XML查询结构连接顺序选择算法分析与优化

张艺濒,谢金晶
(武汉大学 计算机学院,湖北 武汉 430072)

摘 要:如今对XML查询的优化是对XML的热点研究方向。其中的结构连接操作是XML数据库查询的主要操作。和关系数据库中的连接运算一样,结构连接顺序的选择是XML数据库查询优化的核心。文中主要通过对XML查询优化中各种选择连接顺序算法的研究,提出了一种优化的算法,在规模较大的XML查询中能够有效缩减搜索空间,提高效率。

关键词:XML查询优化;结构连接顺序选择;动态规划
中图分类号:TP302.1 **文献标识码**:A **文章编号**:1673-629X(2007)01-0082-03

Analysis and Improvement of Structural Join Order Selection Algorithm on XML Query

ZHANG Yi-bin, XIE Jin-jing
(College of Computer Science, Wuhan University, Wuhan 430072, China)

Abstract: Today, it's hot to analyse XML query optimization. Structural join operations are central in XML query processing. As join operation to relational database, structural join order selection is at the heart of query optimization in an XML database. In this paper, analyze the basic structural join algorithms and introduce a new algorithm, this algorithm can effectively reduce index spaces.

Key words: XML query optimization; structural join order selection; dynamic programming

0 引 言

随着XML在互联网上的广泛应用,XML数据的存储和查询已经成为迫切需要解决的问题。对XML查询的优化更是对XML的热点研究方向。XML查询典型是由多个XPath查询组成,而任何一个复杂的XPath查询均可由若干个连接操作形成[1]。对一个复杂的XPath查询,连接关系的顺序有很多种,而每一种的开销往往不同。所以,如何适当地选择连接顺序是一个至关重要的问题。

文献[2]中提供了穷举式动态规划法、带剪枝的动态规划法、大幅剪枝的动态规划法等选择连接算法。文中将对各个算法作分析和比较,并在此基础上提出一种优化算法。

1 基本概念

在多阶段决策问题中,各阶段的决策依赖于当前状态,又引起状态的转移,这种解决多阶段问题的算法称为动态规划[3]。算法的核心是将问题划分成若干个可以独立解决的子部分,然后再进行汇总。

在XML查询中,一个XPath语句首先被转化为一棵树,称为查询树。查询的求解过程为:

从初始查询树开始,选择查询树中的某一条边,对该边上相连接的两个结点做连接,连接的结果用一个新的结点表示,并代替原树中的两个结点。每次连接时都会使结点减少一个,同时产生一个新的树。当树只包括一个结点时,整个求解过程便结束。

将每一步连接的代价作为查询树每条边的权值,结构连接顺序选择算法便转化为经典的求最短路经问题。把最短路径问题的动态规划算法应用到该模型,就变成了连接顺序求解的动态规划策略。

2 基本算法介绍分析

2.1 穷举式动态规划(exhaustive dynamic programming, DP)

文献[2]中给出了使用动态规划时的查询树求解过程的示意图(如图1所示),初始查询树为S_{00},即求解的XPath语句为A[B/C]/D。按照查询算法,首先连接两个相邻结点,有6种不同的方案,可以得到6个结果S_{10}~S_{15},其中AB结点表示A与B的连接结果。S_{10}~S_{15}再分别继续连接,得到第三层的所有结果。再连接一次,到达最终的结束状态,即结果ABCD。由此总结出查询树的求解过程实际上是从初始状态出发,经过n次连接到达结束状态的过程。连接过程中边的选择顺序即为连接的顺序。

收稿日期:2006-04-13
基金项目:湖北省自然科学基金资助项目(2005ABA238)
作者简介:张艺濒(1985-),女,江西景德镇人,研究方向为信息安全;导师:董文永,副教授,研究方向为智能计算。

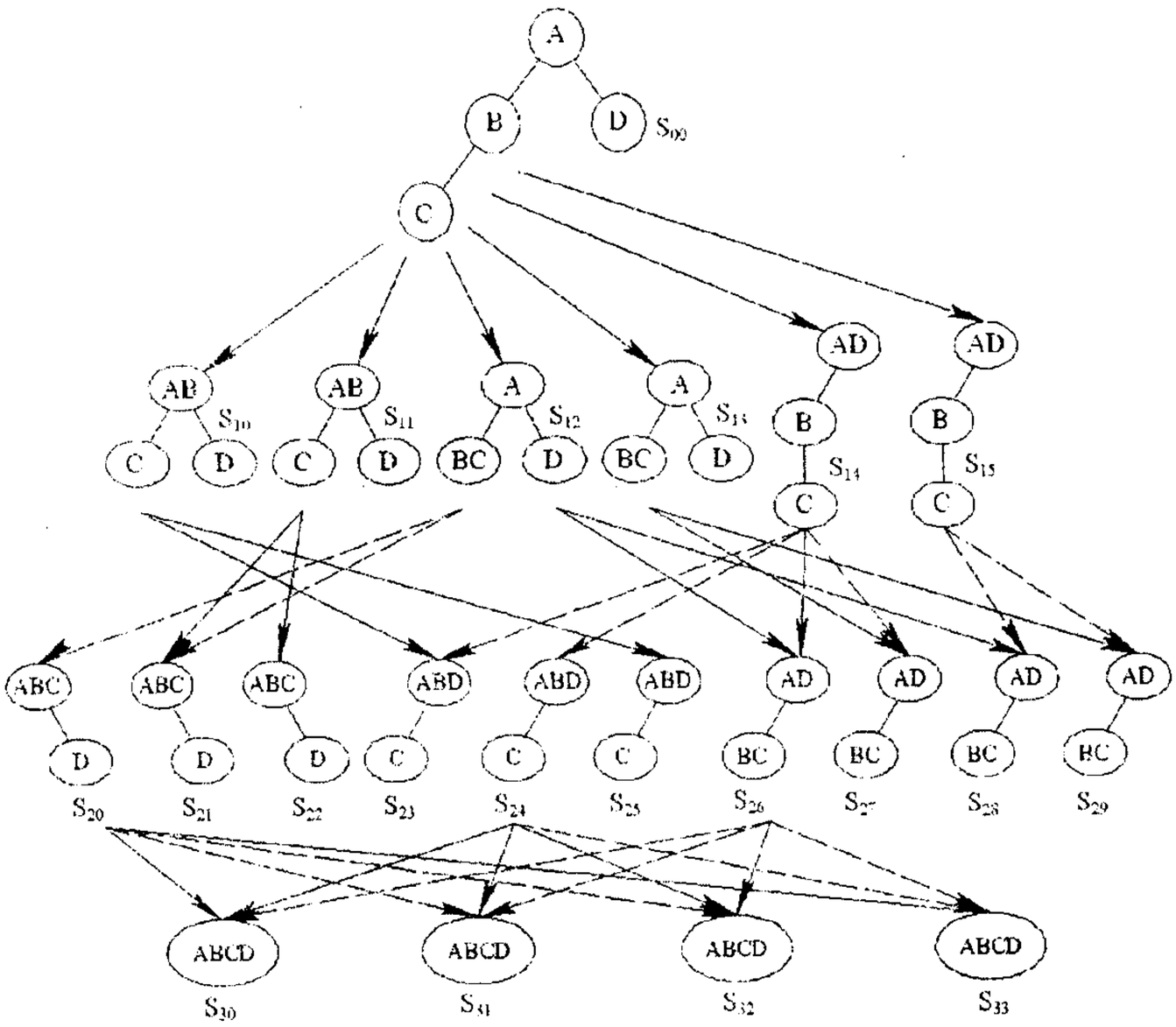

图 1　使用动态规划时的查询树求解图

注意到结构连接运算不符合交换率,参加连接的操作数要求按序排列,当被连接的两个结点集的左右顺序互换时,计算代价也相应发生变化。

考虑一个 n 个结点的查询树,该算法的时间复杂度为:$O(2^n \times n!)$。

2.2　带剪枝的动态规划(dynamic programming with pruning,DPP)

带剪枝的动态规划[4]算法较前者改进之处在于:

(1) 不必一定要等到第 k 层的所有状态都产生才能移动到第 $k+1$ 层。

(2) 对不可能产生最优解的中间状态,不产生下一个状态。

首先,扫描这整棵查询树,得到每一个状态到达最终状态的代价的上限的估计值。在所得的这些状态以及上一步骤下的状态中,选出代价最小的进行下一次连接,对不可能产生最优解的中间状态,不产生下一个状态。如此往复,直到所有结点连接完毕。

文献[2]中对该算法的时间复杂度进行了详细分析:假设这个查询树是一棵棵度为 f,深度为 d 的树,边的总数为 $|E| = \sum_{k=1}^{d} f^k$。层数为 lv 时它的状态数为 $S_{DPP}(lv) = f_{S_{DPP}} \times \left[\frac{lv}{|E|}\right]$。其中,$f_{S_{DPP}} = O(lv^2)$ 表示每种状态下可能出现的排序顺序数。这个算法的所要评估状态个数的最大上限为 $\sum_{lv=0}^{|E|} S_{DPP}(lv)$。

2.3　带大幅剪枝的动态规划(dynamic programming with aggressive pruning,DPAP)

在 DPP 算法的基础上,文献[2]又引入了其他两种启发规则,来对连接代价进行优化。其中一种启发式规则为:定界展开的大幅剪枝动态规划(DPAP with expansion bound)。另一种为:左深树计划上的大幅剪枝动态规划(DPAP on left-deep plans)。

定界展开的大幅剪枝动态规划[5,6]的主要思想为:引入一个参数 Te,限制在每一层能够展开的状态的数目。显然,可以将其理解为一种贪心的思想,虽然不一定能找出最优解,但是确实限制了状态数量,搜索效率得到较大提高。

2.4　算法分析比较

DP 和 DPP 算法能够从整个解空间中找到最优的连接计划,但是其搜索的本质导致计算量与被连接的关系数成指数关系[7]。采用这种方法来寻找 5-6 个关系的最佳连接顺序是合理高效的。当连接结点数超过一定范围,花费在搜索上的时间将呈指数上升。

DPAP 算法能够减少搜索空间,但是可能会错过最优计划。对于中等规模的查询树,不失为一个较好的方案。

3　分树连接的动态规划

基于上述分析比较,各算法的计算复杂程度还是相当大的。为此笔者提出了一个新的算法,即分树连接的动态规划,如图 2 所示。

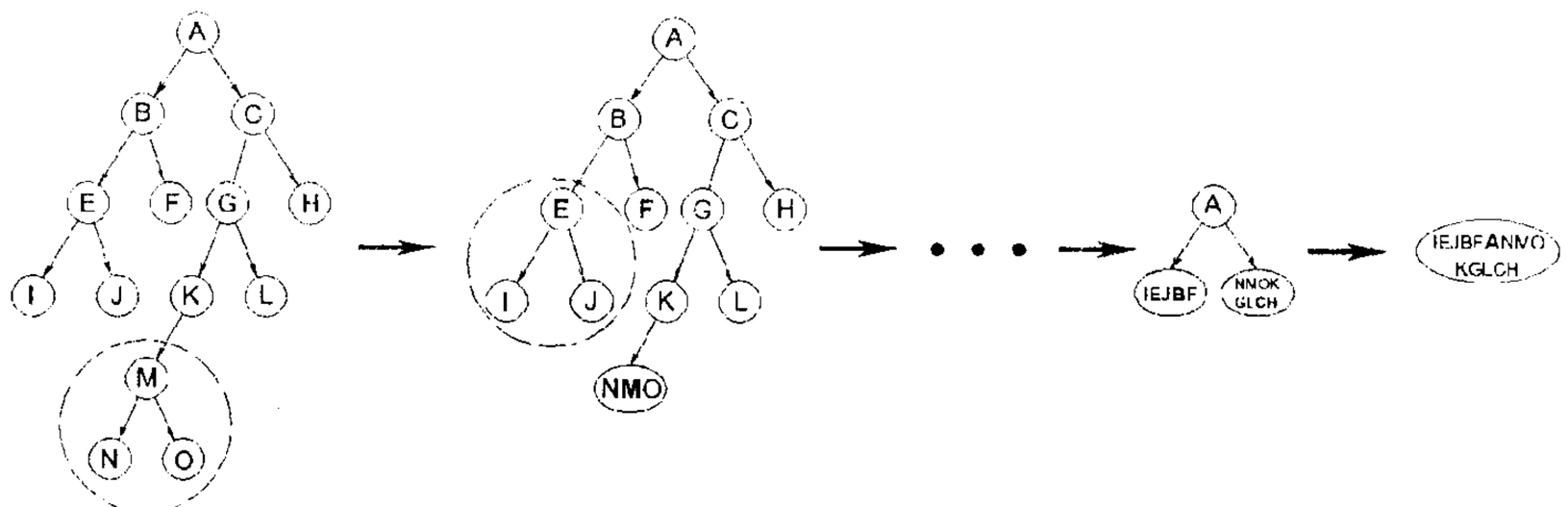

图 2　使用分树连接的动态规划算法过程示意图

图 2 所示的查询过程:先利用某种算法自上而下在当前查询树中分离出一部分结点(M,N,O),使得结点集仍然为一棵树,小树中结点个数上限值为 M。利用 DP 算法(此处及下文中如未特别指出,DP 算法为前述动态规划算法的总称)对小树求解最佳连接顺序(最后一次连接是以树根(M)为序),并将求解出来的结果用一个结点(NMO)替代小树,形成新的查询树。循环找出小树,直至当前查询树中的结点数目 $\leqslant M$。对最后的查询树使用 DP 算法,求出最终连接顺序(IEJBFANMOKGLCH)。

●具体算法如下:

```
function()
{
    输入;
    T //T 为一棵树,即初始查询树
    M=x;//设置 M 的大小
    While(N>M)
    {
        Select(M);//分离出一棵小树,返回小树的结点
        DP();//对小树使用 DP 算法,返回以树根为序的连接
顺序
        Merge();//合并结点
        N=N-M+1;// 修改当前查询树结点个数
    }
    DP();//对最后的查询树使用 DP 算法
}
```

●算法分析:

本算法利用了局部优推导出全局较优的近似算法思想。当查询树结点数增大到一定程度,对整棵树使用动态规划算法会导致状态数目过多、搜索空间过大、花费的时间代价过多。当使用笔者提出的算法,结点数目会在循环的过程中以 M 递减,而每次参与运算的仅仅只是 M 个结点,有效地缩减了搜索空间。同时每个局部的较优也保证了全局的较优。

假设使用 DP 算法对一个结点数为 M 的树进行连接顺序选择,时间复杂度为 $T(M)$。每循环一次查询树结点减少 M 个,故使用本算法总时间复杂度为:$T(M) * \frac{N}{M}$。在这里忽略了分离一个小树的算法的时间复杂度,这是合理的,因为该算法时间复杂度仅在线性级别。与普通算法的复杂度相比,分树连接的动态规划在时空上得到了很大的改进。虽然不能保证找到最优的解,但它缩减了搜索空间,效率很高。

● 对 M 大小的讨论:

本算法总时间复杂度为:$T(M) * \frac{N}{M}$,随着 M 的增大,$T(M)$ 增大,$\frac{N}{M}$ 减小。当 M 增大到 N 或者减小到 1,算法都将退化为普通DP算法。由此看出,选择一个合适大小的 M 对本算法至关重要。笔者经过粗略的分析,M 在 3 ~ 10 之间是比较适宜的,且根据查询树形状的不同,M 有不同的适用值。

● 算法适用条件:

当 N 较小时,使用前述的DPP或者DPAP算法在较短的时间内即可得到较优的连接顺序。若使用本算法,反而将问题复杂化。

当 N 较大时,前述基本算法将不再适用。利用本算法则可以在较短的时间内得到一个较优的解。

4 结束语

通过对现有的选择连接算法的比较分析,针对其不足之处,笔者提出了一种优化策略,并给出新算法的实现过程及性能分析。与普通算法相比,该算法在较大规模的XML 查询中能有效地缩减搜索空间,提高效率。

参考文献:

[1] Garcia - Molina H, UIIman J D, Widom J. Databases System Implementation[M]. 北京:机械工业出版社,2002.

[2] Wu Y, Patel J M, Jagadish H V. Structural Join Order Selection for XML Query Optimization[C]//In: Casati F. Proceedings of the 19th IEEE ICDE International Conference on Data Engineering. Los Alamitos, Bangalore, India: IEEE Computer Society, 2003:443 - 454.

[3] Amer - Yahia S, Cho S, Lakeshmanan L, et al. Minimization of Tree Pattern Queries[C]//In: Proc. of the 2001 ACM SIGMOD Conf. on Management of Data. Santa Barbara, California, USA:[s. n.], 2001:21 - 24.

[4] 万常选. XML 数据库技术[M]. 北京:清华大学出版社, 2005.

[5] Al - Khalifa S, Jagadish H V, Koudas N, et al. Structural Joins: A Primitive for Efficient XML Query Pattern Matching [C]//In: Proceedings of the 18th International Conference on Data Engineering. Los Alamitos: IEEE Press, 2002: 141 - 152.

[6] Lee Y K, Yoo S J, Yoon K. Index structure for structured documents[C]//In: Proceeding of the 1st ACM Int'1 Conf. on Digital Libraries. Net York: ACM Press, 1996:91 - 99.

[7] 刘云生,伍慧敏. XQuery 查询优化中结构连接顺序选择算法[J]. 计算机应用研究,2005(7):87 - 89.

(上接第 81 页)

其存储系统[J]. 合肥工业大学学报,2005,28(6):581 - 584.

[3] Berners - Lee T, Hendler J, Lassila O. the Semantic Web[J]. Scientific American, 2001, 284(5):34 - 43.

[4] 李 涛,董红斌,彭煜伟. 基于语义 web 的组件描述方法[J]. 计算机工程,2004,30(7):39 - 40.

[5] Fallside D C. XML Schema Part 0: Primer[EB/OL]. 2004 - 06 - 05. http://www. w3. org/TR/2001/REC - xml - schema - 0 - 20010502.

[6] 张维明. 语义信息模型及应用[M]. 北京:电子工业出版社, 2002:148 - 172.

[7] Hjelm J. Creating the Semantic Web with RDF[M]. [s. l.]: Villey Computer Publishiing, 2001.

[8] Lassila O, Swich R. Resource Description Framework Model and Syntax Specification[EB/OL]. 1999 - 02 - 22. W3C Recommendation, http://www. w3c. org/TR/1999/REC - rdf - Syntax.

[9] Horrocks I. DAML + OIL: a Description Logic for the semantic web[J]. IEEE Bulletin of the Technical Committee on Data Engineering, 2002, 25(1):4 - 9.