# VEMBP:支持更新的XML树编码方法

覃遵跃[1] 蔡国民[1] 张彬连[1] 汤 庸[2]
(吉首大学软件服务外包学院 张家界 427000)[1] (华南师范大学计算机学院 广州 510631)[2]

**摘 要** 对有序XML文档树进行编码,不需要访问XML原始文件就能够实现对XML数据的管理,提高了XML管理系统的效率。针对查询提出的编码方案具有很高的查询性能,但更新效率很低。为提高更新性能而设计的方案存在查询效率低或者编码空间大等问题。为了在提高更新XML文档效率的同时不对查询性能和编码空间产生负面影响,提出了一种新的编码方法VEMBP(Vector Encoding Method Based of Prime),该方法利用向量表示有序XML节点之间的顺序关系,采用素数表示有序XML文档节点之间的结构信息;并设计了一种算法来实现在没有牺牲查询性能的前提下完全避免更新过程中的二次编码和重新计算,降低了更新代价,同时编码空间也得到了控制。实验结果显示,VEMBP具有较好的查询和更新性能。

**关键词** XML树,VEMBP编码,查询,更新
**中图法分类号** TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.2.034

## VEMBP:A Novel Labeling Method for Updating on XML Data

QIN Zun-yue[1] CAI Guo-ming[1] ZHANG Bin-lian[1] TANG Yong[2]
(School of Software & Service Outsourcing,Jishou University,Zhangjiajie 427000,China)[1]
(School of Computer Science,South China Normal University,Guangzhou 510631,China)[2]

**Abstract** In order to improve the efficiency of the XML management system, some labeling schemas for the orderly XML tree were put forward, which realize processing of XML data under the condition of no need to access the original XML tree. The proposed labeling schemas for query have higher query performance, but the updating performance is poorer. Some novel labeling schemas designed for updating permance sacrifice query efficiency and possesse larger labeling space. For higher updating efficiency and smaller labeling space at the same time no reducing query efficiency, a novel labeling schema called VEMBP(Vector Encoding Method Based of Prime) was proposed, in which vector indicates order relation and a prime indicates structure relation between nodes in XML tree, and then an algorithm was designed which realizes updating in the cases no sacrificing query efficiency and completely avoiding secondary coding. Meanwhile labeling space is also under control. The experimental results show that VEMBP processes better on updating efficiency without sacrificing query performance.

**Keywords** XML tree, VEMBP labeling, Query, Updating

## 1 前言

随着互联网在电子商务、电子政务等各个领域的广泛深入应用,作为Internet事实上的数据交换标准语言,XML得到了迅速的发展并且规模也越来越大。目前提出了很多针对有序XML树的编码方案,在此基础上不需要访问XML原始文件就能够实现对XML数据的查询、更新和修改,提高了XML数据管理系统的效率。

针对有序XML文档查询,主要有路径编码方案和区间编码方案。Dewey编码[1]是一种典型的路径编码方案,这种编码方案在支持有序XML树查询方面具有较高的性能,但当插入节点时,因为需要对很多节点进行二次编码,所以更新代价很高;Li-Moon编码[2]是一种典型的区间编码,该编码方案为有序XML文档的每个节点分配 *start* 和 *end* 两个值,对于节点 $x$ 和 $y$,如果 $start(x)<start(y)$ 并且 $end(x)>end(y)$,则 $x$ 是 $y$ 的祖先节点,这种编码方案采用结构连接算法,获得了较高的查询效率,但插入新节点时也需要对很多其它节点进行二次编码从而降低了更新效率。文献[9]提出概率编码策略DeweyTP,该方案为每个XML节点分配唯一的能够体现节点类型和路径概率的编码,从而减少查询时间并降低存储空间,但DeweyTP方案也不能有效支持XML文档的插入更新计算。

为了避免向有序XML文档插入新节点时对其它节点进行二次编码,提高更新效率,Wu等人提出了素数编码方案Prime[3],在插入新节点时,该方案通过计算节点的SC(Simultaneous Congruence)值来维护节点之间的有序关系,计算

到稿日期:2014-03-27 返修日期:2014-07-11 本文受国家863计划重大项目(2013AA01A212),国家科技支撑计划课题(2012BAH27F05),国家自然科学基金(61363073)资助。

**覃遵跃**(1974-),男,博士生,副教授,主要研究方向为Web数据库技术,E-mail:qzystudy@163.com;**蔡国民**(1976-),男,硕士,讲师,主要研究方向为计算机网络;**张彬连**(1978-),女,硕士,讲师,主要研究方向为计算机应用;**汤 庸**(1964-),男,教授,博士生导师,CCF理事,主要研究方向为数据库、软件工程、协同计算和云服务软件。

SC 值需要耗费 O($n$)的时间复杂度，在总体上增加了插入一个新节点的时间；Neil 等人提出了 OrdPath 编码方案[4]，该编码方案初始化采用奇数表示节点的 self_label 编码，并采用路径方案来表示节点的结构关系，插入新节点后利用预留的偶数来为新节点进行编码。这种方法虽然避免了二次编码以及重新计算，但由于改变了路径编码的性质，因此牺牲了查询性能，总的更新代价也偏高；2009 年 Xu 等人对 Dewey 编码方案进行了扩展，提出了 DDE 编码方案[5]，该编码方案改变了路径方案中通过前缀关系快速判断节点之间结构关系的属性，虽然有效避免了重新编码和重新计算，但判断节点之间结构关系的时间复杂度为 O($n$)($n$ 为路径长度)，降低了查询性能，不适合大规模 XML 数据的查询处理；2010 年，Ko H 等人在 CDBS 的基础上提出了 IBSL 编码方案[6]，该方案采用无穷有序的二进制字符串 BS(Binary String)来表达节点编码，通过算法实现了高效的插入更新计算，但在有序 XML 文档扇出较大的情况下，IBSL 编码长度非常可观，编码长度过长对查询性能产生负面影响。文献[10] 根据二进制小数的特性提出了新的 XML 树编码方案 BSC(Binary Symbol Coding)，该特性既有前缀编码的高效的查询性能，也支持动态更新，但该编码方案在 XML 文档扇出较大或者频繁插入新节点情况下，编码空间的规模增长很快；为了保持 XML 文档高效的查询性能，罗道峰等人[13]根据有序 XML 文档节点数之间的关系在路径编码的基础上提出了预留策略来支持插入更新计算，该预留策略在插入新节点数较少情况下不需要对其它节点进行二次编码，但在极端插入较多新节点下则不能避免二次编码。

2011 年，Yi Jiang 等人提出了连续分数编码方法 CFE (Continued Fraction-based Encoding)[7]，基于该方法构建了基于区间的编码方案 Range-CFE 和基于前缀的编码方案 Prefix-CFE，虽然采用 CFE 编码方法能够避免插入时的二次编码和重新计算，但如文章的引理所证明的一样，判断节点之间的有序关系是一个 O($n$)的时间复杂度，因此该编码方法也牺牲了查询性能。2012 年 Jian Liu 等人提出了动态浮点数 Dewey 编码方案 DFPD (Dynamic Float-point Dewey)[8]，该编码方案对 Dewey 编码进行了扩展，采用浮点数表示节点的 self_label，并采用复杂的算法实现了插入更新计算，虽然完全避免了二次编码和重新计算，但与 DDE 一样，由于改变了前缀编码的结构性质因此也牺牲了查询性能。2013 年，Zhi-Hong Deng 等人针对有序 XML 文档提出了 LAF[12] 编码方法，该方法利用数据库系统存储 XML 节点编码信息提高了检索 XML 关键字的效率，但不能实现高效更新计算。

已经提出的编码方案例如 Dewey 和 Li-Moon 编码虽然查询性能很高，但由于插入新节点需要对很多其它节点进行编码，具有很低的更新效率；为了有效处理更新问题所提出的 IBSL 编码方案存在存储空间过大的问题，并且对查询性能也有负面影响。DDE、DFPD 和 CFE 虽然能够有效处理更新问题，但因为其改变了编码的前缀性质，所以也是以牺牲查询性能为代价。

本文提出了一种新的编码方案 VEMBP(Vector Encoding Method Based of Prime)，该编码方案采用素数表示节点之间的结构关系，初始化时采用整数表示兄弟节点之间的顺序关系，利用向量方法避免插入新节点时的二次编码和重新计算，提高了更新效率，并且具有较小的编码空间。

## 2 VEMBP 编码方案

### 2.1 基本概念

**定义 1**(向量) 既有大小又有方向且遵循平行四边形定则的量叫做向量。

在平面直角坐标系中，分别取与 $x$ 轴、$y$ 轴方向相同的两个单位向量 $i$、$j$ 作为一组基底。$a$ 为平面直角坐标系内的任意向量，以坐标原点 $O$ 为起点作向量 $OP=a$。由平面向量基本定理知，有且只有一对实数$(x,y)$，使得 $a$=向量 $OP=x_i+y_j$，向量 $a$ 的坐标表示 $a=(x,y)$，其中$(x,y)$是点 $P$ 的坐标。

**定义 2**(向量加法) 向量的加法满足平行四边形法则和三角形法则。有 $a$、$b$ 两个向量，向量 $a=(x,y)$，$b=(x',y')$，则 $a+b=(x+x',y+y')$。

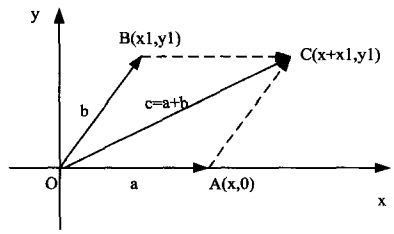


图 1 向量的加法

**定义 3**(向量的角度) 平面直角坐标系中，以坐标原点 $O$ 作为起点的向量 $a$，$a=(x,y)$，则向量 $a$ 的角度 $\sin(a)=y/sqrt(x^2+y^2)$。

在图 1 中，向量 $b$ 的角度 $\sin(b)=y1/sqrt(x1^2+y1^2)$，向量 $c$ 的角度 $\sin(c)=y1/sqrt((x+x1)^2+y1^2)$。

**定义 4**(有序向量 Vector) 对若干个有序节点，第一个节点的向量 $a_1=(1,0)$，第二个节点的向量 $a_2=(2,0)$，…，第 $n$ 个节点的向量 $a_n=(n,0)$。

**定义 5**(向量二元组) 一个向量二元组 $\beta=$(向量序号 $no$，向量 $Vector$)。例如有 4 个有序节点，$A$ 是第 1 个节点，$B$ 是第 2 个节点，$C$ 是第 3 个节点，$D$ 是第 4 个节点，则它们的向量二元组分为是 $\beta_A=(1,(1,0))$，$\beta_B=(2,(2,0))$、$\beta_C=(3,(3,0))$、$\beta_D=(4,(4,0))$。

**定义 6**(向量序$\prec$) 判断两个不同的向量二元组 $\beta$ 和 $\gamma$ 的向量序执行如下算法：

(1)如果 $order(\beta)\prec order(\gamma)$，则 $\beta\prec\gamma$，如果 $order(\gamma)\prec order(\beta)$，则 $\gamma\prec\beta$；否则即 $order(\beta)=order(\gamma)$，然后执行第(2)步；

(2)如果 $\sin(Vector(\beta))\prec\sin(Vector(\gamma))$，则 $\beta\prec\gamma$；如果 $\sin(Vector(\gamma))\prec\sin(Vector(\beta))$，则 $\gamma\prec\beta$。

注：$order(\beta)$计算向量二元组 $\beta$ 的向量序号 $no$；$\sin(Vector(\beta))$计算向量二元组 $\beta$ 的角度。

例如，图 2 所示 5 个有序节点，根据定义 6(1)，因为 $order(a)=1$，$order(b)=2$，$order(c)=3$，$order(d)=4$，$order(e)=5$，所以它们的向量序为 $a\prec b\prec c\prec d\prec e$。

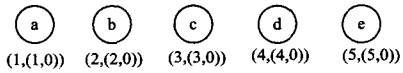


图 2 向量序号不同的有序向量二元组

图 3 显示了向量二元组的向量序号相同的 5 个有序节点，根据定义 6(2)，$\sin(Vector(h)<\sin(Vector(n))<\sin(Vector(k))<\sin(Vector(m))<\sin(Vector(j))$，因此它们的向量

序为 $h \prec n \prec k \prec m \prec j$。

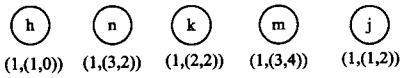


图 3　向量序号相同的有序向量二元组

## 2.2　VEMBP 方案

**定义 7**(VEMBP 编码方案)　对有序 XML 文档，节点之间有结构关系和顺序关系，结构关系通过素数编码方案确定，兄弟节点之间的顺序关系通过向量二元组确定。有序 XML 树的节点 $v$ 的 VEMBP 编码为 3 元组，即 VEMBP($v$)=(节点 $v$ 的结构信息，节点 $v$ 在兄弟节点中的序号，节点 $v$ 的向量二元组)。

图 4 中，节点 $c$ 的 VEMBP 编码 VEMBP($c$)=(1×5,2,(0,2))，则节点 $c$ 结构信息为素数 5，节点 $c$ 在兄弟节点中的序号为 2，向量二元组为(2,0)；节点 $e$ 的 VEMBP 编码 VEMBP($e$)=(1×5×11,1,(1,0))，节点 $f$ 的 VEMBP 编码 VEMBP($f$)=(1×5×13,2,(2,0))，因为 Structure(VEMBP($e$))/Structure(VEMBP($c$))为整数 11，所以 $c$ 是 $e$ 的双亲，Structure(VEMBP($f$))/Structure(VEMBP($c$))为整数 13，因此 $c$ 也是 $f$ 的双亲，及 $e$ 和 $f$ 是兄弟，又因为 order(VEMBP($e$))<order(VEMBP($f$))，所以 $e$ 是 $f$ 的左兄弟。

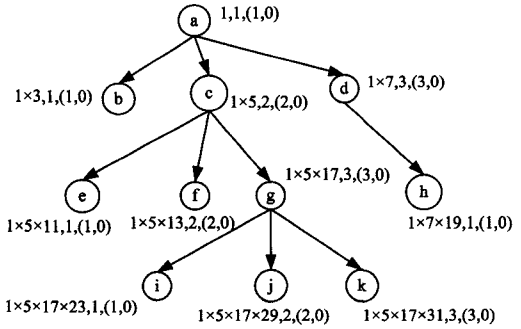


图 4　有序 XML 树的 VEMBP 编码

定义 7 确定的 VEMBP 编码方案中，结构信息利用素数表达，兄弟节点之间的关系用向量二元组表达，能够确定节点之间的结构关系和顺序关系，满足了查询要求，但对有序 XML 文档的操作，除了查询操作还有删除节点、更新节点和插入节点操作，删除节点(该操作将删除其节点的所有子孙节点)和更新节点(该操作更新节点内容)不影响节点之间的结构关系和顺序关系，因此不需要对其它节点进行二次编码，但插入新节点会改变节点之间的有序关系，采用算法 1 可以在不对其它节点进行二次编码的基础上完成更新操作，提高更新效率。

算法 1 在有序 XML 树插入新节点，维持该 XML 树节点之间的有序关系。

**算法 1**　insertNodeInXML(XML, $VEMBP_{left}$, $VEMBP_{right}$)//在有序 XML 文档中插入新节点

输入：有序 XML 树，$VEMBP_{left}$ 和 $VEMBP_{right}$

输出：新的有序 XML 树

1. begin
2. if($VEMBP_{left}$ is not null and $VEMBP_{right}$ is null)
   //左边不为空，右边为空，即在最右边插入新节点
3. Structure($VEMBP_{new}$) = Structure(Parent($VEMBP_{left}$)) × $Prime_{new}$
   //新节点的结构信息
4. order($VEMBP_{new}$) = order($VEMBP_{left}$) + 1;//新节点的向量序号
5. Vector($VEMBP_{new}$) = (order($VEMBP_{new}$),0)
6. else if($VEMBP_{left}$ is null and $VEMBP_{right}$ is not null)
   //左边为空，右边不为空，即在最左边插入新节点
7. Structure($VEMBP_{new}$) = Structure(Parent($VEMBP_{right}$)) × $Prime_{new}$
   //新节点的结构信息
8. if(order($VEMBP_{right}$) == 1)//右边序号为 1
9. order($VEMBP_{new}$) = −1;//新节点的向量序号
10. else//右边序号不为 1
11. order($VEMBP_{new}$) = order($VEMBP_{right}$) − 1;
    //新节点的向量序号
12. Vector($VEMBP_{new}$) = (ABS(order($VEMBP_{new}$)),0)
13. else//左边和右边都不为空，即在两个节点中间插入新节点
14. insert($VEMBP_{left}$, $VEMBP_{right}$)
15. return 有序 XML 树;
16. end

算法 2 在两个有序节点的中间插入新的节点，同时保持这 3 个节点之间的向量序关系。

**算法 2**　insertNode($VEMBP_{left}$, $VEMBP_{right}$)//在两个有序节点的中间插入新的节点，$VEMBP_{left}$ 和 $VEMBP_{right}$ 都不为空

输入：插入位置 $VEMBP_{left}$ 和 $VEMBP_{right}$

输出：新节点的编码 $VEMBP_{new}$

1. begin
2. if(order($VEMBP_{left}$) == order($VEMBP_{right}$))
   //两个有序节点的向量序号相同，表示在同一个坐标系
3. Structure($VEMBP_{new}$) = Structure(Parent($VEMBP_{left}$)) × $Prime_{new}$
   //新节点的结构信息
4. order($VEMBP_{new}$) = order($VEMBP_{left}$);//新节点的向量序号
5. X-axis(Vector($VEMBP_{new}$)) = X-axis($VEMBP_{left}$) + X-axis($VEMBP_{right}$)
   //新节点横坐标的值
6. Y-axis(Vector($VEMBP_{new}$)) = Y-axis($VEMBP_{left}$) + Y-axis($VEMBP_{right}$)
   //新节点纵坐标的值
7. else //两个有序节点的向量序号不相同，表示不在同一个坐标系
8. if(order($VEMBP_{right}$) >= order($VEMBP_{left}$) + 2)
   //右边的向量序号与左边的向量序号的距离大于 1
9. Structure($VEMBP_{new}$) = Structure(Parent($VEMBP_{left}$)) × $Prime_{new}$
   //新节点的结构信息
10. order($VEMBP_{new}$) = order($VEMBP_{left}$) + 1;
    //新节点的向量序号
11. Vector($VEMBP_{new}$) = (order($VEMBP_{new}$),0)
    //新节点的向量
12. else//右边的向量序号与左边的向量序号的距离等于 1
13. Structure($VEMBP_{new}$) = Structure(Parent($VEMBP_{left}$)) × $Prime_{new}$
    //新节点的结构信息
14. order($VEMBP_{new}$) = order($VEMBP_{left}$)
    //新节点的向量序号

15. X-axis(Vector($VEMBP_{new}$))=X-axis($VEMBP_{left}$)
//新节点横坐标的值
16. Y-axis(Vector($VEMBP_{new}$))= Y-axis($VEMBP_{left}$)+order($VEMBP_{left}$)
//新节点纵坐标的值
17. return $VEMBP_{new}$
18. end

根据算法2，在同一个直角坐标系的第一象限中，任意两个以坐标原点 $O$ 为起点的向量 $a$ 和 $b$，它们的和 $c=(a+b)$，如果 $\sin(a)<\sin(b)$，则 $\sin(a)<\sin(c)<\sin(b)$，如图5所示。

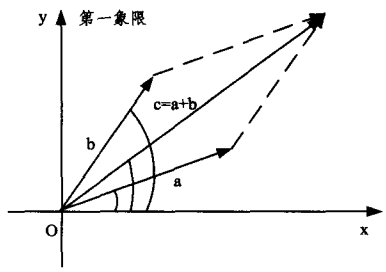


图5 第一象限向量内向量之和

为了说明插入新节点的各种不同情况，图6所示的有序XML树的VEMBP编码方案是假设已经删除了节点 $j$ 与 $k$ 之间序号为3、4、5、6、7、8之后的情况。(1)插入新节点 $x1$，根据算法1的第6行—14行，在最左边插入新节点，$VEMBP_{new}=(1\times5\times37,-1,(1,0))$；(2)插入新节点 $x2$，根据算法1的13行，在两个节点之间插入新节点，根据算法2的第12行—16行，$VEMBP_{x2}=(1\times5\times41,2,(2,3))$；(3)插入新节点 $x3$，在最右边插入新节点，根据算法1的第2—5行，$VEMBP_{x3}=(1\times5\times43,4,(4,0))$；(4)插入新节点 $x4$，因为节点 $j$ 与 $k$ 之间的向量序号的距离等于7(即9—2)，因此根据算法1的第13行、算法2的8行—11行，即 $VEMBP_{x4}=(1\times5\times17\times47,3,(3,0))$；(5)插入新节点 $x5$，因为节点 $f$ 与 $x2$ 都在象限为2的空间(即 $order(f)=order(x2)=2$)，因此按照算法2的第2—6行，$order(x5)=2$，$Vector_{x5}=(2+2,0+2)$，即 $Vector_{x5}=(4,2)$，根据定义6的第二种情况，满足 $f<x5<x2$，因此不需要对其它节点进行二次编码就实现了插入新节点。

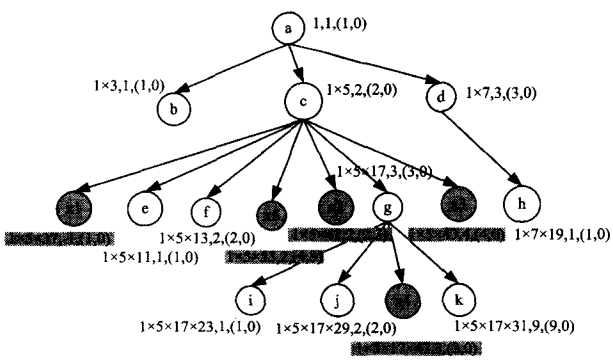


图6 在有序XML树的不同位置插入新节点

## 3 实验分析

为了全面验证VEMBP编码方案，通过实验与其它的编码方案在编码空间、查询性能和更新性能进行了对比。实验环境为Intel CPU G640 2.8GHz，3GB内存，Windows 7操作系统，采用JDK1.6作为开发工具，Oracle 10g作为数据库存储平台。实验数据集如表1所列。

表1 测试数据集

| 数据集 | 主题 | 节点数 | 平均深度 | 平均扇出 |
|---|---|---|---|---|
| D1 | Club | 340 | 3 | 12 |
| D2 | XMark1 | 5679 | 6 | 56 |
| D3 | SigmondRecord | 11526 | 5 | 20 |
| D4 | XMark2 | 15429 | 6 | 78 |
| D5 | Shakespear plays | 179689 | 5 | 48 |

### 3.1 编码空间分析

图7显示了各种不同编码方案节点的平均编码大小。随着节点数的增加，IBSL编码长度逐渐增大，因为IBSL编码的平均长度随着扇出的增大而增大，DDE与Dewey编码采用了相同的编码方案，所以它们的长度一致；OrdPath采用奇数来初始化编码，所以长度要大于DDE和Dewey；Prime编码的初始长度也不大，但为了维护节点之间的有序关系而需要额外的存储空间，所以总的编码长度较大；VEMBP采用三元组进行编码，每个节点的编码长度为20个字节(素数采用长整型表示占8个字节，向量序号采用整型表示占4个字节，向量采用整型表示占8个字节)，并且编码长度不会随着XML树深度和扇出的增加而增大，因此在节点数较少和深度较小时，DDE、Dewey和IBSL的编码长度小于VEMBP，但当扇出和深度增大时，VEMBP的编码空间相对其它编码方案就具有很大的优势。

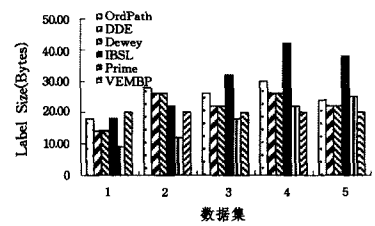


图7 编码空间

### 3.2 查询性能分析

为了测试查询性能，利用D2作为查询测试数据集，表2作为查询测试语句。主要测试结构关系查询、有序关系查询。

表2 查询测试语句

| 序号 | 查询语句 | 查询特征 |
|---|---|---|
| Q1 | /site/catgraph/edge/from=55 | 精确查询 |
| Q2 | /site/categories/dategory[10] | 顺序查询 |
| Q3 | /site/people//person | 正则表达式匹配 |
| Q4 | /site/closed_auctions/close_autction[10 to 15] | 确定兄弟查询范围 |
| Q5 | /site/open_auctions//* | 查询后代轴 |

首先把D2数据集的每个节点转换成对应的VEMBP编码，然后存储在实验数据库中，并利用SQL语言进行查询。对每个查询语句运行11次，取第2-11次运行结果的平均时间作为测试结果，图8显示了各种查询测试语句在数据集D2上的查询时间。DDE、OrdPath和Prime的查询时间较长，因为DDE、OrdPath改变了路径编码的性质，从而导致判断节点之间结构关系比较复杂，而Prime在判断节点之间的有序关系时需要计算SC；虽然IBSL编码利用路径关系可以直接判断节点的关系，但由于它的编码长度较大，因此对查询性能产生了负面影响；VEMBP的查询性能好于其它几种编码方案，因为该编码方案判断结构关系和顺序关系时，只需要对两个VEMBP编码进行常数时间的比较计算。

(下转第181页)

tions on Automatic Control, 2001, 46(2): 353-354
[2] 赵相福,欧阳丹彤. 离散事件系统基于模型诊断的研究进展[J]. 计算机科学与探索,2011,5(2):114-127
[3] Sampath M, et al. Diagnosability of discrete-event systems [J]. IEEE Transactions on Automatic Control, 1995, 40(9): 1555-1575
[4] Contant O, et al. Diagnosability of Discrete Event Systems with Modular Structure[J]. Discrete Event Dynamic Systems, 2006, 16(1): 9-37
[5] Zhou C, et al. Decentralized modular diagnosis of concurrent discrete event systems[C]// 9th International Workshop on Discrete Event Systems. Goteborg, Sweden, 2008: 388-393
[6] Lafortune S, Chen E. A Relational Algebraic Approach to the Representation and Analysis of Discrete Event Systems[C]// Proceedings of American Control Conference. Boston, MA, USA, 1991: 2893-2898
[7] 单锦辉,徐克俊,王戟. 一种软件故障诊断过程框架[J]. 计算机学报,2011,34(2):371-382
[8] 朱荣,徐拾义. 软件测试中故障模型的建立[J]. 计算机工程与应用,2003,39(17):69-71
[9] Kwong R H, Yonge-Mallo D L. Fault Diagnosis in Discrete-Event Systems: Incomplete Models and Learning [J]. IEEE Transactions on Systems, Man and Cybernetics, 2011, 41(1): 118-130
[10] Fijany A, Barrett A C, Vatan F. A fast model-based diagnosis engine[C]// 2012 IEEE Aerospace Conference. Big Sky, MT, 2012: 1-11
[11] Mahulea C, et al. Fault Diagnosis of Discrete-Event Systems Using Continuous Petri Nets[J]. IEEE Transactions on Systems, Man and Cybernetics, 2012, 42(4): 970-984
[12] 王晓宇,欧阳丹彤,赵剑. 不完备模型下的离散事件系统诊断方法[J]. 软件学报,2012,23(3):465-475

(上接第 160 页)

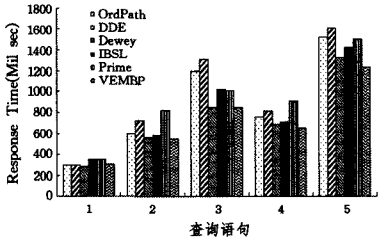


图 8 查询测试

### 3.3 更新性能分析

为了充分测试 VEMBP 编码方案的插入更新性能,利用 D2 作为测试数据集,在 site/people[1]节点之后分别连续插入 10、20、50、80、100 个节点,更新时间如图 9 所示。Dewey 编码的更新时间最长,因为为了维持有序 XML 文档的节点之间的有序关系,该编码需要对插入位置之后的节点以及它的后裔节点进行二次编码;OrdPath、DDE 虽然不需要进行二次编码,但因为它们的查询定位时间长,所以总的更新时间也较长;Prime 编码不需要进行二次编码,但为了维持节点的有序关系需要修改关系索引表而进行 SC 的计算,所以总的更新性能也较低;IBSL 编码更新性能低于 VEMBP 的主要原因是因为 IBSL 编码长度大于 VEMBP 而消耗了更多的查询时间;VEMBP 方案具有最好的更新性能,因为该方案不需要像 Dewey 方案进行二次编码,也不像 DDE 等方案进行复杂的结构关系判断。

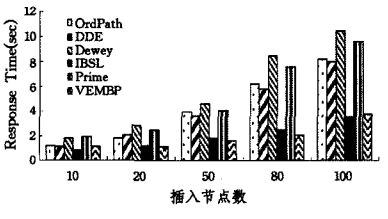


图 9 插入新节点

**结束语** 本文分析了已存在的支持有序 XML 文档操作的主要编码方案的特性,并指出了它们的优点和缺点,提出了 VEMBP 编码方法。该方法利用向量二元组表示节点的顺序关系,采用素数表示节点之间的结构关系,该方法在没有牺牲查询性能的前提下,取得了很好的更新性能,并且编码空间也得到了有效约束。下一步的工作是在该编码方案的基础上研究如何进行 XML 关键字检索。

## 参 考 文 献

[1] Tatarinov S, Viglas D, Beyer K J, et al. Storing and Querying Ordered XML Using a Relational Database System[C]// Proc of the ACM SIGMOD 2002. Los Alamitos, CA: IEEE Computer Society, 2002: 204-215
[2] Li Q, Moon B. Indexing and Querying XML Data for Regular Path Expressions[C]// Proc of the 27th Intl Conf Very Large Data Bases (VLDB). New York: ACM, 2001: 361-370
[3] Wu X, Lee M, Hsu W. A Prime Number Labeling Scheme for Dynamic Ordered XML trees[C]// Proc of the 20th Int Conf Data Engineering (ICDE'04). Los Alamitos, CA: IEEE Computer Society, 2004: 66-78
[4] O'Neil P, O'Neil E, Pal S, et al. ORDPATHs: Insert-Friendly XML Node Labels[C]// Proc of ACM SIGMOD 2004. Los Alamitos, CA: IEEE Computer Society, 2004: 903-908
[5] Xu L, Ling T W, Wu H, et al. DDE: From Dewey to a Fully Dynamic XML Labeling Scheme[C]// Proc. of the 35th SIGMOD International Conference on Management of Data. 2009: 719-730
[6] Ko H, Lee S. A Binary String Approach for Updates[J]. IEEE Transactions on Knowledge and Data Engineering, 2010, 22(4): 602-608
[7] Jiang Yi, He Xiang-jian, Lin Fan, et al. An Encoding and Labeling Scheme Based on Continued Fraction for Dynamic XML[J]. Journal of Software, 2011, 6(10): 2043-2049
[8] Liu Jian, Maa Z M, Li Yan. Efficient labeling scheme for dynamic XML trees[J]. LNCS, 2007, 4653: 515-161
[9] 陈子阳,刘佳,张刘辉,等. DeweyTP:一种面向概率 XML 数据的编码方案[J]. 通信学报,2013,34(11):26-32
[10] 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
[11] 周军锋,孟小峰. XML 关键字查询处理研究[J]. 计算机学报,2012,35(12):2459-2478
[12] Deng Hi-hong, Xiang Yong-qing, Gao Ning. LAF: a new XML encoding and indexing strategy for keyword-based XML search [J]. Concurrency and Computation: Practice and Experience, 2013, 25(11): 1604-1621
[13] 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
[14] 覃遵跃,黄云,蔡国民,等. 支持 XML 插入更新的编码方法[J]. 计算机应用,2012,32(12):3540-3543