CSBTT:一种基于二叉树遍历的 XML 文档编码模式①

万里勇[1,2], 陈 颖[1]

[1](南昌工学院 信息学院, 南昌 330108)
[2](中南大学 信息科学与工程学院, 长沙 410075)

摘 要: XML 文档数据编码模式是 XML 文档查询处理的基础, 好的文档编码模式有利于提高文档的查询效率. 为了解决 XML 数据查询效率低、支持动态更新等问题. 本文在二叉树遍历的编码基础上, 引入二叉树的三叉链表存储结构对 XML 文档结点进行编码. 该编码利用自然数作为编码序号, 因此编码长度较短; 引入结点双亲指针, 方便结点之间结构关系的判定, 结点采用三叉树链式存储, 方便文档的更新操作.

关键词: XML 文档; 编码; 二叉树; 三叉链表; 更新

CSBTT: An XML Document Coding Schema Based on Binary Tree Traversal

WAN Li-Yong[1,2], CHEN Ying[1]

[1](School of Information, Nanchang Institute of Science & Technology, Nanchang 330108, China)
[2](Information Science and Engineering, Central South Unvisuarty, Changsha 410075, China)

Abstract: XML document data encoding scheme is foundation of XML document query processing. A good document encoding mode can improve the efficiency of XML document query. In order to resolve the inefficiency for XML data query and support dynamic updates, etc, this paper has proposed an improved method to encode XML document nodes. On the basis of the binary traversal, it introduces the trigeminal linked list storage structure of binary tree for encoding. It takes natural number as the serial number of the node encoding, so the encoding length is shorter. Besides, node parent pointer is used for encoding, which will facilitate to determinate the relationship between element nodes. Due to trigeminal linked list storage of nodes, the operation of the document update is more convenient

Key words: XML document; coding; binary tree; trigeminal linked list; update

近年来众多的数据索引和查询技术, 都依赖于对 XML 树结点的编码方法. 为了支持和加快数据的查询处理, 人们为此提出了许多编码方案. 常见编码方案主要有: 区域编码、前缀编码、K 分树编码、支持动态更新的编码和二叉树编码等等. 区间编码(如: Zhang 编码、Wan 编码、Dietz 编码等)简单,在针对结果关系查询方面表现出色, 缺点就是不支持更新. 当涉及 XML 频繁更新时, 需要对大量的结点重新编码, 这样增加了查询工作量, 降低了查询的效率. 前缀编码简单且支持更新操作, 由于它运用前缀操作, 其运算速度比算术运算效率要低且编码占用的空间也比较大.

结合 XML 文档的结构特点, 文献[1]提出了一个 PBiTree 编码: 将 XML 文档 T 转化为完全二叉树, 利用“自底向上, 自左向右”顺序对结点编码, 该编码简单, 但不支持数据更新且需要提供单独的层次信息. 文献[2]提出了二叉树遍历的 XML 文档编码模式, 文中阐述该方案较好解决了树遍历的 XML 编码更新和节点关系判断方面存在的不足. 文献[3]提出了一种基于完全二叉树的结构顺序编码,该编码完全支持数据更新且编码长度较短, 判断结点间关系的算法的时间复杂度为 O(log n). 文献[4]提出数据更新编码机制, 利用二叉树中序遍历定义整数新的序关系, 这种做法是重新编排自然数序列将静态编码转化为动态编码.

上述各种编码虽然具有较高的静态编码与查询效

① 基金项目:新世纪优秀人才支持计划(NCET-10-0787);江西省教育厅教学教育改革项目(JXJG-11-88-4)
收稿时间:2012-06-27;收到修改稿时间:2012-08-19

率, 但是有的不适合动态的更新操作, 或者结点的存储空间较大, 或者结点间关系判定不方便等等的问题. 基于此考虑, 本文提出了一种基于二叉树遍历的编码(Coding Schema Based on Binary Tree Traversal, 简称 CSBTT 编码). 它利用三叉链表存储结构对 XML 文档进行编码, 利用自然数作为编码序号, 因此编码长度短和存储空间小特点; 引入双亲结点指针, 方便结点间结构关系的判定, 另外该编码还支持动态更新操作.

1 CSBTT编码方案

XML 编码方案的优劣一般可以从三个方面进行判断[5]: (1)对 XML 数据索引和查询的支持; (2)适应编码更新; (3)编码长度.

1.1 相关的定义

二叉树的存储一般采用顺序和链式存储. 顺序存储就是将二叉树的结点按存储单元依次存储存一维数组中, 这种存储结构比较占用存储空间. 另外一种就是链式结构, 常见的有二叉链式和三叉链式.

XML 文档数据模型通常采用树型结构, 树中的每个节点对应着一个元素, 树中各边则描述了元素之间的关系, 树中各节点是有序的. 在对 XML 文档进行编码的时候, 一般将 XML 文档转化为相对应的 XML 文档树来操作. 图 1 是一个 XML 文档, 图 2 是文档对应的文档结构树(椭圆为元素结点, 矩形为文本结点, 结点间的连线代表着结点之间某种关系).

```
<? xml version =  “1.0” encoding=  “ISO-8859-1” ?>
<dblp>
  <paper ID=  “001”>
    <title>XML</title>
    <author>Mike</author>
  </paper>
  <paper ID=  “002”>
    <title>DB</title>
    <author>John</author>
    <price>20</price>
  </paper>
</dblp>
```

图 1　XML 文档

1.2 XML 编码原理

CSBTT 编码首先将 XML 文档转化成文档树, 然后将文档树转化为二叉树, 再由上自下, 从左到右的次序给树中各结点编号 n(n∈自然数), n 为该结点在树 T 中的信息位置. 借助三叉链表特性, 可以通过双亲指针快速判定结构树中任意结点之间的结构关系.

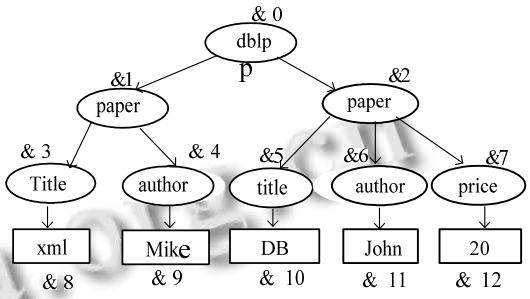

图 2　XML 文档树

1.3 编码过程

1.3.1 将 XML 文档树进行二叉化

将给定的 XML 文档转化为 XML 文档树, 使 XML 文档转化为树的结构; 再对 XML 文档树进行二叉化. 其转化思想采用左孩子-右兄弟的方法. 图 2 的 XML 文档树转化后的二叉树如图 3 所示.

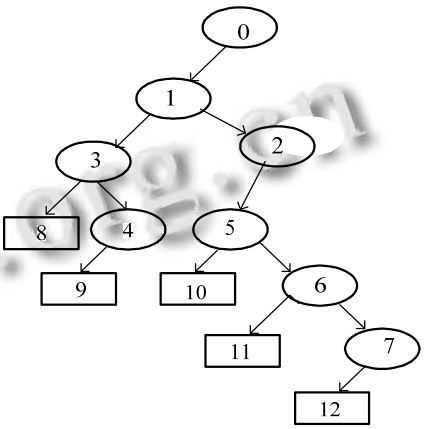

图 3　转换后的二叉树

XML 文档树本身的深度和宽度较大的时候, 转化过程中, 得到的二叉树的深度会非常的大. 对于这种情况, 做法是将生成的二叉树按照一定的规则, 对二叉树进行划断生成相应的多棵二叉树, 且将对应二叉树的根结点统一存放在单链表中或统一的数组中.

1.3.2 二叉树三叉链表存储结构

三叉链表结构中每个结点包含三个指针域, 分别为左孩子指针、右孩子指针和双亲指针, 如图 4 所示.

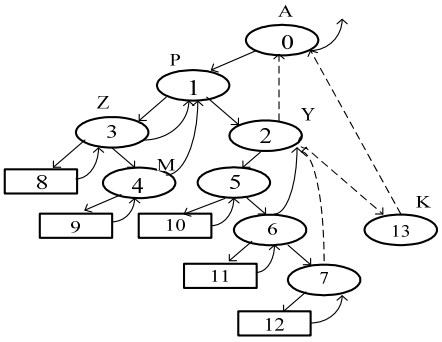

图 4 三叉链表结构图

2 更新操作及实验性能分析

转化后的文档树转变成二叉树, 此时 XML 文档树结点查询和删除操作都是基于对二叉树遍历操作, 其时间和空间复杂度都为 O(n).

2.1 编码的更新操作

2.1.1 插入结点操作

如果 XML 文档需要增加元素时, 则对应文档树中也需增加结点. 新增加的结点是某结点的子结点. 为了避免其他结点的编码修改, 在处理插入子结点插入操作时, 应该考虑将子结点插入文档树中的最后子结点的右子树中[2], 图 4 中的 K 结点即为新插入的结点.

算法 1(查找算法)

输入: 二叉树 T;

输出: 结点 p;

```
BiTree * FindNode(BiTree T)
{   BiTNode   *p;
   if(p= =NULL)   return NULL;
   if (p->info= =x->info)      return p;
      p = FindNode(p->lchild);    //左子树
   if (p = =NULL)return FindNode(p->rchild); //右子树
   else   return p;
}
```

算法 2(插入算法)

输入: 二叉树 T 要插入位置 p 和待插入的结点 y;

输出: 最终的二叉树;

```
Ins (BiTree T)
{   BiTNode   *p ;
   int   n;   //n 为结点的个数
   p=FindNode(T);   //找到待插点位置
   p->rchild=y;
   y->parent=p->parent ;//建立双亲指针
   y->code=n+1; //为新插入的结点编号
}
```

图 4 所示, 假设待插结点为 K, 根据算法描述, 要插入 Y 位置处, 则具体操作为: 将 Y 的右孩子指针指向 K, K 的双亲指针指向 A.

2.1.2 删除结点操作

对于二叉树中结点删除操作, 分两种情况: 删除叶子结点和非叶子结点. 对于叶子结点, 只要将被删结点的双亲结点右孩子指针指向空即可删除; 对于非叶子结点, 只要将被删结点的双亲结点左孩子指针指向被删结点的右孩子.

算法 3(删除算法)

输入: 二叉树 T 和要删除的结点 p ;

输出: 最终的二叉树 ;

```
Del(BiTree T)
{ BiTNode   *p, *q;
p=FindNode(T);    //查找所需删除的结点
if (p==NULL)    //文档树中不存在要删除的结点
   return NULL;
else if (p->rchild= =NULL)        //叶子结点
{ q=p->parent;
   q->rchild=NUL;
   }
else
   p->parent->lchild=p->rchild;   //非叶子结点
   dispose(p) ;      // 回收空间
}
```

图 4 所示, 假定要删除 Z 结点(非叶子元素结点), 只要将 P 的左孩子指针指向 M 结点; 如果要删除 M 结点(叶子元素结点), 则直接修改 Z 右孩子指针让其指向空, 即可删除 M 结点; 为了不影响二叉树其他结点编码, 在删除结点同时还要删除其所有的子结点, 以免产生冗余而浪费空间, 而删除后的结点称为虚结点.

2.2 结点关系判断

命题 1(兄弟关系和父子关系). 设 Vi , Vj 和 Vk 分别为二叉树中的任意的结点, 如果存在 Vi = Vk ->lchild 和 Vj = Vk ->rchild , 那么 Vi 和 Vj 互为兄弟, 同时 Vk 和 Vi (或 Vk 和 Vj)是父子关系.

命题 2(祖先-子孙关系). 设 Vi , Vj 和 Vk 分别为

二叉树中任意的结点，如果存在 Vk->parent=Vj 且 Vj->parent=Vi(或者 Vk->parent->…parent= Vi)，那么 Vi 和 Vk 是祖先与子孙关系.

命题 3(辈差计算). 假定 Vi 和 Vj 为二叉化后结构树的两个结点, Vi ,Vj , 如果 Vi …Vj 路径，那么 Vj 是 Vi 子孙，同时 Vj 到 Vi 路径长度即为辈差.

图 4 中, Z 和 M 都存在指向 P 结点的双亲指针，故 Z 和 M 是兄弟；M 双亲指针指向 P 结点，所以 P 与 M 是父子关系; M 双亲指针指向 P 结点，而 P 双亲指针又指向 K，所以 M 与 K 为祖先与子孙关系.

2.3 编码性能及实验数据分析

2.3.1 编码长度及关系判定方面分析

由文献[2,3]可知其他几种的编码长度为 O(N) 或 O(logN), CSBTT 编码用自然数作为编码，故长度较短. 结点间结构关系方面，前缀编码是通过前缀操作来判定; PBiTree 编码通过等值运算和附加位运算等结合来判定；而 CSBTT 编码则可通过双亲指针的回溯可判断任意结点间的关系(见命题 1, 2)，而且还容易计算出结点辈差(见命题 3). 具体信息见表 1 所示.

表 1　编码长度与结构关系的比较

编码方案	编码长度	祖先/后裔的检测
前缀编码	O(N)	前缀操作
PbiTree 编码	O(N)	两个非等值操作和附加位运算
Zhang 编码	O(logN)	两个非等值操作
CSBTT 编码	O(N)	等值运算操作容易计算辈数差

2.3.2 结点更新实验分析

(1) 实验环境及数据集：数据集为：DBLP1，大小为 131MB 左右；硬件：CPU-Intel Pentium Dual 2.0G, RAM 为 2G；系统：windows XP Professional；开发工具: JDK1.6 和 MyEclipse 7.0.

(2) 利用工具 XMLGen 将数据集生成六个文档：d1, d2, d3, d4, d5 和 d6，其对应的结点个数分别为(单位：个): 725, 14318, 22430, 43968, 132455, 267590.

图 5 显示 XML 文档结点数越多，需二次编码的结点编码率反而低, CSBTT 编码较好支持文档更新操作.

2.3.3 时空性能分析

(1)空间性能方面：与区间编码相比较，本文的 CSBTT 编码具有良好的空间性能， CSBTT 编码用自然数作为编码序号，而区间编码使用一对整数对表示.

(2)时间性能方面：和区间编码相比，CSBTT 编码时间性能也相对占有优势. 这是因为利用 CSBTT 编码在处理 XML 文档时，只需要扫描一遍文档树，而区间编码则至少要扫描两遍文档树. 因此，从这个角度来看, CSBTT 编码所花的时间较区间编码短.

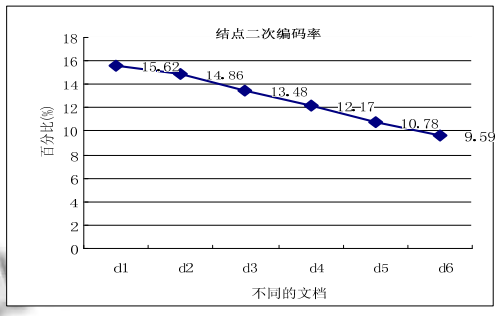

图 5　结点二次编码率

3　结语

CSBTT 编码，就动态更新方面考虑，再对 XML 文档进行更新时，无需对整个文档树进行大面积修改，只要修改相应的指针，这样提高了 XML 文档操作效率；结点关系判定方面，引入双亲指针，方便快捷找出结点间的结构关系；编码长度方面，采用自然数作为结点的编码序号，编码长度短且节省存储空间.

后续工作中需进一步考虑的问题有：(1)由于要将 XML 文档树转化为相应的二叉树，在转化过程中，将花费一定的时间代价，如何降低其开销；(2)结合实际的应用环境，将该编码模式与索引技术相结合，在此编码的基础上建立适合的索引机制.

参考文献

1 Wang W, Jiang HF, Lu HJ, Yu XJ. PBiTree Coding and Efficient Processing of Containment Join. Proc. of the 19th International Conference on Data Engineering. India, 2003.

2 肖厚新,唐常杰,张婷,等.BTCS:基于二叉遍历的 XML 文档编码模式.四川大学学报,2006,3(6):533–537.

3 张鹏,冯建华,房志峰.一种基于二叉树的 Native XML 数据库文档编码机制.计算机应用,2008,28(9):2331–2334.

4 庄灿伟,冯少荣,林子雨,等.XML 数据更新编码机制-ITBI. 计算机应用,2010,30(9):2325–232.

5 孟晓峰.XML 数据管理:概念与技术.北京:清华大学出版社, 2009.