XML 数据的路径表达式查询优化技术*

吕建华+, 王国仁, 于 戈

(东北大学 信息科学与工程学院,辽宁 沈阳 110004)

Optimizing Path Expression Queries of XML Data

LÜ Jian-Hua+, WANG Guo-Ren, YU Ge

(Faculty of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

+ Corresponding author: Phn: 86-24-83681250; E-mail: dbgroup@mail.neu.edu.cn

http://www.neu.edu.cn

Received 2002-08-12; Accepted 2002-10-14

Lü JH, Wang GR, Yu G. Optimizing path expression queries of XML data. *Journal of Software*, 2003,14(9): 1615~1620.

http://www.jos.org.cn/1000-9825/14/1615.htm

Abstract: Path expression is one of the core components of most XML query languages, and many evaluation methods for path expression queries are proposed recently. However, there are few researches on the issue of path expression optimization. In this paper, two kinds of path expression optimizing principles are proposed, named path shorten and path complementing, respectively. The path shorten principle reduces the querying cost by shortening the path expressions with the knowledge of XML schema. While the path complementing principle tends to substitute the user queries with the equivalent lower-cost path expressions. The experimental results show that these two techniques can work on most path expression queries and largely improve the efficiency of path expression query processing.

Key words: XML; path expression; query processing; querying cost; query optimization

摘 要: 路径表达式作为 XML 数据查询语言的核心部分,关于它的计算方法的研究成果已有很多,然而针对路径表达式本身进行优化的研究却相对较少.提出了两种针对路径表达式的优化策略:路径缩短策略和补路径策略,从而提高了 XML 路径查询效率.路径缩短策略根据 XML 文档模式信息,将路径表达式查询长度缩短,从而简化查询本身以降低需要的查询代价;而补路径策略则试图使用代价更小的等价路径表达式来替换原始查询.经过对实验数据的分析,这两种优化策略对于绝大多数路径表达式查询可以应用,并可大幅度地改进路径表达式的查询性能.

关键词: XML;路径表达式;查询处理;查询代价;查询优化

中图法分类号: TP311 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60173051, 60273079 (国家自然科学基金); the Foundation of Teaching and Research Award Program for Outstanding Young Teachers in Higher Education Institution of China (教育部高等学校优秀青年教师教学和科研奖励基金); the Foundation for University Key Teacher by the Ministry of Education of China (教育部高等学校骨干教师资助计划)

第一作者简介: **吕建华**(1977－),男,河北承德人,博士生,主要研究领域为 XML 数据库,索引技术,查询处理技术.

XML(extensible markup language)语言是 W3C 组织提出的一个 Internet 上数据表示和数据交换的新标准.由于正则路径表达式(regular path expression,简称 RPE)可以访问 XML 文档结构的任意层次,大部分 XML 查询语言都采用正则路径表达式作为它们的主要组成部分之一.因此在 XML 查询处理中,RPE 的性能非常重要.

用关系数据库管理 XML 数据的系统优化正则路径表达式的一个通用的方法是:首先根据 XML 文档的模式信息或者数据统计信息将 RPE 重写成若干个简单路径表达式(SPE)的集合;然后计算 SPE,例如,在 VXMLR[1]系统中,包含"//"和"*"的 RPE 查询被重写为简单的路径表达式,再根据关系数据库模式翻译成 SQL 语句进行处理.在 Lore 系统中,有 3 种 SPE 路径表达式的查询策略:自顶向下(top-down)、自底向上(bottom-up)和混合策略(hybrid)[2].作为混合方法的一个改进,外延连接算法得到了广泛的关注.文献[3]提出了一种类似的方法,称为路径连接;文献[4]以路径连接方法为基础,提出了一个索引结构 APEX,并根据查询负载情况建立适应性的路径索引.而文献[5,6]则详细定义了元素外延等概念,并设计了若干索引结构,包括元素索引、属性索引、引用索引和路径索引等,以支持外延连接算法.文献[7]针对对象数据库中的路径表达式查询使用代价模型估算查询代价,以选择最佳的查询方法.但是,它们都没有提出针对路径表达式特性本身的优化方法.

本文基于外延连接算法提出了两个路径表达式查询的优化策略,它们也适用于其他路径表达式计算方法.它们分别是:路径缩短(path shorten)策略和补路径(path complementing)策略.本文第 1 节给出了一些相关定义,它们是进行路径表达式查询处理的基础.第 2 节和第 3 节分别详细讨论了路径缩短策略和补路径策略.第 4 节给出了测试结果以及相关性能分析.第 5 节总结全文.

1 相关定义

1.1 XML数据模型与数据模式

XML 文档可以被表示为标签在节点上的树,即 $T_d=(V_d,E_d,\delta_d,\Sigma_d,root_d,oid,type_d,oid_d)$,其中 V_d 是树中的节点集合;E_d 是树中边的集合,边表示元素节点之间的关系;δ_d 是节点之间的映射函数.每一个节点都有一个类型与一个惟一的标识符相对应,而这些类型和标识符分别属于集合 Σ_d 和 oid,相应的映射函数为 $type_d$ 和 oid_d.每一个 XML 数据树都有一个根节点 $root_d$.如图 1 所示是一个 XML 文档相应的数据树的片断,该文档来源于 XML 测试数据集 XMark[8].XML 文档的数据模式可以被定义为一个有向图,即 $G_t=(V_t,E_t,\delta_t,\Sigma_t,root_t)$.其中 V_t 是图中的节点集合;E_t 是节点之间边的集合,边标明的是元素的嵌套包含关系;δ_t 函数决定 E_t 中每条边的走向;Σ_t 是节点名字的集合,即元素类型集合;图中存在惟一一个仅有出边而没有入边的节点,被称为图的根 $root_t$.在这个模型中,从孩子节点通过 δ_t^{-1} 函数可以访问相应的父亲节点.本文在 XML 数据模型表示中使用下标 d,而在 XML 数据模式表示中使用下标 t.图 2 给出示例文档的数据模式图.

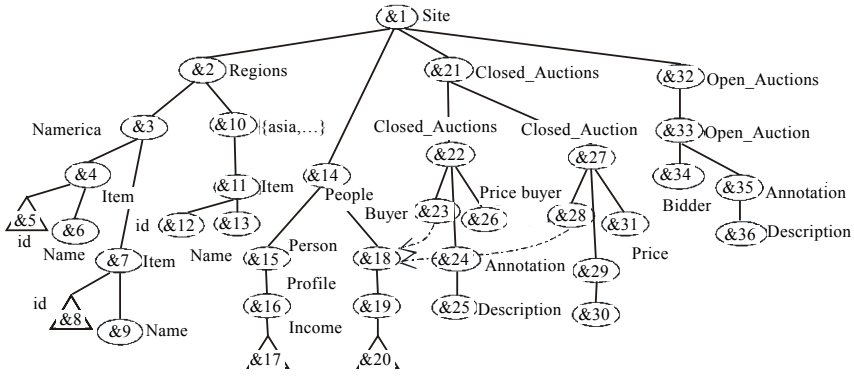

Fig.1 Sample XML document

图 1 一个 XML 文档例子示例

1.2 XML外延

定义 1. 对于一个 XML 文档 $T_d=(V_d,E_d,\delta_d,\Sigma_d,root_d,oid,type_d,oid_d)$,XML 外延 EXT(Etype)是 T_d 中类型为 Etype 的所有元素实例的集合.即 EXT(Etype)=$\{oid_d(e)|\ e\in V_d\wedge type_d(e)\in\Sigma_d\wedge\}.type_d(e)$=Etype.

对于第 1.1 节中介绍的 XML 数据模型,$root_d$ 是文档的惟一入口,因此需要建立适当的索引结构来真正得到 XML 外延.我们采用类似于文献[5]中的 PCX 来索引 XML 外延,为了叙述简便,我们称它为 XML 外延索引(XML extent index,简称 XEX).XEX(Etype)索引的集合就是 EXT(Etype).

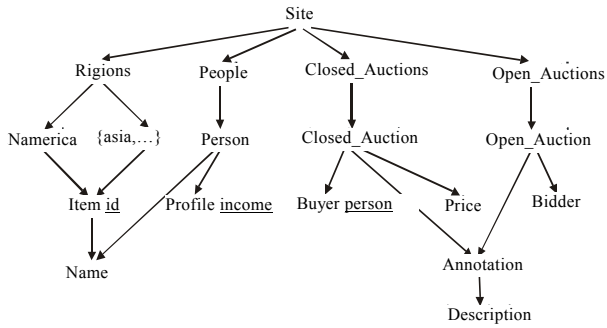

Fig.2 Sample XML schema
图 2 XML 模式图示例

1.3 路径表达式查询

XML 文档可以被抽象成若干路径实例的集合,每一个数据节点都相应地对应于一个路径实例.路径表达式查询用来匹配 XML 文档的路径实例,从而得到符合条件的元素实例.路径查询可以由一个五元组 $Q=\langle G_t,T_d,SE,PE,RS\rangle$来表示,其中 G_t 和 T_d 是查询文档模式及文档数据,PE 是查询路径表达式,SE 是查询的初始元素集,RS 则是查询结果集.由于 SE 是 PE 的起始点,因此 PE 肯定是一个相对路径,则出现在 PE 中的所有路径连接符都只表示元素之间的联系.用户的原始查询形式可以由 $SE+PE$ 得到.

2 路径缩短策略

2.1 路径缩短策略

定理 1. 对于符合 XML 模式 G_t 的 XML 文档 T_d,除了 $root_d$ 所用,其他元素都是它的后代子孙,即$(\forall n)(n\in\{V_d-root_d\}\rightarrow n$ 是 $root_d$ 的后代).

证明略.

定理 2. 如果绝对路径表达式查询 $Q=\langle G_t,T_d,root_t,PE,RS\rangle$是 G_t 中惟一能够访问最后目标的途径,并且我们使用 End(PE)来表示路径 PE 的最后一个元素类型,则 RS=EXT(End(PE)).

证明略.

定义 2. 设 $Q=\langle G_t,T_d,root_t,PE,RS\rangle$是一个路径表达式查询,并且 End($root_t+PE$)=$E_n$,如果满足 $(\forall e)(e\in EXT(E_n)\rightarrow e\in RS)$,则 PE 被定义为元素类型 E_n 的惟一访问路径(unique accessing path),记作 $UAP(E_n)=PE$.

推论 1. 设 P 和 P_2 是两条路径表达式,并有路径表达式查询 $Q_1=\langle G_t,T_d,root_t,P/E/P_2,RS_1\rangle$和 $Q_2=\langle G_t,T_d,E,/P_2,RS_2\rangle$,如果 $UAP(E)=PE+P$,则 $RS_1=RS_2$.

定义 3. 给定 G_t 和 T_d,设 $E_1,E_2\in V_t$,如果 $(\forall e_2\in V_d)(e_2\in EXT(E_2)\rightarrow(\exists e_1)(e_1\in EXT(E_1)\wedge\delta_d^{-1}(e_2)=e_1))$,定义 E_1 是 E_2 的惟一父元素(unique parent),记作 $UP(E_2)=E_1$.

定义 4. 给定 G_t 和 T_d,设 $E_1,E_2\in V_t$,如果 $(\forall e_2\in V_d)(e_2\in EXT(E_2)\rightarrow(\exists e_1)(e_1\in EXT(E_1)\wedge e_1$ is the ancestor of $e_2))$,定义 E_1 是 E_2 的关键祖先(key ancestor),表示访问 E_2 的元素实例必须通过 E_1,记作 $KA(E_2)=E_1$.

定理 3. 如果用 $E_i(1\le i\le n)$ 和 $C_i(1\le i\le n-1)$ 来分别表示 n 步绝对路径表达式 $E_1C_1E_2C_2E_3\ldots C_{n-1}E_n$ 的步和路径连接符,且对于查询 $Q_1=\langle G_t,T_d,E_1,C_1E_2C_2E_3\ldots C_{n-1}E_n,RS_1\rangle$,有 $UAP(E_2)=C_1E_2$,即 Q_1 可以由相对路径查询 $Q_2=\langle G_t,T_d,E_2,C_2E_3\ldots C_{n-1}E_n,RS_2\rangle$替代,如果 E_2 是 E_3 的惟一父元素或者关键祖先(当 C_2 是'/'或'//'时),则 $Q_3=\langle G_t,T_d,E_3,C_3E_4\ldots C_{n-1}E_n,RS_3\rangle$可以代替 Q_1,即 $UAP(E_3)=C_1E_2C_2E_3$.

证明略.

根据定理 3 可知,在一定条件下,Q_3 可以由 Q_2 进行优化得到,从而表明相对路径查询也是可以进行优化的.总结本节的这 3 个定理和一个推论,它们的本质是相同的,亦即根据用户查询的性质,利用 XML 文档模式信息,

把用户的路径表达式查询缩短一部分,从而节省了查询代价.因此,我们把这 3 个定理统一称为路径表达式查询的路径缩短策略.

2.2 路径缩短算法

定理 3 不仅可以用来缩短相对路径表达式,还指出了路径优化策略的执行原则,即对路径表达式查询从起始点开始一步一步地进行路径缩短操作,直到某一步不能优化为止.当路径连接符是'/'的时候,只需判别惟一的父元素就可以了,过程也比较简单;而当路径连接符是'//'时,需要判断关键祖先.下面我们介绍反向路径树(reverse path tree),用以帮助判断关键祖先.

反向路径树 $T_r=(V_r,E_r,\Sigma_r,root_r,E_1,E_2)$被定义为一棵节点上有名字的树,它根据某一 XML 模式 G_t和两个给定的元素类型 E_1和 E_2建成,目的是用来辅助判定 E_1是否是 E_2的关键祖先.我们可以简单地用 $RPT(E_1,E_2)$来表示 $T_r=(V_r,E_r,\Sigma_r,root_r,E_1,E_2)$,它包含了 G_t中从 E_1出发到 E_2结束的所有路径和一部分从 $root_t$到 E_2结束的路径.反向路径树 $RPT(E_1,E_2)$的叶子节点只可能是 E_1和 $root_t$两种.且当 E_1是 E_2的关键祖先时,$RPT(E_1,E_2)$的叶子节点只可能是 E_1.我们可以据此判断一个元素类型是否为另外一个的关键祖先.总结以上两种分别判别惟一父元素和关键祖先的方法,我们得到路径缩短策略的算法,见算法 1.

算法 1. 路径缩短算法.

(1) 对路径 $\sum_{i=1}^{n}C_iE_i$ 的每一步;

(2) 如果 C_i是"/",则检查$\delta_t^{-1}(E_{i+1})=\{E_i\}$是否成立,若成立,则记载当前步可以缩短并继续缩短余下的路径;否则,goto (4).

(3) 如果 C_i是"/",则建立 $RPT(E_i,E_{i+1})$,并检查它所有的叶子是否为 E_i,若是,则记载当前步可以缩短($cs=i$),并继续缩短余下的路径;否则 goto (4).

(4) 优化查询路径为 $\sum_{i=cs+1}^{n}C_iE_i$.

3 补路径策略

路径缩短策略通过优化路径表达式本身来提高查询效率.下面我们将介绍另外一种优化策略,它把那些用户书写的复杂的、高代价的查询路径表达式用一些简单的、低代价的与用户查询等价的路径表达式查询来替代,从而得到比较好的效率.这个优化策略和路径缩短策略一样,依然需要 XML 文档模式信息的知识.

定义 5(互补路径). 对于 XML 模式 $G_t=(V_t,E_t,\delta_t,\Sigma_t,root_t)$和 XML 数据 $T_d=(V_d,E_d,\delta_d,\Sigma_d,root_d,oid,type_d,oid_d)$,如果 $E_2\in V_t,KA(E_2)=E_1$,在 E_1 和 E_2 之间存在着全部 n 条路径为 $P_1,P_2,\ldots,P_n$,且 $\mathrm{End}(P_i)=E_2(1\le i\le n)$,那么 $\bigcup_{j=1}^{i-1}p_j+\bigcup_{j=i+1}^{n}p_j(1\le i\le n)$ 被定义为路径 P_i 相对于元素 E_1 和 E_2 的互补路径(complementary paths).记作 $CP(E_1,E_2,P_i)=\bigcup_{j=1}^{i-1}p_j+\bigcup_{j=i+1}^{n}p_j$.

定理 4(补路径策略). 如果 $CP(E_1,E_2,P_i)=\bigcup_{j=1}^{i-1}p_j+\bigcup_{j=i+1}^{n}p_j$,且路径 P_i 相对应的查询为 $Q_i=\langle G_t,T_d,E_1,P_i,RS_i\rangle$ $(1\le i\le n)$,则有 $(\forall e_2)(e_2\in EXT(E_2)\rightarrow e_2\in\bigcup_{i=1}^{n}RS_i)$,即 $EXT(E_2)=\bigcup_{i=1}^{n}RS_i$,也就是 $RS_i=EXT(E_2)-\bigcup_{j=1}^{i-1}RS_j+\bigcup_{j=i+1}^{n}RS_j$.

证明略.

经过补路径优化策略得到的路径表达式查询也需要使用路径缩短策略来加以进一步的优化.而补路径策略实施的关键问题就是确定查询路径的互补路径,并进行代价估算和查询策略选择.我们采用与文献[9]提出的对象数据库路径表达式查询代价估算方法的改进算法进行 XML 路径表达式查询的代价估算.

算法 2. 补路径算法.

(1) 如果 $KA(E_n)=E_0$,则令 $P'=\mathrm{get}CP(E_0,E_n,P)$,并检查 $Cost(EXT(E_n)-PS(P'))$是否小于 $Cost(PS(P))$,若是,则优

化路径查询为 $EXT(E_n)$–$PS(P')$,算法结束;否则 goto (2);

(2) 对于 $i(1\le i\le n)$;

(3) 如果 $KA(E_i)=E_0$,则令 $tmp_p=\sum_{j=1}^{i}C_jE_j$, P'=get$CP(E_0,E_i,tmp_p)$,并检查 $Cost(EXT(E_n)$–$PS(P'))$是否小于 $Cost(PS(tmp_p))$,若是,则优化路径查询为 S=$(EXT(E_n)$–$PS(P'))$+$\sum_{j=i}^{n}C_jE_j$, 算法结束;否则,重复此步,直至所有路径步已经检查完毕为止.

算法 2 使用反向路径树来判定关键祖先,同时确定互补路径,并使用函数 get$CP(E_1,E_2,P)$表示确定互补路径的过程.该算法接着对得到的各种查询策略进行代价估算,选择其中代价最少的执行策略.注意到,在进行代价估算之前,需要对路径表达式进行路径缩短优化,算法中使用函数 $PS(P)$和 $Cost(P)$分别表示对 P 进行路径缩短和代价估算.另外,该算法从路径最后端开始实施补路径策略,一旦得到比原路径代价小的策略就立刻停止.这样,路径越长,补路径策略的期待效益就越高.

4 性能评价

在这一节中,我们通过一些测试集全面地评价本文提出的路径表达式查询优化策略的性能.测试的硬件平台为一个主频为 933MHZ 具有 384MB 主存的单 CPU PC 机,软件平台为 XBase 系统[5].所有的测试程序使用 MS VC++ 6.0 和 Inada 2.0 编写.所有的测试结果基于两个测试数据集:XMark[8]和 XMach[10].共有 3 种不同的查询方法:外延连接(extent join)、路径缩短(path shorten)和补路径(path complementing).第 1 种是传统的外延连接算法,而后两种分别采用了路径缩短策略和补路径策略来优化某些路径表达式.

4.1 查询优化性能评价

图 5 和图 6 分别显示了路径缩短策略在测试数据集 XMark 与 XMach 上与传统外延连接算法相比的性能收益.我们可以看到,在所有的查询中,路径缩短方法都要比外延连接方法出色.而似乎 XMark 的性能提高并不十分明显,这是因为 XMark 的查询是 XQuery 查询,除了路径表达式之外还包含许多其他操作.但是即使这样,路径缩短方法依然比外延连接方法出色很多.由于查询的特点不同,路径缩短策略带来的性能提高程度也有所不同,造成这种后果的原因也是各种各样的:(1) 有些查询的性能可以被极大地提高(约 10~200 倍),例如 XMark 的 Q5,Q6,Q7, Q18 和 Q20 以及 XMach 中的 Q2 到 Q7,它们可以被缩短很多;(2) 一些查询的性能仅能得到很小程度的提高(约 0.3%~8%),例如 XMark 中的 Q13 到 Q16 以及 XMach 中的 Q1.其中一些查询包含其他需要花费非常大的代价的操作符,另外一些查询本身非常长却只能被缩短很少一部分,还有一些几乎难以进行缩短操作;(3) 其他查询介于上述两种之间,性能提高约 10%~400%,例子有 XMark 的 Q1 到 Q4,Q8 到 Q12 和 Q17 以及 XMach 的 Q8.它们只能被缩短很少因而节省的连接不太多(Q1),或者查询需要其他高代价操作,以至于查询优化的性能不能清楚地显现(Q2 到 Q4 和 Q8 到 Q12).此外,因为测试数据集的查询都是绝对路径查询,根据定理 1,它们的查询至少能被缩短一步.

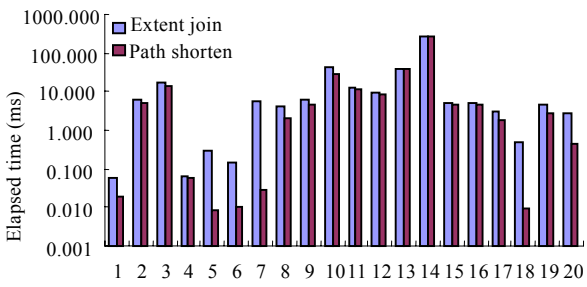

Fig.5 Performance of path shorten (Xmark)

图 5 路径缩短策略性能(XMark)

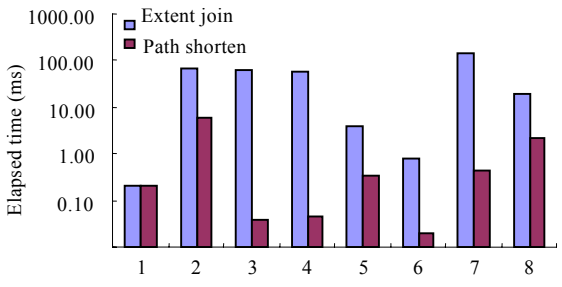

Fig.6 Performance of path shorten (Xmach)

图 6 路径缩短策略性能(XMach)

由于补路径策略需要进行代价估算,它在路径表达式查询中是否适用不仅取决于查询路径自身和 XML 模式,而且还与 XML 文档的统计信息相关.因此不是所有的查询都可以用它们等价的互补路径进行计算.本文基

于测试标准 Xmark,选择 3 条路径进行补路径策略的性能测试,如图 7 所示.由分析结果可知,对于某些路径表达式查询,它可以在路径缩短策略的基础上再将查询性能提高 20%~200%.

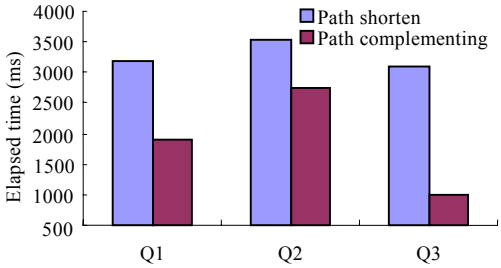

Fig.7 Performance of path complementing

图 7 补路径策略性能

5 结 论

路径表达式是现有 XML 查询语言的共同组成部分,目前针对基于路径表达式进行 XML 数据查询的研究工作基本上都集中在新的查询方法、新的操作符重写策略以及新的索引结构上.本文针对路径表达式本身讨论了 XML 查询的优化问题,提出了两个路径表达式优化策略:路径缩短策略和补路径策略.路径缩短策略根据 XML 模式信息,将路径表达式查询缩短,从而减少需要的查询代价;而补路径策略则是使用代价更小的等价路径表达式来替换原始查询.同时,本文还基于目前得到广泛关注的外延连接算法给出了相关的数据结构和算法.基于模拟数据和实际数据的实验结果都表明,这两个优化技术是现实可行的,可以使查询性能得到显著提升,其中 80%以上的查询平均效率提升 20%~400%或者更多.

References:

[1] Zhou A, Lu H, Zheng S, Liang Y, Zhang L, Ji W, Tian Z. VXMLR: A visual XML-relational database system. In: Franklin MJ, Moon B, Ailamaki A, eds. Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Madison: ACM, 2002. 719~720.

[2] McHugh J, Widom J. Query optimization for XML. In: Atkinson MP, Orlowska ME, Valduriez P, Zdonik SB, Brodie ML, eds. Proceedings of the 25th International Conference on Very Large Data Bases. Edinburgh: Morgan Kaufmann Publishers, 1999. 315~326.

[3] Li Q, Moon B. Indexing and querying XML data for regular path expressions. In: Apers PMG, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass RT, eds. Proceedings of the 27th International Conference on Very Large Data Bases. Roma: Morgan Kaufmann Publishers, 2001. 361~370.

[4] Chung C, Min J, Shim K. APEX: An adaptive path index for XML data. In: Franklin MJ, Moon B, Ailamaki A, eds. Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Madison: ACM, 2002. 121~132.

[5] Lu H, Wang G, Yu G, Bao Y, Lü J, Yu Y. Xbase: Making your gigabyte disk queriable. In: Franklin MJ, Moon B, Ailamaki A, eds. Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison: ACM, 2002. 630.

[6] Lü J, Wang G, Yu JX, Yu G, Lu H, Sun B. Performance evaluation of a DOM-based XML database: Storage, indexing and query optimization. In: Meng XF, Su JW, Wang YJ, eds. Advances in Web-Age Information Management, the 3rd International Conference, WAIM 2002. Lecture Notes in Computer Science 2419, Springer-Verlag, 2002. 13~24.

[7] Gardarin G, Gruser JR, Tang Z. Cost-Based selection of path expression processing algorithms in object-oriented databases. In: Vijayaraman TM, Buchmann AP, Mohan C, Sarda NL, eds. Proceedings of the 22th International Conference on Very Large Data Bases. Mumbai: Morgan Kaufmann Publishers, 1996. 390~401.

[8] Schmidt A, Waas F, Kersten ML, Carey MJ, Manolescu I, Busse R. XMark: A benchmark for XML data management. In: Bernstein PA, Loannidis YE, Ramakrishnan R, Papadias D, eds. Proceedings of the 28th International Conference on Very Large Data Bases. Hong Kong: Morgan Kaufmann Publishers, 2002. 974~985.

[9] Wang GR, Yu G, Zhang B. Selectivity estimation in object-oriented databases. Chinese Journal of Computers, 1998,21 (supplement):171~177 (in Chinese with English abstract).

[10] Bohme T, Rahm E. Multi-User evaluation of XML data management systems with XMach-1. In: Proceedings of the 1st VLDB Workshop on Efficiency and Effectiveness of XML Tools, and Techniques. Hong Kong, 2002. 148~159.

附中文参考文献:

[9] 王国仁,于戈,张斌.面向数据库系统中谓词选择的估算.计算机学报,1998,21(增刊):171~177.