文章编号:1001 - 9081(2008)09 - 2331 - 04

一种基于二叉树的 Native XML 数据库文档编码机制

张 鹏[1], 冯建华[1], 房志峰[2]
(1. 清华大学 计算机科学与技术系,北京 100084; 2. 山东政法学院 信息科学技术系,济南 250014)
(zhangp@tsinghua.org.cn)

摘 要:在对于现有编码机制进行综述的前提下,提出一种新的 XML 文档编码机制,该编码机制基于完全二叉树的结构顺序编码。在该 XML 文档编码机制下,判断节点之间祖先-后裔关系算法的时间复杂度仅为 $O(\log n)$,完全支持更新,并且编码长度较短。

关键词:Native XML 数据库;编码机制;XML 查询;完全二叉树
中图分类号: TP311.13 **文献标志码**:A

New XML document coding scheme based on complete binary tree

ZHANG Peng[1], FENG Jian-hua[1], FANG Zhi-feng[2]
(1. *Department of Computer Science & Technology, Tsinghua University, Beijing* 100084, *China*;
2. *Department of Information Science & Technology, Shandong Institute of Political Science and Law, Jinan Shandong* 250014, *China*)

Abstract: In this paper, a new coding scheme was proposed, which was based on the sequence of its complete binary tree. The scheme is easy to realize and only one positive integer is needed to express the position of the node in XML tree. The time-bounding of identifying the ancestor-descendant relationships is only $O(\log n)$. It also supports XML document update. In the scheme, the length of the code is short.

Key words: Native XML database; coding scheme; XML query; complete binary tree

0 引言

XML 是一种专门为因特网所设计的标记语言,它的重点是管理信息的数据本身,数据的显示交给其他技术解决。随着近些年 Web Service 的蓬勃发展,XML 越来越多地活跃在数据交换和存储领域。

大量 XML 文档的出现促进了 Native XML 数据库的研究,Native XML 数据库已经成为当前数据库理论研究的前沿之一。Native XML 数据库以自然的方式处理 XML 数据,以 XML 文档作为基本的逻辑存储单位,针对 XML 的数据存储和查询特点专门设计适用的数据模型和处理方法[1]。由于查询是数据库最为频繁的操作,因此在 Native XML 数据库的研究中,XML 查询处理器是研究的热点。XML 查询处理器的主要功能在于查询分解,查询优化和查询执行等,主要目的在于查询求解。目前 Native XML 数据库的查询求解方法包括:基于 XML 文档索引或者结构索引的导航遍历算法;基于 XML 文档编码的结构连接算法;基于 XML 文档序列表示的序列匹配算法等。在上述方法中,利用节点编码的方法是主流技术之一。因此,XML 文档的节点编码机制是实现 Native XML 数据库高效查询的重要条件,对 XML 文档节点编码机制的研究具有非常重要的意义。本文对现有编码机制进行分析比较,并提出一种新的基于完全二叉树的顺序编码机制,简称 CBT(Complete Binary Tree Sequence)编码。

1 相关研究

XML 文档编码机制是指为 XML 文档树的每个节点赋予一个特定的编码,以便于能够通过编码直接判断 XML 文档树中的节点之间的祖先-后裔关系,而不需要对原 XML 文档进行导航遍历。依据编码的原理,可以分为基于区间的编码机制和基于路径的编码机制。基于区间的编码机制将每个节点编码为一个区间,而基于路径的编码机制则是依据节点的路径将每个节点编码为一个数字。

1.1 基于区间的编码机制

有关文档编码的早期研究多集中在区间编码法上。其主要思想在于为 XML 文档树中的每一个节点赋予一对正整数 begin 和 end,编码形式为[begin, end]。所有节点的编码区间都是其祖先节点的编码区间的子集。也就是说,节点 a 是节点 d 的祖先的充分必要条件是 a. begin≤d. begin 并且a. end≥d. end。

Li-Moon(XISS)编码机制[2]的主要思想在于:XML 文档树 T 中的每个节点被赋予二元组〈order,size〉,order 是节点的扩展先序遍历序号,size 是节点的后裔范围。因此,文档树中节点 a 和节点 d 存在祖先-后裔关系,当且仅当 a. order < d. order且 d. order + d. size≤a. order + a. size。另外,每个节点被赋予一个大于或者等于零的值 depth,其表示该节点在文档树中的层次。

Zhang 编码机制[3]的主要思想在于:XML 文档树 T 中的每个节点被赋予二元组〈begin, end〉,begin 是节点在 XML 文档中的开始位置,end 是节点在 XML 文档中的结束位置。因此,文档树中节点 a 和节点 d 存在祖先-后裔关系,当且仅当 a. begin < a. end 且 a. end > d. end。另外,每个节点被赋予一

收稿日期:2008 - 03 - 19;**修回日期**:2008 - 06 - 02。
基金项目:浙江自然科学基金资助项目(Y105230);清华大学基础研究基金资助项目(JCqn2005022)。
作者简介:张鹏(1981 -),男,山东莘县人,硕士研究生,主要研究方向:XML 数据库、移动 Ad Hoc 网络、移动 mesh 网络、移动通信; 冯建华(1967 -),男,山西万荣人,副教授,博士,主要研究方向:数据库、信息处理; 房志峰(1977 -),男,山东聊城人,讲师,硕士,主要研究方向:网络数据库。

个大于或者等于零的值depth,其表示该节点在文档树中的层次。

Wan(ApproXQL)编码机制[4]的主要思想在于:XML文档树T中的每个节点被赋予二元组〈order, maxOrder〉,order是节点的扩展先序遍历序号,maxOrder是节点的后裔编码中扩展先序遍历序号的最大值。因此,文档树中节点a和节点d存在祖先-后裔关系,当且仅当a.order < d.order且d.maxOrder≤a.maxOrder。

Dietz编码机制[5]的主要思想在于:XML文档树T中的每个节点被赋予二元组〈pre, post〉,pre是节点的先序遍历序号,post是节点的后序遍历序号。因此,文档树中节点a和节点d存在祖先-后裔关系,当且仅当a.pre < d.pre且d.post < a.post。

基于预留算法的编码机制[6]的主要思想在于:XML文档树T中的每个节点被赋予二元组〈order,size〉,order是节点的扩展先序遍历序号,size是预留的编码空间。基于数据模式和更新模式进行一定运算,从而获得预留的编码空间。

区间编码机制根据区间判断节点关系,其优点是编码简单,节点编码的平均长度为$O(\log(n))$;缺点是使用非等值比较进行节点关系判断,并且需要单独提供层次信息。

1.2 位向量编码机制

位向量编码机制[7]的主要思想在于:XML文档树T中的每个节点与一个n位向量v对应,n是树T中节点的数目。在向量的某个位置i上值"1"惟一标识了第i个节点,并且每一个后裔节点继承了其祖先的所有"1"位。若节点a是d的祖先,当且仅当a.v & d.v = a.v;若节点d是a的后裔,当且仅当d.v | a.v = d.v。位向量编码将节点与n位向量对应,其优点是容易判断节点间的层次关系;缺点是不支持更新操作,编码空间较大。

1.3 PBiTree 编码机制

PBiTree编码机制[8]的主要思想在于:将XML文档树T转化为完全二叉树,然后按照"自底向上,自左至右"的顺序为每个节点进行编号。PBiTree编码按照节点在文档树中的位置进行编码,其优点在于编码简单,节点编码的平均长度为$O(n)$;并且使用等值比较进行节点关系判断。缺点是不支持更新操作,需要单独提供层次信息。

1.4 Dewey 编码机制(前缀编码机制)

Dewey编码机制[9]的主要思想在于:将XML文档树中父节点的编码直接作为其子节点编码的前缀,也称为前缀编码。因此,文档树中节点a和节点d存在祖先-后裔关系,当且仅当a.code是d.code的前缀。Dewey编码将XML文档树中父节点的编码直接作为其子节点编码的前缀,其优点是编码简单,节点编码的平均长度为$O(n)$;支持更新操作。缺点是前缀操作比算术操作运算速度慢,因此该编码方法效率较低。

1.5 素数编码机制

素数编码机制[10]的主要思想在于:在自底向上素数编码方法中,将每个XML文档树的叶节点按顺序赋予不同素数,将父节点编码为子节点编码的乘积;在自顶向下素数编码方法中,将根节点编码为1,将每个节点编码为来自父亲的"父标记(parent-label)"和来自素数列的"个人标记(self-label)"的乘积。另外,当文档树过高的时候,可以对树进行"分解(tree decomposition)"。在自底向上素数编码方法中,文档树中节点a和节点d存在祖先-后裔关系,当且仅当(a.code) mod (d.code) = =0;在自顶向下素数编码方法中,文档树中节点a和节点d存在祖先-后裔关系,当且仅当(d.code) mod (a.code) = =0。素数编码采用整除运算判断祖先-后裔关系,其突出的优点是较好地支持更新操作,并且对于XML树的扇出不敏感;整除判断运算速度较快。其缺点是编码长度较大。

1.6 BBT 编码机制

BBT编码机制[11]的主要思想在于:将根节点编码为1,将某节点的最左儿子节点编码为其父节点编码的2倍,其他儿子节点的编码为其最近左兄弟节点编码乘以2再加1。该文献还提出了支持更新的BBT编码方法,其主要思想在于:提供额外的信息来表示一个节点在其兄弟节点中的位置序号,而不能利用BBT编码本身所包含的兄弟节点的位置信息。另外,为了节省存储空间,BBT编码可以采用分块存储的方法。BBT编码机制的优点是祖先-后裔关系的判断采用计算机常用的移位操作,具有很高的运算速度;较好地支持更新操作,仅在删除非最左叶节点或者添加节点到非最左位置时会导致同一层兄弟节点之间的顺序错误,为了更好地支持更新,可以增加分量记录节点在其所在层中的顺序。其存在的问题在于编码长度较长,对于节点数为n的文档树而言,BBT编码的长度为$O(n)$。

综合上述分析,区间编码机制和位向量编码机制对于更新操作支持较差,运算效率较低。另外,一项理想的编码机制应当满足如下技术指标:可确定性、动态性和简洁性。可确定性指的是,节点之间的祖先-后裔关系可以唯一地、快速地得到确定。动态性指的是该编码机制支持文档树的编码更新,并且更新操作不需要对XML文档树重新编码。简洁性指的是编码应当尽量简短,以便于装入内存。综合考虑上述技术指标,因为数据库查询是数据库最为频繁的操作,所以上述第一项技术指标最为重要。节点之间的祖先-后裔关系唯一确定是数据库查询正确性的要求,必须得以满足。在满足上述要求的前提下,节点之间的祖先-后裔关系可以快速查询是数据库查询效率的要求,确定节点关系的代价最好为$O(\log n)$。当XML文档树插入节点或者删除节点时,尽量少地改动节点的编码。另外编码简短也有利于查询效率的提高。

2 基于完全二叉树的顺序编码机制

鉴于现有编码机制存在的上述缺陷,本文提出一种基于完全二叉树的顺序编码机制,并简称为CBT编码机制,该机制具有如下优点:1)编码形式简单,只需要一个正整数即可充分表示节点在XML文档树中的位置信息;2)可以实现祖先后裔关系的快速查询,时间复杂度约为$O(\log n)$;3)支持XML文档的更新操作;4)编码长度较短。

2.1 CBT 编码原理

CBT编码机制的原理是,将XML文档树转化为二叉树T,然后给树T上的每个节点赋予一个大于等于1的正整数order,节点node的编码形式为〈order〉,它能完全表示节点node在树T中的位置信息。

具体编码步骤如下。

1)采用左子女-右兄弟表示法,将XML文档树T转换为二叉树BT。将T转换为BT的规则为(此处的R、A和D都是文档树T中的节点):

如果 R∈T,且 R 是 T 的根节点,则 BT 的根节点也是 R。

A,D∈T,如果 D 是 A 的第一个儿子,则在 BT 中 A 的左儿子就是 D。

A,D∈T,如果 D 是 A 的直接右兄弟,则在 BT 中 A 的右儿子就是 D。

2)将转换所得的二叉树 BT 的根节点编码为 0。

3)将转换所得的二叉树 BT 的其他节点按照完全二叉树广度优先遍历的编号依次编码,需要强调的是,对于二叉树 BT 的空节点也要预留节点编号。编码步骤的示例如附图 1 所示。

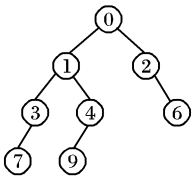

图 1　CBT 编码示例

在图 1 所示的示例中,编码为“2”的节点虽然没有左子女,但是仍然将其编号加以预留,从而其右子女的编码为“6”。同样,编码为“3”的节点虽然没有右子女,但是其编号依然预留,从而编码为“4”的节点的左子女编码为 9。

根据上述编码过程,可以将 CBT 编码机制用下述的算法 1 实现。其中,主要采用队列的方式对完全二叉树中的节点进行顺序编码。

算法 1

```
Coding_XML_Tree(N){
  // 输入: XML 文档树的节点 N
  // 输出: 节点 N 的编码 order
  order = 0;
  if(current != NULL){                    // current 表示当前指针
    p = current; Qu.EnQueue(current);
                                          // 保存当前指针,根指针入队
    while(!Qu.IsEmpty()){
      current = Qu.DeQueue();
      if(当前节点是节点 N){将该节点编码为 order; break;}
      if(LeftChild! = NULL)
        LeftChild();                      // 待访问节点的子女节点入队
      else
        将一个标识符入队;                  // 当没有子女时,入队标识符
      while(current! = NULL){ Qu.EnQueue(current); order++;
      NextSibling();}
    }
  }
  else
    current = p;                                        // p 为树节点
  return order;
}
```

2.2　CBT 编码的性质分析

CBT 编码采用完全二叉树作为数学模型,从根节点开始按照广度优先遍历的顺序从 0 开始编号。CBT 编码存在下述性质(n 为某节点编码)。

性质 1　$Parent(n) = \lfloor (n-1)/2 \rfloor, n \neq 0$。

性质 2　$Leftchild(n) = 2n+1, 2n+1 < n$。

性质 3　$Rightchild(n) = 2n+2, 2n+2 < n$。

性质 4　$Level(n) = \lfloor \mathrm{lb}(n+1) \rfloor$。

其中,性质 1 用以求出编码为 n 的节点的父节点编码;性质 2 用以求出编码为 n 的节点的左子女节点编码;性质 3 用以求出编码为 n 的节点的右子女节点编码。利用这些性质,可以将一棵完全二叉树线性存储,从而实现高效地查询。

2.3　CBT 编码中节点关系判断

在 CBT 编码下,判断节点 a 和节点 d 之间是否存在祖先-后裔关系的算法如下(其中 a.order 和 d.order 分别表示节点 a 和 d 的编码)。

1)如果 a.order > d.order,那么节点 a 不是节点 d 的祖先。

2)如果 a.order < d.order,那么求出节点 d 的父节点的编码 ParentOrder(d):

(1)如果 a.order = ParentOrder(d),那么节点 a 是节点 d 的祖先;

(2)如果 a.order > ParentOrder(d),那么节点 a 不是节点 d 的祖先;

(3)如果 a.order < ParentOrder(d),那么节点 d 指向节点 d 的父节点,然后循环步骤 2)。

在附图 1 所示的示例中,以节点对〈1,7〉为例,a.order = 1,d.order = 7,首先判断得出节点的编码 a.order < d.order,因此求出节点 d 的父节点 d′的编码为 3;然后判断得出节点的编码 a.order < d′.order,因此求出节点 d′的父节点 d″的编码为 1;然后判断得出节点的编码 a.order = d″.order,从而得出节点 a 和节点 d 之间存在祖先-后裔关系。以节点对〈1,6〉为例,a.order = 1,d.order = 6,首先判断得出节点的编码 a.order < d.order,因此求出节点 d 的父节点 d′的编码为 3;然后判断得出节点的编码 a.order > d′.order,从而得出节点 a 和节点 d 之间不存在祖先—后裔关系。

2.4　CBT 编码对文档更新的支持

分析 CBT 编码可知,CBT 编码是以节点位置作为自变量的函数,某个节点的 CBT 编码仅仅与该节点在二叉树位置有关。因此,某位置上的节点的编码具有唯一性和确定性,不受其他节点的影响。因此,当插入新节点或者删除旧节点的时候,仅仅需要对新的节点进行编码,完全无需修改其他节点的编码。因此,CBT 编码对于文档更新具有良好的支持。

3　性能分析

本部分主要采用标准的 Sharks 数据对 CBT 编码进行性能评估,探讨 CBT 编码机制与其他编码机制的性能比较,在对 XML 文档更新的支持程度方面 CBT 编码机制与其他编码机制的比较。实验中的 Sharks 数据采用 Shakespeare2.00 标准[12],执行实验程序的计算机的 CPU 是 Intel Core 2,内存为 512 MB,运行的操作系统是 Windows XP Professional,采用标准 C++程序设计语言编写实验程序。

3.1　时空性能比较

相对于区间编码机制而言,CBT 编码机制有较好的空间性能,这主要是因为区间编码机制需要使用前后两个端点表示一个区间。相对于 BBT 编码机制而言,CBT 编码机制也有较好的空间性能,这主要是因为 CBT 编码机制的编码长度比 BBT 编码机制更短。图 2 显示了对 Sharks 数据中的 hen_v.xml, john.xml, lear.xml, m_wives.xml 分别进行编码后,结果编码所需要的存储空间。从图 2 可以看出,就存储空间的要求而言,CBT 编码机制相对于其他编码机制有一定提高。

CBT 编码机制比区间编码机制时间性能更好,这主要是因为区间编码机制至少需要扫描两遍 XML 文档树而 CBT 编码机制只需要扫描一遍。CBT 编码机制与 BBT 编码机制时间

性能相当,由于CBT编码机制编码过程中需要从队列存取数据,时间性能比BBT编码机制略差。

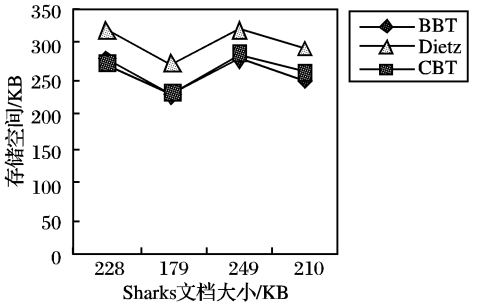

图2 对Sharks文档进行编码所需要的存储空间比较

3.2 更新支持比较

表1显示了在结点的插入和删除操作中,CBT编码机制和以Dietz编码机制为代表的区间编码机制需要重新编码的结点的数目的比较。

表1 更新支持性能比较

结点总数	Dietz算法需要更新的结点数量	CBT算法需要更新的结点数量
24009	20001	12
46094	44101	51
69991	69897	81
94463	92263	115

从表1可以看出,在以Dietz编码机制为代表的区间编码机制下,当XML文档树中插入或者删除结点时,需要重新编码的结点的数目接近于XML文档树结点的总数。而CBT编码机制下,当XML文档树中插入或者删除结点时,需要重新编码的结点的数目微乎其微。因此,CBT编码机制对XML文档更新的支持程度非常好。

4 结语

本文集中讨论了Native XML数据库的文档编码机制,提出了一种新的文档编码机制——基于完全二叉树的顺序编码机制(CBT)。理论分析和实验结果表明,将CBT编码方案和其他编码方案相比较,在上述第三部分提出的三项技术指标上都存在较大优势。就可确定性而言,节点之间的祖先-后裔关系完全可以唯一确定,并且判断节点之间的祖先-后裔关系的时间复杂度为$O(\log n)$。就动态性而言,CBT编码对于节点的插入和删除具有很好的支持,完全无需修改其他节点的编码。就简洁性而言,CBT编码与节点数目呈线性关系。另外,CBT编码简单易行,便于编码实现。这是在Native XML数据库文档编码机制上进行的有效尝试。

在本文的基础上,在Native XML数据库的文档编码机制的研究方面可以进一步展开其他研究工作。例如,本文提出的编码机制需要多次存取队列,如何减少由此带来的访问代价;本文提出的编码机制需要将Native XML数据库的XML文档树转化为二叉树,如何减少由此带来的时间代价等。这些都是值得进一步研究的问题。

参考文献:

[1] 冯建华,钱乾,廖雨果,等. 纯XML数据库研究综述[J]. 计算机应用研究,2006,23(6):1-7.

[2] LI QUAN-ZHONG, MOON B. Indexing and querying XML data for regular path expressions[C]// Proceedings of the 27th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 2001: 361-370.

[3] ZHANG C, NAAGHTON J, De WITT D, *et al*. On supporting containment queries in relational database management systems[C]// ACM SIGMOD Record. New York: ACM Press, 2001, 30(2): 425-436.

[4] WANG CHANG-XUAN, LIU YUN-SHENG. Efficient supporting XML query and keyword search in relational database systems[C]// Proceedings of the 3rd International Conference on Web-Age Information Management. London: Springer-Verlag, 2002: 1-12.

[5] DIETZ P F. Maintaining order in a linked list[C]// Proceedings of the 14th Annual ACM Symposium on Theory of Computing. San Francisco: ACM Press, 1982: 122-127.

[6] 罗道锋,孟小峰,蒋瑜. XML数据扩展前序编码的更新方法[J]. 软件学报,2005, 16(5): 810-818.

[7] TATARINOD L, VIGLAS S D, BEYER K, *et al*. Storing and querying ordered XML using a relational database system[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2002: 204-215.

[8] WANG WEI, JIANG HAI-FENG, LU HONG-JUN, *et al*. PBiTree coding and efficient processing of containment joins[C]// Proceedings of the 19th International Conference on Data Engineering. Bangalore: [s.n.], 2003: 391.

[9] WIRTH N. Type extentions [J]. Acm transaction on programming languages and systems, 1988, 10(2): 204-214.

[10] WU XIAO-DONG, LEE M L, HSU W. A prime number labeling scheme for dynamic ordered XML Trees[C]// Proceedings of the 20th International Conference on Data Engineering. Washington, DC: IEEE Computer Society, 2004: 66.

[11] 冯建华. 纯XML数据库的查询求解关键问题研究[D]. 北京:清华大学,2006.

[12] Shakespeare2.00标准[EB/OL]. [2008-01-01]. http://www.xml.com/pub/r/396.

(上接第2330页)

参考文献:

[1] HALKIDI M. On clustering validation techniques [J]. Journal of Information System, 2001, 17(2): 107-145.

[2] WILLE R. Restructuring lattice theory: An approach based on hierarchies of concepts [M]. Dordrecht-Boston: Reidel, 1982: 445-470.

[3] 史忠植. 知识发现[M]. 北京:清华大学出版社,2002.

[4] 宫秀军,史忠植. 基于Bayes语义模型的半监督Web挖掘[J]. 软件学报,2002, 13(8):134-140.

[5] 刁倩,王永成,张惠惠. 基于神经网络的中文信息概念联想构造算法[J]. 情报学报, 2000, 12(6):183-188.

[6] KARYPIS G. cluto2.0 clustering toolkit [EB/OL]. [2008-03-01]. http://glaros.dtc.umn.edu/gkhome/views/cluto.

[7] VALTCHEV P, MISSAOUI R, LEBRUN P. A partion based approach towards constructing Galois (concept) lattices [J]. Discrete Mathematics, 2002, 256(3): 801-829.

[8] VALTCHEV P, MISSAOUI R. Building concept (Galois) lattices from parts: Generalizing the incremental methods [C]// Proceedings of the ICCS'01, LNCS 2120. Berlin: Springer-Verlag, 2001: 290-303.