XML 查询优化模型 XQO 的研究设计

范新灿
FAN Xin-can

深圳职业技术学院 电信学院,广东 深圳 518055
Electronics & Information Engineering,Shenzhen Polytechnic,Shenzhen,Guangdong 518055,China

FAN Xin-can.Research and design of optimization model XQO based on XML query.Computer Engineering and Applications,2009,45(19):120-122.

Abstract: XML query optimization has become a hot and difficult research topic.This paper presents a model of XQO for XML query optimization,which combine the query processing,illustrating and designing XQO from several optimizing query parse: Logical optimization、physical optimization,query execution.At last,through the experimental data to validate XQO,which can effectively reduce the cost of practical execution and improve querying speed.
Key words: XML;query algebra;query tree;regular path express

摘 要:XML 现有的查询技术不够成熟,效率低下,精确度不高,如何优化查询成为业界热点和难点问题。结合当今查询优化算法技术,设计了一个查询优化模型 XQO,从查询过程的各个阶段进行优化查询解析、逻辑优化、物理优化,设计执行策略和算法,并从实验结果验证优化的效果。
关键词:XML;查询代数;查询树;路径表达式
DOI:10.3778/j.issn.1002-8331.2009.19.036 **文章编号**:1002-8331(2009)19-0120-03 **文献标识码**:A **中图分类号**:TP311

1 引言

作为现今网络信息描述和信息交换的标准,XML 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,当 XML 数据以文档进行存储时,常使用关键字进行查询。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。与传统的查询需求相比,现有的 XML 数据处理通常把 XML 查询转变为数据库查询表达,由查询优化器优化查询并执行,再将查询的结果转变为 XML 数据,但这种多级转换造成效率的降低和查询语义的混淆。XML 查询具有以下特点:

(1)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;

(2)查询表达式中加入了编程语言的嵌套和条件判断思想;

(3)路径中包含不确定因素;

(4)查询对象和返回结果类型不确定。

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 XML 的热点研究方向。结合 XML 查询优化的最新研究,设计了一个 XML 查询优化模型:XQO(XML Query Optimization),结合 XML 查询的过程,对于每个查询阶段设计优化策略和算法进行优化。

2 XQO 模型结构设计

在 XQO 模型中,将 XML 查询优化过程分为 4 个步骤:查询解析、逻辑优化、物理优化和查询执行,查询优化过程的模型及采用的相应优化技术如图 1 所示。

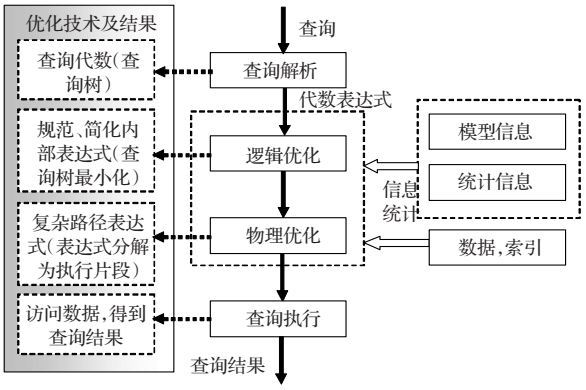

图 1 XQO 查询优化处理

查询解析阶段,将 XML 查询转换为某种内部表达式,即 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,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,并支持对查询结果的重构。此阶段以一种抽象语法树或者查询树的形式出现,为下一步的优化过程铺平道路。

逻辑优化阶段,利用模式信息,规范和简化内部表达式。对于同一查询请求,转换成不同的等价表达方式。XML 代数表达式求解时,操作顺序是影响效率的关键,逻辑优化的重点在于对操作顺序的调整。在查询解析阶段,将路径表达式转换成查

作者简介:范新灿(1978-),男,讲师,研究方向:XML、Web 服务。
收稿日期:2008-04-17 **修回日期**:2008-08-04

询树,查询效率依赖于查询树的规模,查询树的最小化是 XML 逻辑优化的重点。

物理优化阶段,利用代价模型和统计信息计算不同的表达式、不同算法的执行代价,选择最低代价的查询计划。确定表达式的执行顺序和决定每步操作的具体算法。对逻辑优化最小化的查询树,将需要查询的表达式分解为可执行的片段,选择合适的执行顺序和执行算法并执行。确定执行顺序的主要因素是中间结果集的大小,采用并优化复杂路径表达式选择性分析。

查询执行阶段,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。

3 XML 查询代数优化

3.1 代数定义

路径表达式是目前各种 XML 查询语言和过滤语言的核心,这主要是根据 XML 文档具有树状或者图状的逻辑结构决定的。XQO 首先定义一个代数模型,在这个模型的基础上给出路径表达式的语法定义以及它的有限自动机表示方法。

与关系代数类似,XML 查询代数也需要定义一系列的操作符:选择、抽取、序列、连接、分组、聚集、排序、消重、构造。这些操作符可以分为 3 类:(1)结构相关:抽取(模式树匹配)、连接、构造;(2)数值相关:选择(谓词)、分组、聚集、消重、排序;(3)混合:集合、序列。下面列举出各操作符的含义:

选择操作符:$\sigma P, PE(X)=\{x|x<X, PE(x)\}$

抽取操作符:$EPi, Po(X)$

构造操作符:$\chi Pi, Po(X)$

连接操作符:$\flat$ ü $P1, P2, C(X1, X2)=\{j(x1, x2)|X1, x2 \in X2, C(x1, x2)\}$

分组操作符:$\gamma P, f(X)=\{g(v, r(x1, x2, \cdots, xn))|xi \in X, v=f(xi)\}$

序列操作符:$\xi Pi, (op1, op2, \cdots, opn)(X)=\{s(op1(x), op2(x), \cdots, opn(x))|x \in X\}$

聚集操作符:$AP, f(X)=\{a(v, x)|x \in X, v=f(X)\}$,$f$:聚集函数,如 sum、count、average、low、high 等。

排序操作符:$\tau P, f(X)$

消重操作符:$\delta P, f(X)=\{x|x \in X, Pxi \in \delta(X), \text{if}(xi \neq x)]f(xi\} \neq f(x)\}$

集合操作符:$X1 \cup X2=\{x|x \in X1 \vee x \in X2\}$,

$X1 \cup X2=\{x|x \in X1 \wedge x \in X2\}$,

$X1=\{x|x|vX, x \in X \wedge x \beta X1\}$

为了建立 XML 完整的数学模型,需要增加节点类型:root、element、text、comment、PI、CDATASection、attribute 和相应的判断函数:root()、element()、text()、comment()、PI()、ATASection()、attribute();值的数据类型:int、double、char、string、date 和相应的函数:substr()、left()、right()、now()等。运算符:+,一 r*r/;modo 关系运算符:<,<=;>=r>r 二 r!=。数学函数:指数、对数、幂函数、三角函数等。

3.2 路径表达式

一个 XML 文档 XTree 模型示意如图 2 所示。路径表达式是 XML 查询的基本建筑单元。正则路径表达式,是指能够以 Like-SQL 语言的方法表示查询的内容,但却可以结合一些特殊字元或一些特殊符号,来描述 XML 文件的阶层式及树状资料内容,如树状结构子孙关系的描述等。在利用正则路径表达式来下达一些 like-SQL 的指令时,不一定要完全知道目前所查询的 XML 文件资料的全貌,只要针对所查询的资料条件,加以利用正则路径表达式的一些符号或特殊字元来与条件结合描述,就可以轻易的查询所要寻找的 XML 文件内容。

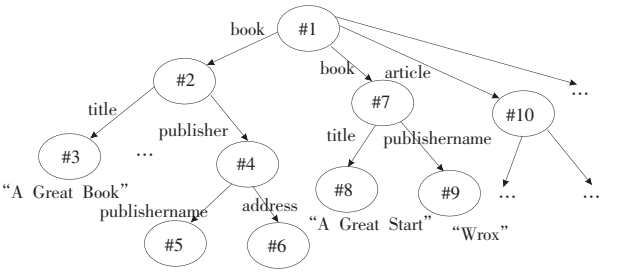

图 2　一个 XML 文档 XTree 模型示意图

针对模型 XTree,路径表达式的形式化定义[3]如下:

(1){}表示一棵空树;

(2){l=>T}表示一棵树,它的根有一条标记为 l 的输出边,下面附着一棵子树 T;其中 T 可以表示叶节点,此时 T 等于某文字值或空元素。由于{l=>T}只表示单一一棵树,可省略括号,简写为 l=>T;

(3){l_1=>T_1, l_2=>$T_2 \cdots$, l_n=>T_n}也表示一棵树,它有 n 个由同一节点发出的、分别标记为 $l_1, l_2, \cdots, l_n$ 的输出边,每个输出边分别附着子树 $T_1, T_2, \cdots, T_n$;

(4)无论是边标签 l,还是子树 T_i(包括叶节点),都可以用变量表示;

(5)2,3 或 2,3 的组合成为路径表达式。

目前的 XML 查询语言如 XPath、XQuery 等,这些查询语言都是基于正则路径表达式。路径表达式分解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。路径表达式的分解释查询优化重要的环节。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。

Lore[5]是 Stanford 大学的研究者提出的能有效存储和查询 XML 数据的系统,在系统中为了提高查询效率设计了 4 种不同的索引结构,分别为 value index、text index、link index 和 path index。value index 是针对 PCDATA 为原子对象建立的,利用 PCDATA 作为搜索键值,建立对应的 B^{+2} Tree 结构。text index 则是针对 PCDATA 为 text 对象建立的,利用 inverted list 结构,对 PCDATA 中较为重要的关键字作索引,并传回所对应的标签名称。link index 则是为了解决在 OEM 图中不支持逆向指针问题,利用 hashing 的方法来记录所有标签名称的父标签。最后,path index 则是记录了部分重要的路径结构及其在 OEM 图的查询结果的对应关系。用户在查询时,系统动态地选择所需要的索引结构,来完成查询。

Lore 只能针对简单路径进行查询计算,用户使用正则路径表达式进行查询时,系统选择索引通常需要同时使用多个索引结构,当正则路径表达式相当复杂时,将会增加查询的负担。可针对正则路径表达式查询来设计索引机制 DataGuid,利用有限状态自动机的技术来解决正则路径表达式查询的问题,减少了使用正则路径表达式必须大量地检查文件中所有可能的路径,达到加快查询的目的。针对 DataGuide 只能支持单个正则路径

表达式查询的局限性进行了改进，提出了一种索引机制 B+2 index。此索引机制通过所定义的路径模板(path template)，利用等价关系将 XML 数据包含的对象分成若干组，再通过构建有限状态自动机来实现，即通过定义路径模板来实现对复杂的查询建立索引。

4　XML 查询树的最小化

查询树的最小化是逻辑优化的研究重点。XML 查询树的优化从语法层次和语义层次两个层次。XQuery 查询表达式如下：

```
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(PTQ)如图 3(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，称这种冗余结点为语法冗余结点。经语法优化后的 PTQ。

如图 3(b)所示，若从模式可知任一 author 均有 name，则 $v6$ 为冗余结点，称这种冗余结点为语义冗余结点，经语义优化后的 PTQ 如图 3(c)所示。

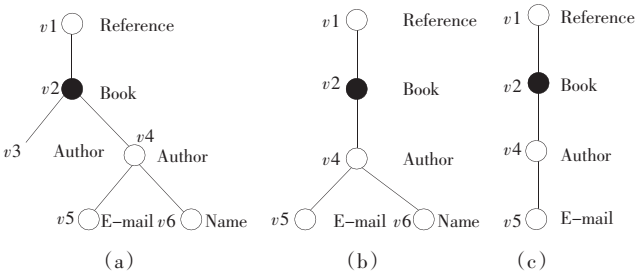

图 3　查询树的语法优化和语义优化过程

5　复杂路径表达式选择性分析

本文设计的 XQO 查询优化模型经过逻辑优化后，结果是一个或多个查询树。下一步的操作需要确定不同查询片段的执行次序，即中间结果集的大小。采用复杂路径表达式的选择性分析来估计结果集规模。XQL 查询优化器通过统计 XML 数据分布情况，基于一定假设估计路径目标结点的中间结果集的大小。

需要进行代价模型[4]的定义：$wkld=\{(Q_i,w_i),i=1,2,\cdots n\}$，这里 Q_i 是一 XML 查询，w_i 是其权重，反映它在负载 $wkld$ 中的重要程度。权重高说明该查询经常发生，或执行的优先级较高。在不同的索引模式下，查询负载的查询代价是不同的。查询负载代价等于在一定索引模式 S 下，查询负载 $wkld$ 中各查询代价与其权重乘积的和，计算公式如下：

$$cost(wkld,S)=\sum_{i=1}^{n} w_i*cost(Q_i,S)$$

这里 S 是索引模式，即路径索引的集合，其初始值为 ϕ。给定索引存储空间限制和查询负载 $wkld$，选择最优索引模式 S，在不超过存储空间限制的前提下，使得 cost($wkld$,S)最小。要找到这个最优解，如果从 n 个路径索引中选择索引模式，就要搜索 $2n$ 个候选索引模式，这是一个 NP 难问题。因此采用贪心算法来找到一个近最优的索引模式(NO IS)，在一定的磁盘空间约束下，该索引模式可最大限度地提高给定查询负载的查询效率。并不关心 $cost(wkld,S)$的精确计算，因为只需要比较不同索引模式(S_1 和 S_2)下查询负载的查询代价($cost(wkld,S_1)$和 $cost(wkld,S_2)$)的差异。若 $cost(wkld,S_1)$小于 $cost(wkld,S_2)$，则表明索引模式 S_1 比 S_2 带来的查询收益大。索引模式 S 的收益计算公式如下：

$$bf(S)=cost(wkld,\phi)-cost(wkld,S)$$

更进一步的计算公式为：

$$bf(S)=\sum_{i=1}^{n} w_i*bf_{ind}^{i}(S)$$

其中 bf_{ind}^{i} 是使用索引模式 S 后查询 Q_i 所产生的查询收益，它的定义如下：

$$bf_{ind}^{i}(S)=\sum_{Q_{mat}^{j}\text{是}Q_i\text{的一部分}} db.cost(Q_{mat}^{j})$$

Q_{mat}^{j} 是指为查询 Q_i 中的任一路径查询构建路径索引的 SQL 语句，$db.cost(Q_{mat}^{j})$是指执行 Q 所需的查询代价，该代价由 RDBMS 的优化器估算（大多数商业化 RDBMS 都提供对此统计功能的支持，如 DB2）。$db_size(Q)$表示执行查询 Q 所得结果所占的存储空间。这样在执行查询 Q 时，就可用路径索引取代原来的 Q_{mat}^{j} 部分的查询，降低 Q 的查询代价。

通过以上公式计算路径索引的代价，进行设计基于代价的路径索引选择构建算法。XML 代价估计中，路径表达式选择性代价估计是核心问题，可将其视为一棵树，其中一个分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，表示为：

$$P=t_1[p_1]/t_2[p_2]/\cdots/t_n[p_n]$$

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。

6　XQO 查询优化实验

基于 XQO 模型优化查询机制，对使用仅 XQuery 语言和结合 XQO 模型进行优化后的 XML 文档进行查询测试比较。测试平台为 P4 2.8 GHz 处理器，512 M 内存，Windows 2000 操作，JDK1.5 编程进行具有大量信息的 XML 文档查询。

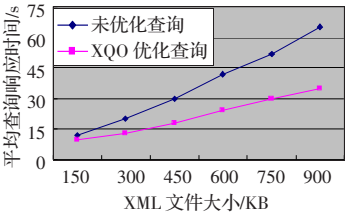

图 4　XQO 查询优化实验结果

选取不同大小的 XML 文档，对比采用 XQO 优化模型和只采用 XQuery 查询语言进行查询。在给定代价参数后，从图中可

(下转 133 页)