能在不恢复版本的前提下直接进行任意版本的结构查询。实验分析比较了它们的查询效能并证明了基于索引的算法能最大程度地避免查询中的冗余。

关键词 XML,索引,多版本XML,结构化连接
中图法分类号 TP311 **文献标识码** A

Structural Query for Multiversion XML Document Based on Index

DING Zheng ZHOU Hong
(Electronics and Information Engineering Department, Soochow Science & Technology University, Suzhou 215011, China)

Abstract This article mainly discussed how to break through the limitation of version recovery and conduct complicated structure inquiry in any version of XML documents. Surrounding the thesis, it introduced the common method of current XML document version management, then brought out and implemented a new index method based on numbering schema of multi-version XML file, finally refered structural joining algorithm into XML version control field and improved three classical structure joining algorithm, all of which support complicated structure inquiry without version recovery. We analyzed and compared their searching efficiency by experiment, and proved that index searching algorithm can avoid redundancy in searching as much as possible.

Keywords XML, Index, Multiversion XML, Structural Join

1 XML文档版本管理的研究现状

XML文档版本管理的目的是不仅可以让使用者操作最新版本的XML文档,还可以快速获取旧版本的XML文档,对XML文档版本发展过程进行追踪。目前已提出的XML文档版本管理技术有Diff-based,Timestamps-based和文档版本整合等方法。

Diff-based[1]方法是利用edit-script语法(insert,delete,update,move)来记录同一份XML文档不同版本之间的变化。不管Diff-based方法经过怎样的发展,仍然无法突破直接获取任意XML文档版本的限制。

Timestamps-based[2]方法是以时间戳与数值范围为XML文档版本中的每个结点进行编码,但没有讨论如何明确定义结点数值范围中的间距值range,也没有说明当新的XML文档版本加入时,例如新增、删除和修改XML文档,原来XML文档结点的数值范围应该如何调整。

文档版本整合[3]是指将XML文档的原始版本及其后续版本整合成一个文档,即多版本XML文档,并设计出新的numbering schema,给予XML树状结构上的每个结点两种范围的编码:空间上的数值范围(startpos:endpos)编码和时间上的生存周期编码(vstart:vend)。图1是一个带结点编码的多版本XML文档示例。

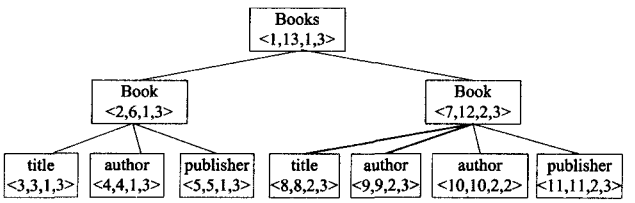

图1 多版本XML文档(版本1-3)以及编码方式

版本1只有books结点和左分支结点,这些结点一直到版本3依然保留,因此每个节点的(vstart:vend)都是(1:3),版本2增加了右边的book结点及4个子结点,因此这些结点的vstart是2,版本3仅删除了book结点的第3个子结点author,因此该author结点的vend是2,其他子结点的vend均为3。图2给出了结点编码信息。

新的Numbering Schema是区间编码的一种扩展,存在如下关系:

到稿日期:2010-05-18 返修日期:2010-09-15

丁 峥(1978—),女,讲师,主要研究方向为计算机应用、数据库技术等,E-mail:dingding_3046@sina.com.cn;周 虹(1958—),女,副教授,主要研究方向为计算机应用、数据库技术等。

(1)利用时间范围(vstart:vend)可撷取满足特定版本的结点。如需要撷取版本 V 的结点 a,只需满足 $a.\text{vstart}<V<a.\text{vend}$ 条件即可。

(2)利用数值范围可判断两个结点之间的结构关系。如给定的两结点 u,v,u 是 v 的祖先当且仅当 $u.\text{startpos}<v.\text{startpos}<u.\text{endpos}$。

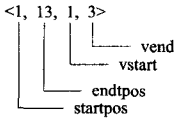

图 2 结点编码信息

以上文献研究的重点在于如何迅速地恢复版本,对于直接查询任意版本,尤其是结构查询,没有涉及。

2 支持多版本 XML 文档的索引机制

在新的 Numbering Schema 基础上,文献[4]首次将结构化连接的方法引入到多版本 XML 文档查询的领域,扩展了 Stack-Tree-Desc 算法,新的算法能在不恢复版本的前提下进行任意版本的结构查询。文献[4]有很重要的意义,但遗憾的是算法的扩展有逻辑错误。文献[5]纠正了错误,给出了正确的 Stack-Tree-Desc-MV 算法。文献[6]提出支持多版本 XML 文档的索引机制的构想,并在此基础上扩展出支持版本的算法 Anc_Desc_B+_MV。但文献[6]仅仅从理论上进行了探索,对于索引机制的结构阐述得并不清楚,算法也仅仅从理论上分析在索引的支持下,能比 Stack-Tree-Desc-MV 算法有更好的查询效能。本文将在之前研究的基础上,更进一步探讨 MVXIS(Multi_version XML Indexing Systerm)索引,并最终实现它。

MVXIS 支持根据元素名称和属性名称来查询值或者结构,为达成这一目标,MVXIS 提供了快速有效达成下列目标的机制:

(1)给定一个元素名称,以迅速找到所有版本具有此元素名称的结点;

(2)给定一个属性名称,以迅速找到所有版本具有此属性名称的结点。

由此可见,利用 MVXIS 处理 XML 版本问题时,能够不恢复版本直接进行查询,尤其是结构查询。MVXIS 的内部机制由 3 部分组成:值列表索引、元素索引、属性索引。

值列表索引:在一个 XML 文档中,不论是元素、属性或文本都是不定长的字符串,不能直接构建索引,需要把这些字符串先转化成一个特有的标志 Nid(Name Identifier),可以用 MD5 算法完成这个工作,MD5 算法是用在数字签名上的算法,我们在这里巧妙地利用该算法把任意字符串转化成不同的 128 位的二进制,用来做元素、属性或者文本的特定标志。

元素索引:元素索引文件的结构是一个特殊的二级 B+树结构(见图 3),在一级 B+树中,以元素的名称 Nid 为键值来建立 B+树,B+树的每一个叶子结点又指向一个二级 B+树,该 B+树是以元素结点的 Startpos 值为键值来建立的,第二级 B+树的叶子结点才真正指向该结点在文件中的位置。

给定一个元素名称,利用 MVXIS 能够快速找到所有具有该元素名称的结点,并且这些结点带有版本信息。具体步骤如下:

(1)根据元素名称去检索值列表索引文件,从中找到该元素名称对应的 Nid;

(2)用该元素的 Nid 去查找元素名称 B+树文件,利用元素 Nid 的 B+树,可以找到相应的叶子结点;

(3)利用该叶子结点的地址信息可以找到第二级 B+树的根结点。

(4)沿着根结点可以得到二级 B+树的第一个叶子结点,从而提取到所有叶子结点,也就是所有具有该元素名称的结点列表。

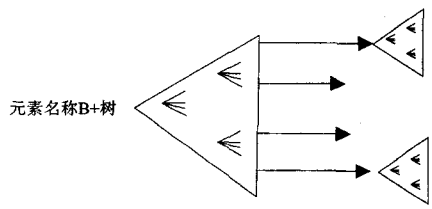

图 3 元素索引的组织结构

由此可见,利用元素索引文件能达成我们上面提出的第一个目标,而这一目标是元素结构连接的前提条件。例如:要完成结构化关系边(e_1,e_2)(父-子或祖先-后代关系)的连接,前提必须是提取出与结点谓词 e_1,e_2 相匹配的点集,即 AList=$\{a_1,a_2,\cdots,a_n\}$和 DList=$\{d_1,d_2,\cdots,d_m\}$。

第二级 B+树不仅仅在提取点集的时候起到作用,而且在进行查询的时候,可以将其读入内存,利用其跳过一些不参加连接的结点,避免了连接的冗余操作。这一点在文献[6]中有详细阐述,在此不再赘述。

属性索引文件的结构与元素索引文件类似。由于篇幅的关系就不赘述了。

3 查询实验及结果分析

XML 文档的结构查询通常被转化为两个结点列表之间的包含关系或文档位置关系的结构连接操作,同时关键字操作也被转化为两个结点列表之间的包含关系的结构连接操作。因此,有效的结构连接是 XML 查询实现的关键。目前,在结构连接方面已经进行了大量卓有成效的研究,提出了一系列有效的结构连接算法,但目前所有的结构化连接算法都不支持 XML 文件版本的查询,我们发现:多版本 XML 文档的版本 V 中,结点 A(startpos,endpos,vstart,vend)是结点 D(startpos,endpos,vstart,vend)的祖先的充要条件是:$A.\text{vstart}<V<A.\text{vend}$ and $D.\text{vstart}<V<D.\text{vend}$ and $A.\text{startpos}<D.\text{startpos}<A.\text{endpos}$。这是一个很重要的命题,也是所有研究的基础。正是在这个命题支持下,文献[5]扩展了 Stack-Tree-Desc 算法,文献[6]扩展了 Anc_Desc_B+算法,由于篇幅的关系,这里不再列出详细的算法过程,文献[5,6]中已有详细叙述。扩展后的算法能在不恢复版本的前提下,从一个多版本 XML 文件中找出某一具体版本的祖孙结点对,但它们仅仅从理论上分析了查询的过程,缺乏实验支持。在本节我们将扩展 Tree-Merge- Desc 算法、Stack-Tree-Desc 算法和 Anc_Desc_B+这 3 个经典的结构连接算法并用实验模

拟查询过程。

3.1 实验环境

完成实验的机器配置为：Inter(R) Pentium(R) 4 2.80 GHz,512M 内存。操作系统是 Windows XP,所用编程工具为 VC6.0。

设计的文档的 DTD 结构如图 4 所示，其中，标有“A”的边表示它所指向的是属性；标有“+”、“*”、“?”的边分别表示它所指向的元素在文档中可以重复出现一次或多次、零次或多次、零次或一次。在实际中也有此 DTD 结构的文档，因此实验数据具有现实合理性。例如：一本书具有若干章，每一章有若干个节，每一节又有若干个小节，因此，section 元素可能嵌套出现零次或多次。

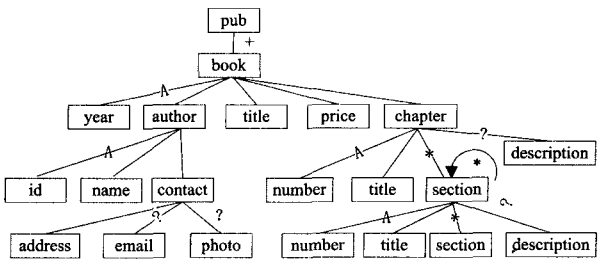

图 4 测试数据集的 DTD

3.2 实验框架以及主要实验步骤

实验的框架如图 5 所示，主要步骤如下：

(1)添加示例文件，版本 1 中有 10 个 book 结点及相应的子结点。

(2)将配置好的 XML 文档(版本 1-1)以及四元组结构存入我们设计的存储模式中。

(3)生成 MVXIS 索引文件(版本 1-1)。

(4)版本 $N+1$ 在版本 N 的基础上增加 10 个 book 结点和相应子结点，生成多版本文件(版本 1-5)。文档大小为 818k,在该多版本 XML 文档(版本 1-5)中，共有 50 个 book 元素结点，30200 个 title 元素结点，396 个 chapter 个元素结点，15150 个 section 元素结点等。

(5)调整索引文件(版本 1-5)。

(6)实现 3 个查询算法，利用 3 个算法对多版本文件(版本 1-5)查询，分别查询每个版本中的元素对之间的结构关系。

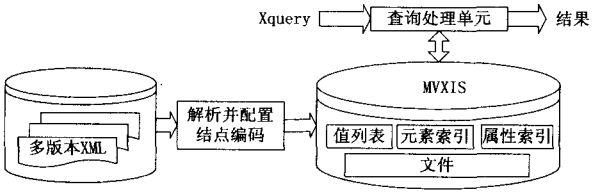

图 5 实验框架图

3.3 实验结果分析

实验是为了测试 3 个扩展算法对多版本 XML 文档的特定版本中的结点之间的结构查询。我们选择多版本 XML 文档(版本 1-5)中具有代表关系的测试实例，section /title。查询每个版本的这个实例的结构关系，选取两个性能指标：扫描元素的个数和查询时间。

扫描元素的个数即查询时读取父结点(section)和子结点(title)元素的数目，这个指标能反映算法跳过不相关元素的能力，如表 1 所列。

表 1 3 个算法扫描元素结点数量的对比(单位：个)

section	TMD-MV	STD-MV	ADB-MV
版本 1	15150	15150	13350
版本 2	15150	15150	11550
版本 3	15150	15150	9750
版本 4	15150	15150	7950
版本 5	15150	15150	6150

title	TMD-MV	STD-MV	ADB-MV
版本 1	90636	30320	28390
版本 2	90676	30320	26580
版本 3	90716	30320	24470
版本 4	90756	30320	22960
版本 5	90796	30320	21150

从分析看出：3 种算法扫描结点的数目不一样，横向比较，算法 ADB-MV 在每个版本的查询中，扫描的父结点和子结点的数目均比其他两种算法要少，因为 ADB-MV 能利用索引跳过冗余的结点，与我们的理论分析一致。

3 种算法针对查询实例的查询时间如表 2 和图 6 所示，可见算法 ADB-MV 效能最好。

表 2 3 个算法扫描时间对比(单位：毫秒)

SECTION/TITLE	TMD-MV	STD-MV	ADB-MV
版本 1	5500	2344	2172
版本 2	5500	2344	1984
版本 3	5500	2344	1781
版本 4	5500	2344	1594
版本 5	5500	2344	1406

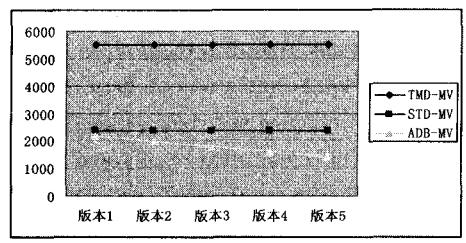

图 6 3 种算法查询时间比较

结束语 以往 XML 版本管理的研究重点在于如何快速恢复版本，而本文另辟蹊径，查询特定版本不必先恢复版本，直接进行结构关系的查询。不满足于一般的查询效率，本文设计出一种支持多版本 XML 文档的高效能的索引机制，并在此基础上扩展现有的一些经典的结构连接算法，扩展后的算法都能支持多版本 XML 文档的查询处理。实验证明，基于索引的结构连接算法比其他算法有更好的效能。

参 考 文 献

[1] Abiteboul M S, Cobena G, Mignet L. Change-centric management of versions in an XML warehouse[C]// Proc. of the 27th VLDB. Rome, Italy, 2001

[2] Chien S-Y, Tsotras V J, Zaniolo C, et al. Storing and Querying Multiversion XML Documents using Durable Node Numbers[C]// Proc. of WISE. 2001

[3] 蔡政霖. XML 文档版本管理[EB/OL]. http://datf.iis.sinica.edu.tw/Papers/2003datfpapers

[4] 贾玉昌，庞引明. 基于结构化联接的多版本 XML 文档查询处理[J]. 计算机工程，2005,41(36):172-174

[5] 丁峥，白云. 基于结构化连接接的多版本 XML 文档查询处理[J]. 苏州科技学院学报：自然科学版，2006,23(4):64-68

[6] 丁峥，孙涌，白云. 多版本 XML 文档的高效能索引机制的研究[J]. 计算机应用与软件，2008,25(6):131-132

[7] 朱庆生，戴刚. 基于 XML 的安全数据交换模型[J]. 重庆工学院学报：自然科学版，2009,23(6):36-39