文章编号:1001 - 9081(2009)10 - 2778 - 03

一种基于有序对的含父子边的小枝模式匹配算法

王 瑞,陶世群
(山西大学 计算机与信息技术学院,山西 太原 030006)
(rita0627@yahoo.cn)

摘 要:随着 Internet 的发展和网上 XML 数据规模的与日剧增,如何准确、高效地查询 XML 数据已经成为研究的热点问题。目前,已经提出了很多小枝模式匹配算法,但没有解决含有父子边的小枝模式查询。针对该问题,提出了一种基于有序对的新算法 PCTwig,通过在查询树和文档树上分别建立父子关系的有序对来进行查询。查询过程中避免了产生中间结果,也不需要进行归并操作,实验证明该算法是有效的。

关键词:XML 文档;XPath;小枝模式匹配;有序对;父子关系
中图分类号: TP311 **文献标志码**:A

Matching algorithm for twig patterns with parent-child edges based on ordered pair

WANG Rui, TAO Shi-qun
(*School of Computer and Information Technology, Shanxi University, Taiyuan Shanxi* 030006, *China*)

Abstract: With the development of Internet and the constantly increasing scale of XML data, how to query the XML data exactly and efficiently becomes a hot issue. At present, there are many algorithms for twig pattern matching, but they don't have good method to solve the twigs which have parent-child edges. The new algorithm called PCTwig was proposed for this problem, which was based on the ordered pair. The twigs were queried through setting the ordered pair of parent-child relationship on query tree and document tree. In query process, it can avoid useless intermediate result and merge operation. The experiment shows the effectiveness of the approach.

Key words: XML document; XPath; twig pattern matching; ordered pair; parent-child relationship

0 引言

随着互联网的发展,XML 已经成为一种网上通用的数据表示与交换的标准。如何有效地查询网上与日俱增的海量的 XML 数据也已成为很有研究价值的重要课题。利用路径表达式来导航 XML 文档的查询是许多 XML 查询语言的共同特点。使用路径表达式查询的方法有两种:一种是结构连接,另一种是小枝模式匹配。

近年来,研究工作者提出了很多查询算法,如直接连接算法 PMGJN[1],EE-Jion/EA-Jion[2]等,它们要重复扫描列表,并会产生大量的中间结果。文献[3]作者提出了一种基于栈的算法 Stack-Tree,[4]提出了一种基于区域划分的结构连接算法 Range Partitioning Jion 等,这些算法都是将查询树分解成一系列二元结构求解,然后再归并,这在一定程度上避免了重复扫描列表,但仍会产生大量的中间结果。为避免产生大量的中间结果,文献[5]作者提出了一种整体的小枝模式匹配算法 TwigStack,它处理仅含祖先后裔(A-D)关系时不会有中间结果产生。文献[6 - 7]作者对 TwigStack 算法进行了改进,TSGeneric + 算法利用索引能够高效地跳过那些不参加连接的节点。文献[8]作者提出了"流"的思想,利用"流"来加速小枝模式处理过程。文献[9]作者提出了 TJFast 算法,它只需访问叶子节点。以上这些算法都是基于归并的,因此所需代价很高。Twig2Stack 算法[10]和 TwigList 算法[11]分别使用了层次栈和队列而不需要归并,因此优于 TwigStack 算法和 TJFast 算法等,但却会产生大量的随机访问,并且算法复杂。文献[12]作者针对查询树的具体特点提出了算法 TwigNM,这些算法都是基于非归并的,效率明显高于基于归并的算法。

以上算法都没有很好地处理父子(P-C)关系,文献[13]作者提出的 TwigStackList 算法能较好地处理小枝模式中的 P-C 关系,但是当分支节点含 P-C 边时,仍有中间结果产生。针对上述算法的不足,提出了一个更好地处理 P-C 关系的查询算法,主要工作如下:

1)将节点存储为有序对形式,不需对节点进行区间编码;

2)根据存储特点,提出一种新的小枝模式匹配算法 PCTwig,能高效处理含 P-C 边的小枝模式查询。

1 相关概念

XML 文档的嵌套结构可以用树形结构的数据模型表示,称为 XML 文档树。查询路径 Xpath 也可表示成树形结构来与它进行模式匹配,给出文档树 D 和查询树 Q 如图 1 所示(简称为 D 和 Q)。

1)小枝模式:将 XPath 查询语句用树的形式来表达,我们称这棵树为小枝模式。例如 XPath = //a/c[d]/e/b 对应的小枝模式为 Q。

节点有序对:某个节点的节点有序对是由该节点和它的父节点组成,没有父节点的设置父节点为 null。例如在 Q 中 a 节点的节点有序对是(a,null),在 D 中 a_1 节点的节点有序对是(a_1,r_1)。

2)节点标签流:在查询树中,一个节点的标签流是按先序遍历文档树中该类型节点的顺序排列,由所有该类型节点

收稿日期:2009 - 04 - 07;**修回日期**:2009 - 05 - 25。

作者简介:王瑞(1985 -),女,山西临汾人,硕士研究生,主要研究方向:XML 数据库技术; 陶世群(1946 -),男,安徽寿县人,教授,博士生导师,主要研究方向:数据库理论与技术、XML 数据管理。

的节点有序对组成。例如 Q 中 a 节点的标签流对应 D 中的 $(a_1,r_1),(a_2,a_1),(a_3,b_2)$。

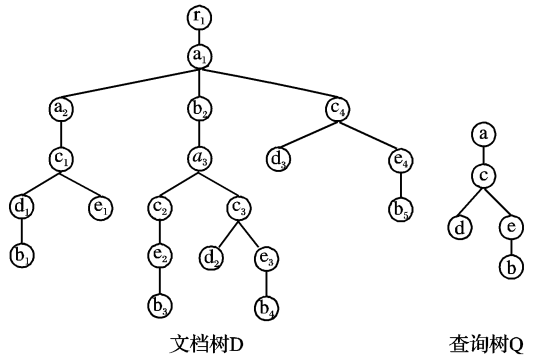

图 1　文档树 D 和查询树 Q

3)查询树标签流:查询树的标签流,是按后序遍历查询树中节点的顺序排列,由树中所有节点的节点有序对组成。例如 Q 的标签流是(d,c),(b,e),(e,c),(c,a),(a,null)。

2　表的存储结构

为查询树建立一个表存放所有查询节点的信息,其中 id 是节点后序遍历进表的顺序,name 和 pname 分别是当前节点名和其父节点名,pnum 是父节点的分支数,当表中父节点分支数不为 1 并且该父节点在表中出现次数等于分支数时,flag 值为当前父节点名,否则为 null。例如 Q 对应的表如表 1 所示,其中查询节点 d 和 e 的父节点都为 c, c 的分支数不为 1,但是由于 d 所对应的 c 在表中第二次出现等于 c 的分支数,所以 flag 值为 c,而 e 所对应的 c 在表中是第一次出现,所以 flag 值为 null。

表 1　Q 表

id	name	pname	pnum	flag
1	d	c	2	null
2	b	e	1	null
3	e	c	2	c
4	c	a	1	null
5	a	null	0	null

对文档树中不同类型的节点分别建立对应的表,例如为 D 中 r、a、b、c、d 和 e,分别建立对应的 r 表、a 表、b 表、c 表、d 表和 e 表。其中 id 是同类型节点先序遍历进表的顺序,name 和 pname 分别是当前节点名和其父节点名,pid 是父节点对应表的进表顺序。D 中节点类型为 a 和 b 的节点对应的表如表 2、3 所示。

表 2　a 表

id	name	pname	pid
1	a	r	1
2	a	a	1
3	a	b	2

表 3　b 表

id	name	pname	pid
1	b	d	1
2	b	a	1
3	b	e	2
4	b	e	3
5	b	e	4

3　PCTwig 匹配算法

该算法是在以上表结构下进行查询的,会得到五个动态链表 L_a、L_b、L_d、L_c 和 L_e 来存储与 Q 标签流匹配的结果。Q 标签流是按后序遍历 Q 的顺序,这样可以确保在查到某节点时,在它之前的节点是匹配的,从而避免无用节点的产生。

通过 Q 标签流所指节点有序对和 D 中对应节点标签流中的节点有序对的匹配来进行查询。例如当 Q 标签流所指节点有序对为(d,c), D 中 d 节点标签流中的节点有序对为$(d_1,c_1),(d_2,c_3)$和(d_3,c_4),通过不同的匹配方法来进行查询。有序对的形式可以很好的将节点与节点连接起来,这样就避免了连接操作。

在查询树中,节点有三种:叶子节点、根节点和中间节点。提出三种 Match 方法对这三种情况的节点进行查询。当是叶子节点时,用 Match1;当是中间节点时,用 Match2;当是根节点时,用 Match3。

输出结果时,按先序遍历 Q 的顺序访问查询节点对应的链表,判断前一链表节点有序对中 name 和 id 是否与当前链表节点有序对中 pname 和 pid 相等来输出结果,碰到当前链表节点有序对中 pname 是分支节点时,找到 pname 对应链表中的节点有序对进行判断。

3.1　PCTwig 算法描述

算法 1　PCTwig(Q,D)

```
while(not eof( T_Q )) do          // T_Q 流的指针没有指向结尾时
  getNext( q ) find T_q
  //获得查询节点 q,并且找到与 q 对应的 T_q 流
  if ( q.name = =last( q ).pname) then
    //查询节点名和上个查询节点的父节点名相同时
    if ( q.pname = =null) then Match3( T_q,q )
    //查询节点的父节点名为 null 时
    else Match2( T_q,q )
    end if
    else Match1( T_q,q )
    end if
  if ( q. flag! =null) then Compare(branch)
  //查询节点的标记不为 null,比较分支
  end if
  advance( T_Q )                  //将 T_Q 流指针指向下一个节点
end while
showSolutions( )
//按先序遍历查询树中节点对应链表的顺序输出
```

算法 2　Match(T_q,q)

```
//查询节点有序对和对应标签流中节点有序对匹配
Match1( T_q,q )                          //查询节点是叶子节点
while(not eof( T_q )) do
  if ( T_q. pname = = q. pname) then add( T_q,L_q )
    if ( q. pnum >1) then branch.pname = T_q. pname
      branch.pid = T_q. pid branch.name = T_q. name
    end if
  advance ( T_q )
  else advance ( T_q )
  end if
end while
Match2( T_q,q )                          //查询节点是中间节点
while(not eof( T_q )) do
  if ( T_q. pname = = q. pname&&T_q .id 等于 last( q )对应链表
    中的 pid) then add( T_q,L_q )
    if ( q. pnum >1) then branch.pname = T_q. pname
      branch.pid = T_q. pid branch.name = T_q. name
    end if
    advance ( T_q )
    else advance ( T_q )
  end if
end while
Match3( T_q,q )                          //查询节点是根节点
while(not eof( T_q )) do
  if ( T_q. id 等于 last( q )对应链表中的 pid) then add( T_q,L_q )
  advance ( T_q )
```

```
    else advance ( T_q )
    end if
end while
```

算法 3　Compare(branch)

```
find branch. pname = = q . flag
//找到父节点名为当前标记的所有分支
if ( branch. name not equal&&branch. pid not equal)  then
  //如果不同名分支没有相同 pid
  在 branch. name 对应链表中删除 pid 为 branch. pid 的节点
end if
```

算法中其他函数：

eof (T_Q)：T_Q 流的指针指向结尾时，返回 true；

getNext (q)：返回 T_Q 流中指针所指内容并赋值给 q；

last (q)：返回 q 的上一个查询节点；

add (T_q,L_q)：将当前 T_q 流指针所指节点有序对放入相应的链表中；

advance (T_q)：将 T_q 流指针指向下一个节点。

3.2　PCTwig 算法执行

用图 1 中的例子来说明该算法的工作原理，执行步骤如图 2 所示。

1)查询节点有序对为(d,c)，对应 T_d 流中节点有序对，d 为叶子节点，通过 Match1 进行匹配。

2)查询节点有序对为(b,e)，对应 T_b 流中节点有序对，b 为叶子节点，通过 Match1 进行匹配。

3)查询节点有序对为(e,c)，对应 T_e 流中节点有序对，e 为中间节点，通过 Match2 进行匹配。因为 e. flag 不为 null，对分支进行比较后删除父节点不同的分支，在 L_d 链表中删除 (d_1,c_1)，在 L_e 链表中删除(e_2,c_2)。

4)查询节点有序对为(c,a)，对应 T_c 流中节点有序对，c 为中间节点，通过 Match2 进行匹配。

5)查询节点有序对为(a,null)，对应 T_a 流中节点有序对，a 为根节点，通过 Match3 进行匹配。

按先序遍历 Q 的顺序，访问查询节点对应的链表，最终输出结果为：(a_1,r_1)、(c_4,a_1)、(d_3,c_4)、(e_4,c_4)、(b_5,e_4)和(a_3,b_2)、(c_3,a_3)、(d_2,c_3)、(e_3,c_3)、(b_4,e_3)。

3.3　PCTwig 算法分析

PCTwig 算法可以输出整个小枝模式的匹配结果。不同算法在对 XML 文档查询时，都需要将文档和查询语句解析存入表中，所以这里不考虑构造表结构的时间和空间开销，只考虑算法在数据库表中查询的时空效率。有 n 个节点的小枝模式查询，最坏的情况下，PCTwig 算法的 I/O 和 CPU 的时间复杂度与 n 个查询节点对应的 T_q 流的大小和最后匹配结果大小的总和成线性关系，即 $O(|T_1|+\cdots+|T_n|+|output|)$。算法的空间复杂度(即链表的大小)，由于采用动态链表，从而动态分配空间，和基于栈的算法的空间效率相同。最坏的情况下，是查询树标签流和它对应节点标签流匹配节点的大小。

4　实验设置和结果

在 Windows XP 操作系统下使用 VB6.0 编写的实验系统实现了 TwigStackList 和 PCTwig 算法。选择 TwigStackList 算法和本文提出的 PCTwig 算法进行对比，是因为 TwigStackList 算法是处理 P-C 关系一个比较经典的算法。实验中使用一个生成的数据集作为实验用的 XML 文档。硬件条件为：Inter Core 2 @2.0 GHz，1 GB RAM，160 GB 硬盘的 PC 机。软件环境为：Windows XP Service Pack2，编程工具为 VB6.0。实验所使用的小枝模式查询如表 4 所示。

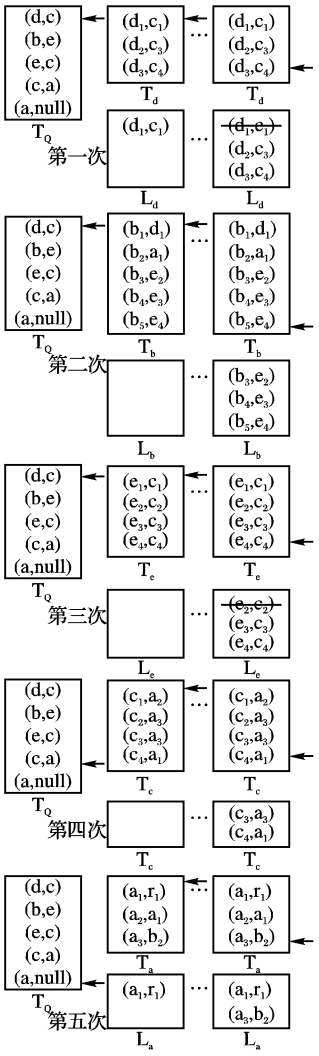

图 2　PCTwig 算法执行实例

表 4　实验中所用的小枝模式查询

Name	QueryTrees	ResultSize
TQ1	//S/V/P/N	1 213
TQ2	//S/V[/P/M]/Q	866
TQ3	//S[/W]/V[/P[/N]/M]/Q	153
TQ4	//S[/W]/V/P[/N]/M	335
TQ5	//S/V/P[/N]/M	1 156

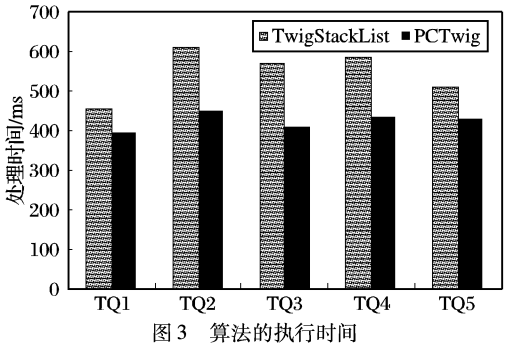

图 3　算法的执行时间

图 3 给出了 TwigStackList 和 PCTwig 算法在表 4 所示查询模式上的执行时间。由图 3 可以看到 PCTwig 比 TwigStackList 算法的执行效率高，尤其是当小枝模式的分支结构越复杂时，PCTwig 算法明显优于 TwigStackList 算法，如 TQ2，TQ3 和 TQ4 所对应的直方图所示。(下转第 2790 页)

5 结语

随着RFID等自动识别技术的推广应用,复杂事件处理机制会受到更广泛的关注和应用。本文针对RFID数据的特点以及目前提出的RFID复合事件处理方法的不足,提出了ERD事件处理方法,将事件检测和中间结果共享机制有机地统一起来。仿真实验表明,ERD在复合事件检测数量和检测效率方面有着比较好的效果。在ERD检测方法中,存储区空间的及时回收对其检测效率有着重要影响。计划今后从空间回收机制入手,进一步研究如何及时有效地减少中间结果,从而提高ERD的检测效率。

参考文献:

[1] WANG FUSHENG, LIU SHAORONG, LIU PEIYA, *et al*. Bridging physical and virtual worlds: Complex event processing for RFID data stream[C]// the 10th International Conference on EDBT' 2006, LNCS 3896. Berlin: Springer, 2006: 588 - 607.

[2] HU WENHUI, YE WEI, HUANG YU, *et al*. Complex event processing in RFID middleware: A three layer perspective[C]// Third 2008 International Conference on Convergence and Hybrid Information Technology. Washington, DC: IEEE, 2008, 1: 1121 - 1125.

[3] JIN X, LEE X, KONG N, *et al*. Efficient complex event processing over RFID data stream[C]// Seventh IEEE/ACIS International Conference on Computer and Information Science. Washington, DC: IEEE Computer Society, 2008: 75 - 81.

[4] WU E, DIAO Y, RIZVI S. High-performance complex event processing over streams[C]// Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2006: 407 - 418.

[5] ZANG CHUANZHEN, FAN YUSHUN, LIU RENJING. Architecture, implementation and application of complex event processing in enterprise information systems based on RFID[J]. Enterprise Information Systems, 2007, 1(1): 3 - 23.

[6] FORGY C. Rete: A fast algorithm for the many patterns/many objects match problem[J]. Artificial Intelligence, 1982, 19(1): 17 - 37.

[7] BERSTEL B. Extending the RETE algorithm for event management [C]// TIME 2002: Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning. Washington, DC: IEEE Computer Society, 2002: 49.

[8] DUTTA K, RAMAMRITHAM K, KARTHIK B, *et al*. Real-time event handling in an RFID middleware system[C]// Proceedings of 5th International Workshop on Databases in Networked Information Systems, LNCS 4777. Berlin: Springer, 2008: 232 - 251.

[9] EPCglobal Application Level Events[EB/OL]. [2008 - 02 - 27]. http://www.epcglobalinc.org.

[10] WALZER K, GROCH M, BREDDIN T. Time to the rescue-Supporting temporal reasoning in the Rete algorithm for complex event processing[C]// The 19th International Conference on Database and Expert Systems Applications, LNCS 5181. Berlin: Springer, 2008: 635 - 642.

[11] YONEKI E, BACON J. Unified semantics for event correlation over time and space in hybrid network environments[C]// LNCS 3760: OTM Confederated International Conference. Berlin: Springer, 2005: 366 - 384.

[12] BERNHARDT T. Esper[EB/OL]. [2008 - 02 - 21]. http://esper.codehaus.org.

(上接第2780页)

5 结语

本文提出了一种新的算法PCTwig,该算法在处理仅含P-C边的小枝模式时是非常有效的。算法的优点在于不用结构连接和归并操作,并且避免了中间结果的产生。今后的工作是怎样将基于有序对的小枝模式匹配算法应用于含有A-D边的小枝模式,而不是限于仅含P-C边的小枝模式。

参考文献:

[1] ZHANG CHUN, NAUGHTON J, De WITT 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.

[2] LI QUAN-ZHONG, MOON B. Indexing and querying XML data for regular path expressions[C]// Proceedings of the 27th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 2001: 361 - 370.

[3] AI-KHALIFA S, JAGADISH H V, KOUDAS N, *et al*. Structural joins: A primitive for efficient XML query pattern matching[C]// Proceedings of the 18th International Conference on Data Engineering. Los Alamitos: IEEE Press, 2002: 141 - 152.

[4] 王静,孟小峰,王珊.基于区域划分的XML结构连接[J].软件学报,2004,15(5):720 - 729.

[5] BRUNO N, KOUDAS N, SRIVASTAVA D. Holistic twig joins: Optimal XML pattern matching[C]// Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Madison: ACM Press, 2002: 310 - 321.

[6] JIANG HAI-FENG, LU HONG-JUN, WANG WEI, *et al*. XR-tree: Indexing XML data for efficient structural joins[C]// ICDE: Proceedings of the 19th International Conference on Data Engineering. Bangalore: IEEE Computer Society, 2003: 253 - 264.

[7] 李素清,陶世群.一种改进的基于小枝模式的XML数据库查询算法[J].计算机应用,2007,27(12):3021 - 3025.

[8] CHEN TING, LU JIA-HENG, LING T W. On boosting holism in XML twig pattern matching using structural indexing techniques[C]//Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. Baltimore: ACM Press, 2005: 455 - 466.

[9] LU JIA-HENG, LING TW, CHAN C-Y, *et al*. From region encoding to extended Dewey: On efficient processing of XML twig pattern matching[C]// VLDB: Proceedings of the 31st International Conference on Very Large Data Bases. Trondheim: ACM Press, 2005: 193 - 204.

[10] CHEN SONG-TING, LI HUA-GANG, TAEMURA J, *et al*. Twig^{2}stack: Bottom-up processing of generalized-tree-pattern queries over XML documents[C]// VLDB: Proceedings of the 32nd International Conference on Very Large Data Bases. Seoul: ACM Press, 2006: 283 - 294.

[11] LU QIN, JEFFREY X Y, DING BO-LIN. TwigList: Make twig pattern matching fast[C]// DASFAA: Proceedings of the 12th International Conference on Database Systems for Advances Applications. Bangkok: [s.n.], 2007: 850 - 862.

[12] 陶世群,富丽贞.一种高效的非归并的XML小枝模式匹配算法[J].软件学报,2009,20(4):795 - 803.

[13] LU JIA-HENG, CHEN TING, LING T W. Effcient processing of XML twig patterns with parent child edges: A look-ahead approach [C]// CIKM: Proceedings of the ACM Conference on Information and Knowledge Management. Washington, DC: ACM Press, 2004: 533 - 542.