# 一种支持数据更新的前缀编码方案①

魏东平，贾 楠，徐瑞敏

(中国石油大学(华东) 计算机与通信工程学院，东营 257061)

**摘 要**：目前大部分前缀编码方案都不能很好的支持 XML 文档的数据更新。提出的前缀编码方案不仅能高效地支持结构查询，快速准确的判断 XML 文档结构树中任意两个结点之间的父子、先后代以及兄弟关系，而且对插入的结点采用新的编码规则，避免了更新操作带来的编码调整问题，能有效支持 XML 文档更新。

**关键词**：XML；前缀编码；数据更新

## A Novel Prefix-Labeling Scheme for Update-Supporting

WEI Dong-Ping[1], JIA Nan[1], XU Rui-Min[1]

(College of Computer and Communication Engineering, China University of Petroleum, Dongying 257061, China)

**Abstract:** Most prefix-labeling scheme nowadays cannot support XML document updating fully. Based on the comparison of the current prefix-labeling scheme, a novel prefix-labeling was proposed. This scheme does not only efficiently support structure queries, which can rapidly and accurately judge the parent-child, ancestor-descent and sibling relationships between any two nodes of the XML document tree, it also adopts the new labeling rules to avoid an update coding adjustment operation that may bring about when new nodes are inserted, and effectively support update for XML document.

**Keywords:** XML; Prefix labeling scheme; update-supporting

XML 关键字查询(XML Keyword Search)[1]是当前 XML 领域的一个新的研究热点，而大部分的 XML 全文查询算法均使用前缀编码方案对 XML 文档进行编码，因此，前缀编码的研究，特别是如何解决文档更新的问题，对于解决 XML 研究领域的许多问题都有重要的参照意义。

本文对现有的前缀编码方案进行比较分析，针对不能很好支持文档更新的问题，提出了一种新的前缀编码方案。该方案利用 Dewey 编码对原始 XML 文档进行编码，并通过改进编码规则对插入的新结点进行编码，从而完全支持文档的更新。

## 1 相关研究

Dewey 编码[2]是最早提出的前缀编码方案。其查询效率较高，但不支持更新。之后提出的简单前缀(Simple Prefix, SP)编码[3]使用二进制字符串标记 XML 树的节点，但编码码长很大，也不能避免重新编码。

DeweyIDs 编码[4]初始编码时，只将正奇数分配给结点作为层标识。在对 XML 文档树进行动态更新(插入新结点)时，在预留空间充裕的情况下，只将正奇数分配给新结点作为层标识；当预留空间不足时，则引入偶数作为标识。

ORDPATH 编码[5]是一种扩展的 Dewey 编码，它使用奇数进行初始编码。它能够部分解决更新操作造成的重新编码问题。但是编码的规模很大，查询效率低下，并且会出现编码溢出。

DLN 编码[6]也是在 Dewey 上的一种改进编码方案，对于层标识使用了可变长的编码，可以解决层标识溢出问题；在更新时，引入了子层编号和相应的子层分隔符。但这却降低了存储效率，也增加编码的复杂度。同样 DLN 也存在最左端插入问题，并且在更新时需要使用子层编码，造成空间效率下降。

通过分析，本文认为 XML 文档更新没有解决的问题主要集中在以下两个方面：

① 收稿时间:2010-07-02;收到修改稿时间:2010-07-26

①编码要支持高效的查询操作；

②编码同时要有效支持动态更新，避免重新编码；

## 2 改进的前缀编码方案EX_DEWEY

### 2.1 编码原理

为了能够更好的支持文档的更新操作，我们将 Dewey 编码进行改进，提出一种新的支持数据更新的编码方案。编码规则为：XML 文档树中的每一个结点被赋予一个二元组<Ex_Dewey_id, depth>，其中 Ex_Dewey_id 为结点的编码，其初始形式与 Dewey 编码相同，也是将一个结点的双亲结点的编码作为其前缀，双亲结点与孩子结点之间用分隔符“.”隔开；depth 为该结点在树中所处的层数，用来加速结构连接的操作。图 1 为一个编码的实例。

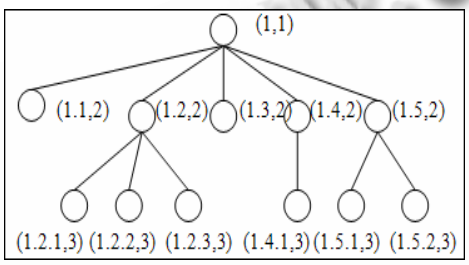


图 1　初始 EX_DEWEY 编码实例

定义 1. (编码序)假设给定两个 EX_DEWEY 编码 A（$a_1.a_2\ldots a_m$，$d_1$），B（$b_1.b_2\ldots b_n$，$d_2$），如果 A < B 有且仅当满足：

$\exists k \le \min(m,n)$，使 $\frac{a_1}{b_1}=\frac{a_2}{b_2}=\ldots=\frac{a_{k-1}}{b_{k-1}}$ 且 $a_k \times b_1 \le b_k \times a_1$；

定理 1. (编码序的传递性)假设给定三个 EX_DEWEY 编码 A（$a_1.a_2\ldots a_i$，$d_1$），B（$b_1.b_2\ldots b_m$，$d_2$），C（$c_1.c_2\ldots c_n$，$d_3$），如果 A < B并且B < C，那么 A < C。

证明：由定义 1 可知，

由 A < B，可得 $\exists j \le \min(l,m)$，使得 $\frac{a_1}{b_1}=\frac{a_2}{b_2}=\ldots=\frac{a_{j-1}}{b_{j-1}}$ 且 $a_j \times b_1 \le b_j \times a_1$；

由 B < C，可得 $\exists k \le \min(m,n)$，使得 $\frac{b_1}{c_1}=\frac{b_2}{c_2}=\ldots=\frac{b_{k-1}}{c_{k-1}}$ 且 $b_k \times c_1 \le c_k \times b_1$；

我们假定 $\frac{a_1}{b_1}=\beta$，$\frac{b_1}{c_1}=\gamma$。

当 j < k 时，因为 $\frac{a_1}{c_1}=\frac{a_1}{b_1}\times\frac{b_1}{c_1}=\beta\times\gamma$，$\frac{a_1}{c_1}=\frac{a_2}{c_2}=\ldots=\frac{a_{j-1}}{c_{j-1}}=\beta\times\gamma$。因为 $a_j \times b_1 \le b_j \times a_1$ 且 $\frac{b_1}{c_1}=\frac{b_j}{c_j}$，有 $a_j \times c_1 \le \frac{b_j \times a_1 \times c_1}{b_1}=\frac{a_1 \times b_1 \times c_j}{b_1}=c_j \times a_1$，因此，A < C.

同理可证当 j = k, ， j > k 时， A < C。

定义 2.(插入结点的编码）假设给定两个兄弟结点 EX_DEWEY 编码 A（$a_1.a_2\ldots a_m$，d），B（$b_1.b_2\ldots b_m$，d），A + B 定义：$A+B=((a_1+b_1),(a_2+b_2)\ldots(a_m+b_m),d)$

定理 2.(插入结点编码正确性) 假设给定两个兄弟结点 EX_DEWEY 编码 A（$a_1.a_2\ldots a_m$，d），B（$b_1.b_2\ldots b_m$，d），则 $A<(A+B)<B$。

证明：因为 A 和 B 是兄弟结点，所以 $\frac{a_1}{b_1}=\frac{a_2}{b_2}=\ldots=\frac{a_{m-1}}{b_{m-1}}$，因此A<B,即 $a_m \times b_1 \le b_m \times a_1$

假定 $\frac{a_1}{b_1}=\beta$,也就是$a_1=\beta\times b_1$。

现证明 $A<(A+B)$，有 $\frac{a_1}{a_1+b_1}=\frac{\beta\times b_1}{\beta\times b_1+b_1}=\frac{\beta}{\beta+1}$，即 $\frac{a_1}{a_1+b_1}=\frac{a_2}{a_2+b_2}=\ldots=\frac{a_{m-1}}{a_{m-1}+b_{m-1}}=\frac{\beta}{\beta+1}$ 又有 $a_m\times(a_1+b_1)=a_m\times a_1+a_m\times b_1\le a_m\times a_1+b_m\times a_1=(a_m+b_m)\times a_1$,所以$A<(A+B)$

同理可证 $(A+B)<B$

所以， $A<(A+B)<B$。

### 2.2 编码对结构查询的支持

基于该编码方式，通过等值和附加整数运算操作即可在常数时间内判断结点间的祖孙关系、父子关系，兄弟关系，时间复杂度为 O(n)。

假设给定两个 EX_DEWEY 编码 A（$a_1.a_2\ldots a_m$，$d_1$），B（$b_1.b_2\ldots b_n$，$d_2$），有如下判断命题：

#### 2.2.1 祖孙关系的判断

命题 1. 结点 A 和结点 B 存在祖先后代关系当且仅当 m < n 并且 $\frac{a_1}{b_1}=\frac{a_2}{b_2}=\ldots=\frac{a_m}{b_m}$。

证明：当 m = 1 < n 时，A 是文档的根结点，由此可知任何结点都是 A 的祖孙结点；

当 m ≠ 1 < n 时，满足 $\frac{a_1}{b_1}=\frac{a_2}{b_2}=\ldots=\frac{a_m}{b_m}$，由此可知 A 和 B 处于不同层次，并且编码序为 A < B，即 A 是 B

的祖先结点，A 和 B 满足祖先后代关系。

2.2.2 父子关系的判断

命题 2. 结点 A 和结点 B 存在父子关系当且仅当 A 和 B 满足祖先后代关系并且满足 $d_1 = d_2 - 1$。

证明：先证明 A 和 B 满足祖先后代关系(见证明 1)，又因为 $d_1 = d_2 - 1$，即 B 位于 A 的下一层，由此可知，A 和 B 满足父子关系。

2. 2. 3 兄弟关系的判断

命题 3. 结点 A 和结点 B 存在兄弟关系当且仅当 $m = n$ 并且 $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \ldots = \frac{a_{m-1}}{b_{m-1}}$。

证明：当 $m = n$ 时，由编码特点只知，A 和 B 处于同一层次。又因为且 $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \ldots = \frac{a_{m-1}}{b_{m-1}}$，所以 A 和 B 具有相同的父亲节点，即 A 和 B 满足兄弟关系。

### 2.3 编码对数据更新的支持

对于动态 XML 文档，由于频繁插入、删除结点，如何最大限度地降低更新代价，是一个很重要的问题。EX_DEWEY 编码的一个重要特点就是可以在任意的两个结点之间插入一个新的结点，而不需要重新编码。一个更新操作实例见图 2。

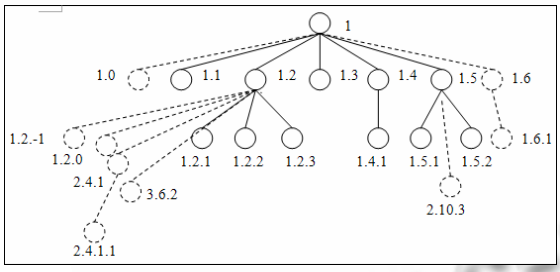


图 2 EX_DEWEY 编码更新操作实例(省略层次信息)

2.3.1 插入结点

在不同位置插入新节点，其编码的生成的规则不相同。

1) 最左儿子结点之前插入

为了支持这种插入，需要引入负数概念。当结点 A 是第一个孩子结点时，编码为，那么在它左边插入的新结点编码为。

2) 最右儿子结点之后插入

当结点 A 是最后一个孩子结点时，编码为，那么在它右边插入的新结点编码为。

3) 连续的兄弟结点中间插入

当在两个连续兄弟结点 A、B 中间插入新的结点时，编码为 A+B。

2.3.2 删除结点

删除结点及其子结点后，都不会对 XML 文档中的其他结点造成影响，故不需重新编码。

2.3.3 更新算法

Algorithm：Updating_XML_Tree(T, left, right)

Input：T 为 EX_DEWEY 编码树，left 是插入位置左侧结点的编码(可以为 null)，right 是插入位置右侧结点编码(可以为 null)；

Output：新插入的结点 m 的编码

Description：

```
if N 的插入在最左兄弟之前
    { n.ex_dewey_id=left.ex_dewey_id;
    n.depth=left.depth;
ReplaceEnd(n.ex_dewey_id,
GetEnd(n.ex_dewey_id)-1);}
  //ReplaceEnd(m, k)修改层标识符 m 最后一个整数 k;
  // GetEnd(m)返回层标识符 m 的最后一个整数;
    else  if N 的插入在最右兄弟之后
          { n.ex_dewey_id=left.ex_dewey_id;
           n.depth=left.depth;
ReplaceEnd(n.ex_dewey_id,
GetEnd(n.ex_dewey_id)+1); }
          else N 的插入在兄弟结点之间
            {if(left.depth=right.depth)
{ n.ex_dewey_id=left.ex_dewey_id+right.ex_dewey_id;
                    n.depth=left.depth; }
              else
                  return null;
            }//END
```

## 3 实验和分析

这一节主要对 EX_DEWEY 编码方案在查询性能、更新性能等方面进行性能评估，并将其与 ORDPATH 编码比较。执行实验程序的计算机的 CPU 是 Intel CoreTM2，主频 2.33GHZ，4GB 内存，320GB 的 SATA 硬盘，操作系统为 Windows XP Professional。仿真数据集采用通用数据集 Xmark[7]和 Treebank[8]，相关参数如表 1 所示。

表 1　测试数据集

| 数据集 | 大小(MB) | 结点数 | 最大深度 |
|---|---|---|---|
| Xmark | 113 | 1666315 | 12 |
| Treebank | 85.4 | 2437666 | 36 |

### 3.1 查询性能编码原理

EX_DEWEY 编码比 ORDPATH 编码结构查询响应时间较快，因为 EX_DEWEY 编码在结点关系判断时只需要采用等值和整数运算，并且通过附加层次信息 depth 来加速结构查询，所以查询响应时间较短；而 ORDPATH 编码在编码比较时，由于涉及到偶数结点判断，层次关系判断比较复杂， 可能需要进行多次数值比较，因此相应的查询响应时间较长。

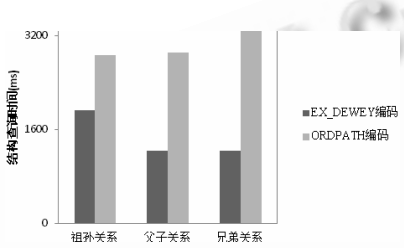


图 3　两个编码方案在查询上的对比

### 3.2 更新性能

EX_DEWEY 编码在插入结点数目发生变化时，XML 文档编码大小基本趋于不变，这是由于采用新的结点生成规则，插入结点后，基本上完全避免了重新编码；而 ORDPATH 编码采用负数和偶数作为预留空间，只是部分解决更新操作造成的重新编码问题，当更新频繁时，仍可能带来部分重新编码，所以不能完全解决 XML 文档的动态问题。

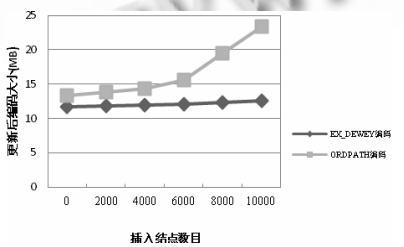


图 4　两个编码方案在结点更新后编码大小的对比

## 4　结论

本文分析了现有的几种 XML 文档前缀编码方案，研究了不同前缀编码方案解决数据更新问题的策略，提出了一种新的支持数据更新的前缀编码方案 EX_DEWEY 编码方案。它不仅在静态编码中具有很好的时空效率,而且本文更注重的是:在更新时,它可以完全避免重新编码任何已有结点,并且只需要将插入结点相邻的两个结点的层标识之和的作为插入结点的层标识,通过实验分析证明，更新效率较高。

## 参考文献

1 S Amer-Yahia JS. XMLfull-text search: Challenges and opportunities. Proc. of the 31st international conference on Very large data bases, Trondheim, Norway, 2005: 1386－1387.

2 Tatarinov I, Viglas SD, Beyer K, et al. Storing and Querying Ordered XML using a Relational Database System. In Proceedings of 21th ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, 2002: 204－215.

3 Cohen E, kaplan H, Milo T. Labeling Dynamic XML Tree. Proc. of the 21st ACM Symposium on Principles of Database Systems. Madison, Wisconsin, USA, 2002: 271－281.

4 Duong M, Zhang Y. LSDX: A New Labelling Scheme for Dynamically Updating XML Data. Proc. of the 16th Australasian Database Conference, 2005:185－193.

5 O'Neil P, O'Neil E, Pal S. ORDPATHS:Insert-Friendly XML Node Label. Proc. of the ACM SIGMOD, 2004: 903－908.

6 Tatarinov I,Viglas S D, Beyer K, et al. Storing and Querying Ordered XML using a Relational Database System. Proc. of 21th ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, 2002:204－215.

7 University of Washington XML Repository. http:// www. cs. washington.edu/research/xmldatasets/

8 The XML Benchmark Project. http://www.xml-benchmark.org/