第 34 卷 第 23 期 计 算 机 工 程 2008 年 12 月
Vol.34 No.23 Computer Engineering December 2008
·软件技术与数据库· 文章编号：1000—3428(2008)23—0074—02 文献标识码：A 中图分类号：TP311.13

基于 P2P 端点目录的索引方法 APNI

程崇燕[1]，张 晶[2]
(1. 天津师范大学计算机与信息工程学院，天津 300387；2. 中国民航大学计算机学院，天津 300300)

摘 要：针对在 P2P 环境中可扩展标记语言的端点目录管理的要求，提出一种 APNI 索引方法，该索引方法利用素数索引方法(Prime Number Index)并加以改进，借以素数索引表和层次索引表辅助。实验结果表明，APNI 查询结果比 Prime Number Index 更快。
关键词：可扩展标记语言；P2P 网络；APNI 索引方法；素数索引方法

APNI——Index Method of Catalog in P2P

CHENG Chong-yan[1], ZHANG Jing[2]
(1. Computer and Information Engineering College, Tianjin Normal University, Tianjin 300387;
2. College of Computer Science & Technology, Civil Aviation University of China, Tianjin 300300)

【Abstract】 In order to meet catalog administration of Extensible Markup Language(XML) in P2P, this paper proposes an index method Advanced Prime Number Index(APNI), which uses the prime number index method and improves it with accessorial structures of Prime Number Index table and hierarchy index table. The purpose is that it queries XML more flexibly. Results of the experience show that it queries XML using APNI more quickly than using Prime Number Index.
【Key words】 eXtensible Markup Language(XML); P2P; Advanced Prime Number Index(APNI); Prime Number Index

1 概述

随着互联网络的蓬勃发展，eXtensible Markup Language(XML)[1]作为一种数据表现和交换的格式在WWW上得到广泛的应用。在P2P[2]网络中利用XML在本地缓存应用数据，每一个Peer端点都会不断地处理XML文档，因而需要一个目录来管理这些XML文档，基于此需要一种基于XML目录管理的索引方法便于快速地定位和查询，这样的目录索引要满足以下条件：

(1)便于更新：在 P2P 端点的 XML 文档会不断地增加或者删除，因而 XML 目录索引要便于更新。

(2)便于查询和定位：XML 目录索引要能够快速定位到所要查找到的 XML 文档，查询的方法要灵活多样。

Advanced Prime Number Index(APNI)能满足P2P每一个peer目录索引的要求。APNI索引是在Prime Number Index[3]索引基础上的改进，继承了Prime Number index特点，使用素数编码每一个XML节点，另外又使用了其他结构辅助查询，以便能很快地定位查询的节点，并适应不同的查询要求。

2 Prime Number Index 的问题及改进的 APNI 方法

2.1 Prime Number Index

定义 1 每一个节点用唯一的 label 来标记，label 的值是 parent label 和 self label 的乘积所得，如图 1 所示。

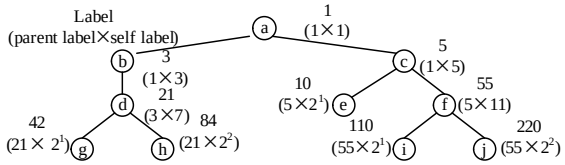

图 1 素数编码策略

在图 1 中，parent label 是父节点的 label 值。self label 是每一个非叶节点唯一的素数编码值，叶节点的 self label 用 2^n 来表示。

2.2 素数编码策略的问题

素数编码适用于XML文档的更新，特别是插入不要求次序的情况，可直接插入，无需对整个文档重新编码。素数编码可判断 2 个节点是否具有祖先后代关系，而不能直接找到所要查找的节点，因为每个节点的素数编码与整体的XML文档的编码没有关系，只与它的父节点有关。这也使得它不能像其他的数据编码一样例如区域编码、先序编码等从编码上就能直接判断出节点之间的先后关系，因而对于XPath[4]轴的查询就较为困难。基于此要辅助其他的结构使得素数编码更好地判断出节点之间的关系。

2.3 APNI 索引结构

定义 2 APNI 索引方法是在原先的素数编码策略的基础上在每一层添加层数的记录，利用 APNI 索引结构是便于配合素数索引表和层次索引表记录节点的信息和节点之间的关系，最终方便查询的目的。其结构如图 2 所示。

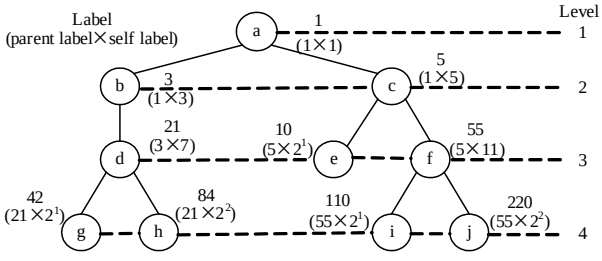

图 2 素数编码的改进结构

素数索引编码表如表 1 所示，包括素数编码列(Prime

作者简介：程崇燕(1982－)，女，硕士研究生，主研方向：XML 数据库应用；张 晶，硕士研究生
收稿日期：2008-03-08 **E-mail：**chengcyan@126.com

Number)、素数编码对应的节点在 XML 文档中的第几层(Level)、节点在 XML 文档对应层次的位置(Pos)、节点父节点的素数编码(Parent)、节点的类型(Kind)及其节点的信息(Name)等。在素数索引表中，以素数的编码从小到大来排序。

表 1　素数索引表

Prime Number	Level	Pos	Parent	Kind	Name	…
1	1	1		Elem	a	
3	2	1	1	Elem	b	
5	2	2	1	Elem	c	
10	3	2	5	Elem	e	
21	3	1	3	Elem	d	
42	4	1	21	Elem	g	
55	3	3	5	Elem	f	
84	4	2	21	Elem	h	
110	4	3	55	Elem	i	
220	4	4	55	Elem	j	

层次索引表如表 2 所示，记录的是每层对应的素数节点编号(label)。

表 2　层次索引表

Level	Prime Number
1	1
2	3,5
3	10,21,55
4	42,84,110,220

由于 APNI 索引结构是在 Prime Number Index 的基础上改进，因此它继承了 Prime Number 素数编码索引更新节点的优势。APNI 在更新 XML 文档节点的时候，要对素数索引表和层次索引表进行局部微调。当在 XML 文档中插入节点时，APNI 只需比 Prime Number 素数编码索引在素数索引表中多添加一行信息，并把节点对应的素数编码添加到层次索引表当中去。如果同层节点的顺序有所变动，则修改素数编码表中的对应信息。

层次索引表包括 XML 层次和层次对应的节点素数编号。通过 2 个表的联合使用进行 XPath 轴的查询。

2.4　APNI 查询方法

2.4.1　子孙节点的查找

如果节点 v 是 u 的子孙，则 $label(u)$是奇数，$label(v)$ mod $label(u)=0$。

定理 1 如果要查找 u 全部的子孙，首先通过素数索引表找到 u 对应的 level 值 n，再在层次索引表中从 n+1 的 level 找起一直到第 m 层有 u 的子孙而第 m+1 层没有为止，如果 v 是 u 的子孙，则 $label(v)$ mod $label(u)=0$。

u 的子孙一定是在 u 的下一层开始出现，如果第 m 层有 u 的子孙而第 m+1 层没有 u 的子孙了，说明已经查找完毕。

算法 1 查找节点 u 全部子孙的算法

输入：XML 文档和 XPath 表达式//u/*

输出：输出 u 节点和其子孙的 XML 文档

Begin

(1)把 XML 文档进行分解，为每个节点添加属性记录节点唯一的 label

(2)XML 文档分解的同时,把节点按照 26 个字母(不区分大小写)进行分组，并同时生成素数索引表和层次索引表

(3)对于 XPath 表达式进行分解，在分组中找到所要节点 u 的 label，判断 label(u)是否是奇数，如果是则找 u 的子孙，如果不是则输出 u 为文本节点

(4)初始化 i 为 u 的下一层,初始化 t 为 true，表示这一层有 u 的子孙，t 为 false 表示这一层没有 u 的子孙

(5)在层次表中查找 u 的子孙

```
while i 不是最后一层 and t==true do
        t=false
        for each level[i][j] then
            if level[i][j]mod Label(u)==0 then
              t=true
            end if
        end for
    利用二分查找在素数索引表中找 u 的子孙具体信息
        BinarySearch(handler.node_p, handler.level_array[i][j])
i++
    end while
```

(6)以 XML 格式输出 u 的子节点

2.4.2　祖先节点的查找

在素数编码中，如果 u 是 v 的祖先当且仅当 $label(u)$为奇数，$label(v)$ mod $label(u)=0$。如果要查找 v 的全部祖先，有 2 种查找的办法：

(1)在素数索引编码表中首先找到 v 对应的 parent 的素数编码，并记录下来，再通过这个 parent 编码找他的 parent 编码并记录，以此类推，最后找到根。这些都将是 v 的祖先节点。

(2)在素数索引编码表中首先找到 v 对应的 level 值 n，再从层次索引表当中找到 level 是 n-1 的值，比较 n-1 的 level 对应的素数值 u 是否为奇数，$label(v)$ mod $label(u)=0$。如果满足则 u 是 v 的祖先，使用这种方法查找 n-2 的 level 对应的素数值，一直查找到第 2 层，因为根肯定是 v 的祖先。

2.4.3　前序节点的查找

u, v 有 3 种节点关系：u, v 同时是叶节点，有一个是叶节点，都不是叶节点。判断 u 是否是 v 的前序节点，后 2 种节点关系判断方法如下：

首先判断 $label(v)$ mod $label(u)\neq 0$，且 $label(u)$ mod $label(v)\neq 0$ 都不等于 0，则可判定 u, v 无父子关系。再在素数索引表当中找到 u 的 level 值 $level_u$ 和 v 的 level 值 $level_v$，比较 $level_u$ 和 $level_v$ 的大小，如果 $level_u > level_v$，则查找 u 的祖先，且 u 的祖先 p 要与 v 有相同的 level 值，如果 u 的祖先 p 与同层的 v 相比，位置在同层更靠前 $pos_p < pos_v$，那么 u 是 v 的前序节点。如果 $level_u = level_v$ 且 $pos_u < pos_v$，那么也可以判定 u 是 v 的前序节点。

2.4.4　后序节点的查找

查找后序节点的操作与查找前序节点的方法相类似，不再赘述。

2.5　层数的估计

实际上XML文档树的层数将会影响APNI索引结构的查询速度，层数越多，查询越复杂。但是通过对大量XML文档结构的分析[5]，在超过 190 000 棵XML文档树中，有 99%的XML文档树的层数小于 8，在剩余的 1%当中大部分XML文档树的层数少于 30 层。因此，不必担心XML的层数影响查询速度。

3　实验结果分析

为了评价算法的性能，本文做了大量的实验，并对其进行分析。

3.1　实验设置

所有的算法都用在 Java 实现，所有实验都在 Pentuim IV2.8 GHz, 1 GB RAM, 80 GB 硬盘的台式计算机上运行，底层操作系统是 Windows XP，实验文档使用 XMark 生成。

3.2　实验结果比较分析

图 3 是 APNI 和 Prime Number Index 查询比较，查询的

（下转第 88 页）