Computer Science and Application 计算机科学与应用, 2011, 1, 149-153
http://dx.doi.org/10.12677/csa.2011.13029 Published Online December 2011 (http://www.hanspub.org/journal/csa)

XML Query Technology Research Based on Node-Set

Lianqiang Niu, Shichao Dong, Shengnan Zhang
College of Information Science and Engineering, Shenyang University of Technology, Shenyang
Email: niulq@sut.edu.cn; dianxiaoke@163.com; zsncjr@sina.com
Received: Oct. 16th, 2011; revised: Nov. 1st, 2011; accepted: Nov. 22nd, 2011.

Abstract: XML query languages take complex path expressions as their core. According to the long path expression, especially Production of a large amount of intermediate results; the larger price can be spent on joining operation of indexical nodes. This paper put forward a model which supports projection operation; choose operation and high efficient structure connection in node set. Meanwhile, this model provides path query processing with more choices.

Keywords: XML Path Query; Structural Join; XML Query Pattern

基于节点集的 XML 查询技术研究

牛连强，董世超，张胜男
沈阳工业大学信息与工程学院，沈阳
Email: niulq@sut.edu.cn; dianxiaoke@163.com; zsncjr@sina.com
收稿日期：2011 年 10 月 16 日；修回日期：2011 年 11 月 1 日；录用日期：2011 年 11 月 22 日

摘　要：复杂路径表达式查询是 XML 查询的核心研究内容。针对长路径表达式，尤其是在查询中间结果较多，节点索引的连接操作代价较大的情况下，提出了一种基于节点集的索引方式。该方式支持节点集间的投影操作、选择操作以及高效的结构连接，从而实现数据的快速查询并形成基于节点集的 XML 查询模式。同时该模式为 XML 基于路径的查询处理提供了更多的选择。

关键词：XML 路径查询；结构连接；XML 查询模式

1. 引言

随着 XML 应用的日益广泛，XML 数据管理和查询问题也引起了人们的普遍关注，并成为研究的热点。尽管 XML 有其各种不同的表示和用途，但其本质仍然是基于层次的数据结构，并可映射为相应的 XML 关系树或者 XML 关系图[1]。当前已有学者提出了多种 XML 查询语言，例如 Xpath、Xquery 等，这些查询都将路径表达式[2]作为其研究的重点，通过节点等价、路径等价等技术，得到比原始文档小得多的索引树，进而通过索引树的遍历得到查询结果。基于路径的索引技术可以很好地支持简单路径表达式的查询，但是对于正则路径表达式，效果不太理想。目前被广泛采用和接受的是分解连接查询[3]，其执行策略的基本思想是首先定位所要查询的节点集，然后在节点集之间加入结构连接从而形成新的 XML 关系树。由于在查询过程中需要进行大量节点之间结构连接的操作，因此高效的结构连接算法是实现 XML 查询的关键。本文侧重于研究如何获得合理的节点集，如何在结构连接的基础上加入一些基本的关系操作从而提高查询的效率。结合关系树模型，本文提出了以最大相关节点集为主要操作对象的查询模式，该方法充分利用集合的基本操作，增大了基本查询片段的粒度，从而减少了产生的部分中间结果。同时在提高结构连接效率基础上引入选择、投影等操作，实现相关节点集的筛选连接操作，形成查询路径获得查询结果。

2. 关系树模型

XML 是层次化数据结构，采用当前比较成熟的层次化数据模型 DOM (Document Object Model)对 XML

模式进行建模，即将 XML 层次结构转化为一棵有向的 XML 关系树。DOM 主要描述文档本身的结构和内容，同时为了增加通用性，DOM 同时支持 DTD 和 XML Schema 两种 XML 模式语言。

在 XML 关系树中，规定树中节点的方向为自顶向下，节点之间存在父子，兄弟、祖孙联系，将这种 XML 关系树称之为 SLT 树[4]。在 SLT 树中定义两种节点，一种为数据节点，它包括元素、元素的属性以及属性的取值；一种为结构节点，它主要包括元素之间先后顺序，元素之间的继承关系等。图 1 为一个 SLT 实例，树中每一个节点都有两个属性：max_occr 和 min_occr，max_occr 表示为出现的最高频率，min_occr 表示为最低频率。

DTD example：

<! ELEMENT Company(employees,wages)>

<ATTLIST Company name CDATA

#REQUIRED>

<! ELEMENT employees(employee)*>

<! ELEMENT forms (form)*>

<! ELEMENT wages (position,wage)+>

<!ELEMENT employee (work_id,position？)>

<! ELEMENT work_id (#PCDATA)>

<! ELEMENT position (#PCDATA)>

<! ELEMENT wage(#PCDATA)>

在 SLT 树中，还需要定义以下三个概念：

1) 路径：从一个节点 i 到另外一个节点 j 所经过树的分支称为 i 到 j 的路径，其中路径上两个相邻节点只能为父子或者子父关系，不能为兄弟或者祖孙关系。

2) 度数：一个节点的度数为 SLT 树中从根节点到目标节点 Max_occr 的值大于 1 的节点个数。例如在图 1 中，comment 的度数为 1，其路径为(company-employees-employee)其中对应的 Max_occr 为(1, 1, more)。

3) 相关节点集：相同度数的节点构成相关节点集，但是相同度数的节点不一定在同一个相关节点集中。图 1 中，(comments, forms)为一个相关节点集。

4) 最大相关节点集：相关节点集的最大集合称之为最大相关节点集。图 1 中，(wenjuan, name, comments, forms)则为一个最大相关节点集。

3. XML 查询处理

3.1. XML 查询树构造

XML 查询树的构造采用如下步骤：

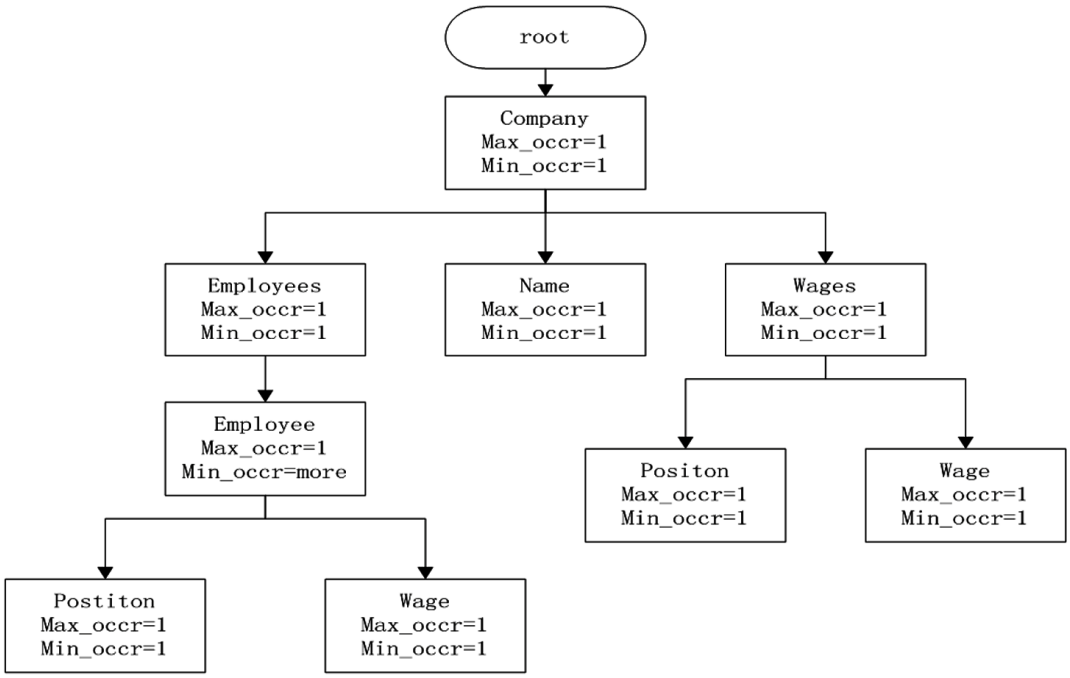

Figure 1. An example of SLT
图 1. SLT 实例

1) 对 SLT 树进行手工节点划分；

2) 将 SLT 划分成若干个最大相关节点集；

3) 由最大相关节点集构成一棵新的树，称之为 Node 树。

例如对图 2 中的 SLT 树进行划分，获得 A、B、C，3 个最大相关节点集：

A = (company, name, employees, wages)

B = (employee, position, wage)

C = (wages, wage, position)

3.2. 结构连接

XML 路径查询是利用其 XML 生成树各枝节之间的关系进行查询，在查询过程中，可能产生多余的枝节，这样将导致查询效率的下降，消耗大量的时间和空间。以目标节点为导向的查询主要是通过利用查询树中的目标节点来避免中间枝节的传递[5]，而基于最大相关节点集的查询则不同，它是通过节点之间的联系，即通过查找相关节点集来代替路径查询中的路径遍历，以查询尽量少的节点来获得查询结果。在查询过程中，由于 XML 已经转化为 Node 树，则查询过程即是利用最大相关节点集构造一棵新的 Node 树的过程，查询结果则为构造后的 Node 树的叶子节点。

当前存在的结构连接算法主要有：归并连接算法[6]和非归并连接算法[7]，其中归并连接算法又包括直接归并连接算法和基于缓存的归并连接算法[8]，归并连接算法的基本思想是：设参加连接的两个关系表为 Alist 和 Dlist，首先对 Alist 中的第一个节点 a1，在 Dlist 中顺序搜索到可能与节点 a1 连接的第一个节点，称为扫描始点，然后在 Dlist 中从扫描始点开始顺序扫描，对满足 begin < a1.end 条件的所有元组 dj 再判断是否满足连接条件。若满足，则产生连接结果节点对(a1.dj)；继续对 Alist 中的第二个元组 a2 重复上面的步骤，直到 listA 或 listD 中元组连接完毕。直接归并连接算法缺点是可能存在重复的元组，从而导致重复扫描，最坏的情况下效率很低。

基于缓存的归并连接算法[9]是对于直接归并连接算法的一种改进，即在其对比的过程中加入队列或者栈等数据结构进行元组的比较，查询效率较高，但是要利用额外的空间。

本文利用归并连接思想，对最大相关节点集进行扩充，即在最大相关节点集中加入最大相关节点集的父节点。对于包含根节点的最大相关节点集，则再次加入根节点，以此来表明该最大相关节点集是 Node 树中的根节点；对于其他最大相关节点集则是加入其根结点。例如扩充后的最大相关节点集 B 为(employees, employee, position, wage)。其结构连接如下，对于 Node 节点 A 和 B 进行连接，首先利用 B 中的节点与 A 中的节点进行匹配，如果匹配成功，则通过匹配成功的节点进行连接，如果未能匹配成功，则说明 A、B 节点在 Node 树中非父子或者子父关系，即构造失败。

在对其进行结构连接时，结构结构连接的实现是在其各个最大相关节点集之间进行的，首先获取结构

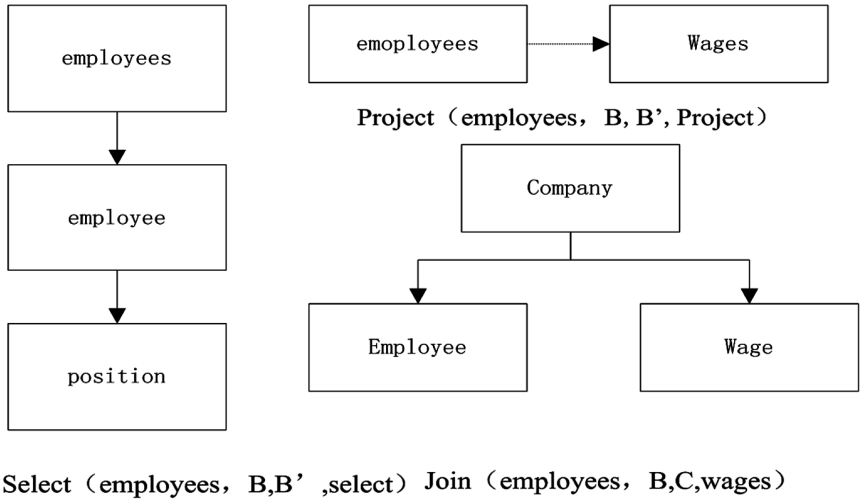

Figure 2. All kinds of node operation
图 2. 各种节点操作

连接的两个最大相关节点集 A 和 B，选取 B 中的一个节点去匹配 A 中的节点，如果匹配成功则 A、B 节点连接成功；如果匹配不成功，则选择 B 中的其他节点进行匹配。若 B 中所有节点都未能匹配成功，则连接失败。如要结构连接的是兄弟节点或者祖孙节点，则需要在结构连接过程中利用其他的节点，称之为桥梁节点。其思想为首先找到叶子节点，然后与上层节点，或者其父节点匹配；接着其父节点再进行上层匹配，重复匹配上层节点，最终得到根节点，则连接成功。

其上算法为：

```
F-SNode Struct_link 算法：
If (！Node){
For (F in Node and S in Node)
    If (F.node == S.node)
{F&S；Node.add (F, S)}
        Else{printf (“Node can not
link”)；}}
B-B’ Node or G-G’ Node 结构连接：
If (！Node){
For (B in Node and B’ in Node)
    If (C in Node)
    {C.node = B.Node;C&B;
Node bc = Node.add(B, C);
C.node=B’.Node;C&B’;
Nodebc’=Node.add(C,B’);
 Node.add(bc,bc’)
}
Else{printf(“C Node not exsit”);
        }
}
```

3.3. XML 查询

当前 XML 查询主要是通过路径的遍历来完成的，而在查询过程中可能会产生一些无效的路径查询操作[10]。而基于相关节点集的查询利用其逆向查询，获得查询路径得到查询结果，在查询的过程中减少了部分中间结果。使得查询效率得到一定的提高。

设有四元组(r, s, s’, v)，其中 s 和 s’为最大相关节点集，r 为 s 的根节点，v 为相关操作，利用其相关操作实现其目标节点的选择，然后在目标节点选择后加入投影、选择、结构连接等操作实现数据的查询。

3.3.1. 选择(select)

选择操作(select)是指在查询的过程中，利用其查询的目标节点定位到其所在的最大相关节点集，在其定位的最大相关节点集合中进行节点的操作。在四元组中选择操作的具体实现为：在 Node 树中，选择要查询的是最大相关节点集 B 中的 position 节点，利用选择操作选择该节点，则 B 中只剩余该最大相关节点集中的根节点和其 position 节点。对于剩余的节点结合进行连接，获得一个 Node 树中的单分支，进行多次选择获得多个分支，提供其查询过程中的各个路径分支。实现上述操作的四元组为：Select (employees, B, B’, select)如图 2。

3.3.2. 投影(Project)

投影操作是指在 XML 层次中对于某一些属性值，即 XML 生成的 Node 树中，投影某些 Node 节点，对于 Node 树上的节点进行多节点的操作。在 Node 树中，为了获得同一层次的信息，可以利用投影操作。例如在 Node 树中，利用投影操作，获得某层次的某些节点集合。例如在对于上边给定的 Node 树中选择其第二层次的所有节点进行投影操作，其实现如下: Project (employees, B, B’, Project)如图 2。

3.3.3. 连接(Join)

连接操作，该操作是查询的关键，主要是查询某个节点时通过利用逆向方式获得连接路径，而最后获得所要查询的节点位置。例如为了获得查询的整个路径，主要是利用连接操作通过对于各个不同相关节点集合中的连接，对于 Node 树进行裁剪获得一棵新的 Node 树而该树则为 XML 查询的路径生成树。例如在给定的 Node 树下查询 employees 中某部门和 wages 下工资相结合的内容，连接实现查询的操作为 Join (employees, B, C, wages)如图 2。

4. 实验示例

基于以上理论，在最大相关节点集的基础上，对其单个或者多个最大相关节点集进行操作。简单来说，对于单个相关节点集利用选择操作，获得某些节点而对于多个最大相关节点集的操作则是先进行选择、投影操作，利用获得的一些相关节点进行利用最大相关

Table 1. Comparison times of query node
表 1. 查询所需节点比较次数表

Layer	nodes	Xpath	X_Nodes
1	1	0	0
2	2	1	1
2	4	3	1
4	8	14	3

节集的连接操作，实现路径的逆向连接。最后获得所需要的查询路径。从而生成以最大相关节点集为基础的 XML 查询技术。

以下我们将以图 1 中给出的 DTD 文档作为源模式文档，通过将其文档转化为 SLT 树再对其生成的 SLT 树划分得到一个 Node 树，将该文档分割为问卷的元素信息和格式信息，元素的类型需要根据元素类型中的选项给出，而对应的题号通过匹配相应类型得到。在目标模式文档中，希望得到的是题号为 X 的节点，则利用其最大相关节点集的 XML 查询方式，利用上边定义的四元操作实现如下：首先生成最大相关节点集如下，

A = (company, name, employees, wages)；

B = (employees, employee, position, wage)；

C = (wages, wage, positon)；

获得题号为 X 则通过对最大相关节点集进行查询获得 B’ = (employee, wage)，通过结构连接操作，将 B’与其上层父 Node 节点进行连接通过共有的 company 节点进行匹配连接，获得其路径为 company-employees-employee 则查找成功，若没有匹配结果则查找失败。

同时通过对于不同层次的 XML 查询连接进行试验比较，通过对路径查询和基于最大相关节点集的查询比较，对于层次较少或节点较少的 XML 其比较次数和效率基本相同，但是对于多枝节或者是层次较多的 XML 来说，利用最大相关节点集的比较次数要明显小于利用路径查询进行直接归并连接的次数，XML 不同查询方式下次数比较对比如表 1。

其中 layer 表示 XML 层次，nodes 表示 XML 转化后生成 SLT 树后的节点数，Xpath 则是表示路径查询过程中节点的比较次数，X_Nodes 则表示本文所将的最大相关节点集的 XML 查询方式。通过比较可以看出，在层次单一，节点较少的情况下两者的比较次数基本相当，但是随着层次和节点的增加，其比较次数在传统的 XML 路径查询方式中将会变的越来越多，从而导致效率的下降，而对于基于最大相关节点集合的节点比较虽然也是逐渐增加但是明显要小于传统的 XML 路径查询过程中的比较次数。而在进行 XML 查询的过程中基于路径的查询相当于对于 XML 生成树的遍历，如何以最快的速度获得要查询的 XML 内容即在 XML 生成树上的节点则是查询的关键，要实现以最快的速度获得要查询的内容，比较次数越少则实现的效率相当来说就会越高。所以通过基于最大相关节点集合的想法可以在很大程度上减少比较的次数，缩短遍历的路径从而提高查询的效率。

5. 总结

如何减少路径查询过程中的计算是 XML 路径查询的关键问题。本文结合最大相关节点集的概念，提出了逆向路径查询的方法。该方法在进行查询连接的过程中通过扩展最大相关节点集的思想，实现利用最大相关节点集进行连接。连接的过程中通过减少大量的中间枝节提高了查询的效率。同时在进行查询的过程中加入选择、投影运算减少在确定其目标节点的前提下其它无用节点的影响，提高其命中效率。本文的查询工作主要是针对层次结构明显同时查询节点之间有密切联系的 XML 进行查询，而对于匹配不成功的处理方式和对于松散的，联系不密切的查询将是下一步的主要工作。

参考文献 (References)

[1] 张澎, 徐红云, 王鲁达等. 一种基于 XML 的树型代数[J]. 计算机仿真, 2008, 5(25): 147-151.

[2] 沈剑沧, 鲍培明. XML 查询方法的设计与研究[J]. 计算机工程, 2007, 21(33): 63-65.

[3] 孔令波, 唐世渭, 杨冬青等. XML 数据的查询技术[J]. 软件学报, 2007, 6(18): 1400-1418.

[4] 杨文军, 李涓子, 王克宏. 基于关系树模型实现数据转换[J]. 计算机科学, 2004, 11(31): 114-117.

[5] 王静, 孟小峰, 王宇. 以目标节点为导向的 XML 路径查询处理[J]. 软件学报, 2005, 5(16): 828-837.

[6] 门爱华, 周立柱, 张亚鹏. XML 数据库结构连接算法之分析[J]. 计算机科学, 2007, 6(34): 136-139.

[7] 孔令波, 唐世渭, 杨冬青等. XML 数据的查询技术[J]. 软件学报, 2007, 6(18): 1400-1418.

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

[9] 王国仁, 乔百友, 韩东红. 基于分片的 XML 快速结构连接算法[J]. 计算机学报, 2008, 1(31): 78-90.

[10] M. Fernadez, W.-C. Tan and D. Suciu. SilkRoute: Trading between relations and XML. Computer Network, 2000, 33(1-6): 723-745.

