一种新的支持XML文档更新的编码方法

付 鹏 蒋夏军 皮德常
(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 提出了一种新的支持XML文档更新的编码方法——DVLS(Dynamic Vector Labeling Scheme)。DVLS仅由3个向量组成,克服了传统前缀编码中编码长度随着XML文档树深度的增加而增长的缺陷,其主要思想是:利用向量的加法来支持XML节点数据的更新,并分别针对静态和动态XML文档提出优化方案,以提高查询效率。在向量序的基础上,通过与DDE编码的对比实验,验证了DVLS编码的高效性。

关键词 可扩展标记语言,前缀编码,动态向量编码方案,向量,XML文档更新
中图法分类号 TP311 **文献标识码** A

Novel Labeling Scheme for XML Update-supporting

FU Peng JIANG Xia-jun PI De-chang
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract A novel labeling scheme called DVLS(Dynamic Vector Labeling Scheme) was proposed to process update in dynamic XML data. DVLS is made up of three vector codes for overcoming the default of increace of the label length with the depth of the XML document tree that the trandtional prefix shceme can't avoid. The main idea of DVLS is that it makes use of vector addition to support the update of the XML node data and can simplify for not only static but also dynamic XML document in order to improve the efficiency of the query. The comparative experiments on DDE and DVLS confirm that the DVLS is more efficient based on vector order.

Keywords XML, Prefix labeling scheme, Dynamic vector labeling scheme, Vector, XML document update

1 概述

随着信息技术的发展,XML(extensible markup language)技术在网络环境中的应用愈加广泛。如何对XML数据进行高效的索引和查询,成为当前研究的热点之一。对XML树按照某种方法进行编码,是目前大多数针对XML数据索引和查询技术的基础[1],其中比较常用的一类是前缀编码。

前缀编码方法又称为基于路径的编码方法,其父节点的编码是孩子节点编码的前缀。传统的静态前缀编码方法,如文献[2-4],通过对编码进行比较,进而支持节点间位置关系及结构关系的计算,其中比较典型的是Dewey编码[2]。但是传统的前缀编码不能很好地支持XML文档的动态更新,并且随着XML文档深度的增加,编码的长度也会增长,影响了查询的效率。

近年来针对前缀编码方法的研究,主要集中在使其如何支持动态XML文档方面,如LSDX编码[5]、ORDERPATH编码[6]、FPES[7]编码等,但是当文档不更新或者少更新时,其查询效率不及Dewey编码等静态前缀编码方法。DDE编码[8]利用向量对Dewey编码进行扩展,不仅保持了Dewey编码对静态XML文档的处理性能,而且能够支持XML文档的动态更新;IDD[9]中针对DDE编码如何利用已删除节点编码问题进行了改进。但是上述编码仍未解决编码长度随XML文档深度增加而增大的问题。

本文提出了一种新的前缀编码方法——DVLS(Dynamic Vector Labeling Scheme),该方法不仅能够有效支持XML文档的动态更新,而且针对前缀编码随着XML文档深度的增加编码长度也增长这一缺陷进行改进,通过实验与DDE编码进行比较,验证了其静态性能及动态性能方面的高效性。

2 相关研究

2.1 静态前缀编码

在传统的静态前缀编码中,比较典型的就是Dewey编码,给定一个节点 v,令其编码为 $L(v)$,则其第 i 个孩子节点 u_i 的编码为 $L(u_i)=L(v).i$,其中"."为连接符,如图1所示。

给出两个Dewey编码 $A:a_1.a_2.a_3.\cdots.a_m$ 和 $B:b_1.b_2.b_3.\cdots.b_n$,按照自然序"<"和"="对两个编码进行逐位比较,可以判断出节点间是否具有祖先后代(AD)关系、父子(PC)关系以及兄弟(Sibling)关系;按照字典序(Lexicographical order)对两个编码进行比较,可以判断出节点之间的位置(Document Order)关系。

到稿日期:2013-05-25 返修日期:2013-07-21 本文受航空科学基金(20111052010)资助。
付 鹏(1984-),男,硕士,主要研究方向为XML文档编码,E-mail:fp3696@sina.com;蒋夏军(1973-),男,博士,讲师,主要研究方向为时空数据库、计算机仿真等;皮德常(1971-),男,博士,教授,博士生导师,主要研究方向为数据挖掘、数据库技术等。

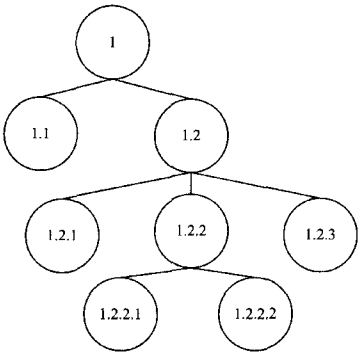


图 1　Dewey 编码

Dewey 编码的查询效率较高，但是不支持更新，并且随着 XML 文档深度的增加编码的长度也会增长，影响了查询的效率。

2.2　动态前缀编码

为了支持 XML 文档的动态更新，一些新的编码方法被提出来。

LSDX 编码按照路径为 XML 文档节点赋予相应的数字和字符组合。在插入新节点时，按照一定的规则在相关节点编码上添加相应的字符作为新节点的编码。该编码虽然能够支持文档的动态更新，但是在进行各种关系的判断时需要对字符进行比较，查询效率要差于利用数字进行编码的编码方式。

ORDERPATH 编码以 Dewey 编码为基础，利用奇数进行初始编码，在更新时再利用奇数之间的偶数来支持节点插入。它能够部分解决更新操作造成的重新编码问题，但是其编码规模很大，查询效率不高，并且会出现编码溢出问题。

上述两种编码虽然能够支持 XML 文档的动态更新，但在静态查询效率上不及 Dewey 编码，在插入节点较多时会出现编码溢出问题，并且均未解决编码长度随着 XML 文档深度的增大而增长的缺陷。

DDE 编码在静态时与 Dewey 编码相同。在动态更新时，DDE 编码利用向量的加法支持节点插入，例如在相邻兄弟节点 $A:a_1.a_2.a_3.\cdots.a_n$ 和 $B:b_1.b_2.b_3.\cdots.b_n$ 之间插入新节点，则新节点编码为 $A+B:(a_1+b_1).(a_2+b_2).(a_3+b_3)\cdots(a_n+b_n)$。同时，为了支持动态更新后节点间关系的判断，DDE 编码对字典序的先于“$\prec_d$”的关系也进行了重新定义。

当且仅当满足下列两个条件之一时，DDE 编码 $A\prec_d B$：

- $m<n$，并且 $a_1/b_1=a_2/b_2=\cdots=a_m/b_m$；
- 存在 $k\in[0,\min(m,n)]$，使得 $a_1/b_1=a_2/b_2=\cdots=a_{k-1}/b_{k-1}$ 并且 $a_k\times b_1<b_k\times a_1$。

利用向量的加法以及新的序关系，DDE 编码在保持 Dewey 编码静态查询高效性的基础上，又能有效地支持 XML 文档的动态更新，并且有效避免了编码的溢出问题。IDD 编码在 DDE 编码的基础上利用 Stern-Brocot 树，对删除节点的编码进行了有效的利用。但是，这两种编码方式也没有解决编码长度随文档深度的增大而增长的缺陷，随着 XML 文档深度的增加，动态效率也受到影响。

2.3　向量和向量序(Vector Order)

文献[10]中将向量和向量序应用至 DDE 编码等各种编码方式中，有效提高了各种编码的动态性能，这也是本文所提出编码的基础。

2.3.1　向量码和向量序

这里所提出的向量是一个形如(x,y)的二元组，其中 $x>0$。在坐标系中，向量(x,y)可以用一条以原点为起点，以(x,y)的有向线段为终点来表示，因为 $x>0$，所以所有的向量码均落在坐标系的一、四象限。图 2 中给出了向量码(3,2)和(2,3)的示例。

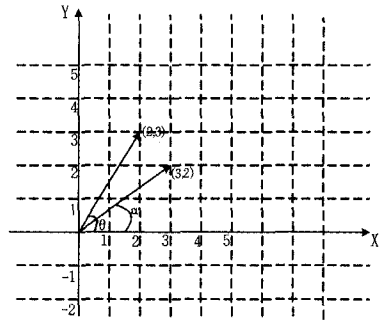


图 2　向量码和向量序

对两个向量进行大小的比较，需要利用向量序，即通过比较向量码与 X 轴夹角的正切值(“tan”值)来确定向量码之间的大小关系，并且用“$\prec$”表示“向量序小于”，用“$=$”表示“向量序等于”。因为向量与 X 轴的夹角范围为$-90°\sim90°$，所以对应的 tan 值的范围就为$-\infty\sim+\infty$。

例如，图 2 中向量码(3,2)与 X 轴的夹角为 α，向量码(2,3)与 X 轴的夹角为 θ，$\tan\alpha=2/3$，$\tan\theta=3/2$。显然，$\tan\alpha<\tan\theta$，因此$(3,2)\prec(2,3)$。

2.3.2　向量加法(Vector addition)

两个向量相加，即是将两者的横坐标和纵坐标分别相加，例如：给出两个向量 $A(x_1,y_1)$和 $B(x_2,y_2)$，则 $A+B=(x_1+x_2,y_1+y_2)$。相加之和为这两个向量码所组成平行四边形的对角线，所以两个向量相加之和永远在这两个向量之间，如图 3 所示。

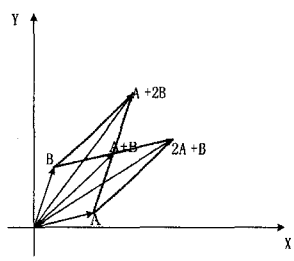


图 3　向量码的加法

再按照向量序进行比较即可得到：

给出两个向量 A 和 B，如果 $A\prec B$，那么 $A\prec\cdots3A+B\prec2A+B\prec A+B\prec A+2B\prec A+3B\prec\cdots\prec B$。

利用向量的加法，可以在两个向量之间插入无限个向量，并且所插入的向量在初始的两个向量之间，按照向量序“$\prec$”有序排列。这一性质，也正是通过向量能够支持 XML 文档动态更新的基础。

3　DVLS 编码

3.1　初始编码

如图 4 所示，DVLS 编码机制中，XML 文档的节点(除根

节点外)u 的编码 $L(u)$可以用“*Vlabeling*. *Vparentlabeling*. *Vlocalorder*”的形式来表示,其中以分隔符“.”所划分的每一部分均为一个向量(x,y),并且初始时所有“x” 的值均为“1”。“*Vlabeling*”是节点唯一标记,初始时其“y”值为深度优先遍历 XML 文档时所得到的数值,其中根节点为“1”,每访问下一个节点时其值加“1”;“*Vparentlabeling*”为该节点的父节点的“*Vlabeling*”;“*Vlocalorder*”为该节点在其兄弟节点中所处的位置,初始时其“y”值为该节点在兄弟节点中所对应的序数。对于根节点,因为其没有父节点以及兄弟节点,所以需要进行特殊处理,将其初始化为向量“(1,1)”。

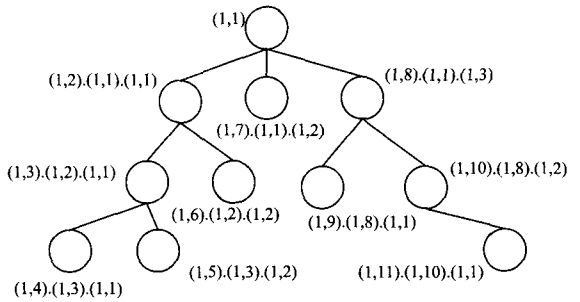


图 4　DVLS 编码

DVLS 编码利用向量序支持节点间关系的判断,规则如下:

• 位置关系(Document Order)的判断

节点 u 位于节点 v 之前$\Leftrightarrow L(u).Vlabeling \prec L(v).Vlabeling$;

• 父子关系(PC)的判断

节点 u 是节点 v 的父节点$\Leftrightarrow L(u).Vlabeling \equiv L(v).Vparentlabeling$;

• 兄弟关系(Sibling)的判断

节点 u 和节点 v 是兄弟节点$\Leftrightarrow L(u).Vparentlabeling \equiv L(v).Vparentlabeling$;

• 祖先后代(AD)关系的判断

节点 u 为节点 v 的祖先节点,当且仅当满足下列 3 个条件之一:

(1)若节点 u 有右兄弟节点:

假设与 u 相邻的右兄弟节点为 m,如果 $L(u).Vlabeling \prec L(v).Vlabeling$ 并且 $L(v).Vlabeling \prec L(m).Vlabeling$,则节点 u 为节点 v 的祖先节点;

(2)若节点 u 无右兄弟节点,但是 u 的祖先节点中至少有一个具有右兄弟节点:

假设在 u 的所有具有右兄弟节点的祖先节点中,距离 u 的路径最短的祖先节点的相邻右兄弟节点为 n,如果 $L(u).Vlabeling \prec L(v).Vlabeling$ 并且 $L(v).Vlabeling \prec L(n).Vlabeling$,则节点 u 为节点 v 的祖先节点;

(3)若 u 无右兄弟节点,并且 u 的所有祖先节点均无右兄弟节点:

如果 $L(u).Vlabeling \prec L(v).Vlabeling$,则节点 u 为节点 v 的祖先节点。

3.2　对静态 XML 文档的支持

因为初始时,DVLS 编码的所有向量的“x”值均为“1”,所以此时向量的“tan”值与该向量的“y”值是相等的,因此静态时可以将 DVLS 编码进行简化,即可以将所有向量的“x”省略,编码可简化为“*label*. *parentlabel*. *order*”,将利用向量序“$\prec$”和“$\equiv$”进行的向量大小比较转化为利用自然序“$<$”和“$=$”的整数比较,从而提高了节点间各种关系查询的效率,如图 5 所示。其中“*label*”、“*parentlabel*”、“*order*”分别为 DVLS 编码中向量“*Vlabeling*”、“*Vparentlabeling*”、“*Vlocalorder*”的“y”值。

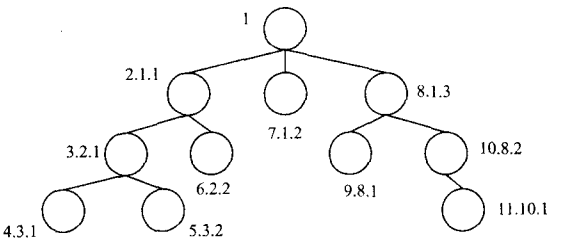


图 5　对静态 XML 文档的支持

3.3　对 XML 文档动态更新的支持

DVLS 利用向量的加法支持 XML 文档的插入操作,新插入节点的 DVLS 编码可以简化为“*Vlabeling*. x_{po}. *parentlabel*. *order*”,其中“*Vlabeling*”部分为向量(x,y),“x_{po}”为向量“*Vparentlabeling*”和“*Vlocalorder*”的“x”部分,“*parentlabel*”和“*order*”分别为向量“*Vparentlabeling*”和“*Vlocalorder*”的“y”部分,如图 6 中节点 A。下面将介绍各插入节点的计算方法。

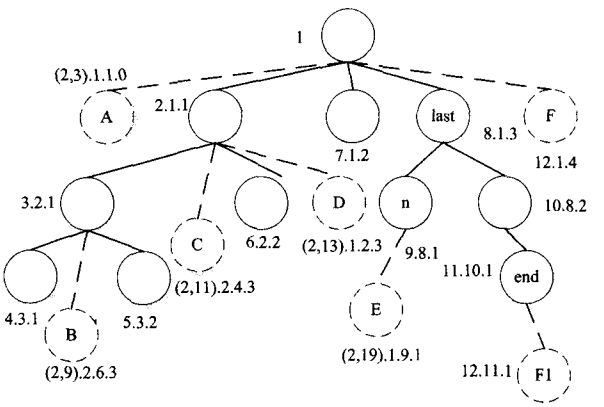


图 6　对 XML 文档动态更新的支持

在讨论节点插入之前,先给出两个函数的说明,对于节点 m:

Pre (m)返回值为 XML 文档序中与节点 m 相邻的前一节点;

Next(m)返回值为 XML 文档序中与节点 m 相邻的后一节点。

当插入节点 u 时,假设 u 的左兄弟节点为 ls,右兄弟节点为 rs:

(1)如果 u 的左兄弟节点不存在,如图 6 中节点 A,则

$L(u).Vlabeling \equiv L(rs).Vlabeling + L(rs).Vparentlabeling$;

$L(u).Vparentlabeling \equiv L(rs).Vparenglabeling$;

$L(u).Vlocalorder \equiv L(rs).Vlocalorder + (0,-1)$。

(2)如果 u 的左右兄弟节点均存在,并且 $Pre(rs)$为节点 ls,如图 6 中节点 B,则

$L(u).Vlabeling \equiv L(ls).Vlabeling + L(rs).Vlabeling$;

$L(u).Vparentlabeling \equiv L(ls).Vparenglabeling +$

$L(rs).Vparentlabeling$；

$L(u).Vlocalorder \equiv L(ls).Vlocalorder + L(rs).Vlocalorder$。

(3)如果 u 的左右兄弟节点均存在，但是 $Pre(rs)$ 不为节点 ls，如图 6 中节点 C，则

$L(u).Vlabeling \equiv L[Pre(rs)].Vlabeling + L(rs).Vlabeling$；

$L(u).Vparentlabeling \equiv L(ls).Vparenglabeling + L(rs).Vparentlabeling$；

$L(u).Vlocalorder \equiv L(ls).Vlocalorder + L(rs).Vlocalorder$。

(4)如果 u 的右兄弟节点不存在，并且 u 的左兄弟节点不是原 XML 文档中的最后一个节点，如图 6 中节点 D，则

$L(u).Vlabeling \equiv L(ls).Vlabeling + L[Next(ls)].Vlabeling$；

$L(u).Vparentlabeling \equiv L(ls).Vparentlabeling$；

$L(u).Vlocalorder \equiv L(ls).Vlocalorder + (0,1)$。

(5)如果 u 为原 XML 文档的叶子节点 n 的子节点，并且节点 n 不是原 XML 文档的最后一个节点，如图 6 中节点 E，则

$L(u).Vlabeling \equiv L(n).Vlabeling + L[Next(n)].Vlabeling$；

$L(u).Vparentlabeling \equiv L(n).Vlabeling$；

$L(u).Vlocalorder \equiv (L(n).Vlabeling.x,1)$。

(6)如果在原 XML 文档的最后一个节点 end 之后插入节点 u，则

$L(u).Vlabeling \equiv L(end).Vlabeling + (0,1)$；

a)若在根节点的最后一个子节点 last 之后插入节点 u，如图 6 中节点 F，则

$L(u).Vparentlabeling \equiv L(last).Vparentlabeling$；

$L(u).Vlocalorder \equiv L(last).Vlocalorder + (0,1)$。

b)若 u 为 end 的子节点，如图 6 中节点 $F1$，则

$L(u).Vparentlabeling \equiv L(end).Vlabeling$；

$L(u).Vlocalorder \equiv (L(end).Vparentlabeling.x,1)$。

4 实验及结果分析

因为在利用向量和向量序后，DDE 编码是各种编码方式中综合性能最好的编码，所以实验选用 DDE 编码与本文提出的 DVLS 编码进行比较，各种编码用 C# 语言在 VS2008 中实现，实验平台为 2.5GHz Intel 酷睿双核处理器，2G 内存，Windows7 操作系统，选用的数据集见表 1，其中 D1－D4 源自文献[11]，D5 源自文献[12]。

表 1 测试数据集

数据集	文档名称	文档大小	节点数	最大深度	平均深度
D1	Customer	503kB	13502	3	2.9
D2	UWM	2MB	66735	5	3.9
D3	Nasa	23.8MB	476646	8	5.6
D4	Treebank_e	82MB	2437666	36	7.9
D5	Xmark	113MB	1666315	12	6

4.1 静态性能分析

4.1.1 静态编码时间

图 7 是两者编码时间的对比，可以看出，DVLS 编码要稍优于 DDE 编码。因为 DDE 在静态编码时随着深度的增加，其编码长度也增长，而 DVLS 在静态编码时每个节点的编码固定为 3 个整数，可以直接利用自然数进行编码。

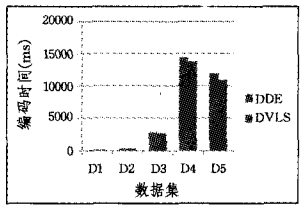


图 7 静态编码时间

4.1.2 查询性能对比

在 XML 查询中，位置关系(Docement order)、父子关系(PC)、兄弟关系(Sibling)、祖先后代关系(AD)是 4 种常见的关系计算。实验选用文档 Treebank_e 的前 10000 个节点并计算每两个节点间是否具有上述 4 种关系。图 8 显示了两种编码在计算 4 种关系时所需要的时间。因为 DDE 编码在进行各种关系的比较时需要利用字典序，编码的长度对其影响很大，尤其在进行位置关系和祖先后代关系判断时，需要将编码长度较小的节点编码逐位与另一编码进行比较，所以这两种关系计算相比父子关系和兄弟关系效率较差；而 DVLS 编码的编码长度固定，并且进行各种关系的计算时可以直接利用整数以及自然序进行比较，因此具有较高的查询效率。

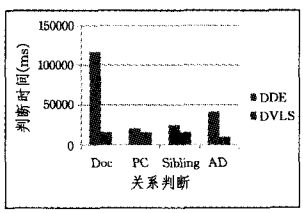


图 8 静态关系判断

4.2 动态性能分析

在 Treebank_e 文档的第 1、5、10、14 层各随机选取两个相邻节点，分别进行均匀插入和倾斜插入，测试动态性能。

首先对均匀插入和倾斜插入进行以下说明：

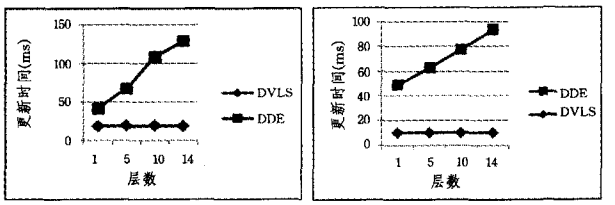
均匀插入：在两个相邻节点 ls 和 rs 间进行 n 次均匀插入操作，插入节点数为 2^n-1。即当进行第一次均匀插入时，在 ls 和 rs 之间插入 1 个节点；进行第 n 次均匀插入操作时，是在第 $n-1$ 次均匀插入基础上，在 ls 和 rs 之间每两个节点中间再插入一个新节点。实验中 $n=12$，即在各层分别进行 12 次均匀插入操作，每层插入 4095 个节点。

倾斜插入：在同一节点之前或之后连续插入 n 个节点。实验中，在各层分别倾斜插入 2000 个节点。

4.2.1 更新时间

图 9(a)、(b)分别为两种插入操作下 DDE 编码与 DVLS 编码更新时间的对比，可以看出随着文档深度的增加，DDE 编码的更新时间增长，而 DVLS 编码的更新时间基本保持不变。这是因为随着 XML 文档深度的增加，节点的 DDE 编码长度也会增长，更新过程中需要逐位进行相加操作，所以更新时间也受到影响；而 DVLS 编码在每层节点上的编码均固定为 3 部分，更新时最多只需要进行 6 次相加，并且均可利用自然数直接进行操作，因此更新时间几乎不受 XML 文档深度

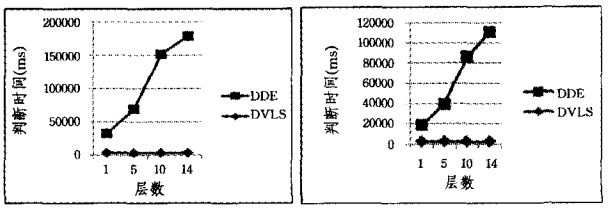
的影响，效率较高。



(a) 均匀插入更新时间　　(b) 倾斜插入更新时间

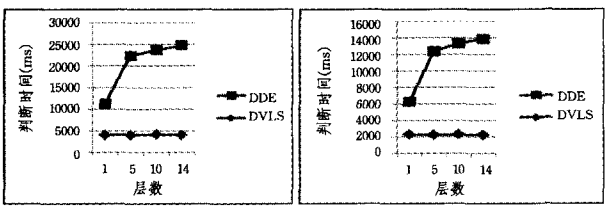
图 9　编码更新时间

4.2.2　查询性能对比



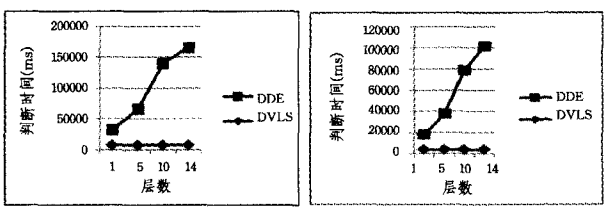
(a) 均匀插入位置关系判断　　(b) 倾斜插入位置关系判断

图 10　位置关系查询



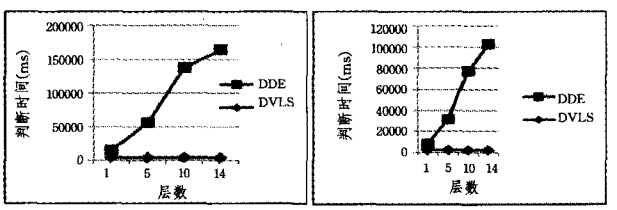
(a) 均匀插入父子关系判断　　(b) 倾斜插入父子关系判断

图 11　父子关系判断



(a) 均匀插入祖先后代关系判断　　(b) 倾斜插入祖先后代关系判断

图 12　祖先后代关系判断



(a) 均匀插入兄弟关系判断　　(b) 倾斜插入兄弟关系判断

图 13　兄弟关系判断

图 10—图 13 分别为在两种情况下，DDE 编码和 DVLS 编码对各层所有插入节点两两之间计算前述 4 种关系所需要的时间对比。可以看出，DDE 编码查询效率受 XML 文档深度的影响仍然较大，而 DVLS 编码几乎不受影响。因为在进行各种关系的计算时，DDE 编码需要利用字典序进行比较，受编码长度影响较大；而 DVLS 需要利用向量序，虽然也受编码长度影响，但是因为其编码长度不受 XML 文档深度的影响，各层的查询时间变化不大，并且相对 DDE 编码具有较高的查询效率。

结束语　本文提出了一种新的 XML 文档编码方法——DVLS 编码，并针对静态以及动态 XML 文档分别提出了优化方案。该编码克服了传统前缀编码随着 XML 文档树深度的增加长度也增长的缺陷，并且利用向量、向量序以及向量加法支持 XML 文档的动态更新。实验结果验证了编码的有效性和可行性。下一步将针对编码的存储格式展开研究。

参 考 文 献

[1] 孟小峰. XML 数据管理：概念与技术[M]. 北京：清华大学出版社，2009：58-75

[2] Tatarinov I, Viglas S D, Beyer K, et al. Storing and Querying Ordered XML using a Relational Database System[C]// 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 Trees[C]// Proceedings of 21th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (SPDS). 2002

[4] Abiteboul S, Alstrup S, Kaplan H, et al. Compact Labeling Scheme for Ancestor Queries[J]. SIAM J. Computing Systems, 2006(40): 55-99

[5] Duong M, Zhang Y. LSDX: A New Labelling Scheme for Dynamically Updating XML Data[C]// Proceedings of the 16th Australasian Database Conference. Australia, 2005: 185-193

[6] O'Neil P, O'Neil E, Pal S. ORDPATHS: Insert-Friendly XML Node Label[C]// Proceedings of the 2004 ACM SIGMOD international conference on Management of data. Paris, France, 2004: 903-908

[7] 刘先锋，周舟，刘萍，等. 一种分数前缀 XML 编码方案[J]. 计算机工程，2012，38(12)：29-31

[8] Xu Liang, Ling T W, Wu Huayu, et al. DDE: From Dewey to a Fully Dynamic XML Labeling Scheme [C]// Proceedings of SIGMOD'09. New York, USA: ACM Press, 2009: 719-730

[9] 庄灿伟，冯少荣，林子雨，等. IDD：DDE 编码改进方法[J]. 计算机研究与发展，2010，47(增刊)：119-126

[10] Xu Liang, Tok Wang Ling, Wu Hua-yu. Labeling Dynamic XML Documents: An Order-Centric Approach[J]. IEEE transactions on knowledge and data engineering, 2012(24): 100-113

[11] Univ. of Washington XML Repository, http://www.cswashington.edu/research/xmldatasets/, 2010

[12] XMark. An XML Benchmark Project[OL]. http://monetdbcwi.nl/xml/downloads.html, 2011