第 30 卷第 9 期 计算机应用 Vol. 30 No. 9
2010 年 9 月 Journal of Computer Applications Sep. 2010

文章编号:1001 - 9081(2010)09 - 2324 - 05

XML 数据更新编码机制——ITBI

庄灿伟,冯少荣,林子雨,张东站
(厦门大学 计算机科学系,福建 厦门 361005)
(cwzhuang0229@163.com)

摘　要: 编码技术是可扩展标记语言(XML)查询处理的基础,传统编码技术利用自然数进行编码,很难支持 XML 动态更新。提出了更新支持的编码方法——ITBI,该方法将整数映射到完全二叉树,利用二叉树的中序遍历定义整数新的序关系,通过新的序关系重排自然数序列将静态编码转化为动态编码。同时,基于 ITBI 前驱、后继、距离等定义,设计了最短位长动态编码分配算法,有效控制更新过程中编码位长的增加。最后通过实验验证了编码的有效性。

关键词: 可扩展标记语言;动态更新;编码技术
中图分类号: TP311.13　　**文献标志码:** A

ITBI: Labeling scheme for XML data update

ZHUANG Can-wei, FENG Shao-rong, LIN Zi-yu, ZHANG Dong-zhan
(*Department of Computer Science, Xiamen University, Xiamen Fujian* 361005, *China*)

Abstract: Labeling scheme is the basis for Extensible Markup Language (XML) query processing. The traditional labeling schemes use numbers based on natural order, which is hard to support XML updating. A new labeling scheme, called ITBI (Inorder Traversal Based Integer), was proposed. ITBI created a mapping between integer and complete binary tree, and a new partial order based on inorder traversal of binary tree was defined, which just needed reordering the natural numbers to support dynamic XML. Meanwhile, based on the conceptions of previous ITBI, next ITBI, ITBI distance, the algorithm for assigning dynamic labels with the smallest size was presented, which controlled the increase in label size efficiently. The experimental results verify the validity of the proposed method.

Key words: Extensible Markup Language (XML); dynamic update; labeling scheme

0　引言

可扩展标记语言(Extensible Markup Language, XML),其目标是定义出计算机和人都能方便识别的数据类型。随着网络应用的快速发展,XML 数据成为当前主流的数据形式。如何对 XML 数据进行有效存储、并支持高效查询变得越来越重要。

在以树为模型的 XML 数据中,快速确定节点间结构关系是高效查询的关键。一个显而易见却又低效的方式是导航遍历,而如果在文档树中事先嵌入编码机制保存结构信息将大大提高查询效率。编码技术已经成为 XML 查询处理的基础和关键。

目前主流的编码技术包括基于前缀的编码[1-4]和基于区间的编码[5-8]。传统的前缀编码和区间编码利用了自然数对 XML 节点进行顺序标识。利用自然数标识顺序关系,具有序关系简单、处理高效、冗余度小、压缩比高等优点,但是当 XML 文档发生更新时,需要调整已有节点的编码以维持结构关系判定,更新代价高,减少编码更新代价的一个有效方法是利用新的偏序关系而不是自然数标识顺序。本文将整数与完全二叉树上的节点建立一一映射,利用二叉树中序遍历的无限可插性,提出了支持 XML 动态更新的偏序关系——ITBI(Inorder Traversal Based Integer),与已有动态编码方案相比,ITBI 无论在静态编码还是动态编码方面都有很好的性能。

1　相关研究

编码机制广泛应用于 XML 查询处理,目前提出了很多 XML 编码机制,其中两大类得到广泛应用,基于前缀的编码和基于区间的编码,这两大类根据动态更新时需不需要调整原有节点编码又可分为静态编码和动态编码。

DeweyID[1]是一种静态前缀编码,它首先对第 i 个孩子节点编号整数 i,然后与父节点编码合并,即第 i 个孩子节点的编码为"父节点编码.i"。当需要判断两节点的祖先后代关系时,只需判断一节点的编码是不是另外一个节点编码的前缀即可。Zhang 编码[5]是一种基于区间的静态编码方法,其编码形式构成一个区间($start$,end),其中,$start$、end 代表该节点在遍历顺序中的起始编号和结束编号。节点 u 是节点 v 的祖先节点,当且仅当 $u.start < v.start$,$u.end > v.end$。

静态前缀编码和区间编码都能高效支持节点间结构关系查询,但是当 XML 频繁更新时,需要对大量节点重新编码,这大大影响查询效率。减少编码更新代价的关键是实现两个已有编码之间可以插入无穷多个新编码。基于这个思路,国内外学者提出了很多动态编码方法。

Amagasa 等人提出了浮点数区间编码方式[7],它使用浮点数而不是整数来标识序关系。在两个浮点数之间可以插入

收稿日期:2010 - 03 - 10;修回日期:2010 - 04 - 09。　基金项目:国家自然科学基金资助项目(50604012)。
作者简介:庄灿伟(1984 -),男,福建泉州人,硕士研究生,主要研究方向:XML 数据库、数据挖掘;　冯少荣(1964 -),男,河北南宫人,副教授,博士,主要研究方向:XML 数据库、数据挖掘;　林子雨(1978 -),男,吉林柳河人,讲师,博士,主要研究方向:数据库、实时主动数据仓库、数据挖掘;　张东站(1974 -),男,江苏新沂人,副教授,博士,主要研究方向:数据挖掘。

很多浮点数,当 XML 动态更新时,可以不修改已有编码。但是,浮点数机器表示的长度是有限的,使得浮点数编码很容易产生溢出现象。另外,浮点数的频繁比较也会牺牲 XML 数据库查询性能。

BSC[3] 是使用定点小数的前缀编码方式,利用了两个定点小数间可以无限插入的性质,较好地支持更新操作。当需要往两个已有小数的中间插入时,只需要将两个小数之和的一半作为对新节点的编码,更新效率较高,但 BSC 只讨论了单节点插入,对批量更新没有讨论,如果将批量更新转化为单节点一个个插入,编码效率较差。

OrdPath[4] 也可以保证两个 OrdPath 码之间无限插入新的 OrdPath 码,其原理是利用奇数作为顺序标志,而偶数作判断标志,例如插入操作发生在 1 和 3 之间,这时用介于 1、3 的偶数 2 连接奇数 1,即"2.1"表示新插入的码值。这种方式在一定程度上避免了重新编码,但是 OrdPath 码中偶数只作为判断标志,编码空间存在大量冗余,另外,这些偶数会造成同层节点编码段数的不一致,影响查询效率。

CDBS[9-10] 是一种高效的动态编码方式,每个 CDBS 码是以"1"结束的二进制位串,两个 CDBS 码之间通过字典序来确定大小关系。任意两个 CDBS 码之间存在无穷个满足字典序的 CDBS 码,使得 CDBS 可以支持 XML 动态更新。当对 XML 静态编码时,CDBS 所需的编码长度和用连续的整数进行编码所需位长相同,保证了 CDBS 具有较高的静态编码效率。但是当动态编码时,为了有效利用已被删除节点编码,CDBS 需要讨论多种情况的插入操作,时间效率不高,而且也不能完全利用到被删节点的编码。例如,当删除一棵子树又随即恢复该子树,两次更新之后,XML 文档和原来一样,但往往需要增加编码长度。

综上,静态前缀编码和区间编码利用自然数进行编码,不能支持 XML 动态更新。动态前缀编码和区间编码一定程度支持 XML 更新操作,但和静态编码相比,往往需要更大的时间开销和更长的编码位数。本文定义了一种新的偏序关系——ITBI,有效支持 XML 动态更新,且具有较短的编码位数和较好的时间性能。

2　ITBI 编码

XML 更新问题的关键是找到新的偏序关系,使两个编码间可以无限插入新编码。ITBI 是一种支持 XML 动态更新的偏序关系,本章介绍 ITBI 支持 XML 动态更新的原理、ITBI 编码过程和存储表示。

2.1　原理

ITBI 的思想始于对完全二叉树编号的观察。把正整数和完全二叉树上的节点建立映射 f: 完全二叉树的根节点赋值 1,然后按层次优先的顺序从左向右、从上到下对二叉树上的节点赋予正整数编号。这样,正整数和完全二叉树上的节点建立起一一映射的关系。基于映射 f,可得到正整数的两种序关系: 当对完全二叉树上的节点进行广度优先遍历时,得到了一个有序序列,这个序列的序关系是自然数序关系。而如果对这些节点进行中序遍历,将得到一种新的序关系,把这种新的序关系定义为 ITBI 偏序关系。例如,1 ~ 12 这 12 个数自然数序关系为"1、2、3、4、5、6、7、8、9、10、11、12"。而 ITBI 序关系为: "8、4、9、2、10、5、11、1、12、6、3、7"。

用二进制表示的 ITBI 码偏序关系"$<$"可形式化如下。

定义 1　偏序关系 $<$。给定任意 $S_L, S_R \in A$:

1) $S_L = S_R \Leftrightarrow S_L$ 与 S_R 完全相同。

2) $S_L < S_R \Leftrightarrow$

a) 从左向右按位比较 S_L 和 S_R 的当前位,直到当前位不相同时停止比较,如果此时 S_L 对应的当前位是 0,而 S_R 对应的当前位是 1,则 $S_L < S_R$;

b) 如果 S_L 是 S_R 的前缀,且 S_R 去除和 S_L 相同的位后,第一位是 1,则 $S_L < S_R$;

c) 如果 S_R 是 S_L 的前缀,且 S_L 去除和 S_R 相同的位后,第一位是 0,则 $S_L < S_R$。

3) $S_L > S_R \Leftrightarrow S_R < S_L$。

ITBI 序关系下,给定正整数 n,则有 $2n < n < 2n+1$。ITBI 序关系下的正整数有一些良好的性质。

定理 1　ITBI 序关系下,不存在最小正整数,也不存在最大正整数。

证明　假设 a 是最小的正整数,而 $2a < a$,与假设矛盾。同理证明不存在最大正整数。　证毕。

定理 2　ITBI 序关系下,任给两正整数 a, b,其中 $a < b$,存在正整数 m,满足 $a < m < b$。

证明　如果自然数序上 a 比 b 小,令 $m = 2b$,则 $a < m < b$。这是因为,在完全二叉树中,m 是 b 的左孩子,依中序遍历的规则,先遍历完 b 的左孩子,再遍历 b 自身,所以 $m < b$; 又因为在 ITBI 序关系下介于 m 和 b 之间的正整数在自然数序上只能比 b 大,而现在自然数序上 a 小于 b,推出 $a < m$,所以 $a < m < b$。同理证明如果自然数序上 b 小于 a,令 $m = 2a + 1$,则 $a < m < b$。　证毕。

定理 1 和定理 2 表明在 ITBI 序关系下,给定有序的正整数序列,可以往任意位置插入新的正整数,使得新的序列仍然有序,这是 ITBI 支持 XML 更新操作的理论基础。

2.2　编码过程

ITBI 与传统静态编码成正交关系,首先用静态编码方法产生自然数序关系的编码,然后将自然数序关系转化为 ITBI 序关系就可。ITBI 编码的关键是将整数 $1 \cdots n$ 转化为 ITBI 序关系,利用整数与完全二叉树的对应关系,将具有 n 个节点的完全二叉树进行中序遍历,就可以得到 n 个 ITBI 序关系的整数,如算法 1 所示。

算法 1　*StaticAllocate*(n)　　//ITBI 静态编码分配算法

输入: 正整数 n

输出: n 个 ITBI 码

```
Arr [1,n]
i ← 1
Sta_DFS( n,Arr,1,i)
return Arr
Procedure Sta_DFS( n,Arr,S,&i)
/*  递归函数,n 是节点总数,Arr 是数组,S 是 ITBI 码,i 是引用型
    数组下标,初值为 1* /
if S > n return
Sta_DFS( n,Arr,2* S,n,i)
Arr [i++] ← S
Sta_DFS( n,Arr,2* S + 1,n,i)
```

算法时空复杂度都为 $O(n)$。由于 ITBI 编码只是对自然数进行重新排列,而不会增加编码长度,所以 ITBI 有着与自然数编码相同的编码效率。

和自然数编码一样,ITBI 可以应用于各种编码机制,图 1

是 Students 文档树及其 ITBI 编码示例，每个元素节点上方是 ITBI 前缀编码，下方是 ITBI 区间编码。进行 ITBI 前缀编码时，当只有一个孩子节点时，产生 ITBI 序列“1”，当有两个孩子节点时，产生 ITBI 序列“2,1”，然后依前缀编码规则对每个节点赋予编码。进行 ITBI 区间编码时，为 6 个节点产生 1 ~ 12 的 ITBI 序列“8、4、9、2、10、5、11、1、12、6、3、7”，然后依遍历顺序对每个节点赋予编码。

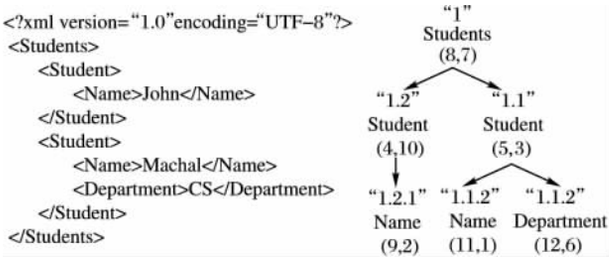

图 1 Students. xml 文档及其对应的 ITBI 编码

2.3 存储表示

ITBS 码是一个整数，一个 ITBS 码的有效位是其对应的二进制整数，存储这个有效位的方法包括定长、变长、压缩等多种方式，这些方式所需空间占用各不相同。如表 1，静态分配 12 个 ITBS 码，其有效二进制位的总位长是 37，定长存储需要 50 位，利用(位长，有效位)进行存储需要 75 位，UTF8 存储需要 96 位。

表 1 ITBS 存储表示示例

自然数序	ITBS 序	ITBS 有效二进制位	定长存储(长度 = 4)	变长存储 1(位长，有效位)	变长存储 2(UTF8)
1	8	1000	1000	100,1000	00001000
2	4	100	0100	011,100	00000100
3	9	1001	1001	100,1001	00001001
4	2	10	0010	010,10	00000010
5	10	1010	1010	100,1010	00001010
6	5	101	0101	011,101	00000101
7	11	1011	1011	100,1011	00001011
8	1	1	0001	001,1	00000001
9	12	1100	1100	100,1100	00001100
10	6	110	0110	011,110	00000110
11	3	11	0011	010,11	00000011
12	7	111	0111	011,111	00000111
总位长		37	48 + 2 = 50	73 + 2 = 75	96

在讨论编码长度时，需用哪种存储方法进行讨论。一般而言，有效二进制位越长，其所需的定长或变长存储所需空间也越大，所以在动态更新时，为新节点的编码应具有最短有效二进制位。不失一般性，本文讨论编码长度指有效二进制位的长度。

3 动态编码分配

任意两个 ITBI 码之间存在无穷个介于其间的编码，设计 ITBI 动态编码分配算法的核心是从无穷的中间编码中产生 n 个新编码，满足这 n 个新编码总位长最短。本节首先讨论 $n = 1$ 的单个动态编码分配算法，然后讨论 $n \geqslant 1$ 的批量动态编码分配算法。

3.1 单个动态编码分配算法

ITBI 码的集合 $A = \{0,1,2,3,\cdots\}$，其中 0 并不存在于实际编码，用 0 表示虚拟的无穷小或虚拟的无穷大，可以方便地统一最左插入，最右插入等各种操作。

定义 2 层次 i。根节点层次定义为 1，每下降一层，i 值加 1。

定义 3 位长 $size$。对于任意 $S \in A$，$size(S)$ 指 S 的二进制位长，在数值上等于完全二叉树上 S 所在的层次。特别的，$size(0) = 0$。

定义 4 前驱 pre。对于任意 $S \in A$，$pre(S)$ 是指中序遍历给定层次的完全二叉树，在遍历序列中，位于 S 之前的上一个 ITBI 码，其中 $pre(0)$ 指遍历序列的最后一个 ITBI 码。

定义 5 后继 $next$。对于任意 $S \in A$，$next(S)$ 是指中序遍历给定层次的完全二叉树，在遍历序列中，位于 S 之后的下一个 ITBI 码，其中 $next(0)$ 指遍历序列中的第一个 ITBI 码。

例如，给定完全二叉树层次 3，中序遍历该完全二叉树，得到序列 4、2、5、1、6、3、7，有 $size(1) = 1$，$pre(1) = 5$，$pre(0) = 7$，$next(1) = 6$，$next(0) = 4$。

给定层次为 i 的完全二叉树，非叶节点 S 的前驱是其左孩子的最右下方叶子节点，其后继是其右孩子最左下方叶子节点，特别的，无穷小 0 的后继是完全二叉树的最左叶子节点，无穷大 0 的前驱是完全二叉树的最右叶子节点。

基于以上定义，设计了算法 2 得到最短位长中间编码。

算法 2 $MiddleWithSmallestSize(S_L,S_R)$ // 计算中间编码

输入：$S_L < S_R$。最左插入，S_L 取 0；最右插入，S_R 取 0。

输出：S_M，满足 $S_L < S_M < S_R$ 且 S_M 具有最短位长

$i \leftarrow 1 + \max(size(S_L),size(S_R))$

$S_L \leftarrow next(S_L,i)$

$S_R \leftarrow pre(S_R,i)$

return S_L 与 S_R 的最长公共前缀

Procedure $size(S)$ /* 求 S 的二进制位数 */

  if $S = 0$ return 0

  return $1 + lb(S)$

Procedure $pre(S,i)$ /* 求 S 的前驱，只实现对非叶节点求前驱 */

  if $size(S) >= i$ ERROR

  if $S = 0$ return 1^i

  return $S \oplus 01^{i-size(s)-1}$ // ⊕表示连接操作，1^i 表示 i 个 1

Procedure $next(S,i)$ /* 求 S 的后继，只实现对非叶节点求后继 */

  if $size(S) >= i$ ERROR

  return $S \oplus 10^{i-size(s)-1}$

算法时空复杂度都为 $O(1)$。

定理 3 算法 2 得到的 S_M 满足：$S_L < S_M < S_R$ 且 S_M 具有最少的二进制位数。

证明 由命题 2，$size(S_M)$ 满足 $size(S_M) \leqslant 1 + \max(size(S_L),size(S_R))$，给定完全二叉树层次 $i = 1 + \max(size(S_L),size(S_R))$。$S_L$，$S_R$ 自身不满足 S_M 定义，所以 S_M 只能存在于 S_L 的后继 S_L' 与 S_R 的前驱 S_R' 之间，求介于 S_L，S_R 间的 S_M，等价于求介于 S'_L，S'_R 间的 S_M，而 S'_L，S'_R 都是叶子节点，满足两叶子节点的 S_M 是其最近共同祖先，所以算法 2 得到正确的结果。 证毕。

3.2 批量动态编码分配算法

为得到介于任意两个 ITBI 码之间最短总位长的 n 个新编码，引进距离的定义。

定义 6 距离 $dist$。S_L，S_R 距离定义为：中序遍历给定层次的完全二叉树，遍历序列中，介于 S_L，S_R 之间 ITBI 码个数加 1。

例如，给定完全二叉树层次 3，中序遍历序列为 4、2、5、1、6、3、7，则 $dist(1,7) = 3$。

S_L,S_R 都是叶子节点时,由于叶子节点的后继是非叶节点,非叶节点的后继是叶子节点,所以两个叶子节点的距离等于这两个叶子节点对应自然数距离的两倍,即 $dist(S_L,S_R) = 2\times(S_R - S_L)$。当 S_L,S_R 不是叶子节点时,因为 S_L 后继和 S_R 前驱是叶子节点,仍可以转化为求两叶子节点的距离。

当完全二叉树层次增加一层,所得到的距离将是原来的两倍,这种线性递增的特性使得距离比是常量。例如,当 $i = 2$ 时,$dist(2,3) = 2$,$dist(2,1) = 1$,$dist(2,3)/dist(2,1) = 2$;当 $i = 3$ 时,$dist(2,3) = 4$,$dist(2,1) = 2$,$dist(2,3)/dist(2,1) = 2$。这样,距离有了更形象化的意义。

利用 ITBI 距离的定义,设计了 ITBI 批量动态编码分配算法,其思想是每次贪心的计算位数最少的中间数,将其保存在数组合适位置,然后递归的求得所需要的 n 个 ITBI 码。

算法 3　$DynamicAllocate(S_L,S_R,n)$　// 批量动态编码分配

输入: 正整数 n,ITBI 码 S_L,S_R 且 $S_L < S_R$

输出: n 个介于 S_L,S_R 间的 ITBI 码

```
Arr[0,n+1]                          //n 个有效编码保存在 Arr[1,n]
Arr[0] ← S_L
Arr[n+1] ← S_R
Dyn_DFS(Arr,0,n+1)
return Arr[1,n]
Procedure Dyn_DFS(Arr,l,r)
  /* 递归函数,Arr 是数组,l、r 是数组左右下标 */
  if l+1 >= r return
  i ← 1 + max(size(Arr[l]),size(Arr[r]))          // 层次 i
  S_M ← MiddleWithSmallestSize(Arr[l],Arr[r])     // 中间数 S_M
  d1 ← dist(Arr[l],S_M,i)
  d2 ← dist(S_M,Arr[r],i)
  m ← round(l + [d1/(d1+d2)] * (r-l))   //S_M 保存位置 m
  Arr[m] ← S_M
  Dyn_DFS(Arr,l,m)
  Dyn_DFS(Arr,m,r)
Procedure Dist(S_L,S_R,i)
  /* 求 S_L,S_R 在层次 i 下的距离 */
  if i < max(size(S_L),size(S_R)) ERROR
  tmp ← 0
  if size(S_L) < i then S_L ← next(S_L,i) tmp ++ end
  if size(S_R) < i then S_R ← pre(S_R,i) tmp ++ end
  return 2*(S_R - S_L) + tmp
```

算法时空复杂度都为 $O(n)$。

定理 4　动态 ITBI 编码分配算法产生了 n 个介于 S_L,S_R 间的 ITBI 码,且这 n 个 ITBI 码二进制总位数最短。

证明　显然算法产生了 n 个介于 S_L,S_R 间的 ITBI 码,这里证明定理的后半部分。令 $S_M = MiddleWithSmallestSize(S_L,S_R)$,则 S_M 是 n 个 ITBI 码里的一个,因为若 S_M 不在这 n 个 ITBI 码里,则将 S_M 替换序列里的任意一个,可以得到总位数更少的 ITBI 码,所以问题的关键是将 S_M 保存在数组中哪个位置 p。设 $Arr[l,r]$ 共 $n+2$ 个元素,其中 $Arr[l] = S_L$,$Arr[r] = S_R$,n 个有效 ITBI 码保存在 $Arr[l+1,r-1]$ 中,令 $d_1 = dist(S_L,S_M)$,$d_2 = dist(S_M,S_R)$,依距离的定义可知,为了使总位数最小,中间数 S_M 在数组中的位置 p 应满足: p 对 l,r 划分的距离比等于 S_M 对 S_L,S_R 划分的距离比,即 $(p-l)/(r-p) = (d_1/d_2)$,解得 $p = l + [d_1/(d_1+d_2)]\times(r-l)$(采用四舍五入法),递归的求得满足要求的 n 个 ITBI 码。　证毕。

如图 2,往 Students 文档插入子树,前缀编码产生介于 2 和 1 之间的最短位长 ITBI 码 5,得到新子树的根节点编码 1.5,接着编码其孩子节点。区间编码产生介于 10 ~ 5 总位长最短的 6 个 ITBI 码 84、42、85、21、86、43,接着为每个节点按遍历顺序赋予编码。

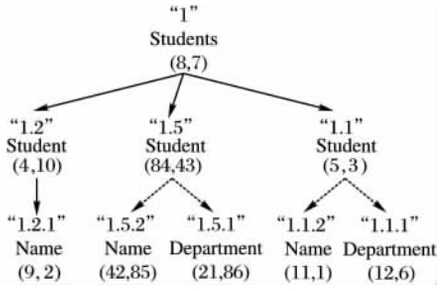

图 2　ITBI 对更新的子树编码

4　实验

称自然数编码为 BINARY 编码,选用 BINARY 区间编码、CDBS 区间编码与 ITBI 区间编码进行对比。实验用 Tinyxml 对 XML 文档进行解析,测试数据集及其属性见表 2,其中 D1、D2 源自文献[11],D3、D4、D5 源自文献[12]。

表 2　测试数据集

数据集	文档名称	总节点数	最大深度	平均深度
D1	Hamlet	6 636	6	4.79
D2	All_shakes	179 690	7	5.58
D3	Nasa	476 646	8	3.16
D4	Lineitem	1 022 976	3	2.94
D5	Treebank	2 437 666	36	7.87

4.1　静态性能分析

评价静态性能好坏的两个主要指标是完成编码所需要的时间和节点平均编码长度。实验用 CDBS、ITBI、BINARY 分别对上述 5 个数据集进行编码,测试它们的静态编码时间和编码长度。

图 3(a)是三者编码时间对比,从图中可以看出,BINARY 编码具有最好的时间性能,这是因为 BINARY 直接利用连续的自然数对节点编码,而 ITBI 和 CDBS 都需要先计算出新的码值序列,然后利用新码值序列对节点编码。和 CDBS 相比,ITBI 具有更好的时间性能。这是因为 CDBS 需要对两个已有码值进行比较后才能计算出新的 CDBS 码,而 ITBI 不需要比较操作,在时间上花销较小。图 3(b)是三者的平均编码长度对比,对于同一个 XML 文档,虽然每种编码方式对每个节点编码的二进制位串并不一样,但对整个文档编码的总位数一样的。

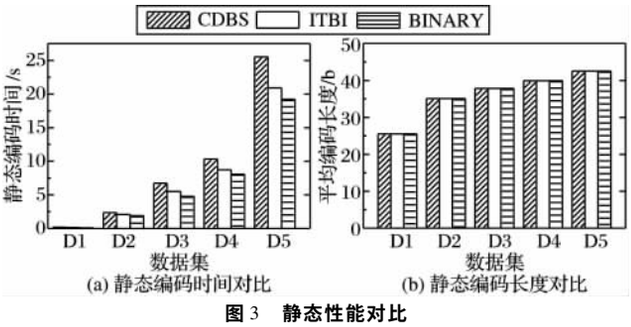

图 3　静态性能对比

4.2　动态性能分析

CDBS 和 ITBI 都可以较好支持 XML 动态更新,实验比较 CDBS 和 ITBI 的动态编码时间和编码效率。

实验 1　测试 CDBS 和 ITBI 在 XML 文档发生更新时所需要的动态编码时间。数据集 Hamlet. xml 共有元素节点 6 636个,其根节点下有 5 个 act 元素代表的子树,更新操作首先删除第二个 act 子树,然后测试在删除的位置分别插入 D1 ~ D5 数据集时 CDBS 和 ITBI 所需要的编码时间。图 4(a) 表明,ITBI 优于 CDBS,这是因为 CDBS 动态分配算法需要分别讨论各种插入情况,而 ITBI 可以很好地进行统一操作,减少判断时间的开销。另外,CDBS 动态编码位数较长,也影响了编码时间。

实验 2　选用数据集 Hamlet. xml 测试 CDBS 和 ITBI 动态编码空间占用。对 Hamlet 文档动态更新过程如下: 首先删除第一个 act 子树,随即又恢复该子树,这是第一次更新。第二次更新在第一次更新的基础上删除第二个 act 子树,随即又恢复,按同样方式处理剩下 3 个 act 节点。每次更新之后,Hamlet 文档和初始一样,但编码却发生了变化,统计 Hamlet 文档在每次更新后节点平均编码长度变化。从图 4(b) 可以看出,在更新之前,CDBS 和 ITBI 平均编码长度是一样的,但当删除又插入一棵子树时,CDBS 并不能得到原有编码,而是需要更长的编码位数。可见,当 XML 动态更新时,CDBS 并不总能得到最短的二进制位串,而 ITBI 能很好地重用已删编码,具有较好的更新效率。

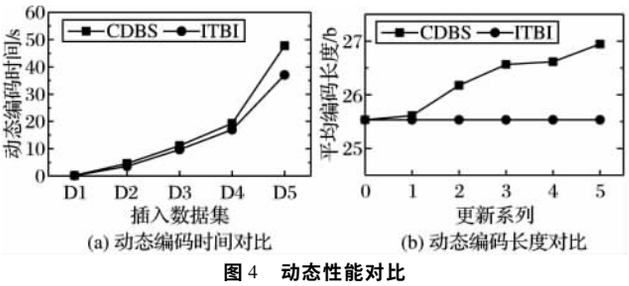

图 4　动态性能对比

5　结语

编码机制广泛应用于 XML 查询处理,设计支持 XML 动态更新的编码机制具有重要的实际意义。本文提出了支持 XML 动态更新的编码机制——ITBI,其在编码效率、更新支持、编码时间等方面都有较好的性能。在静态编码效率上,ITBI 具有和自然数编码相同的编码长度;在动态编码效率上,ITBI 可以产生最短的二进制位长为新插入的节点编码;在更新支持上,ITBI 可以处理单个叶子节点更新、批量节点更新而不修改已有编码;在编码时间上,ITBI 利用机器整数类型进行编码,具有较好时间性能。但是,ITBI 并不能完全避免 XML 动态更新时的重新编码,当进行中间节点更新时,ITBI 也需要修改已有编码,怎么有效处理中间节点更新,是未来要思考的问题。

参考文献:

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

[2] KHA D D, YOSHIKAWA M, UEMURA S. An XML indexing structure with relative region coordinate [C]// Proceedings of the 17th International Conference on Data Engineering. Washington, DC: IEEE Computer Society, 2001: 313 - 320.

[3] 汪陈应,袁晓洁,王鑫,等. BSC: 一种高效的动态 XML 树编码方案[J]. 计算机科学,2008,35(3):76 - 78.

[4] O'NEIL P, O'NEIL E, PAL S, *et al.* ORDPATHs: Insert-friendly XML node labels [C]// Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2004: 903 - 908.

[5] ZHANG C, NAUGHTON J, DEWITT D, *et al.* On supporting containment queries in relational database management systems [C]// Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2001: 425 - 436.

[6] GRUST T. Accelerating XPath location steps [C]// Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2002: 109 - 120.

[7] AMAGASA T, YOSHIKAWA M, UEMURA S. QRS: A robust numbering scheme for XML documents [C]// Proceedings of the 19th International Conference on Data Engineering. Washington, DC: IEEE Computer Society, 2003: 705 - 707.

[8] 罗道峰,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.

[9] LI C Q, LING T W, HU M. Efficient processing of updates in dynamic XML data [C]// Proceedings of the 22nd International Conference on Data Engineering. Washington, DC: IEEE Computer Society, 2006: 13 - 22.

[10] LI C Q, LING T W, HU M. Efficient updates in dynamic XML data: From binary string to quaternary string [J]. The VLDB Journal, 2008, 17(3): 573 - 601.

[11] NIAGARA experimental data [EB/OL]. [2009 - 12 - 23]. http://www.cs.wisc.edu/niagara/data.html.

[12] University of Washington XML repository [EB/OL]. [2009 - 12 - 23]. http://www.cs.washington.edu/research/xmldatasets.

(上接第 2323 页)

[6] LIU H, YU L. Towards integrating feature selection algorithms for classification and clustering [J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(4): 491 - 502.

[7] PARK J S, SHAZZAD K M, KIM D S. Toward modeling lightweight intrusion detection system through correlation-based hybrid feature selection [C]// CISC 2005: Proceedings of the First SKLOIS Conference on Information Security and Cryptology, LNCS 3822. Berlin: Springer, 2005: 279 - 289.

[8] YANG Y, PEDERSEN J O. A comparative study on feature selection in text categorization [C]// Proceedings of the Fourteenth International Conference on Machine Learning. San Francisco, CA: Morgan Kanfmann Publishers, 1997: 412 - 420.

[9] HAN JIAWEI, MICHELINE K. Data mining concepts and techniques [M]. 2nd ed. Beijing: China Machine Press, 2007.

[10] LI FEIXIONG, LIU QUAN. An improved algorithm of decision trees for stream data based on VFDT [C]// 2008 International Symposium on Information Science and Engineering. Shanghai: [s. n.], 2008: 597 - 600.

[11] FAN WEI. Systematic data selection to mine concept-drifting data streams [C]// Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2004: 128 - 137.