ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.2, February 2007, pp.323−331
DOI: 10.1360/jos180323

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

XML 数据物化模式的生成与优化技术*

张 亮+, 李 然, 汪 卫, 施伯乐

(复旦大学 计算机与信息技术系,上海 200433)

A Technique to Generate and Optimize the Materialized Model for XML Data

ZHANG Liang+, LI Ran, WANG Wei, SHI Bai-Le

(Department of Computing and Information Technology, Fudan University, Shanghai 200433, China)

+ Corresponding author: Phn: +86-21-65643501, Fax: +86-21-65642219, E-mail: zhangl@fudan.edu.cn, http://www.cit.fudan.edu.cn

Zhang L, Li R, Wang W, Shi BL. A technique to generate and optimize the materialized model for XML data. *Journal of Software*, 2007,18(2):323−331. http://www.jos.org.cn/1000-9825/18/323.htm

Abstract: One way to improve the performance of XML (extensible markup language) management systems is to materialize part of the XML documents and store them aside in cache memory. In this paper, a method is presented to characterize query sets of XML data as schema graph, which is a technique to generate materialized plan based on the distribution of user's queries. Experimental results demonstrate its performance gain in XML cache management.

Key words: XML (extensible markup language); relational database; storage; query

摘 要: 根据用户查询的分布情况,基于缓存以及在硬盘上对 XML(extensible markup language)数据进行物化是提高 XML 数据存储与查询系统性能的主要方法之一.提出了一种 XML 查询集合的描述方法,即查询模式图,并以此为基础提出了一种能够充分考虑查询优化策略的物化方案生成方法.实验结果表明,该方法可以快速地生成物化方案,能够满足缓冲区管理等应用领域的需要.

关键词: XML(extensible markup language);关系数据库;存储;查询
中图法分类号: TP311 文献标识码: A

可扩展标记语言(extensible markup language,简称 XML)[1]是互联网联盟(World Wide Web Consortium,简称 W3C)提出的一种通用的数据标记语言.由于 XML 具有较强的数据描述能力和通用性,所以,XML 不仅应用于 Web 信息发布,还广泛地应用于信息集成与交换、数字图书馆等领域.由于越来越多的信息系统采用 XML 格式进行表示和存储数据,对 XML 数据的存储与查询技术的研究已成为近年来数据库领域的研究热点.

目前,XML 数据管理系统主要有两种形式:一种是所谓的"纯 XML 数据库",如 Software AG 公司推出的 Tamino 和 Stanford 大学研制的 Lore 等.它们针对 XML 数据的树型结构,结合 XML 查询中结构信息查询的要求,提出特定的查询方法,其中最主要的技术是以结构化连接[2]为基础的查询方法.这种方法对 XML 数据的结点进行合理的编码,通过对不同标签在文档中的位置进行比较,获得标签间的结构关系,从而完成查询;另一种

* Supported by the National Natural Science Foundation of China under Grant Nos.60303008, 69933010 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2002AA4Z3430 (国家高技术研究发展计划(863)

Received 2004-11-22; Accepted 2006-03-31

形式仍然采用原有的关系数据库引擎,以关系表存储经过拆分的 XML 数据.当用户进行 XML 查询或其他处理时,数据库重新组合这些数据.这个过程的关键是选择和设计 XML 在关系数据库中的存储模式,即物化视图的方案选择方法.良好的方案将会极大地提高 XML 数据的查询效率[3−6].

这两种方法各有特点:前者对 XML 数据具有很好的针对性,特别适用于数据的模式不稳定,甚至没有固定模式的情况;后者可以借助传统数据库的成熟技术,但要求 XML 数据模式相对比较固定.

第 1 种方法虽然在存储时不需要对 XML 数据的模式进行转换，但由于采用结构化联接,难以充分利用缓冲区来提高系统的性能.为了充分利用系统的资源,对 XML 数据中的一部分进行物化,提高 XML 数据的整体查询性能的做法已成为人们关注的问题[7].这一过程的关键是根据用户的查询特征确定需要进行物化的数据集合.

本文针对 XML 数据查询的特点,提出了一种新的 XML 数据查询特征的描述方法和查询集合的查询代价计算方法.该方法从查询处理优化的角度出发,较好地反映了查询的代价.在此基础上,提出了一种快速地生成物化方案的方法,以适应缓冲区管理的需要.本文后续章节将针对 XML 数据的物化方案生成过程分别进行论述,最后给出相应的实验.

1 相关研究

对 XML 物化方法的研究一直是 XML 存储与查询技术的重要组成部分.但是,由于 XML 数据库与关系数据库在模型上的差异,在将 XML 数据存储到关系数据库时,一般会导致产生大量的关系表.对于这个问题,美国宾夕法尼亚大学的 STORED 系统利用关系数据库与半结构化数据(overflow 图)相结合的方式存储 XML 数据[3].Wisconsin-Madison 大学研究了各种 XML 数据的关系数据库存储方法,提出了 Basic Inline,Shared Inline 和 Hybid Inline 三种方法[4].这几种方法主要是从 XML 数据的结构出发构造物化方案,其目标是建立 XML 数据模式与关系模式之间的对应关系,并没有考虑用户的查询集合对查询性能的影响.

在很多信息系统中,例如在数字图书馆的应用[8]中,用户的查询往往呈现一定的规律性,系统对性能的要求也较为苛刻.因此,如何根据用户的查询分布进行物化或设计缓冲区的数据存储策略,便引起了人们的重视.杨良怀[9]等人基于频繁模式生成算法对用户的 XML 数据查询集合进行分析.该方法首先获取其中的频繁模式,经过对这些频繁模式的综合与分析,产生 XML 模式树的一个子集;然后对这个子树上的节点进行物化,并将物化结果存放在缓冲区中,从而达到提高系统查询性能的目的.

为了获取用户查询模式的分布情况,需要对 XML 查询之间的关系进行分析.Dong 等人在文献[10]中对 XML 查询之间的包含关系进行了研究,提出了判断嵌套 XML 查询之间包含关系的方法,并进行了计算复杂性分析.

2 XML 数据物化模式的生成和优化过程

本文的 XML 数据物化模式生成过程如图 1 所示.

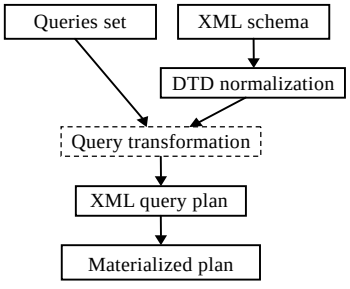


Fig.1 The process to materialize XML data

图 1 XML 数据物化存储方案生成过程

该系统首先对 XML 数据的模式(DTD(document type definition)或者 Schema)进行平坦化,生成一个简化的 XML 数据模式.简化模式去除了原来 XML 数据模式中与存储模式无关的部分.同时,经过对 XML 数据进行监控,可以得到相关的一些统计信息,如 XML 数据的规模、特定标签的出现频率等,同时也可以获得用户的查询集合.通过对这些用户查询信息的分析,可以生成用户的查询模式图.该图综合了 XML 查询集合中的各种信息.然后使用改进的贪心算法,基于 XML 查询的代价分析公式生成新的数据库模式.新生成的模式能够较好地提高系统整体的查询性能,并适应缓冲区计算的要求.

3 XML 数据的平坦化

XML 数据模式具有很强的描述能力.各 XML 元素之间具有不同的前后次序和重复方式.然而,其中很多信息却是与 XML 数据存储无关的,因而可以首先对 XML 模式进行简化.本文针对 XML 数据的一种模式描述形式——DTD,仿照文献[4]的转换规则进行了描述.针对 Schema 的转换规则是相似的.

DTD 结构的复杂性主要来源于 XML 文档中元素说明的复杂性.如$\langle$!ELEMENT $a((b|c|(d,d^*)^*)?,(e|(f+|c?)^*)^{**})\rangle$,其中 b,c,d,e,f 都是其他元素.在很多应用中,对 XML 进行存储时不需要保存其次序信息,因此可以简化 DTD,使之转换成便于关系数据库存储的只有关系和属性两级结构的关系模式.

转换规则如下:

规则 **1**. $e?\rightarrow e$; $e^+\rightarrow e^*$; $e^{**}\rightarrow e^*$; $e^*?\rightarrow e^*$; $e?^*\rightarrow e^*$.

规则 **2**. $(e_1|e_2)\rightarrow e_1,e_2$; $(e)^*\rightarrow e^*$.

规则 **3**. $e_1|e_2|e_3|\ldots|e_n\rightarrow e_1,e_2,e_3,\ldots,e_n$; $\ldots,e,\ldots,e,\ldots\rightarrow e^*,\ldots$; $\ldots,e?,\ldots,e?,\ldots\rightarrow e^*,\ldots$;

$\ldots,e?,\ldots,e^*,\ldots\rightarrow e^*,\ldots$; $\ldots,e^*,\ldots,e?,\ldots\rightarrow e^*,\ldots$; $\ldots,e^*,\ldots,e^*,\ldots\rightarrow e^*,\ldots$

规则 **4**. $(e_1,e_2,\ldots,e_n)^*\rightarrow MidE^*$, $\langle$!*ELEMENT MidE*$(e_1,e_2,\ldots,e_n)\rangle(n\geq 2)$.

规则 1 是对一元操作符进行简化的转换,即将多个一元操作符简化为单个一元操作符;规则 2 是对嵌套的表达式进行平坦化操作的转换,即将嵌套的表达式转换成平坦的表达式,也就是说,任何操作符的操作对象内部都不包括“,”和“|”这些二元操作符;规则 3 是对相同子元素进行集中的转换,即将表达式中多次出现的相同子元素进行集中表达;规则 4 是对多个子元素进行保持顺序语义的转换,即对同类子元素的多次出现进行合并.

这样,$\langle$!*ELEMENT* $a((b|c|(d,d^*)^*)?,(e|(f+|c?)^*)^{**})\rangle$将简化成$\langle$!*ELEMENT* $a(b,c^*,d^*,e^*,f^*)\rangle$.

4 查询转换

对于非限定模式的 XML 数据查询,由于 XML 模式的差异,自然会形成不同的查询实例.但在很多应用中,同一主题的 XML 数据由于其语义的相似性,尽管在结构上有所不同,但其描述的信息和用户查询的信息也是相同的.为了使系统具有更好的扩展性,可以根据 XML 数据模式(DTD)的相似性对用户的查询进行转换,将相似的查询转换成同一个.在文献[11]中,提出了 DTD 之间相似性的评价方式,并提出了建立 XML 查询模式与 XML 数据模式之间距离的函数,实现模糊查询的方法.这种方法同样可以实现不同的 DTD 上 XML 查询的转换.

5 用户查询模式图

5.1 XQUERY查询语句的模式树

与 SQL 相似,XQUERY 同样也是一种结构性很强的查询语言.为了方便查询处理和分析,需要将 XQUERY 转换成模式树的形式.本文描述的 XML 查询是以文献[12]中的模式树为基础的.

模式树中的每个结点对应于 XML 数据模式中的一个结点,每个结点的结构包括以下信息:对应的标签名和属性(过滤或构造).其中:过滤的意思是该路径出现在 Where 子句中,用于对查询结果进行过滤;构造的意思是该路径用于构造返回结果.在模式树中,从根结点到叶结点的路径对应于 XQUERY 语句中的路径.在模式树中,每条边上也包含了属性字段,该字段描述了该边是父子边还是祖先边.例如,对下面这个查询:

FOR $*p* IN document(“auction.xml”)//person, $*l* IN $*p*/profile
WHERE $*l*/*age*>25 and $*p*// watches/watch=‘Piaget’
RETURN ⟨result⟩{$*p*//watch}{$*l*/interest}⟨/result⟩
对应的模式树如图 2 所示.

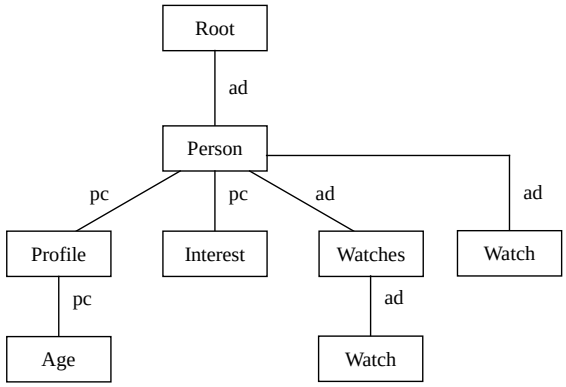


Fig.2 A sample of schema tree
图 2 模式树的例子

在上面的例子中,root//person/profile/age 是过滤部分,其余两个分支是构造部分,即与查询结果相关的部分.

下面定义在模式树中各条路径之间的关系.

定义 **1**. 如果模式树中路径 *A* 对应的 XML 文本中的结点均包含于路径 *B* 对应的结点中,则路径 *A* 包含于路径 *B* 中.如 root//person//watch 包含 root//person//watchs/watch.

定义 **2**. 如果在一个 XQUERY 查询中,路径 *A* 属于过滤部分,而路径 *B* 属于构造部分,则路径 *A* 决定路径 *B*.如 root//person /profile/age 决定 root//person/intrest.

定义 **3**. 如果路径 *A* 和路径 *B* 均属于同一个 XML 文档或同属于一个查询的构造部分,则路径 *A* 和路径 *B* 具有相邻关系.如 root//person//watchs/watch 和 root//person/intrest 相邻.

5.2 用户查询模式图

用户查询模式图从模式树分支的层次上记录 XML 查询集合中的所有查询语句,以及与查询相关的各种属性.查询模式图用于对查询集合的整体查询性能进行计算.在一棵查询模式树中,各条路径在最终查询的过程中所起的作用是不同的,查询优化过程中的一个基本原则是尽早地缩减查询结果的规模.从目前的研究结果来看,XML 的管理和查询方法无论是基于结构化联接,还是基于关系数据库,大多遵循这一原则.

定义 **4**. 用户查询模式图 $G=(V,E)$,其中:V 是结点的集合;$E=EC\cup ED\cup ER$ 是边的集合.XQUERY 模式树中的每条分枝对应于 G 中的一个结点.如果两个结点之间存在包含关系,则这两个结点之间存在一条边属于 *EC*;如果两个结点之间存在决定关系,则这两个结点之间存在一条边属于 *ED*;如果两个结点之间存在相邻关系,则这两个结点之间存在一条边属于 *ER*.

在用户查询模式图中,结点的结构为:对应的路径(*path*),在查询集合中过滤部分出现的频率(*node_ff*),在构造部分出现的频率(*node_fc*),该结点的计算代价(*node_cost*)和结点对应的 XML 文本中的元素数(*node_length*).

EC 边(u,v)为有向边,其结构为:包含关系在查询集合中出现的频率(*ec_f*);包含的结点 v 对应的元素数在被包含的结点 u 对应的元素数中所占的比例(*ec_r*),这个比例可以通过所统计的 XML 数据的信息来获得.

ED 边(u,v)为有向边,其结构为:决定关系在查询集合中出现的频率(*ed_f*),用于计算物化的效果,即如果物化 u 对应的路径,则在计算 v 对应的路径时,计算量减少的程度(*ed_r*).这个比例可以结合不同的查询实现方法,通过计算满足判断条件的结点在整个数据集中占的比例求得.

ER 边的无向边,其结构为:相邻关系在查询集合中出现的频率(*er_f*),如果两个结点出现在同一个 DTD 中并

具有父子或兄弟关系,则 *er_f*=1;否则为 0,或它们出现在同一个查询的构造部分的比例.

假设查询集合中包含如下两个查询,它们出自同一个 DTD,分别占整个查询集合的 80%和 20%.

查询 1:

FOR $*p* IN document(“auction.xml”)//person, $*l* IN $*p*/profile

WHERE $*l*/*age*>25

RETURN ⟨result⟩{$*p*//watch}{$*l*/interest}⟨/result⟩

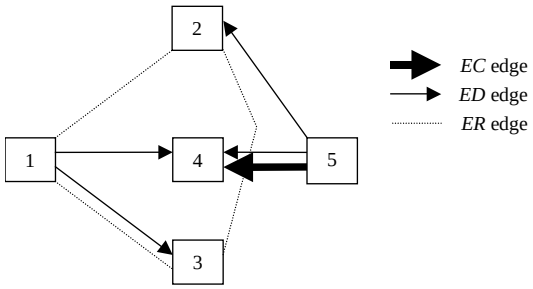
查询 2:

FOR $*p* IN document(“auction.xml”)//person, $*l* IN $*p*/profile

WHERE $*p*//watches/watch=“Piaget”

RETURN ⟨result⟩{$*p*//watch}{$*l*/job}⟨/result⟩

它们对应的查询模式如图 3 所示.



1: Root//person/profile/age 2: Root//person/profile/job 3: Root//person/profile/interest
4: Root//person/profile//watch 5: Root//person/profile//watches/watch

Fig.3 A sample of schema graph for user’s query

图 3 用户查询模式图的例子

6 物化收益的计算

通过物化收益的计算获得物化方案的思想,最早出现在数据立方体中对物化视图的选择上[7,13].随后,在物化视图选择方面出现了一批相应的研究成果.但是在数据立方体的查询中,查询模型比较简单,对查询代价的估算以数据立方体格中单元的体积进行计算,不需要考虑查询优化.在 XML 查询处理方面,目前还没有一种查询代价的估算方法,文献[9]中主要通过模式树的出现频率对部分 XML 树进行物化,放在缓冲区中,以提高查询的性能.该方法没有考虑查询优化和查询分解技术.此外,文献[9]中给出了一种根据缓冲区中的查询结果改写 XML 查询语句、提高查询系统的整体性能的方法.但是从整体上讲,目前还缺乏成熟的 XML 查询代价估测方法.

本文提出一种基于用户查询模式图计算物化收益的方法.

针对一个查询集合的用户查询模式图,可以认为整体查询的代价是查询模式图中各个结点的计算代价之和.这是由结构化联接的方法所决定的.

定义 **5**. 对于用户的查询模式图 $G(V,E)$和已经物化的结点集合 M,对于结点 $r\notin M$,如果将结点 r 物化,那些由结点 r 决定或包含结点 r 的查询的计算代价会变小.在当前物化选择 M 下,r 相对于结点 q 的物化收益 $Bv(r,q,M)$就被定义为查询 q 的计算代价的减少量,即

$$Bv(r,q,M)=C(q,M)-C(q,M\cup\{r\}).$$

其中,$C(q,M)$代表在物化结点集 M 的基础上完成查询 q 的代价.

于是,限定模式的 XML 数据物化方案选择问题就可以定义为:给定存储空间限制 S,用户查询模式图 $G(V,E)$,查询集 Q 以及 Q 中每个查询 q 的详细信息,如查询频率 $f(q)$等,要求找出一个物化结点集 $M\subseteq V$,使 Q 的

物化收益 $B(M)$最大.相应的 M 称为最佳物化选择方案.

当 $B(M)$最大时,所有查询的加权计算代价 $\sum_{v\in V} f(q)C(v,M)$ 也达到最小.这样,整个 XML 数据存储系统的性能也最好.

下面讨论查询模式图中节点之间的各种关系对查询代价的影响.

包含关系:如果结点 u 到结点 v 具有包含关系,则结点 u 的物化将导致结点 v 的查询代价下降的比例为 $ec\_r\times ec\_f$.所以,基于 u 计算结点 v 的代价为 $ec\_r\times ec\_f\times(v.node\_ff+v.node\_fc)\times v.node\_cost$.

决定关系:如果结点 u 到结点 v 具有决定关系,则由于结点 u 的物化将导致结点 v 的查询代价下降的比例为 $ed\_r\times ed\_f$.所以,基于 u 计算结点 v 的代价为 $ed\_r\times ed\_f\times(v.node\_ff+v.node\_fc)\times v.node\_cost\times u.node\_ff$.

在基于一个物化结点的集合计算某个结点的代价时,应从所有的物化结点中寻找那个带来计算量最大的物化结点,并以此进行计算.在物化选择的算法方面大致可以分为 3 类:穷举搜索法、遗传算法和贪心法.其中:前两种方法的计算代价比较大,不适合缓冲管理等联机应用.为此,我们主要基于贪心法的应用及其改进方法.

首先看一下基于简单贪心法的算法,参见算法 1.

算法 **1**. 简单贪心算法.

输入:空间限制 S,查询模式图 $G=\{V,E\}$.

输出:满足空间限制的物化选择 M.

过程:

$M=\varnothing$;

While (S>0) Do

    Max*Benefit*=0;

    For Each $u\in V,u\notin M$ Do

        If (Max*Benefit*<$B(u,M)/u.node\_length$)

$v=u$;

Max*Benefit*=$B(u,M)/u.node\_length$;

        End

    End

$M=M\cup\{v\}$;

对所有结点 $p\in M$,如果 p 与 v 之间有 ER 边相连且 $er\_f>THEADHOLD$,则将 p 与 v 放在同一个关系中.

//因为 v 与 p 具有语义相关性,经常会同时访问,所以一起存放可以提高查询性能

//*THEADHOLD* 为用户定义的阈值

$S=S-S(v)$;

End

Return M

算法的基本思想是:在每次选入一个结点之前,都要重新计算所有尚未入选的结点的单位收益(收益/关系的体积),从中选出最好的那个.这保证了算法通常会得到比较好的结果,但缺点是算法复杂度太高.

事实上,当一个结点入选之后,不需要重新计算所有尚未入选的候选结点的单位收益,就能知道下一步中单位收益最大的候选关系是哪一个.在我们目前采用的代价模型下,物化收益是满足单调性的(随着算法的进展,单位收益率逐渐下降).其主要原因是选择了一个物化视图以后,这个视图可以用于降低某些查询的计算代价,因此会降低其他候选视图的收益.由此特性我们可以得到:随着结点的选入,未入选候选关系的物化收益只可能下降而不可能上升.

由于随着视图的选入,未入选视图的物化收益只可能下降而不可能上升,另外,因候选关系的体积不变,其单位收益也只下降而不上升.我们只要按照原来的单位收益从大到小依次检查未入选的候选关系,如果它的新单位收益没有下降,或是仍比其他视图的单位收益(新单位收益或老单位收益)大,它就必然是当前单位收益最

大的那个;否则,将新单位收益代替老单位收益,并检查下一个候选关系.实验表明:大部分情况下,只要依次检查少数几个候选关系,就可以确定下一步应该选择哪个候选关系.

算法 **2**. 改进的贪心算法.

输入:空间限制 S,查询模式图 $G=\{V,E\}$.

输出:满足空间限制的物化选择 M.

过程:

$M=\varnothing$;

计算所有结点的单位收益 $B(v,\varnothing)/S(v)$,并按从大到小的顺序排列成一个链表,在链表中同时记录结点老的收益率;

While (S>0) Do

Max*Benefit*=0;

对链表从前向后逐个计算 Do

If (Max*Benefit*<$B(u,M)/u.node\_length$)

$v=u$;

Max*Benefit*=$B(u,M)/u.node\_length$;

End

Else

修改结点 u 的收益率,并放到链表的相关位置.

If (Max*Benefit*>当前结点老的收益率)

Break;

End

$M=M\cup\{v\}$;

对于所有结点 $p\in M$,如果 p 与 v 之间有 ER 边相连且 $er\_f>THEADHOLD$,则将 p 与 v 放在同一个关系中.

//因为 v 与 p 具有语义相关性,经常会同时访问,所以放在一起可以提高查询性能

//*THEADHOLD* 为用户定义的阈值

$S=S-S(v)$;

End

Return M

改进的贪心算法有 3 个主要阶段:第 1 阶段计算所有结点的单位收益;第 2 阶段是按照单位收益进行排序,并建立降序链表;第 3 阶段是逐个选入结点,与此同时,每次选入一个结点之后,调整当前计算代价,维护链表,使链表从头到尾各结点的单位收益保持降序,直到当前的收益率小于最大收益率为止.由于改进的贪心算法只是减少了每个候选物化视图收益的计算代价,对收益的值没有影响,所以不会改变简单贪心法对物化视图的挑选顺序.所以,改进的贪心算法的计算结果与简单贪心法是完全一致的.

7 实验及结果分析

为了检验改进的贪心算法的性能,我们用模拟测试程序进行了实验.实验之前,采用 IBM 提供的 XML 数据生成模拟器产生实验所需要的 XML 数据集,并随机生成查询集合.

实验环境为配置赛扬 1G,256 兆内存的计算机,操作系统为 Windows 2000.

实验 **1**. XML 数据模式不同的结点数对算法开销的影响.

本实验的主要参数如下:结点总数变化范围是 300~1000;物化比例(P)为 0.5.实验结果如图 4 所示.由图 4 我们发现:随着结点总数的增加,简单贪心算法的时间开销随着总字段数的增加而明显增大;而改进的贪心算法的时间开销则增加极少.所以,改进的贪心法的性能较好.

实验 **2**. 不同物化比例对计算代价和算法开销的影响.

在本实验中,我们主要研究在不同物化比例(物化视图空间/原始数据集)下,两种算法的表现.主要实验参数如下:结点总数为 500;物化比例(P)的变化范围是 0.1~0.7.实验结果如图 5 所示.从图 5 中可以看到,优化算法的执行时间明显低于简单的贪心法,而且增长速度基本上是呈线性的.

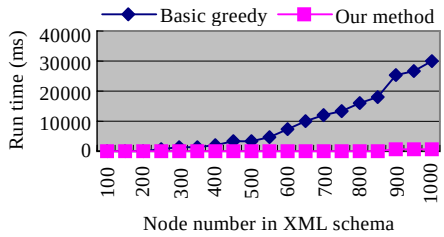


Fig.4 The run time on different schema sizes

图 4 不同模式规模下计算代价的比较

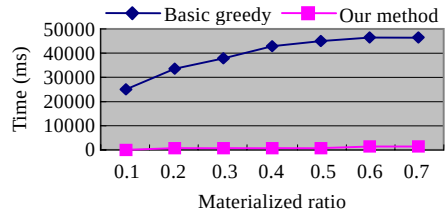


Fig.5 The run time on different materialized ratios

图 5 不同的物化比例下计算量的比较

由于在建立查询模式图期间已经将查询的相关信息存储到查询模式图中,所以查询集合的大小对算法没有明显影响.

实验 **3**. 物化对查询性能的影响.

在本实验中,我们针对 XML 数据的模式随机生成了 5 个查询集合 $Q1$~$Q5$,其中的查询数目分别为 5,15,25,30,8.在实验中,物化的空间为原始空间的 40%~50%.实验结果如图 6 所示.实验说明,物化后查询时间与物化前相比有一定的提高.

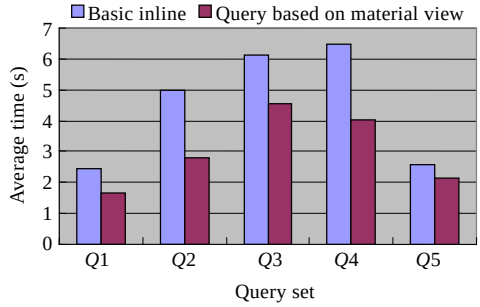


Fig.6 Query time gain after materialization

图 6 物化后查询时间的改进

8 结 论

本文提出了一种基于查询优化思想的 XML 物化方案生成技术.该方法可以应用到不同的 XML 查询方法中.今后,我们将对基于这种思想的 XML 查询引擎技术进行研究.

References:

[1] Extensible markup language (XML) 1.0. 4th ed., W3C, 2006. http://www.w3.org/TR/xml

[2] Al-Khalifa S, Jagadish HV, Koudas N, Patel JM, Srivastava D, Wu YQ. 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 Computer Society, 2002. 141−154.

[3] Deutsch A, Fernandez M, Suciu D. Storing semistructured data with STORED. In: Davidson SB, Faloutsos C, eds. Proc. of the 1999 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1999. 431−442.

[4] Shanmugasundaram J, Tufte K, He G, Zhang C, de Witt D, Naughton J. Relational databases for querying XML documents:

Limitations and opportunities. In: Atkinson MP, Orlowska ME, Valduriez P, Zdonik SB, Brodie ML, eds. Proc. of the 25th Int'l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 1999. 302−314.

[5] Abiteboul S. Querying semi-structured data. In: Afrati FN, Kolaitis PG, eds. Proc. of the Int'l Conf. on Database Theory. LNCS 1186, London: Springer-Verlag, 1997. 1−18.

[6] Tian F, DeWitt DJ, Chen JJ, Zhang C. The design and performance evaluation of alternative XML storage strategies. ACM SIGMOD Record, 2002,31(1):5−10.

[7] Gupta H. Selection of views to materialize in a data warehouse. In: Goos G, Hartmanis J, Leeuweneds JV, eds. Proc. of the 6th Int'l Conf. on Database Theory. LNCS 1186, Heidelberg: Springer-Verlag, 1997. 98−112.

[8] Arms WY, Write; Shi BL, Zhang L, Wang W, *et al*., Trans. Digital Libraries. Beijing: Publishing House of Electronics Industry, 2001 (in Chinese).

[9] Yang LH, Lee ML, Hsu W. Efficient mining of XML query patterns for caching. 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. San Francisco: Morgan Kaufmann Publishers, 2003. 69−80.

[10] Dong X, Halevy AY, Tatarinov I. Containment of nested XML queries. In: Nascimento MA, Ozsu MT, Kossmann D, Miller RJ, Blakeley JA, Schiefer KB, eds. Proc. of the 30th Int'l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 2004. 132−143.

[11] Lu Y, Zhang L, Wang W, Duan QY, Shi BL. DTD ranking in the smart XML query. Journal of Computer Research and Development, 2003,40(11):1579−1585 (in Chinese with English abstract).

[12] Jagadish HV, Lakshmanan LVS. TAX: A tree algebra for XML. In: Goos G, Hartmanis J, Leeuwen JV, eds. Proc. of the 8th Int'l Workshop on Databases and Programming Languages LNCS 2397, Heidelberg: Springer Berlin, 2002. 149−164.

[13] Shukla A, Deshpande PM, Naughton JF. Materialized view selection for multidimensional datasets. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 1998. 488−499.

附中文参考文献:

[8] Arms WY,著;施伯乐,张亮,汪卫,等,译.数字图书馆概论.北京:电子工业出版社,2001.

[11] 路燕,张亮,汪卫,段起阳,施伯乐.XML 查询中 DTD 的排序技术.计算机研究与发展,2003,40(11):1579−1585.

张亮(1963－),男,湖北武汉人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数字图书馆,多媒体技术,信息集成,生物信息学.

汪卫(1970－),男,博士,教授,博士生导师,主要研究领域为数据库原理,复杂结构数据管理,数据挖掘.

李然(1973－),男,硕士,主要研究领域为数字图书馆,XML 管理.

施伯乐(1936－),男,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论与应用,数字图书馆,数据仓库,数据挖掘.