对左兄弟/右兄弟结构连接算法的研究与改进

王治和
（西北师范大学数学与信息科学学院　兰州 730070）

摘　要　结合区间编码和结点模型映射方法提出一种用于关系数据库的扩展存储模式。通过按结点编码中的广度遍历序号建立聚集索引，实现左兄弟/右兄弟关系结构连接算法的改进。改进后的算法降低了内存空间的开销，缩小了列表的扫描范围，明显提高了查找匹配速度，达到了查询优化的目的。

关键词　XML，查询优化，扩展存储模式，左兄弟/右兄弟，结构连接算法

Research and Improvement for the Preceding-sibling/Following-sibling Structural Join Algorithm

WANG Zhi-He
(College of Mathematics and Information Science, Northwest Normal University, Lanzhou 730070)

Abstract　By the use of the region coding and node model mapping method, an extended storage schema is presented for relational-database. We establishe clustered index for the optional breadth traversal serial number of node coding to improve the structural join algorithms for processing preceding-sibling/ following-sibling relationships. The proposed method has advantages of saving memory, shrinking the scanning area of list and remarkably improving the rate matching of lookup. This algorithm improves the efficiency of XML data query.

Keywords　XML, Query optimization, Extended storage schema, Preceding-sibling/following-sibling, Structural join algorithm

1　改进的扩展存储模式

本文在 XRel[1] 和 XParent[2] 两种模式的基础上，设计了一种基于结点模型映射方法的 XML 数据的关系存储模式——改进的扩展存储模式（extended storage schema）。该模式是通过区间编码方案来反应 XML 文档模型的，目的是实现对文档树中任意两个结点对之间的祖先/后裔关系、双亲/孩子关系以及文档位置关系的检测，并加速 Xpath 路径表达式的计算。其基本思想是：对 XML 文档树中的所有结点分别进行一次深度优先遍历和一次广度优先遍历，分别产生这些结点的深度扩展遍历序号和广度遍历序号，以深度扩展遍历序号作为结点的标识。XML 文档树中的结点包括元素结点、属性结点以及属性值，它由四个关系表组成，如表 1 所示。

Element 表、Attribute 表分别用来存储元素结点和属性结点的内容；Value 表用来存储 XML 文档的内容，即所有文本结点的内容及属性结点的值；Path 表存储 XML 文档中所有从根开始到另一个元素或属性的路径。

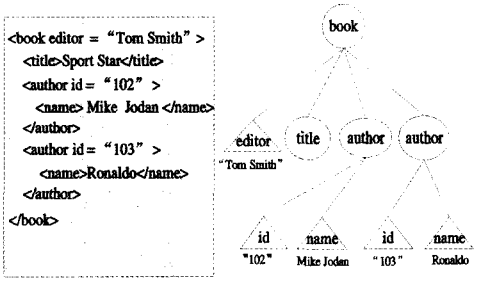
表 1　改进的 XML 文档关系存储模式

Element	(docID, order, maxorder, depth, pathID, parentorder, parentman, gdorder)
Attribute	(docID, order, maxorder, depth, pathID, parentorder, parentman, gdorder)
Value	(dorID, order, pathID, Value)
Path	(pathID, pathexp)

其中，docID 是该结点所在文档的文档标识；order 是该结点的深度扩展遍历序号；maxorder 是以该结点为根的子树中所包含的所有元素结点、属性结点以及文本结点中最大的 order，以反映结点之间的祖先/后裔关系和之前/之后关系（注：只要是不小于最大 order 的任意一个整数，为便于给后裔对象的插入预留序号的空间，该结点之后的第一个结点的 order 取值只要大于该结点的 maxorder 值即可）；depth 是该结点在文档树中所处的层数以反映祖先/后裔关系中的嵌套层数关系；pathID 是从根开始到该结点的路径表达式的标识，以支持对简单绝对路径表达式按路径索引的方法（即路径法）进行计算；parentorder 和 parentmax 分别是该结点的双亲元素结点的 order 和 maxorder；parentorder 用来支持结点之间双亲/孩子关系、左兄弟/右兄弟关系的计算，parentmax 用来加速结构连接的计算；gdorder 是广度遍历序号，用于支持兄弟结点关系的计算。pathexp 域存储标记路径。

Element 表、Attribute 表和 Value 表中（docID、order）是主键，pathID 是外键；Path 表中 pathID 是主键。

图 1 所示的是一个 XML 文档片断及相应的数据模型。



(a)XML 文档片断　　(b)相应数据模型

图 1　XML 文档片断及相应的数据模型

表 2 反映的是该 XML 文档的扩展存储模式的编码结

果，包括元素和属性结点的(order，maxorder，depth，gdorder)字段。

表2 XML文档片断的扩展存储模式的编码结果

元素或属性名	(order，maxorder，depth，gdorder)
book	(1,150,0,2)
editor	(2,6,1,2)
title	(7,27,1,3)
author	(39,66,1,4)
id	(40,40,2,6)
name	(41,45,2,7)
author	(67,94,1,5)
id	(68,68,2,8)
name	(69,73,2,9)

2 左兄弟/右兄弟结构连接算法的改进

2.1 XPath加速器

为了有效地支持XPath各种查询轴的计算，包括包含关系的查询轴和文档位置关系的查询轴，T. Grust提出了一种为了加速XPath定位步计算的一种数据库索引结构，称为XPath加速器(XPath Accelerator)[3]。其基本思想就是通过索引快速地确定XPath定位步的记录扫描范围。

2.2 基于B+树XPath加速器的索引嵌套循环连接算法

假设计算一个XML文档树的右兄弟轴计算代价。列表P存储的是所有上下文结点，即P是左兄弟元素结点列表，a是following-sibling轴(选择上下文结点的所有在其后(右)的兄弟)，f是一个元素的标记，列表F是右兄弟元素结点列表，两个列表P和F都是基于深度遍历的结点序号建立了聚集索引。那么对于XPath查询定位步P/a::f，利用XPath加速器的计算过程是：对于列表P中的每一个上下文结点p，首先，在列表F中通过索引定位它的扫描始点，然后，在列表F中从扫描始点开始扫描结点p的following域，判断所扫描的每一个结点f是否是上下文结点p的右兄弟。此处理过程类似于关系数据库中的索引嵌套循环连接。

如图2反映的是一个样例XML文档树。其中，正方形表示列表P中的结点，椭圆形表示列表F中的结点，六边形表示同时属于列表P和F中的结点，三角形表示其它结点。带阴影的正方形和六边形结点表示它们存在右兄弟结点。

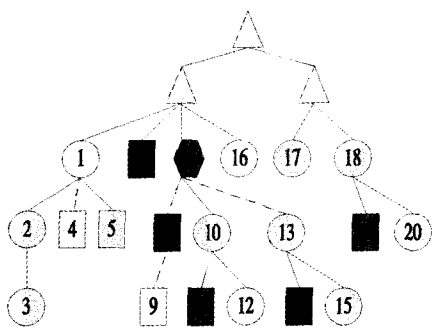


图2 一个XML文档样例

该算法关于F的扫描代价如图3所示，其中，使用n表示列表F中的结点n，使用(n)表示列表P中的结点n。图(a)反映列表P和F的内容，并在列表F中标出了连接结果；图(b)反映的是对列表F的扫描代价，带箭头的线段表示对列表P中的给定结点在列表F中的扫描范围，例如，对于列表P中的(4)、(5)号结点，它们在列表F中无右兄弟，(6)号结点的右兄弟为7，但是它们扫描范围都是结点7、10、…、20。

显然，对于XPath加速器，在计算右兄弟(following-sibling)轴时，对F列表进行了大量的无用扫描或重复扫描。本文对此算法提出了相应的改进算法。

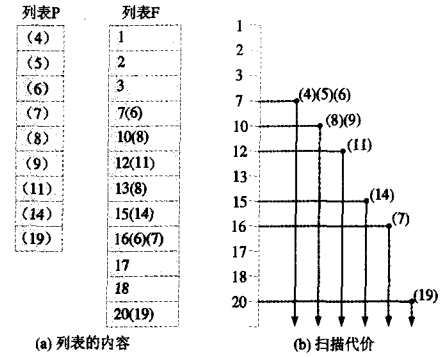


图3 计算右兄弟的扫描代价

2.3 改进的左兄弟/右兄弟结构连接算法

假设列表P′存储所有上下文结点，列表F′存储所有兄弟结点，将两个列表基于扩展存储结点编码中的关于结点的广度遍历序号gdorder建立聚集索引，则图2将转换为图4所示。

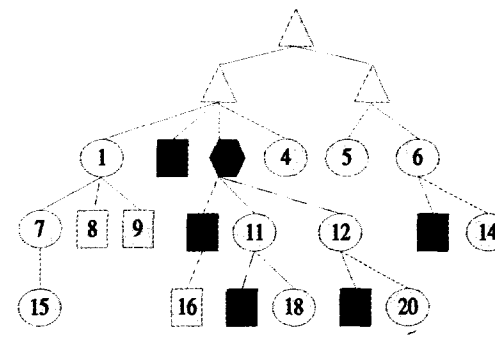


图4 改进后的XML文档样例

那么对XPath查询定位步P/a::f，改进后的算法处理过程如下：

(1)记P′列表中当前结点为r。

(2)在列表F′中通过索引定位它的扫描始点。即定位在列表F′中聚集索引大于r的聚集索引的第一个结点上。

(3)如果扫描始点是r的右兄弟转(4)，否则转(5)。

(4)从第一个可能的右兄弟开始，在F′列表中进行兄弟关系匹配，并输出匹配的结点对，直到第一个非r的右兄弟结点为止。

(5)转第(1)步，对P′列表中的下一个结点进行处理。

算法

输入：参与连接的左兄弟列表P′和右兄弟列表F′。

输出：按左兄弟/右兄弟关系连接的连接结果output。

```
Gdorder-Following-Sibling(P′,F′)
/*假设P′列表和F′列表中的所有结点都有相同的docID*/
1 p=P′→firstNode;r=p;output=NULL;
2 while (P′and F′ are not empty){
3  在列表F′中通过索引定位它的扫描始点为f;
4  if (r.parentorder = = f.parentorder){
5    append(r,f) to output;f= f→nextNode;
  }// 输出r的连接结果
```

```
6  r=r->nextNode;
7}
```

该算法关于 F′的扫描代价如图 5 所示。

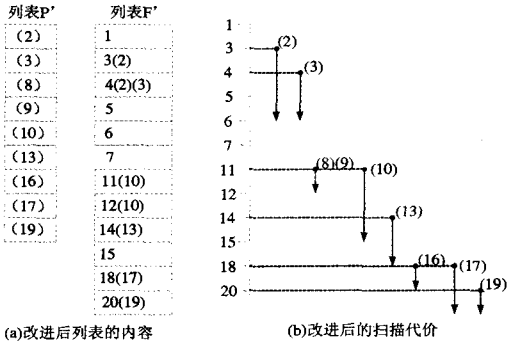


图 5　改进后的左兄弟扫描代价

从图 5(b)中可以看出，箭头的线段表示对列表 P 中的给定结点在列表 F 中的扫描范围，例如，对于列表 P′中的(2)号结点，它在列表 F′中的扫描范围是结点 3 到结点 5；(3)号结点，它在列表 F′中的扫描范围是结点 4 到结点 5；(8)号、(9)号结点，它们在列表 F′中的扫描范围都是结点 11 等等。

2.4　算法分析及结论

改进后的左兄弟/右兄弟算法中的聚集索引是按广度遍历序号建立的，所以每一个上下文结点的右兄弟的索引序号一定大于它的索引序号且是连续的，这样对每一个上下文结点进行扫描时，只需从定位的扫描始点即索引序号大于它的第一个右兄弟元素开始，扫描到第一个非它的右兄弟结点为止，即每一遍的扫描过程都只多扫描一个结点，并不需要像改进前的算法一样需扫描到右兄弟列表的末端，从而避免了大量的重复扫描。

我们通过扫描结点数量的多少来反映算法跳过不匹配元素结点的能力，评估算法性能。

在上例中，改进前对右兄弟列表总的扫描范围为 3×9+2×8+1×7+1×5+1×4+1×1=60；而改进后对右兄弟列表总的扫描范围为：

1×3+1×2+2×1+1×3+1×2+1×1+1×2+1×1=16。

改进后的扫描范围比改进前的扫描范围大大减少了。

假设右兄弟列表的结点数为 n，则改进前的扫描结点范围的数量级为 $n^2/2$，而改进后的扫描结点范围的数量级为 n，从而可以看出改进后的算法性能有很大提高。

参 考 文 献

1　Yoshikawa M, Amagasa T, Shimura T, et al. XRel: A path-based approach to storage and retrieval of XML documents using relational databases. ACM Trans on Internet Technology, 2001, 1(1):110～141

2　Jiang H, Lu H, Wang W, et al. XParent: An Efficient RDBMS-Based XML Database System. In: Proceedings of the 18th International Conference on Data Engineering, 2002

3　Grust T. Accelerating XPath location steps. In: Franklin M J, Moon B, Ailamaki A, eds. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). Madison: ACM Press, 2002. 109～120

4　World Wide Web Consortium. XML Path Language(XPath)1.0. W3C Recommendation. 16 November 1999. http://www.w3.org/TR/xpath

5　Jiang H F, Lu H J, Wang W, et al. XR-Tree: Indexing XML data for efficient structural joins. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc of the 19th Int'l Conf. on Data Engineering. Los Alamitos: IEEE Press, 2003. 253～264

6　Fan C, Funderburk J, Lam Hou-in, et al. XTABLES: bridging relational technology and XML. IBM Systems Journal, 2002, 41, 4

7　刘云生，万常选，徐升华．基于关系数据库有效的实现 RPE 查询．小型微型计算机系统，2003，24(10)：1764～1771

8　Al-Khalifa S, Jagadish H V, Koudas N, et al. Structural joins: A primitive for efficient XML query pattern matching. In: Agrawal R, Dittrich K, Ngu AHH, eds. Proc. of the 18th Int'l Conf. on Data Engineering. Los Alamitos: IEEE Press, 2002. 141～152

(上接第 84 页)

类子网内节点间的通信时间远高于聚类子网间节点的通信时间，并且超级节点的处理能力、存储容量和通信速率远高于普通节点的平均通信速率。实验数据采用 Ramen 和 Nimda 的样本模拟特征。

对传统基于网络联动的预警方式和 HPOWP 的蠕虫预警模型的仿真实验结果如图 6 所示，HPOWP 模型可以更早地发现并抑制蠕虫行为。

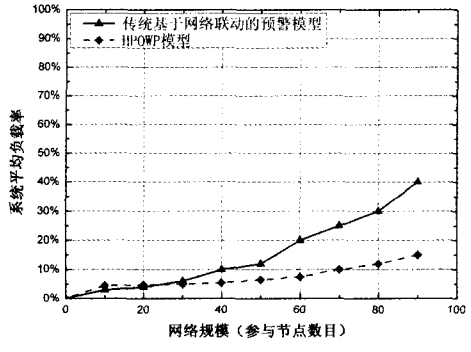


图 7　系统平均负载变化图

随着网络规模的增加，系统节点需要获取并处理更多的可疑信息，负载会不断增加。系统平均负载随网络规模变化的仿真试验结果如图 7 所示，HPOWP 模型的系统平均负载增长缓慢。

结论　本文提出的基于 P2P 的网络蠕虫预警模型(HPOWP)模型能够有效地对蠕虫行为进行预警响应。由于 HPOWP 采用了分层的 P2P 体系结构和基于 DHT 的分布式数据聚合算法，系统具有良好的伸缩性，其系统平均负载不会随网络规模的扩大而迅速增加。因此，HPOWP 更能适应大规模网络环境下的蠕虫防御要求。

参 考 文 献

1　Hathai Tanta-ngai, McAllister M. A peer-to-peer expressway over Chord [J]. Mathematical and Computer Modelling, 2006, 44: 659～677

2　卿斯汉，文伟平，蒋建春．一种基于网状关联分析的网络蠕虫预警新方法[J]．通信学报，2004，25(7)：62～70

3　Kim H, Karp B. Autograph: Toward automated, distributed worm signature detection [C]. In: Proc. of the USENIX Security, San Diego, CA, 2004. 271～286

4　Singh S, Estan C. Automated worm fingerprinting [C]. In: Proc. of the 6th ACM/USE2NIX Symposium on Operating System Design and Implementation (OSDI), San Francisco, CA, 2004. 45～60