文章编号:1001 - 9081(2005)12 - 2879 - 03

一种适用于顺序 XML 树的前缀编码方法

张剑妹,陶世群
(山西大学 计算机与信息技术学院,山西 太原 030006)
(jmzhmm@yahoo.com.cn)

摘 要:在对 XML 数据模型和 XML 查询语言中的顺序性进行分析的基础上,提出了一种用于顺序 XML 树的前缀编码方法,并从唯一性、确定性、动态性、灵活性和简洁性五个方面论证了这种编码的正确性和有效性;同时,运用分层编码的思想解决当 XML 文档规模增大时编码长度增加的问题。

关键词:XML 路经表达式查询;结构关系;区域编码方法;前缀编码方法
中图分类号: TP311.13 **文献标识码**:A

Prefix encoding scheme for ordered XML trees

ZHANG Jian-mei, TAO Shi-qun
(*College of Computer and Information Technology, Shanxi University, Taiyuan Shanxi* 030006, *China*)

Abstract: On the basis of analyzing order in XML data model and XML query language, a prefix encoding scheme for ordered XML trees was proposed, its efficiency and correctness was demonstrated from uniqueness, determinacy, dynamic, flexibility and conciseness. Meanwhile, the idea of layering encoding was used to solve the problem that the size of labels increased when the size of XML document augments.

Key words: XML path expression query; structural relationship; interval-based encoding scheme; prefix encoding scheme

0 引言

随着可扩展标记语言 XML 迅速成为 Web 上数据表示和数据交换的标准,XML 数据的查询变得越来越重要。与此同时,许多 XML 查询语言如 XPath[1]、XQuery[2]、XML-QL、XQL 和 Lorel 等应运而生,这些查询语言的共同特点是利用路径表达式来导航 XML 文档的查询。执行路径查询的主要技术是结构连接,即确定节点之间的结构关系(祖先后代关系或父子关系),而节点之间的结构关系的确定主要依赖于有效的节点编码方法。衡量一种节点编码方法性能好坏的主要技术指标如下:

1) 唯一性:在同一个 XML 文档中没有任何两个节点的编码相同。

2) 确定性:通过节点的编码能快速、唯一地确定两个节点之间的结构关系。

3) 动态性:当更新 XML 文件时不会引起 XML 节点的重新编码,至少能将重新编码的范围缩小到一个较小的范围内。

4) 灵活性:编码能支持各种 XPath/ XQuery 的查询功能。

5) 简洁性:编码的长度应尽可能小。

其中,唯一性和确定性是衡量某种编码方法正确与否的唯一标准,也就是说任何编码方法都必须具有唯一性和确定性,否则将失去对 XML 节点编码的基本意义。而编码动态性、灵活性和简洁性却是相互矛盾的,如果一种编码长度较小,那么这种编码就不可能完全支持 XML 文档的动态更新和在有序 XML 文档上的数据查询功能;反之亦然。到目前为止,在考虑节点顺序的情况下,没有任何一种节点编码方法能充分适应 XML 文档的动态更新。

1 XML 元素的编码方法

目前,XML 元素的编码方法有两种:区域编码方法[3,4]和前缀编码方法[5,6]。

区域编码方法采用深度优先遍历的方法给 XML 文档中的每个节点赋予一对整数值,使它们形成的区间包含其后代节点的编码范围,这样对节点间的结构关系的判断等价于区间包含关系的判断。最先使用区域编码确定树节点之间的结构关系的编码方法是 Dietz 的编码方法[7],它给每个节点赋予一个(pro,post)编码,pro 和 post 分别是节点的前序遍历值和后序遍历值。与文献[7]不同,文献[3]给每个节点赋予一个(start,end)编码,分别是深度优先遍历树时第一次访问到该节点时的顺序值和第二次访问到它时的顺序值。区域编码能有效地支持各种 XML 查询,但它不能适应 XML 文档动态更新。当在文档中插入和删除节点时将会导致文档树的重新遍历。尽管 XISS[4]采用预留空间的办法在一定程度上解决了动态更新的问题,但当插入的节点数超出预留空间时,仍然需要重新遍历文档树。

前缀编码保存了元素的路径信息,任何元素的编码是其后代元素编码的前缀,这样元素间祖先后代关系的确定就等价于前缀子串包含关系的判断。前缀编码方法从根节点开始对 XML 文档树中的每个节点的出边用一组无前缀的二进制串编码,每个节点编码都继承其父节点的编码作为其入边编码的前缀,根节点的编码为空[5]。为了保证编码的平均长度最小,文献[5]采用哈夫曼编码给每个节点的出边赋值,这样编码可以使后代节点最多的节点的编码最短,但却不能保存 XML 文档的顺序信息,也不能充分适应文档的动态更新。文献[6]对节点的出边的编码作了改进,对每个节点的第 1 个出边用“0”编码,对第二个出边用“10”编码,以此类推,对第 i 个节点用“$111^{i-1}0$”编码。这样既保证了每个元素的出边编码的无前缀性,又能适应 XML 文档的动态更新,但这种编码方法不能完全支持对顺序 XML 树的查询,而这类 XML 查询

收稿日期:2005 - 06 - 06;**修订日期**:2005 - 09 - 01

作者简介:张剑妹(1970 -),女,山西屯留人,讲师,博士研究生,主要研究方向:XML 数据库技术; 陶世群(1946 -)男,安徽人,教授,博士生导师,主要研究方向:数据库理论与技术.

在实际查询应用中出现的概率很高。文献[8]使用的 Dewey 编码本质上是一种基于整数的前缀编码方法,为了保存节点的顺序信息,这种节点编码方法使用“.”作为各层节点编码之间的分隔符,从而增加 XML 文档的存储量。

为了既支持所有的在顺序 XML 文档上的查询,又能适应 XML 文档的动态更新,本文提出了一种适用于顺序 XML 树的前缀编码方法。

2 XML 数据模型和查询语言的顺序性

2.1 XML 数据模型的顺序性

一个 XML 文档可以被看成是一个标签在节点上的树,树中的内部节点与元素或属性相对应,叶子节点和数据值(文本)相对应,边表示节点之间的嵌套关系。在 XML 数据模型中,元素必须按在 XML 文档中元素的顺序出现,例如在 XML 出版发行数据库中一本书的作者、章节等的先后顺序是不可颠倒的。

2.2 XML 查询语言的顺序性

XPtah 是一种导航 XML 查询的路径表达式语言,其路径表达式的基本语法是:

Path::=/Step1/Step2/…/Stepn

其中 XPath 中的 Step 定义为:

Step::=Axis::Node-test[Predicate*]

Axis 指定 XML 文档的导航方向,Xpath 支持十三个 Axis,其中的 following-sibling、preceding-sibling、following、preceding 四个 Axis 隐含着 XML 文档中元素的顺序信息;“Node-test”表示在 Axis 方向上要查询的节点名,如路径步 following-sibling::author 是选择当前节点之后的所有名为 author 的兄弟节点;使用 Xpath 还可以查询满足一定条件的节点,这些条件由“Predicate”指定,其中 position()=n 表示查询第 n 个节点,如 child::chapter[position()=2]查询当前节点的第二个名为 chapter 的孩子节点。XPath 还支持缩简形式的路径表达式,缩减路径表达式使用“//”表示祖先后代关系,“/”表示父子关系,用“[n]”表示 position()=n,如//book/author[3]查询所有 book 的第三作者。

XPath 是 XQuery 的子集,其顺序功能在 XQuery 中依然成立。此外,XQuery 还支持范围查询,例如://book/author[2 to 4]将返回所有 book 的第二、第三和第四作者。

无论是从 XML 数据模型看还是从 XML 查询语言上看,顺序性都是 XML 数据的主要特征。在 XPath 和 XQuery 中所有带 following-sibling、preceding-sibling、following、preceding 或 position()的查询,在查询执行时都要使用 XML 元素的顺序信息。

3 适用于顺序 XML 树的前缀编码

3.1 编码方法

在这种编码方法中,任何节点(根节点除外)的编码都由节点顺序码、节点的前缀码和节点顺序码的长度码三部分组成,各部分均用十进制编码,根节点的编码为空。每个节点的第一个孩子节点的顺序码为“1”,第二个孩子节点的顺序码为“2”,当节点的顺序码全为“9”时在顺序码后补加“0”,这样第九个孩子的顺序码为“90”,第十个孩子的顺序码为“91”依此类推。每个节点编码都继承其父节点的顺序码和前缀码作为其顺序码的前缀。为了便于从节点编码中提取节点顺序码,在每个节点编码的前面用固定长度的编码存储节点顺序码的长度。图 1 给出文档的编码示例(这里假定节点顺序码的长度码为一位十进制位,第一位为长度码)。

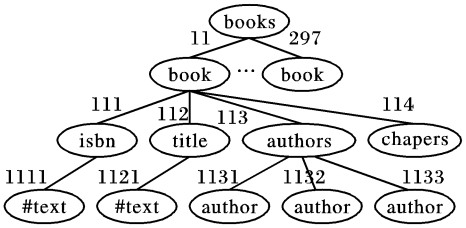

图 1 十进制前缀编码示例

下面两个定理说明了这种编码方法的正确性。下文中用到的基本符号如下:

$order(v)$ 表示节点 v 顺序码;

$precode(v)$ 表示节点 v 的前缀码;

$l(v)$ 表示节点顺序码的长度码,即在节点 v 的编码中节点顺序码 $order(v)$ 的位数;

$label(v)$ 表示节点 v 的编码,它由三部分组成,即:$label(v)=l(v).precode(v).order(v)$,其中“.”表示字符串连接符;

$Prefixes(label(v))$ 表示节点 v 从其祖先节点继承的所有前缀编码的集合,如图 1 的编码“1132”的前缀编码集 $Prefixes(1132)=\{1,13\}$;

$T=(r,V,E)$ 表示一个 XML 文档树;其中树的根节点为 r,V 是节点的集合,E 是节点间边的集合。

定理 1 给定一个 XML 文档树 $T=(r,V,E)$,T 中不存在任何两个节点的编码相同,即:

$(\forall u,v\in V)(u\neq v\rightarrow label(u)\neq label(v))$

证明:对 XML 树中的任何节点 u 的编码 $label(u)=l(u).precode(u).order(u)$,其中节点的顺序码 $order(u)$ 是集合$\{1,2,\cdots,8,90,91,\cdots,990,\cdots\}$的一个元素,由于$\{1,2,\cdots,8,90,91,\cdots,990,\cdots\}$的元素是一组无前缀的编码,根据节点的编码方法 $precode(u)$ 是节点 u 的父节点的编码(不包括长度码),由此可见 $precode(u).order(u)$ 是一种正确的前缀编码。因此,在一个 XML 文档树中不存在两个编码相同的节点。证毕。

定理 2 给定一个 XML 文档树 $T=(r,V,E)$,T 中任意两个节点 u,v,u 是 v 的祖先当且仅当 $precode(u).order(u)$ 是 $precode(v).order(v)$ 的一个前缀,即:

$(\forall u,v\in V)(u$ is an ancestor of $v\leftrightarrow precode(u).order(u)\in prefixes(label(v))$

这个定理的正确性时显然的,根据定理 1,任何节点 u 的 $precode(u).order(u)$ 都是一种前缀编码,因此可以通过判断 $precode(u).order(u)$ 和 $precode(v).order(v)$ 前缀子串的包含关系来确定两个节点之间的祖先后代关系。

3.2 编码对顺序 XML 文档查询的支持

和前缀编码相同,节点之间的结构关系可以通过节点编码(除去长度码后)的前缀子串的包含关系确定。此外,这种编码方法支持所有在顺序 XML 文档上的查询。为了便于分析,将 XPath 和 XQuery 的隐含元素顺序信息的查询分为两类,一类是有 following-sibling、preceding-sibling、following 或 preceding 的方向轴的查询,执行这类查询时只需要获得元素的相对顺序即可;另一类是使用 position()的 XML 查询(注意在缩减路径表达式中用[n]代替 position()=n),这类查询的执行需要获得元素在兄弟节点中的绝对顺序号。XML 文档树中任意两个节点相对顺序都可以通过如下方法确定:从长度码之后按数位一位一位地比较直到找到不相同的数码为

止,然后根据该数码的大小确定节点间的先后关系。例如:在图 1 所示文档中,如要确定“1132”和“114”的先后关系,则从第二位开始比较,当发现 3 <4 时,就可以得到编码位“1132”的节点必在编码位“114”的节点之前。

下面的定理给出了根据节点的编码和节点在兄弟节点中的顺序号之间的关系。

定理 3　给定 XML 文档树 $T=(r,V,E)$,$\forall u \in V$,节点 u 的编码 $label(u)=l(u).precode(u).order(u)$,$N(u)$ 是节点在兄弟节点中的顺序号,则 $N(u)=9\times(l(u)-1)+(order(u) \bmod 10)$。

证明:从 $order(u)$ 的编码规律可以发现,在编码“1,2,…,8,90,91,…,990,991,992,…”中,“90”是第九个编码,“990”是第十八个编码,“9990”是第二十七个编码……,编码位数每增长一位,编码的序号加 9;而这些编码模十得到的结果依次是“0,1,2,…,8”(除长度为 1 的编码外),因此 $N(u)=9\times(l(u)-1)+(order(u) \bmod 10)$ 成立。证毕。

从以上定理不难看出,对 XML 文档中的任何一个节点,首先根据其编码中长度码的值取其编码的后缀,这个后缀即为该节点顺序码,然后使用定理 3 便可得该节点在其兄弟节点中的顺序号。

3.3　编码的动态性分析

在这种编码方法中,节点的顺序码是一组不受长度限制的无前缀编码,因此,这种编码能有效地适应文档的动态更新。当插入一个新节点时,如果不考虑节点顺序,则直接按本文的编码规则给它赋予一个顺序码,而不需要修改任何原节点的编码;如果考虑节点的顺序,这时只需要修改插入节点之后的兄弟节点及其子孙的编码。由此可见,在不考虑节点顺序的情况,这种编码的动态插入代价和前缀编码的动态插入代价相同;当考虑元素顺序时,这种编码的动态插入代价要远远小于区域编码的动态插入代价。如图 2 所示,其中虚线表示的节点为新插入的节点,虚线框内为插入新节点后需重新编码的节点。

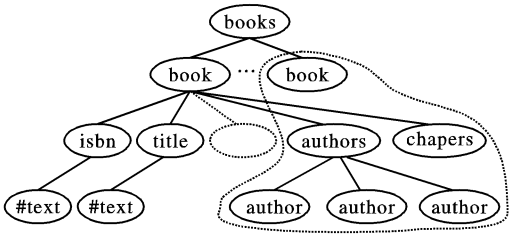

图 2　区域编码的插入

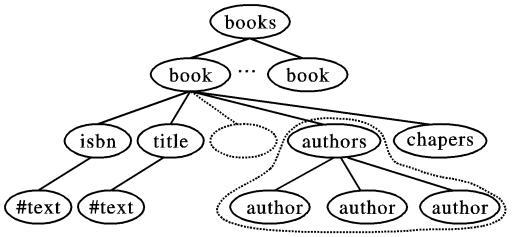

图 3　十进制前缀编码的插入

3.4　编码的长度分析及其改进方案

Dewey 编码[2]是一种既保存文档顺序又能适应文档动态更新的一种编码方法,但 Dewey 编码在各层节点编码之间使用“.”分隔符,这样增加了 XML 文档的存储代价。而本文的编码方法仅用固定的一到两位数码表示顺序码的长度,因此 XML 文档的存储代价远远小于 Dewey 编码。与前缀编码一样,随着节点扇出度的增加,编码长度将指数级增长,为此,采用了分层编码的思想[9]。同时注意到,在 XML 文档中,扇出度较大的节点往往是一些集合型的抽象实体(如图 1 中的 books),这一类节点的扇出度受 XML 文档规模的增加而增加,并且其子节点的顺序无关紧要。当文档规模很大时,这一类节点的扇出度可能会成千上万,这样将会导致前缀编码长度急剧增加。改进的编码方案如下:首先从具有集合类的抽象节点层将 XML 文档树分为两层,该节点为上一层树的叶节点,该节点的子节点成为下一层的若干子树的根节点,这样该 XML 文档树被分成若干个 XML 文档树。然后对上层子树和下层各子树采用本文的编码方法分别独立编码,但下层子树的根节点继承其公共最低祖先节点的编码作为前缀。采用这种分层编码方法可以避免当文档规模很大时,编码长度的急剧增加。

4　结语

设计 XML 数据编码方法的主要目标是在不遍历文档树的前提下唯一地确定节点之间的结构关系。为了既能有效支持对顺序的 XML 文档树的查询,又能充分适应 XML 文档的动态更新,本文提出了一种对顺序 XML 树的十进制前缀编码方法。在不考虑节点顺序的情况下,这种编码方法动态更新代价和前缀编码的动态更新代价相同;在考虑节点顺序的情况下,当有新节点插入时,该编码方法将需要重新编码的节点范围缩小到插入节点之后的兄弟节点及其子孙节点,从而大大减小了动态更新的代价。在编码长度方面,这种编码的长度远远小于既保存文档顺序又能适应文档动态更新的 Dewey 编码。通过与区域编码、后缀编码以及 Dewey 编码的比较,证明这种编码是有效的。但由于这种编码本质上仍然是一种前缀编码,因此其编码长度仍然受节点的扇出度影响。

前缀编码保存了节点的路径信息,今后我们将进一步探讨基于前缀编码的 XML 索引方法,以期有效地利用节点编码中的路径信息减小索引的大小。

参考文献:

[1] World Wide Web Consortium. XML Path Language(XPath) 1.0 [EB/OL]. http://www.w3.org/TR/xpath, 1999.

[2] World Wide Web Consortium. Xquery 1.0: An xml query language [EB/OL]. http://www.w3.org/TR/ xquery, 2001 -08.

[3] ZHANG C, NAUGHTON JF, DEWITT DJ, *et al*. On supporting containment queries in relational database management systems[A]. Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data[C]. Santa Barbara, California, USA, 2001.125 -436.

[4] LI QZ, MOON B. Indexing and querying XML data for regular path expressions[A]. Proceedings of the 27th International Conference on VLDB[C]. Rome, Italy, 2001.361 -370.

[5] KAPLAN H, MILO T, SHABO R. A Comparison of Labeling Schemes for Ancestor Queries[A]. Proceedings ACM-SIAM Symposium on Discrete Algorithms, 2002.

[6] COHEN E, KAPLAN H, MILO T. Labeling Dynamic XML Tree[A]. Proceedings of the 21st ACM Symposium on Principles of Database Systems[C]. Madison, Wisconsin, USA, 2002. 271 -281.

[7] DIETZ PE. Maintaining order in a linked list[A]. Proceeding of 14th Annual ACM Symposium on Theory of Computing[C]. San Francisco, California, May 1982.122 -127.

[8] TATARINOV L, VIGLAS SD, BEYER K, *et al*. Storing and Querying Ordered XML Using a Relational Database System[A]. Proceedings of SIGMOD[C]. 2002.

[9] ABITEBOUL S, KAPLAN H, MILO T. Compact Labeling Schemes for Ancestor Queries[A]. Proceedings ACM-SIAM Symposium on Discrete Algorithms[C]. 2001.