ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.10, October 2006, pp.2069−2086
DOI: 10.1360/jos172069

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

XML 查询优化研究*

孟小峰[1+], 王 宇[2], 王小锋[1]

[1](中国人民大学 信息学院,北京 100872)
[2](河北大学 计算中心,保定 071002)

Research on XML Query Optimization

MENG Xiao-Feng[1+], WANG Yu[2], WANG Xiao-Feng[1]

[1](Information School, Renmin University of China, Beijing 100872, China)

[2](Computer Center, Hebei University, Baoding 071002, China)

+ Corresponding author: Phn: +86-10-62519453, E-mail: xfmeng@ruc.edu.cn, http://www.ruc.edu.cn

Meng XF, Wang Y, Wang XF. Research on XML query optimization. *Journal of Software*, 2006,17(10): 2069−2086. http://www.jos.org.cn/1000-9825/17/2069.htm

Abstract: XML has become the de-facto standard for data representation and exchange on the World-Wide Web. Due to the nature of information on the Web and the inherent flexibility of XML, it is expected that much of the data encoded in XML will be semi-structured. Data on the internet is increasingly presented in XML format which enables researches on various kinds of XML storage model. Meanwhile, XML query optimization has become a hot research topic in database field. This paper gives an overview of the current status of technology for XML query optimization. The features of XML query optimization and key problems of research are also discussed deeply. Main aspects of current work on XML query optimization include XML algebra, cost model, complex path selectivity estimation, statistics information, and so on. Finally, this paper prospects future research directions and presents some viewpoints of XML query optimization.

Key words: XML; query optimization

摘 要: XML 已经成为网络上信息描述和信息交换的标准.由于网络上信息的本质特性和 XML 数据内在的灵活性,很多用 XML 编码的数据都是半结构化的.随着 XML 应用得越来越广泛,人们提出了多种 XML 数据的存储模型.与此同时,XML 的查询优化也是数据库领域研究的一个重要课题.综合论述了 XML 数据查询优化技术的现状,指出了 XML 查询优化的特点和研究的关键性问题.描述了查询优化技术各个方面的重要研究成果以及存在的问题,进一步展望了未来的研究方向,并在此基础上提出了对 XML 查询优化方法的一些观点.

关键词: XML;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.60073014, 60273018 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2003CB317000 (国家重点基础研究发展规划(973)); the Key Project of Chinese Ministry of Education under Grant No.03044 (国家教育部科学技术重点项目); the Program for New Century Excellent Talents in University (国家教育部新世纪优秀人才支持计划)

Received 2006-01-19; Accepted 2006-04-17

XML 已经成为网络上信息描述和信息交换的标准.早期的 XML 数据以文档方式存储,以关键字查询等信息检索手段查询,简单、易用.由于缺乏系统的存储和查询机制的支持,造成查询能力低,不能满足复杂条件的查询,更谈不上查询优化.一些现有的商业数据库系统扩充了处理 XML 数据的功能,利用现有数据库成熟的技术,把 XML 查询要求转变为数据库查询表达,由查询优化器优化查询表达并执行,再将查询的结果转变为 XML 数据.这种方法在一定程度上解决了查询复杂性的要求,但多级转换带来的问题是效率的降低和查询语义的混淆.

与传统的数据库数据相比,XML 数据具有如下特点:

- 数据是自描述的,内容与结构混杂在一起;
- 数据具有完整的嵌套层次;
- 数据是有序的.

XML 数据的不规则性是对传统统计信息方法的重要挑战,其数据分布情况使得一些传统的分布假设难以成立.为了达到所需的代价估计精度,需要更多的统计信息.而结构的复杂性又为获得相对精确的统计信息带来存储和计算上的困难.XML 的有序性制约了转换规则的灵活性.XML 数据的上述问题无论是对关系数据库或是对面向对象数据库的现有查询优化技术都是严峻的挑战.

与传统的查询需求相比,XML 查询具有如下特点:

- 以长路径表达式为查询的核心语句,路径复杂,包含分支路径;
- 嵌套的查询表达,查询表达式中加入编程语言的嵌套和条件判断思想;
- 路径中包含不确定因素,这在之前的查询需求中未出现过;
- 查询对象和返回结果类型不确定.

面向对象数据库已有一些处理复杂长路径表达式的经验,但无法处理 XML 查询中的路径表达式中的不确定情况;关系数据库中已有很多处理嵌套查询的方法,但对掺杂编程语言风格的 XML 查询语言却难以适应.

综上所述,来自数据结构和查询需求两方面的问题导致基于关系和面向对象数据库的查询处理和查询优化技术均不能适应 XML 查询的需要.目前,对 XML 查询优化的研究正成为热点.本文的内容就是对 XML 查询优化技术现状的综合论述,指出了 XML 查询优化的特点和研究的关键性问题,描述了查询优化技术各个方面的重要研究成果和有待进一步解决的问题.

1 XML 查询优化研究问题

查询优化是数据库技术中重要的研究问题,是实现高效查询的关键性因素.对传统数据库查询优化的研究已经形成相对成熟的技术和方法,其中基于代价的优化是主流.查询语言首先被转换成为一种内部表达形式(通常是某种代数,如关系代数等),根据变换规则得到等价表达式,计算不同形式的表达式的执行代价,然后选择一个最小的执行方案.当把这种方法用于 XML 查询优化时,研究者遇到如下问题:

(1) 完善的查询代数标准

众所周知,关系数据库统治数据管理领域长盛不衰的法宝就是描述性查询语言 SQL 及其运行基础关系代数.关系代数的目的之一是给出明确的查询语义,之二是用于支持查询优化.关系代数的优势来自于简单、明确的数据模型——关系,具有完善的数学基础和系统的转换规则.后来的数据模型都以关系代数为蓝本,定义了不同的运算,如面向对象数据模型等,但效果并不尽如人意.XML 数据模型本身具有的半结构化特点是定义完善的代数运算的最大障碍.而 XML 查询语言中的不确定性和一些编程思想的引入是另一个难以克服的困难.

(2) 精确的代价估计

关系模型中,表中的记录是无序的、大小相等的,代价计算时依据的一些分布假设是稳定的.而且,由于其记录大小相等,对时间的估计可以转换为对 I/O 次数的估计,进而转换为对中间结果大小的估计.而在 XML 模型中,数据是有序的,数据聚集的方式不定,每个数据的大小相差悬殊,中间结果大小与 I/O 次数之间的对应关系没有明显的规律.简单地沿用传统的代价计算方法必然导致误差的产生,从而影响精确的代价估计.

(3) 足够的统计信息

足够精确的统计信息是保证查询优化有效性的基础;缺乏足够的统计信息,是造成估计与实际情况产生误差的重要因素.传统的统计信息多是对值的统计,如对平均值、最值、记录个数等的统计.这些对 XML 查询是不够的.XML 数据本身缺乏模式的支持,对数据结构信息的统计显得更加重要.XML 数据中的数值分布在类似树状结构的树叶上,即使相同类型的数据,由于半结构化特点,其分布情况也可能完全不同.因此,需要把对结构的统计信息和对值的统计信息结合到一起,才能得到足够精确的统计信息.

2 XML 查询处理结构

与传统的关系数据库系统的查询处理结构类似,我们可以将 XML 查询处理分为 4 个大的阶段.如图 1 所示.中间方框表示查询处理步骤;左侧方框为使用的相关技术或方法;右侧方框为查询优化和执行时需要的信息.

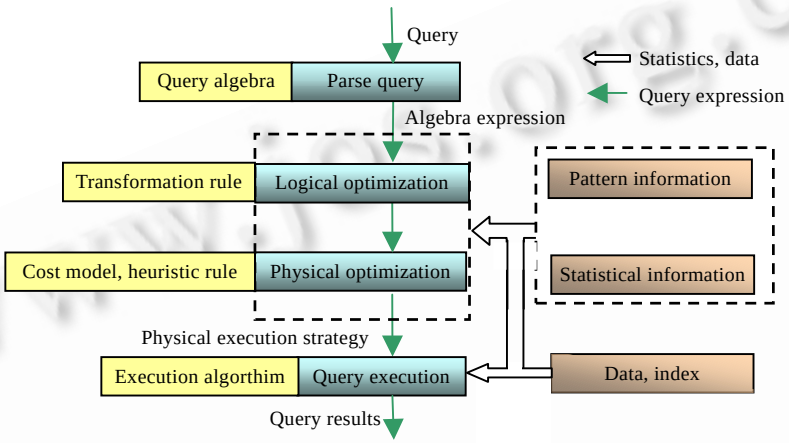


Fig.1 Query processing

图 1 查询处理过程

第 1 阶段查询解析,将查询转换为某种内部表达方式,以便于机器处理,并为下一步的优化过程铺平道路.这种内部表达方式通常以一种抽象语法树或者查询树的形式出现.以传统数据库为基础的查询引擎的做法是转换成关系代数,然后由关系数据库优化器完成剩下的优化工作.而 Native 的数据库系统则采用不同的 XML 代数系统.我们在第 3 节中将会介绍目前流行的几种 XML 代数.

第 2 阶段逻辑优化,利用模式信息,规范和简化内部表达式.在这一阶段中,系统不考虑实际数据的值和数据的存储情况.同一查询请求,可以转换成不同的等价表达方式,其中有一些比原有查询更高效.为了进行这种转换,优化器需要一些转换规则,我们将在第 4 节中讨论这些转换规则.

第 3 阶段物理优化,利用代价模型和统计信息计算不同表达式、不同算法的执行代价,选择最低代价的查询计划.在这一阶段中,需要解决两个问题:确定表达式执行顺序和决定每步操作的具体算法.对于 XML 查询树而言,首先需要将查询表达式分解为可执行的片断,然后选择合适的执行顺序和执行算法并执行.中间结果集的大小是决定执行策略是否高效的关键因素,与实际的数据分布密切相关.综合考虑数据的存储、索引和数据值的分布情况,准确地估计复杂路径选择性是其中的难点.对于一个给定的执行策略,通常会有多个可能的执行算法,产生所有的执行算法的组合造成选择本身的代价过大.因此,会有一些启发式规则用来控制其空间规模,并采用一些空间搜索技术加速选择的过程.我们将在第 5 节中详细加以讨论.

第 4 阶段查询执行,根据物理优化确定的执行策略和算法,访问数据并得到查询结果.由于 XML 数据复杂和变化的结构,需要高效的数据访问算法.

3 XML 代数

XML 代数是对遵循一定数据模型的 XML 文档集合的操作集.XML 代数提供根据请求在文档集合中选择

一个或多个文档或者文档片段的能力.XML 代数应支持对查询结果的重构.

目前对 XML 代数的研究主要集中在对查询代数的定义和从查询语言到查询代数的转换方面.查询代数定义查询对象的类型,可以执行的操作和不同操作之间的转换规则.查询语言经分解转换为由查询代数的操作表达构成的操作树或者操作序列.不同的代数表达可以有相同的语义和执行结果,构成代价空间.

3.1 XML代数定义

目前产生很多种 XML 代数,风格各异,其主要思想来源于关系代数、面向对象代数、半结构化代数和功能化编程语言等.由于篇幅有限,不能在这里一一介绍,我们介绍其中具有代表性和影响力的几种.表 1 列出了不同代数之间特点的比较.

Table 1 Comparisons among XML query algebra

表 **1** XML 查询代数比较

	Data structure	Document order	Node order	Reference supported	Logical operation	Physical operation	Transformation rule
AT&T	Directed graph	×	√	√	√	×	×
IBM	Directed graph	√	√	√	√	×	×
FS	Xquery data model	√	√	√	√	Seldom	√
Lore	OEM	×	√	√	√	√	√
TAX	Tree	×	√	×	×	√	×

Oracle,IBM 和 MS 联合提出的一个 XML 代数标准是文献[1].该标准把 XML 文档看作有向标记图(如果忽略引用,可以看作有向标记树).用五元组 $G(V,E,A,R,O)$表示.其中:V 表示结点,有两种类型:element 和 value.E 表示 element 到 element;A 表示 element 到 value,即属性;R 表示引用.上述 3 种为边的类型.O 表示次序.在这个模型上,规定了导航、选择、连接、构造等操作,其导航操作提供在有向标记图中的遍历操作,包括正向遍历和反向遍历;其连接操作语义类似关系代数中的连接操作,根据相同的值连接不同的文档.该代数采用类似关系代数的表达形式.

Bell Labs.和 AT&T Labs.的 Mary 等人提出的 XML 查询代数[2],基于简化的 XSLT 数据模型[3],增加了引用结点,合并了属性和元素结点,删除了注释结点.其主要思想来源于曾用于半结构化和面向对象数据库的代数——嵌套关系代数,并增加了对正则表达式的操作.在嵌套关系中,数据由多个元组和多个链表组成,并可多级嵌套,其采用 list comprehension 方法表达导航、笛卡尔积、嵌套和连接操作.List comprehension 根据一系列过滤和 generator 操作,得到满足条件的结果链表.Generator 操作与导航操作相对应.该代数还支持结构递归等程序语言特点,用 Haskell 程序语言作为表达形式.

上述两种代数还仅停留在逻辑层次,没有考虑与之相对应的物理代数和查询优化策略,其优势是具有较高的描述性和丰富的语义,与查询语言有密切的转换关系;但其操作中对路径的处理并不完善,形成过多的递归结构[3]或者遍历操作[2],给下一步的优化带来困难,但其处理 XML 查询的方法和思路被后来的 XML 代数规则大量采用.

基于文献[2,3]等,W3C 于 2001 年公布了一个 XML 查询代数标准 XQuery 1.0 Formal Semantics[4],用于规范查询语言语义.该标准遵循简化的 XQuery 1.0 和 Xpath 2.0 Data Model[5].比照关系代数给出了 XML 数据模型的投影、选择和连接等操作的定义,还引入结构递归、条件判断等编程语言的概念.Formal Semantics 的一个特点是代数表达与 XQuery 查询语言相同,成为 XQuery 的核心语法;另一个特点是操作有不同的层次,高层次的操作可以转换为低层次的操作.标准中还给出了少量表达式转换规则,有待近一步扩充.目前已经有了应用该标准的 XML 查询引擎,如 Galax[6].

Standford 大学开发的 XML 数据库 Lore 系统[7]针对系统本身提供的访问操作和索引情况,提出了一套独特的代数操作,包括逻辑代数、物理代数和相应的转换规则.Lore 以该代数系统为基础提出了查询优化方法[8],但由于代数操作定义过多地依赖于 Lore 本身独特的数据存储和索引技术,该代数很难应用于其他系统.

Timber 数据库[9]中应用的 TAX 代数[10],其数据模型为无序的树的集合,树中的数据是有序的.直接针对树和

树枝规定了一系列操作无须中间结构的转换.把 XML 数据模式看作树,把查询语句也看作树,二者之间作模式匹配,得到满足查询树条件的结果树集合.TAX 在处理连接操作时对操作的顺序未做明确规定,不适应严格要求文档顺序的情况.

XAL[11]基于集合概念构造了逻辑代数操作集,其操作分为 3 种:抽取操作从 XML 文档中获得必要的数据,如选择、投影等;元操作控制表达式求值过程,并非针对 XML 数据的抽取或者构造操作,而是为其他操作符准备输入或者控制其他操作的操作,如映射、迭代等;构造操作用于构造查询结果.XAL 为查询优化提供了一组启发式转换规则.

其他如 XOM 代数[12],是完整的操作集,包含 6 种对象操作,但不支持优化;OPAL 代数[13]基于半结构化数据模型,操作对象为多个链表,将有限状态自动机用于生成执行计划;还有 SAL 等[14−17].根据代数的操作方式,可将上述不同的方法分为两种:一种是面向集合的代数,其操作对象是某种类型的集合,如树的集合、值的集合等.这种方法具有很好的优化基础,但可能丢失数据的顺序;另一种称为导航的代数,其操作对象是单个的数据.这种方法不利于进一步的优化.代数定义应是逻辑操作与物理操作的有机结合.但目前的 XML 代数研究或是把他们混合在一起,或是虽然分开但缺乏相应的转换规则.

早期对 XML 代数的研究工作重点在于规范 XML 查询语义,并未考虑查询优化因素,这些代数具有明显的程序化思想,很难进一步优化,只能利用遍历方法求解查询,造成查询效率的低下,不适应大规模 XML 数据的查询需求.而基于数据库思想提出的一些面向“集合”的代数,具有很好的优化基础.因此,目前查询优化的研究工作也多以这些代数为背景,但也存在一些问题.

首先是表达式嵌套问题.在 XQuery 查询中,由于表达式可以任意嵌套,谓词可以出现在任意地方.谓词是有作用域的,同一个谓词在不同的地方会产生不同的查询结果.基于集合的代数需要将嵌套的查询转换为逻辑树形式,不可避免地面临嵌套结构的非嵌套化问题.虽然在关系数据库中有一些方法可以借鉴,但由于 XML 数据结构的复杂性,解决这个问题变得更加困难.

其次是 XML 数据的有序性问题.除非查询语句特别指定,否则应该保持结点在源文档中的结点顺序.而原来一些处理连接的算法,比如排序连接、Hash 连接等,会打乱结点顺序,在连接完成后,需要对连接结果做一个额外的根据结点顺序的排序操作.如果用 nest-loop 连接算法,则可以省去这趟额外的排序.在进行代价估计的过程中,这个额外的排序操作要被考虑在内,这就给进行查询优化的过程带来新的考虑因素.一种可能的解决方法是在所有操作中,忽略结点的有序性,在最后构造结果的时候再对结点按照文档顺序排序.究竟是使用 nest-loop,还是最后增加额外的排序的方法,这是查询优化的一个研究点.

最后,不同的代数标准.在 XML 代数研究中一个值得重视的问题是:目前已经出现了一些查询代数标准,这些标准在风格上相去甚远,很难有共通性.而对执行方法的研究还远远不够,不能形成完整的系统.而且从逻辑代数到物理代数的转换也将是未来研究的一个重要问题.

3.2 复杂路径表达式分解

目前的 XML 查询语言有很多,如 XPath[18],XQuery[19],XML-QL[20],XQL[21],Quilt[22]等.它们的一个共同的特点就是对复杂路径表达式的支持.路径表达式分解是根据一定的转换规则,把用查询语句表达的、复杂的、不确定的路径表达式转换为简单的、明确的、系统可识别的方式,如 XML 代数.路径表达式分解是查询转换的难点,也是查询优化的重要一步,是代价估计的前提条件.根据不同的规则,路径表达式可能分解为不同的等价形式,其中有的代价高,有的代价低,形成代价空间.路径分解的原则是能够产生有限的代价空间,有利于利用分解的结果搜索代价最小的执行方案.

在路径分解时,有两种不同的思路:一种思路是把路径细分为两两的祖先后代或父子对,如 lore,XISS[23]等.这样做分解算法简单,可利用数据物理存储信息,分解的结果容易转换为结构连接运算或者运用系统提供的各种索引.如果查询语句中出现通配符(如*,?,//),可以利用索引直接定位数据,也可以借助模式信息确定通配符所代表的各种可能情况扩展路径.经过分解,路径表达式转换为连接、投影、选择、导航等不同的代数运算;另一种思路[24]是将复杂路径表达式用树的方式表示:从根开始,在树中搜索最长的确定路径(不含*或//)称为一次分

解,其路径构成树的一个子串.以这个点为起点,用上述原则再分解路径,得到确定路径(子串)的集合,称为一个最小分解.在 XML 文档的查询中,确定路径的查询是相对容易的;而不确定路径的查询是比较困难的,尤其是在没有模式或索引的情况下,可能要将中间结果合并才能得到全部的结果.将复杂的、不确定的路径分解为确定的、简单的路径处理.这种分解方法在没有模式信息的情况下处理不确定路径具有一定的优势.

4 逻辑优化

在传统数据库技术中,逻辑优化是指通过一系列转换规则,将原始的查询表达式转换为等价且更高效的形式.关系代数表达式求解时,操作顺序是影响效率的关键.因此,逻辑优化研究的重点并不在于对冗余操作的分析,而是对操作顺序的调整.而 XML 代数的核心是由路径表达式转换的查询树,查询效率依赖于查询树的规模.因此,查询树的最小化是 XML 逻辑优化研究的重点,也是目前研究的热点问题.而对操作顺序的调整因为更多地依赖于物理存储的情况,因此与物理优化相关联.

从层次上看,逻辑优化可分为两个层次:语法层次和语义层次.语法层次的优化是指不依靠任何其他信息,独立地分析查询表达式中分支或结点间的逻辑包含关系,删除冗余部分;语义层次的优化是指通过数据库提供的模式信息,如 DTD,XML Schema 等,或者语义包含、结构包含等完整性约束,查找查询表达式中的冗余分支或结点.下面的例子进一步说明二者之间的区别.若有如下的 XQuery 查询表达式:

for $*a* in reference/book,
  for $*b* in $*a*/author, $*c* in $*a*/author/ name, $*d* in $*a*/author/email
    where $*b* and $*c* and $*d*
return ⟨book⟩$*a*⟨book⟩

其 Pattern Tree(以下简称 PTQ)如图 2(a)所示,其中,实心圆为查询返回结点.若数据满足$*c*,同时必然满足$*b*,$*b* 分支对查询返回结果没有任何影响,是冗余结点.我们称这种冗余结点为语法冗余结点.经语法优化后的 PTQ 如图 2(b)所示.若从模式可知任一 author 均有 name,则 v_6 为冗余结点.我们称这种冗余结点为语义冗余结点.经语义优化后的 PTQ 如图 2(c)所示.

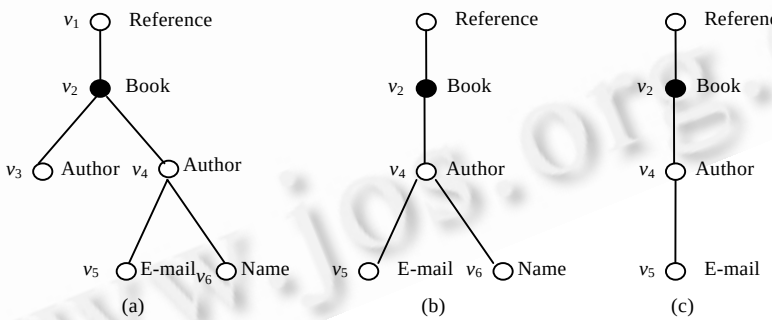


Fig.2 Syntax and semantic optimization

图 2 语法优化和语义优化

下面我们分别介绍语法优化和语义优化的研究现状.

4.1 语法优化

对 PTQ 语法优化问题基于对路径等价性问题[25−28]的研究.最早提出 XPath 最小化的 Wood[29]指出:一个 Xpath 可以表示为合取范式,对 XPath 等价性检查的复杂度等价于对合取范式的等价性检查.而这已经证明是一个 NP 完全问题.如果对 XPath 路径表达式的复杂程度加以一定的限制,XPath 最小化问题的复杂度可以达到多项式级.目前已经提出了对不同类型的 PTQ 的最小化方法.

(1) 简单 PTQ{/,[],*}

文献[29]中提出了对只包含{/,[],*}的简单路径的最小化方法.其思想为:设原始 PTQ 为 *P*,在所有与之等价

的 PTQ 中找到 Q,使得 Q 中的结点个数$|NQ|$最小,则 Q 为 P 的最小化 PTQ.

这种方法的关键是通过对包含映射(containment mapping)的判断完成 PTQ 的等价性判断.不存在祖先后代关系(//)的简单 PTQ 的最小化 PTQ 为其子树.查找等价 PTQ 的范围限制在其子树的范围内,是保证最小化算法的复杂度为多项式级的关键因素.文献[22]中给出了复杂性证明,虽然并未给出算法细节.

(2) PTQ{/,//,[]}

文献[30]提出了 PTQ 的 CIM(constraint independent minimization)算法.与文献[29]相同,其算法的基础也是包含映射.路径中缺少"*"的 PTQ 的最小化问题,仍旧可以在其子树范围内解决.CIM 算法的思想为:从叶结点开始,判断结点在 PTQ 中是否冗余:若某结点是冗余结点,则删除这个结点,继续处理其他叶结点直到所有叶结点判断完毕;若 PTQ 中结点个数为 n,则 CIM 算法的最大复杂度为 $O(n^4)$.

文献[31]提出了一种最大复杂度为 $O(n^2)$的改进算法,通过结点间的二元关系 Simulation 判断冗余性.改进的 CIM 算法与 CIM 算法最大的不同点在于:前者只关心后代结点之间是否相同;而后者还要关心祖先结点之间是否相同.改进的 CIM 算法通过正向遍历查找冗余结点,如果某结点的一个儿子结点与另外儿子结点之间有 Simulation 关系,则这个儿子结点为冗余结点.这样,在一次遍历可以对多个结点的冗余性进行判断,从而提高了 PTQ 最小化的效率.

(3) PTQ{/,//,[],*}

与限制条件下 PTQ 所不同的是,普遍意义下的 PTQ 在语法优化时遇到的一个关键问题是:最小化 PTQ 并非原始 PTQ 的子树,而是由以原始 PTQ 的根为根,连接多个 PTQ 子树构成的 PTQ.其中每个子树均为原始子树的最小化部分.这个问题也是导致其复杂度上升的关键.

文献[32]给出普遍意义下的 PTQ 最小化算法.其算法思想是:递归地在原始 PTQ 的子树中查找最小子树并连接它们,在这个过程中冗余的分支被删除了.文献[32]证明其算法的复杂度为 NP 完全,并指出:在对 PTQ 的分支个数加以一定限制的情况下,改进的算法复杂度可以达到多项式级.但我们有理由相信,这样的改进意义并不大,因为这要求用户在写查询语句时必须注意查询的分支情况,否则将导致某些查询无法优化.

目前,语法优化都以判断结点之间的包含映射关系为基础,分析路径等价性.在查询树中不断地修剪冗余的分支或结点,达到减少查询树规模的目的.普遍意义的 PTQ 语法优化是一个 NP 完全问题,研究者通过对 PTQ 复杂度加以一定限制,提出了多种高效的算法.语法优化不涉及 XML 模式信息,可以利用模式信息进一步简化 PTQ,这种优化称为语义优化.

4.2 语义优化

最早提出语义优化的是关系数据库系统,利用表格属性值之间的约束关系把查询表达式转换为等价但更高效的形式.chase 方法[33]是其中的代表.其思想为:把完整性约束作为冗余条件插入到查询表达式中,与已存在的冗余操作合并,使得组合后的条件符合某些事先定义好的等价转换规则,利用这些等价转换规则重写查询表达式以达到优化的目的.这是一个巧妙的先膨胀再收缩的方法.

但是,应用 Chase 方法于 XML 查询的语义优化时面临下述严重的挑战:

数据结构的变化:与平面表结构不同,XML 数据具有嵌套性.原有的主键、外键等完整性约束不能表达结构上的嵌套关系,缺乏匹配的转换规则.

数据类型的变化:关系模型中,数据有严格的类型;而在 XML 半结构化模型中,数据没有严格的类型约束,同名的结点可以出现在不同的位置,可以有不同的子结点.传统的转换规则的应用方法不适应这种情况,引发的问题就是产生递归的转换,导致路径的无限增长.

查询语句的复杂性:SQL 语句清晰、明确,关系代数操作均为输入参数明确的一元或二元运算.而 XML 查询语句中包含//,*等不确定因素,并以包含多个分支长路径为特点,原有的转换规则不适应于 XML 语义优化.

基于上述挑战,一些研究者提出改进的 Chase 方法,而另一些研究者则从对图分析处理的角度出发,研究 XML 查询的语义优化问题.

(1) 改进的 Chase 方法

Wood 等人[29]最早将 Chase 方法引入简单 XPath 语义优化.文献[29]在 DTD 上定义了 3 种结构约束关系,分别为儿子约束、父亲约束和兄弟约束.若某个查询树中结点 n 为上述约束关系中的主体,且其约束的结点不在查询树中,则在查询树中相应位置加入客体结点.当所有约束关系应用完毕,再用语法优化的方法对查询树进行修剪,得到最小化查询树.为了讨论简单起见,文献[29]中方法只适用于不包含“*”和“//”的简单路径,其复杂度为多项式级.

文献[30]则认为,XML 数据中的结构完整性约束可用儿子约束、后代约束和类型约束概括.为了得到正确的优化结果,他们对 Chase 方法做了 3 个方面的改进:首先,假设约束集合是闭包的;其次,为了保证优化能够完成,约束条件只应用于 PTQ 中原有结点,如果某结点是由于应用某约束条件加入 PTQ 的,则不对其应用任何约束条件;最后,由于应用约束条件加入的结点是冗余的,因此,需要在算法结束时删除这些临时结点.ACIM 算法分为 3 个步骤:首先,应用约束集合中的约束条件放大 PTQ;然后,应用 CIM 算法语法删除冗余结点,在删除时保证不检查被加入临时结点的冗余性;最后,删除所有在第一步中加入的临时结点.若 PTQ 中结点个数为 n,则 ACIM 算法的最坏计算复杂度为 $O(n^6)$.一些冗余结点是容易识别的,如果提前删除这些容易识别的冗余结点然后再应用 ACIM 算法,可以有效地提高优化的效率.算法 CDM 在 PTQ 中遍历地查找并删除这样的冗余结点,其计算复杂度为 $O(n^2)$.实验证明,在使用 ACIM 之前使用 CDM,比直接应用 ACIM 可更为有效地节省时间.

文献[31]在文献[30]的基础上扩充了子类约束,并利用语法优化中的 TPQSimulation 和 TPQMinimization 改进了 ACIM 算法,使计算复杂度达到 $O(n^4)$.

(2) 基于 DTD 的路径等价类方法

文献[34]提出了一种基于模式的 XPath 路径表达式的语义优化方法.其思想是:把 XML 文档模式(DTD)划分为若干个路径等价类,每个类中的路径等价;将 XPath 转换为由简单路径构成的合取范式形式,利用路径等价类中的最短路径代替表达式中的路径.通过这种方法,可以实现 3 个方面的优化:首先,删除冗余的谓词条件;其次简化路径;最后,判断表达式的条件是否满足.如果某个分支的条件不满足模式中的约束关系,则整个表达式的查询结果为空.整个优化过程分为 4 部分:分解、扩展、优化和重构.在分解过程中,XPath 表达式被转换为合取树(XCT);重构 XPath 表达式则将优化的 XCT 转换为 XPath 路径表达式.

改进的 Chase 方法的优点是:语法优化与语义优化相结合,优化过程无须对 PTQ 进行转换.问题是难以保证彻底的优化:首先,PTQ 中存在非叶冗余结点,而语法优化只能在删除叶结点后,让非叶结点变为叶结点的情况下才能判断其冗余性.加入约束条件后的 PTQ 语法优化,不能在不删除叶结点的情况下,判断非叶结点的冗余性;其次,膨胀后的 PTQ 难以压缩,采用对转换规则加以一定限制的方法限制膨胀后 PTQ 规模的方法,会导致优化的不彻底.目前改进的 Chase 方法都是针对一定复杂程度的 PTQ 的优化策略,普遍意义上的 PTQ 的语义优化研究还需深入;最后,从 DTD 中获得的约束条件并不充分,这也是导致优化不彻底的一个因素.如何抽取更多的语义约束条件,是未来研究的一个重要问题.

基于 DTD 的优化方法的优点是:不但能够删除冗余分支,还能够缩短路径长度和直接判断路径是否满足.问题主要是:首先,需要对 PTQ 进行转换,占用大量优化时间;其次,需要不确定路径的确定化,这实际上也是一种路径膨胀,难以保证优化的结果小于优化之前.

两种方法都需要首先扩大路径规模,造成优化的不彻底和效率的丧失,这是目前语义优化面临的一个重要问题.

5 物理优化

逻辑优化的结果是一个或多个查询树.如何确定其中不同查询片断的执行次序,是 XML 物理优化的核心问题.确定执行次序的主要因素是中间结果集的大小.复杂路径表达式选择性分析就是用来估计结果集规模的.其主导思想是:统计 XML 数据的分布情况,基于一定假设估计路径目标结点中间结果集的大小.这种方法一般忽略执行算法的不同和数据的物理存储.本节从统计信息抽取、存储、压缩、维护和统计信息计算等几个方面论述目前这一技术的发展情况和所面临的问题.

5.1 代价估计方法研究

代价估计是对查询物理运算时间的估计.目前,代价计算方法主要有 3 种:基于参数的方法、基于取样的方法和基于统计信息计算的方法.基于参数的方法[35]无须统计数据信息,根据数据分布情况假定其满足包含某些参数的分布函数,通过计算函数的值估计查询计划的执行代价.这种方法在处理有规律分布的数据(如学生成绩)时,可以大量节省统计信息空间和 I/O 代价,但对分布无明显规律的数据会有很大的误差;基于取样的方法[36,37]也无须统计数据信息,做法是从数据集中提取具有代表性的样本,比较不同的查询计划的执行情况,获得代价最小的方案.这种方法的精确性决定于取样的代表性.最简单的方法是随机取样,一些优化的方法根据数据的密度取样.取样方法的缺点是代价估计本身占用时间可能很大;最常用也是研究最多的方法是基于统计信息的方法[38],需要统计估计所用的各种信息,利用统计信息计算不同方案的执行代价.这种方法的精确性取决于统计信息的正确性.但是,统计信息过大又会导致统计时间过长.利用有限的时间和空间得到相对小的执行方案,是代价统计的基本原则.不同的系统支持的物理运算算法不同,代价模型也不同.

在关系模型中,进行代价估计时有两个通用的前提:独立性假设和均匀分布假设.前者是指各谓词之间没有相互依赖关系;后者是指如果关系在某个属性上没有直方图,则认为该属性的各值在数据库中均匀出现.

XML 数据的不规则性是对传统统计信息方法的重要挑战.其数据分布情况使得一些传统分布假设难以成立.在 XML 中,相同名字的结点可能在同一个文档的不同部分出现但却具有截然不同的语义.如同为 name 结点,在 person 下和在 city 下出现其意义就完全不同,这可以称为元素之间的结构依赖性;同时,在 XML 文档中,嵌套在不同祖先下的同类结点的个数差别也很大,如 book 结点下的 author 个数是不确定的,这可以称为元素之间的结构相关性.为了达到所需的代价估计精度,需要更多的统计信息.而结构的复杂性又为获得相对精确的统计信息带来存储和计算上的困难.XML 的有序性制约了转换规则的灵活性.所有这些问题,都使得在 XML 中采用传统的代价估计方法不切实际,会带来很大的误差.针对 XML 数据的特点,我们应该寻求一种新的代价估计方法.

5.1.1 代价模型

查询计划的执行代价主要来自 3 方面的因素:CPU 计算代价、I/O 代价和数据传输代价.CPU,磁盘和网络的速度差距悬殊.当不考虑数据的分布性因素时,影响代价的决定性因素是 I/O 代价.I/O 代价受众多因素制约,主要来自 3 个方面:一是数据库系统参数,如页面大小、内存使用情况等;二是数据集本身因素,如数据存储空间大小、索引情况、每个元素占用空间情况和元素的聚集情况等;三是查询请求因素,如查询条件的选择性等.

目前,对 XML 代价模型的研究并不充分,代价模型相对简单,这也是造成代价估计误差的一个原因.Lore 的代价模型没有考虑聚集情况,不能判定不同的数据是否在同一页面上,因此,其假设每次 I/O 操作只能获得一个对象,把对 I/O 时间的估计转换为对中间结果大小的估计.由于 Lore 中数据本身无聚集,这种方法可以获得较好的效果,但对其他 XML 数据库系统参考意义并不大.文献[39]提出了一种新的代价模型,其基本思想是利用查询反馈信息来调整参数.如图 3 所示,在用户提交查询之前,先人工找出影响查询执行时间的特征,再利用线性回归模型计算出各个特征对查询代价的影响系数,即得到一个形如 $cost(q)=f(v_1,v_2,\ldots,v_d)$ 的函数模型,当用户提交查询时,利用函数和事先统计的特征值进行计算.但是,文中提出的方法只是针对 CPU 代价估计的,没有扩展到 I/O 代价的估计;而且只考虑了一个 XNav 操作符,至于如何扩展到其他情况,文献[39]中并未提及.人工抽取特征具有主观性,如何让系统自动地抽取特征是下一步研究的重点.

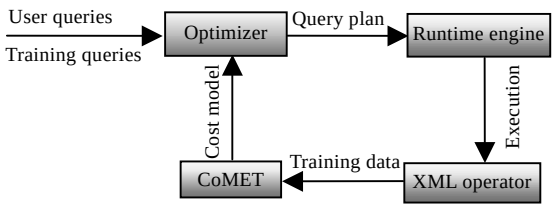


Fig.3 CoMET optimization system

图 3 CoMET 优化系统

5.1.2 代价空间搜索技术

代价空间搜索算法首先通过某种计算方法量化代价空间、构造搜索函数,根据函数值的变化判断是否继续搜索.在众多的空间搜索技术中,最简单的是随机搜索方法,随机地或者按照某个顺序在搜索空间中计算代价.但这种方法效率低下,在实际的系统中很少被采用.爬山算法是应用广泛的一种搜索算法,在以某步长对函数进行搜索的过程中按照逐步接近的方式,定位局部最优执行计划,搜索的效率与初始值和步长相关.当搜索函数非单调时,这种方法找到的是局部极值,而非全局最值.遗传算法是解决局部最优的一种新颖的空间搜索方法.用杂交的方法搜索不同的最优执行计划,适用于有多个极值的搜索函数.这种方法在关系数据库查询优化中有一定的应用意义,在 XML 查询优化的代价空间搜索技术中应用遗传算法的难点在于适应度函数的构造.

单个查询物理计划形成的代价空间可能非常庞大,尤其是对路径很长的情况,其代价空间呈幂次级增长.为了减少代价估计时间,需要利用启发性规则约束代价空间.Lore 的做法是:分别将每一个逻辑操作转换为最优物理子计划,并在转换时应用启发性规则.例如:TargetSet 操作只在路径表达式的起始结点是标记名并且只在路径结束结点上有变量约束时使用;当查询中有多个路径表达式时,不改变其间的顺序.值得注意的一条规则是,选择操作总在最后做.这条规则和关系查询优化的启发性规则正好相反.这是由于在 Lore 中,选择运算总是基于变量绑定运算.

在 XML 代价估计研究中,路径表达式选择性代价估计是核心问题,也是在 XML 查询优化研究中份量最重的一个领域,值得我们特别关注.在第 5.2 节中我们将做专门的论述.

5.2 路径表达式选择性代价估计

XML 路径表达式可视为一棵树,其中的一个主支为从起点到目标点的主路径,其余分支为约束主支的谓词条件(如图 4 所示),表示为 $P=t_1[p_1]/t_2[p_2]/\ldots./t_n[p_n]$.其中:$t_i$ 为结点名;p_i 为谓词,默认存在量词布尔表达式.路径表达式的选择性估计是对满足分支条件的主支数据个数的估计.对 XML 路径表达式的估计需要数据结构的统计信息与分布在结构内部的值的统计信息的结合,以计算路径的选择性.

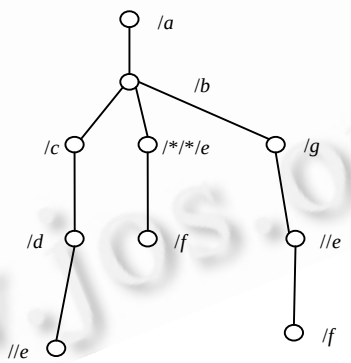


Fig.4 /a/b[c/d//e][g//e/f]//*/*/e/f

图 4 /a/b[c/d//e][g//e/f]//*/*/e/f

5.2.1 数值统计

Chen 等人[40]把数值结点作为普通结点看待,这样,估计 $a=3$ 与 $a.3$ 是等价的,简化统计结构,适用于数值量较少的情况.如果数值量庞大,统计每一个数值的个数会占据大量的空间,导致统计信息过多,影响查询代价估计的效率.这种方法对等值的定点查询可以使用,却很难估计范围查询的代价.如估计 $a>3$ 的代价,需要遍历统计信息中所有同类数值结点,判断其值是否满足条件.

Freire 等人[41]用直方图等方法分别统计不同类型的数值信息,如最大值、最小值、平均值、个数等信息.即适用于范围查询,也适用于点查询.其缺点是:数据类型过多会导致统计信息的膨胀;并且,XML 数据的特点是自描述性,其值和结构有密切的语义关系,分开统计必然导致分别估计,造成估计误差,在文献[41]中有关于这方

面的详细论述.

5.2.2 数据结构抽取

XML 数据为有序有向图.对图的结构抽取有两种极端的方法[42]:

标记分裂图(label-split graph)方法粗略地统计模式信息,其思想是合并所有的标记名相同的结点为一个结点,记录合并结点的个数作为标记的统计信息.这种方法占用空间相对较小,但不能精确地反映数据分布的真实情况,尤其是同名结点出现在不同位置上时,可能包含错误的路径或者圈信息;

B/F 类似图(B/F-bisimiar graph)中,只有所有入边和出边集合相同时才合并同名的结点,保证准确地统计路径表达式所有的分支情况.这种方法的缺点是占用空间大.

查询优化的统计信息控制在一定的范围内,现有系统的抽取方法都是介于两个极端的情况之间.

Lore 的 DataGuide[43]抽取模式的方法是保证每条路径只出现一次,其最小模式是标记分裂图的一个例子.文献[43]指出:这种方法统计路径信息能够精确判断路径是否存在,但并不能更精确地统计不同结点的值在不同路径中的分布情况.如图 5 所示:图 5(c)为图 5(a)的最小 DataGuide.如果对结点 19 的统计信息为 t,根据图 5(c)无法判断是路径 A.C 或者 B.C 的结点个数;图 5(b)为图 5(a)的强壮 DataGuide.如果对结点 12 和 13 的统计信息为 t_{12} 和 t_{13},根据图 5(b)判断路径 A.C 的个数为 t_{12},B.C 的个数为 t_{13}.

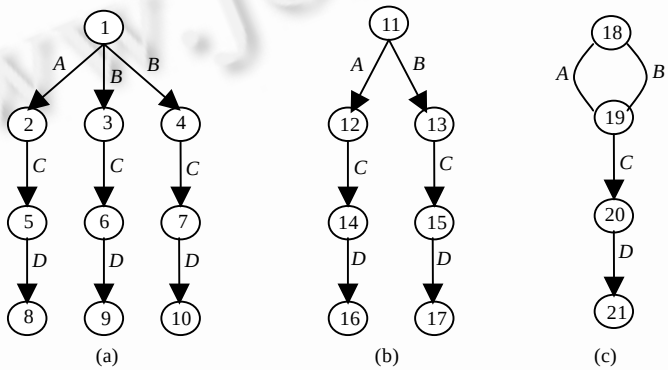


Fig.5 XML data and its corresponding DataGuides

图 5 XML 数据和 DataGuides

StatiX[41]根据 XML Schema 统计结构信息,它是一种折衷的方法.出边不同但同类型的结点合并为一个,系统对不同类型的结点标记以不同区域的值,按照区域分别统计不同路径的结点个数,保留了分支结点的路径分布信息.儿子结点的分布情况保留在父亲结点的统计信息中.每个结点的统计信息不再是单个的值,而是一个复杂的结构.导致的问题一是代价信息的增长,二是代价估计算法的复杂性.

Xsketch[42]也是一种折衷的方法,但只统计结构中每个结点对应数据中结点的个数信息.其特别之处在于对结点入边和出边的信息的统计.如果对任意结点 $v\in V$,存在 $u\in U$,在数据集中都有边(u,v),则 V 对 U 向后固定.如果对任意结点 $u\in U$,存在 $v\in V$,在数据集中都有边(u,v),则 U 对 V 向前固定.这种统计信息用于在合并不同分支与主支之间选择性代价估计的计算.

上述方法统计的只是路径中父子之间的关系,而没有统计同父的兄弟之间的相关性.为了便于计算相对路径的代价和统计路径之间的相关性,Chen 等人[40]吸收了信息检索中在文档中检索子串的采用后缀树的方法[44]统计路径信息,称为相关子路径树(correlated subpath tree).从 XML 数据中抽取所有到叶子的可能路径,形成路径集合,其中间结点的结点名为不可分割子串;叶结点为数值或者字符串值,为可分割子串.每个结点存储子路径在数据中出现的次数和路径特征矢量.我们将在后面的选择性计算部分介绍后缀树方法的代价计算方法;在信息统计部分介绍后缀树信息的存储和修剪维护方法.

XSketches[45,46]在文献[42]的基础上增加了对边的信息和值的信息的统计,从而在一定范围内(TSN)能够处理结构和值或值和值的相关性问题.但增加信息的同时,也使得构造 XSketches 结构代价加大.在 XSEED[47]中提

到:为 100M 的 XMark[48]文档构造一个 XSketches 结构需要超过一天的时间,这在实际中变得不可接受.

针对 XSketches 的缺点,XSEED[47]提出了一种新的处理思路,即抽取最粗略的信息,称为 XSEED 内核,一般只占文档大小的 2%左右;然后,再利用查询反馈的信息把误差最大的那部分路径选择率的精确值存储起来,而这部分存储的信息的多少是根据内存大小来确定的.但是,这种方法只能处理 XPath.

如果把 XML 数据看作树,对树中的结点按照(start,end)[49]编码,以 start 为横坐标,end 为纵坐标,则树中所有结点可看作是平面空间上的点,路径上的祖先和后代关系满足:$x_{ancestor}<x_{descendant}<y_{descendant}\leq y_{ancestor}$,把整个空间区域分成多个方格,统计结点在每个格中的分布个数;Wu 等人[50]的思想是:把不同结点映射为一维坐标上的线段,祖先线段和后代线段的起点及长度满足某种偏序关系,把整个区间分成多个小区间,分别计算落在不同区间内的线段之间的关系.这种方法减少了稀疏分布时的误差,但不能完全避免.这两种统计方法适用于对结构连接运算的代价估计.

5.2.3 选择性计算

当 XML 数据结构复杂、每个文档的数据量很大时,精确地统计信息是不可能的.在计算查询代价时,需要用一些假设来弥补统计信息的不足.目前的路径选择性计算方法分为 3 种:

(1) 基于马尔可夫链的方法

Lore[43]的方法基于马尔可夫链思想,用于计算没有分支条件的简单路径.系统中只保留短路径的选择性统计信息,基于父子结点分布的独立性假设计算长路径选择性.如有长路径 $t_1/t_2/t_3/\ldots/t_n$,其选择性计算公式可以是

$$\hat{\sigma}(t_1t_2...t_n)=\left(\prod_{i=1}^{n-1}\frac{f(t_i,t_{i+1})}{f(t_i)}\right)\times f(t_{n-1},t_n).$$

此时,只需保留长度为 1 和 2 的路径信息.也可以是

$$\hat{\sigma}(t_1t_2...t_n)=\frac{f(t_1t_2...t_{n-1})\times f(t_2t_3...t_n)}{f(t_2t_3...t_{n-1})},$$

则需保留长度为 $n-2$ 和 $n-1$ 的路径信息.路径信息越长,组合个数越多,占用空间越大,计算值越精确.实验表明,当路径信息较短时,加长路径信息导致明显的计算精确性,且空间代价增长相对缓慢;当路径信息较长时,加长路径信息导致空间代价的爆炸性增长,而精确性提高缓慢.如果在路径的某个结点上有对简单值的选择,则根据兄弟分布独立性原则,计算不同选择性再做交集运算.

Aboulnaga 等人[51]对 Lore 的方法进行了改进,提出了路径(path)树和 Markov 表来估计简单路径的选择性.用路径树可以表示原文档的结构,树中的每一个结点代表了从文档的根结点开始的路径,并记录了相应结点出现的次数.但当树变得很大以至于不能放在内存的时候,就需要对树进行剪枝,根据一定的算法,删去那些出现不频繁的结点,然后在这个剪枝过的树上进行选择率的估算.Markov 表存储长度为 m (m>=2)的不同路径,如果查询的长度和 m 相等,直接就可以从表中读出相应的值,误差为 0;当长度大于 m 时,用公式

$$f(t_1/t_2/.../t_n)=f(t_1/t_2/.../t_m)\times\prod_{i=1}^{n-m}\frac{f(t_{1+i}/t_{2+i}/.../t_{m+i})}{f(t_{1+i}/t_{2+i}/.../t_{m+i-1})}$$

进行计算.同样地,当表不能放入内存时,删除那些不频繁路径.

Xsketch[42]增加了对边的固定性统计信息,并对 Lore 的方法做了改进,以适用于更一般的路径表达式代价估计.如果主路径是向后固定的,则统计信息为精确信息,无须进一步计算;如果主路径中有些点不是向后固定的,则在这些点上把主路径分为多个子路径,根据路径独立性假设,用类似 Lore 的公式,以子路径统计信息为参数,计算长路径的选择性.如果分支路径是向前固定的,则其选择性为 1,无须参与计算;如果分支路径中有些点不是向前固定的,则在这些点上把分支路径分为多个子路径,计算不同选择性,再做交集运算.为了提高计算的精确性,其代价路径分解方法为最大交叉方法.

XSketches[45,46]对文献[42]的工作进行了扩展,可以计算 twig 匹配的个数.XSketch 在原来模型的基础上增加了边的分布信息,从而能够从细节上把握数据的分布.这种方法的特点是:先抽取 XML 文档的结构建立 XSketches,然后利用边的稳定性和边的分布概率来估计 twig 的匹配个数,如果边的确是分布均匀的话,那么这

种方法的准确率就比较高.

XSEED[47]方法在 XSEED 核结构上增加了对递归结点的处理,递归结点是指在 DTD 中有类似 $A(A+,B*)$的定义,则 A 结点是一个递归结点.XSEED 核结构在边上记录了在递归的不同层相应的父亲、孩子的结点个数.因此,这种方法可以很好地处理带有递归表达式的 XPath 查询.

马尔可夫链思想中的一个关键性假设是父子结点分布独立性和兄弟结点分布独立性假设.而实际上,很多 XML 数据的父子结点、兄弟结点之间的相关性非常强,应用上述计算方法会导致误差.集合哈希方法统计相同分支之间的相关性信息,更准确地计算分支路径的选择性.

(2) 集合哈希方法

集合哈希方法[40]的核心思想来自蒙特卡罗技术的 Min-Wise independent permutations[44].这种方法用于估计两个集合之间的相似性,集合的特征通过设置哈希函数的集合获得.其优势在于集合的哈希函数值占用存储空间很小.对集合 A,其特征矢量为

$$\text{sig}A=(\min\{\pi_1(A)\},\min\{\pi_2(A)\},\ldots,\min\{\pi_l(A)\}).$$

其中,$U=\{1,\ldots,n\}$;π是 U 的排列;l 是哈希函数的个数.

$$\text{sig}_{A_1\cup\ldots\cup A_k}[i]=\min\{\text{sig}_{A_1}[i],\ldots,\text{sig}_{A_k}[i]\}.$$

假设 A_j 为势最大的集合,则 $\hat{\gamma}=\dfrac{|A_j|}{|A_1\cup\ldots\cup A_k|}$,而

$$|A_1\cup\ldots\cup A_k|=\frac{|\{i\,|\,\min\{\pi_i(A_1)\}=\ldots=\min\{\pi_i(A_k)\}\}|\bullet|A_j|}{\hat{\gamma}}.$$

关于集合哈希函数的构造方法在文献[44]中有详细的论述.

利用集合哈希方法计算路径表达式的代价的方法为:用后缀树统计所有可能出现在查询中的子路径的特征矢量;分解查询为多个相关子路径;应用公式计算选择性.文献[40]中提供了 3 种路径分解的方法,并比较了不同方法之间的优劣.这种代价计算的精度对路径的长度敏感:随路径长度的增长,统计精度下降.并且,特征矢量本身是一种信息压缩的方法,用来计算相关性时的精确性是值得商榷的.

上述两种算法都是根据确定路径上的父子关系统计信息计算路径表达式中后代结点的选择性,没有直接计算后代,也没有根据某些后代估计满足条件的祖先结点的选择性.在执行计划中,存在大量的向前遍历的算法.如何估计这些算法的代价,是一个需要解决的问题.

上面所提到的各种方法的处理能力各有不同,仅从方法能够支持的各种情况(Xpath,Xquery,//,*,value)来看,可以总结为表 2.

Table 2 Comparison among various methods' processing

表 2 各种方法处理能力比较

	Query	//	*	Branch	Value	Correlation
Statix	Simple twig + value	N	Y	Y	Y	N
Lore	Simple path	N	Y	N	N	N
CST	Simple twig + prefix string matching	N	Y	Y	Prefix string matching	Y (set hashing)
Path tree	Simple path	N	Y	N	N	N
Markov table	Simple path	N	Y	N	N	N
Xsketch	Simple twig	N'	N'	Y	N	N
Xsketches	Simple twig + value	N'	N'	Y	Y	Y (md methods)
XSEED	Path	Y	Y	N	N	N

(3) 位置直方图方法

Wu 等人[50]采用位置直方图的方法统计组先后代的分布信息,其代价计算分为基于祖先的代价估计和基于后代的代价估计.基于祖先的代价估计根据每一个祖先的格,累计其满足条件的后代的格中的结点个数;基于后代的代价估计则与其正好相反.根据不同区域的祖先后代结点的偏序关系,分别计算.如图 6 所示:A 为某祖先格,则 B 区域中所有的格的所有结点均为 A 中所有结点的后代;C,E 区域中所有格的部分结点为 A 中部分结点的后代;而 D,F 区域中只有左上半角区域中存在结点且部分结点为 A 中部分结点的后代.如果考虑自身的嵌套,则 A

中部分结点是 A 中部分结点的后代.据上所述,满足条件 P_1 的 A 的满足条件 P_2 的后代个数估计公式为

$$Est_{P_{1,2}}[A]=Hist_{P_1}[A]\times\left\{\frac{1}{4}\times Hist_{P_2}[A]+Hist_{P_2}[B]+Hist_{P_2}[C]+Hist_{P_2}[E]+\frac{1}{2}\times(Hist_{P_2}[D]+Hist_{P_2}[F])\right\}.$$

对每个格而言,其后代格限定在较小区域内,避免多余的统计和计算.但在计算区域的边缘,并非所有结点满足上述关系.根据平均分布的假设给计算结果加以某个系数是产生误差的主要因素,误差在空间结点分布稀疏时变得很大.一个改进的方法是分别统计不同类型的结点的分布情况,但统计信息占用空间很大.这种方法解决的问题是已知集合间的祖先后代关系的估价,未涉及路径中单个谓词的选择性计算问题.

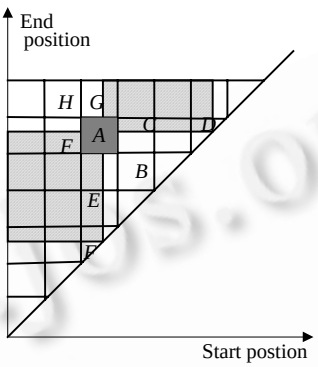
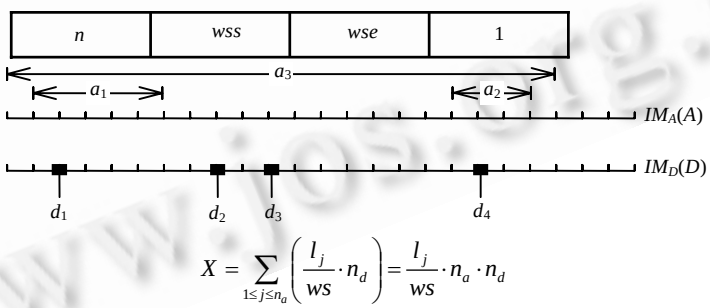


Fig.6 Two-Dimention histogram

图 6 二维直方图示例

Jiang 等人[52]的思想是:把不同的结点映射为一维坐标上的线段,祖先线段和后代线段的起点和长度满足某种偏序关系.把整个编码空间[*c*min,*c*max]分成多个小区间,然后在每个小区间内估计覆盖的线段/起始点的对数.如图 7 所示:统计信息 n 表示每个桶中的线段/起始点的对数;*wss* 表示桶的起始坐标;*wse* 表示桶的终点坐标;l 表示桶中线段的平均长度;匹配的祖先-后代个数在图中用 X 表示.这种方法减少了稀疏分布时的误差,但不能完全避免.



$$X=\sum_{1\le j\le n_a}\left(\frac{l_j}{ws}\cdot n_d\right)=\frac{l_j}{ws}\cdot n_a\cdot n_d$$

Fig.7 PL histogram and statistics

图7 PL直方图及统计信息

上述方法在计算路径选择性时,只考虑有谓词约束的结点,跃过其他结点,直接估计后代或祖先的代价,简化了计算的复杂度,在对模糊路径的代价估计中有一定的优势.与马尔可夫方法相同的是:当路径中出现多个谓词时,假设不同的谓词条件是独立的,没有计算不同的谓词之间的相关性.集合哈希方法计算不同谓词之间的相关性,但后缀树占用空间过大,尤其是在 XML 数据中包含大量数值时.而查询优化的一个原则就是在有限的时间和空间内精确地估计代价.这时,必须对统计信息进行压缩,以保证优化的效率.

5.3 统计信息研究

XML 数据结构复杂,分布不均匀.数据量庞大时,在有限的空间内统计足够多的信息是统计信息研究的难点.当数据更新频繁时,高效的信息维护技术显得更为重要.

5.3.1 统计信息存储

在前文介绍数值统计和数据结构抽取时,已经涉及到统计信息的存储问题,但讨论集中在逻辑层,并未涉及存储的形式.XML 数据统计信息的存储结构主要有以下几种:

树或图[40−42]:用于存储模式信息,如在文献[42]中模式树中每个结点的个数,或者文献[40]中结点之间的固定关系等.由于其数据结构与 XML 原始数据相同,可用对数据本身的存取方法存取统计信息.如果 XML 数据模式结点个数为 n,则树方法存储的代价为 $O(n)$;后缀树的存储代价为 $O(n!)$.

马尔可夫表[51]:用于存储路径信息,不同的路径对应其在数据中出现的次数.为了查找方便,一般在表上再加以一定的索引.如果只统计小于 k 长度的路径,则其存储代价为 $O(nk)$.

直方图[38]:关系数据库中普遍应用的一种统计信息方法.将某属性的值域划分为多个连续或不连续的部分,分别统计不同部分区域内数据的分布情况.对于简单数据,采用一维直方图方法;若统计相关的不同属性的分布情况,需要二维或者更多维数的直方图.在对 XML 数据作统计时,主要采用直方图和树相结合的方法.如果某个结点的值为树值型,则用直方图统计这个结点数值的分布情况.文献[41]中,对于结点之间的父子关系也采用直方图的方法统计.必须首先唯一地标记所有的数据结点,并且,不同类型结点标记的值域没有交叉.当路径中的某些点有谓词约束时,通过统计信息无法知道那些满足条件的结点的标记,从而无法进一步估计其后代的个数.

位置直方图[50]:这是一种二维直方图,其值是离散的,把结构嵌套转换为平面位置关系.如果按类型每维分为 k 格,则其存储代价为 $O(nk^2)$.文献[52]等采用一维直方图方法保存结点的位置信息,但需要根据结点在连接过程中是祖先或后代来保存不同的统计信息,实际上冗余很大.

5.3.2 统计信息压缩

对直方图压缩方面的研究在关系数据库查询优化中已经有一定的研究基础,普遍采用的是小波压缩[53]和 DCT(discrete cosine transform).两种方法均能把直方图中大量的“桶”压缩为少量的系数,同时丢失很少的信息.小波压缩方法在代价估计时,要重新构造与查询相关的部分直方图.Yan[35]等人采用密度函数方法压缩直方图,用数据密度函数直接模拟实际的数据的分布情况.这种方法适用于离散的或是连续的数值型数据.

树的修剪和马尔可夫表的压缩思想是从统计信息中删除对代价估计结果影响相对较少的结点.结点在数据中出现次数越少,则越缺乏统计价值.文献[51]中提供了 4 种不同的方法,并通过实验分析了几种方法在不同的数据分布情况和查询情况时的占用空间和代价估计精度关系,但并未得到相对稳定的方法.

5.3.3 统计信息维护

据我们所知,目前对 XML 统计信息维护的研究很少见.文献[54]提出一种马尔可夫表的在线维护方法,其思想是:在查询开始时没有统计信息,利用查询反馈逐步生成和细化统计信息.其细化规则有两个:重尾规则(heavy-tail)和增量规则(delta).重尾规则是在计算选择性时,给接近查询路径末端的路径的选择性计算加以较高的权重.这样做基于如下考虑:接近路径末端的路径选择性对整个路径的选择性影响更大;增量规则是一种错误减少学习方法.文献[54]的优点在于在线维护统计信息.文献[55]提出了基于 StatiX[41]框架 IMAX 增量维护算法,包括结构信息和值信息的增量维护,但是不易扩展到其他系统上.

6 总结及展望

随着 Internet 的发展,XML 数据规模与日剧增,准确、高效地查询 XML 数据成为目前研究的热点问题.近年来,XML 查询优化研究方兴未艾,主要集中在如下几个方面:对 XML 代数的研究、对根据 XML 代数分解查询语句的研究、对路径选择性估计的研究、对结构连接代价估计的研究等.XML 查询优化是一门综合性强的研究领域,需要吸收众多其他技术的思想,其中对 XML 查询优化影响深刻的主要有:关系数据库查询优化技术、面向对象数据库查询优化技术、信息检索技术、数据仓库和数据挖掘技术、图像处理和压缩技术、人工智能

技术、概率论和统计技术等.

从查询优化的过程来讲,XML 查询优化和其他数据库查询优化技术并无不同之处.从优化的不同环节的技术上看,XML 查询优化具有其独特之处:既要适应更复杂的数据结构和更灵活的变化,又要适应更丰富的查询语义.目前对 XML 查询优化的研究工作还远未成熟,仍有众多尚待解决的问题或需要完善的技术.因此,未来的 XML 查询优化研究将以更广泛、更深入的方式展开.

在 XML 代数研究中,一个值得重视的问题是逻辑操作与物理操作的分工不明确.大量的工作在于制定不同的代数标准,这些标准在风格上相去甚远,很难有共通性.而对真正执行的物理代数的研究还远远不够,而且不能形成完整的系统.另外,从逻辑代数到物理代数的转换也将是未来研究的一个重要的问题.

关系数据库中一些约定俗成的启发式转换规则是在大量的实践基础上形成的.而目前对 XML 数据查询的各种不同的执行方法之间的优劣比较的工作还刚刚开始,并未形成共识性的规则.由于 XML 数据本身的灵活性,找到一些普遍适用的规律是很困难的.在今后的一段时间内,相信会有更多的研究工作在这方面展开.

复杂路径表达式选择性代价估计是 XML 查询优化研究的核心问题,目前已有大量的成果.这些研究或对数据分布进行大量的假设,或对查询表达式的复杂性加以一定的约束.尤其是在相关路径选择性的研究方面,仍有一些尚待解决的关键性问题.

一直以来,统计信息维护是代价估计的基础.但是,关于 XML 数据统计信息的维护问题的研究还处于起步状态.由于 XML 数据统计信息在数据结构上与传统的统计信息有本质的不同,很难直接利用现有的统计信息维护的技术.又由于目前在 XML 统计信息研究中采用了大量的压缩技术,为统计信息维护增加了难度.

Native XML Database 的研究是目前在 XML 研究领域的一个热点,已经出现一批相对独立的系统,这些系统采用的查询和处理方法也将对 XML 查询优化技术产生越来越重要的影响.

References:

[1] Beech D, Malhotra A, Rys M. A formal data model and algebra for XML. In: Beech D, Malhotra A, Rys M, eds. Note to the W3C XML Query Working Group. 1999. 1−26. http://www-db.stanford.edu/infoseminar/Archive/FallY99/malhotra-slides/malhotra.pdf

[2] Fernandez M, Simeon J, Suciu D, Wadler P. A data model and algebra for XML query. 1999. http://www.cs.bell-labs.com/wadler/topics/xml.html#algebra

[3] Kay M. XSL transformations (XSLT), Version 1.0. W3C Recommendation, 1999. http://www.w3.org/TR/xslt

[4] Fankhauser P, Fernandez M, Malhotra A, Rys M, Simeon J, Wadler P. XQuery 1.0 formal semantics. W3C Working Draft, 2002. http://www.w3.org/TR/query-semantics/

[5] Fernandez M, Robie J. XQuery 1.0 and XPath 2.0 data model. W3C Working Draft, 2002. http://www.w3.org/TR/query-datamodel/

[6] Mary FF, Jérôme S, Byron C, Amélie M, Gargi S. Implementing xquery 1.0: The galax experience. In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 1077−1080.

[7] McHugh J, Abiteboul S, Goldman R, Quass D, Widom J. Lore: A database management system for semistructured data. SIGMOD Record, 1997,26(3):54−66.

[8] McHugh J, Widom J. Query optimization for XML. In: Atkinson MP, Orlowska ME, Valduriez P, Zdonik SB, Brodie ML, eds. Proc. of the 25th Int'l Conf. on Very Large Data Bases. Edinburgh: Morgan Kaufmann Publishers, 1999. 315−326.

[9] Jagadish VH, Al-Khalifa S, Lakshmanan L, Nierman A, Paparizos S, Patel J, Srivastava D, Wu YQ. Timber: A native XML database. The VLDB Journal, 2002,11(4):274−291.

[10] Jagadish VH, Al-Khalifa S, Lakshmanan L, Srivastava D, Thompson K. Tax: A tree algebra for XML. In: Ghelli G, Grahne G, eds. Database Programming Languages, 8th Int'l Workshop, DBPL 2001. Frascati: Springer-Verlag, 2001. 149−164.

[11] Frasincar F, Houben GJ, Pau C. XAL: An algebra for XML query optimization. In: Zhou XF, ed. Proc. of the 13th Australasian Database Conf. (ADC 2002). Melbourne: Monash University, 2002.

[12] Zhang D, Dong YS. A data model and algebra for the Web. In: Proc. of the 10th Int'l Workshop on Database & Expert Systems Applications. Florence: IEEE Computer Society, 1999. 711−714.

[13] Liefke H. Horizontal query optimization on ordered semistructured data. In: Cluet S, Milo T, eds. Proc. of the ACM SIGMOD Workshop on The Web and Databases (WebDB'99). Philadelphia: ACM Press, 1999. 61−66.

[14] Mukhopadhyay P, Papakonstantinou Y. Mixing querying and navigation in MIX. In: Agrawal R, Dittrich K, Ngu AHH, eds. Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Computer Society, 2002. 245−254.

[15] Paparizos S, Al-Khalifa S, Jagadish HV, Nierman A, Wu YQ. A physical algebra for XML. Technical Report, University of Michigan, 2002.

[16] Christophides V, Cluet S, Moerkotte G. Evaluating queries with generalized path expressions. In: Jagadish HV, Mumick IS, eds. Proc. of the 1996 ACM SIGMOD Int'l Conf. on Management of Data. Montreal: ACM Press, 1996. 413−422.

[17] Buneman P, Fan W, Simen J, Weinstein S. Constraints for semistructured data and XML. ACM SIGMOD Record, 2001,30(1): 47−45.

[18] World Wide Web Consortium. XML path language (XPath) Version 1.0. W3C Recommendation, 1999. http://www.w3.org/TR/xpath.html

[19] Chamberlin D, Clark J, Florescu D, Robie J, Siméon J, Stefanescu M. XQuery 1.0: An XML query language. Technical Report, World Wide Web Consortium, W3C Working Draft, 2001.

[20] Deutsch A, Fernandez M, Florescu D, Levy A, Suciu D. A query language for XML. 2003. http://www.research.att.com/~mff/files/final.html

[21] Robie J, Lapp J, Schach D. XML query language (XQL). http://www.w3.org/TandS/QL/QL98/pp/xql.html

[22] Chamberlin D, Robie J, Florescu D. Quilt: An XML query language for heterogeneous data sources. In: Suciu D, Vossen G, eds. The World Wide Web and Databases, 3rd Int'l Workshop WebDB 2000. LNCS 1997, Springer-Verlag, 2001. 1−25.

[23] 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. VLDB 2001, Proc. of the 27th Int'l Conf. on Very Large Data Bases. Roma: Morgan Kaufmann Publishers, 2001. 361−370.

[24] Chan C, Felber P, Garofalakis M, Rastogi R. Efficieng filtering of XML documents with Xpath expressions. In: Agrawal R, Dittrich K, Ngu AHH, eds. Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Computer Society, 2002. 235−244.

[25] Wood PT. On the equivalence of XML patterns. In: In: John W L, Verónica D, Ulrich F, Manfred K, Kung-K L, Catuscia P, Luís M P, Yehoshua S, Peter J S, eds. Proc. of the 1st Int'l Conf. on Computer on Computation Logic. LNAI 1861, Berlin: Springer-Verlag, 2000. 1152−1166.

[26] Florescu D, Levy AY Suciu D. Query containment for conjunctive queries with regular expressions. In: Popa L, ed. Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. Seattle: ACM Press, 1998. 139−148.

[27] Calvanese D, Giacomo GD, Lenzerini M. On the decidability of query containment ander constraints. In: Popa L, ed. Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. Seattle: ACM Press, 1998. 149−158.

[28] Wadler P. A formal semantics of patterns in XSLT. Markup Languages archive, 1999,2(2):183−202.

[29] Wood PT. Minimizing simple XPath expressions. In: Mecca G, Siméon J, eds. Proc. of the 4th Int'l Workshop on the Web and Databases, WebDB 2001. Santa Barbara: ACM Press, 2001. 13−18.

[30] Amer-Yahis S, Cho S, Lakshmanan LV, Srivastava D. Minimization of tree pattern queries. In: Aref WG, ed. Proc. of the SIGMOD 2001 Electronic. Santa Barbara: ACM Press, 2001. 497−508.

[31] Ramanan P. Efficient algorithms for minimizing tree pattern queries. In: Franklin MJ, Moon B, Ailamaki A, eds. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data. Madison: ACM Press, 2002. 299−309.

[32] Flesca S, Furfaro F, Masciari E. On the minimization for Xpath queries. In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. VLDB 2003, Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 153−164.

[33] Popa L, Deutsch A, Sahuguet A, Tannen V. A chase too far? In: Chen WD, Naughton JF, Bernstein PA, eds. Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of Data. Dallas: ACM Press, 2000. 273−284.

[34] Kwong A, Gertz M. Schema-Based optimization of XPath expressions. Technical Report, University of California, 2001.

[35] Lynch CA. Selectivity estimation and query optimization in large databases with highly skewed distributions of column values. In: Bancilhon F, DeWitt DJ, eds. 14th Int'l Conf. on Very Large Data Bases. Los Angeles: Morgan Kaufmann Publishers, 1988. 240−251.

[36] Haas PJ, Swami AN. Sequential sampling procedures for query size estimation. SIGMOD Record, 1992,21(2):341−350.

[37] Ling Y, Sun W. A supplement to sampling based methods for query size estimation in a database system. ACM SIGMOD Record, 1992,21(4):12−15.

[38] Muralikrishna M, DeWitt DJ. Equi-Depth histograms for estimating selectivity factors for multi-dimensional queries. SIGMOD Record, 1988,17(3):28−36.

[39] Zhang N, Hass PJ, Josifovski V, Lohman GM, Zhang C. Statistical learning techniques for costing XML queries. In: Böhm K, Jensen CS, Haas LM, Kersten ML, Larson PK, Ooi BC, eds. Proc. of the 31st Int'l Conf. on Very Large Data Bases. Trondheim: ACM Press, 2005. 289−300.

[40] Chen ZY, Jagadish HV, Korn F, Koudas N, Muthukrishnan S, Ng RT, Srivastava D. Counting twig matches in a tree. In: Young DC, ed. Proc. of the 17th Int'l Conf. on Data Engineering. Heidelberg: IEEE Computer Society, 2001. 595−604.

[41] Freire J, Haritsa JR, Ramanath M, Roy P, Siméon J. StatiX: Making XML count. In: Franklin MJ, Moon B, Ailamaki A, eds. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2002. 181−191.

[42] Polyzotis N, Garofalakis MN. Statistical synopses for graph-structured XML databases. In: Franklin MJ, Moon B, Ailamaki A, eds. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2002. 358−369.

[43] Goldman R, Widom J. DataGuides: Enabling query formulation and optimization in semistructured databases. In: Jarke M, Carey MJ, Dittrich KR, Lochovsky FH, Loucopoulos P, Jeusfeld MA, eds. VLDB'97, Proc. of the 23rd Int'l Conf. on Very Large Data Bases. Athens: Morgan Kaufmann Publishers, 1997. 436−445.

[44] Chen ZY, Korn F, Koudas N, Muthukrishnan S. Selectivity estimation for Boolean queries. In: Popa L, ed. Proc. of the 19th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems. Dallas: ACM Press, 2000. 216−225.

[45] Polyzotis N, Garofalakis M. Structure and value synopses for XML data graphs. In: Bressan S, Chaudhri AB, Lee ML, Yu JX, Lacroix Z, eds. Proc. of the 28th VLDB Conf. Hong Kong: Morgan Kaufmann Publishers, 2002. 466−477.

[46] Polyzotis N, Garofalakis M, Iosnnidis Y. Selectivity estimation for XML twigs. In: Titsworth F, ed. Proc. of the 20th Int'l Conf. on Data Engineering, ICDE 2004. Boston: IEEE Computer Society, 2004. 264−275.

[47] Zhang N, Ozsu MT, Aboulnaga A, Ilyas IF. XSEED: Accurate and fast cardinality estimation for XPath queries. In: Ling L, Andreas R, Kyu-Y W, Jianjun Z, eds. Proc. of the 22nd Int'l Conf. on Data Engineering. Atlanta: IEEE Computer Society, 2006. 61.

[48] Schmidt AR, Waas F, Kersten ML, Florescu D, Manolescu I, Carey MJ, Busse R. The XML benchmark project. Technical Report, INS-R0103, CWI, 2001.

[49] Zhang C, Naughton JF, DeWitt DJ, Luo Q, Lohman GM. On supporting containment queries in relational database management systems. In: Aref WG, ed. Proc. of the 20th ACM SIGMOD Int'l Conf. on Management of Data. Santa Barbara: ACM Press, 2001. 425−436.

[50] Wu YQ, Patel JM, Jagadish HV. Estimating answer sizes for XML queries. In: Jensen CS, Jeffery KG, Pokorný J, Saltenis S, Bertino E, Böhm K, Jarke M, eds. Proc. of 8th Int'l Conf. on Extending Database Technology. Prague: Springer-Verlag, 2002. 590−608.

[51] Jiang HF, Lu HJ, Wang W, Yu JX. Containment join size estimation: Models and methods. In: Halevy AY, Ives ZG, Doan A, eds. Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. San Diego: ACM Press, 2003. 145−156.

[52] Aboulnaga A, Alameldeen AR, Naughton JF. Estimating the selectivity of XML path expressions for Internet scale applications. In: Apers PMG, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass RT, eds. Proc. of the 27th Int'l Conf. on Very Large Data Bases. Roma: Morgan Kaufmann Publishers, 2001. 591−600.

[53] Matias Y, Vitter JC, Wang M. Wavelet-Based histograms for selectivity estimation. In: Haas LM, Tiwary A, eds. SIGMOD'98, Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Seattle: ACM Press, 1998. 448−459.

[54] Lim L, Wang M, Padmanabhan S, Vitter JS, Parr R. Xpath learner: An on-line self-tuning Markov histogram for XML path selectivity estimation. In: Bressan S, Chaudhri AB, Lee ML, Yu JX, Lacroix Z, eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong: Morgan Kaufmann Publishers, 2002. 442−453.

[55] Ramanath M, Zhang LZ, Freire J. Incremental maintence of schema-based XML statistics. In: Donald F. Shafer, eds. Proc. of the 21st IEEE Int'l Conf. on Data Engineering. Tokyo: IEEE Computer Society, 2005. 273−284.

孟小峰(1964－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为 Web 数据集成,XML 数据库,移动数据管理.

王小锋(1980－),女,硕士生,主要研究领域为 XML 数据库.

王宇(1973－),女,博士,主要研究领域为 Web 数据管理,XML 数据库.