一种基于代价的 XML 查询优化操作模型

黄寿孟
（三亚学院计算机教学部 三亚 572022）

摘 要 随着 XML 数据库技术研究的深入，关于 XML 查询优化的研究日益增多，但至今其仍是 XML 数据库的薄弱环节。从传统的查询估算模型中找出原子操作，在物理优化时通过分析估算操作，采用基于统计学习的方法找出操作代价和这些影响因素之间的函数关系，从而建立起基于代价的操作模型。

关键词 XML 查询优化，基于代价，操作模型，算法分析
中图法分类号 TP393 **文献标识码** A

Query Optimized Operation Model of XML Based on Cost

HUANG Shou-meng
(Department of Computer Instruction, Sanya University, Sanya 572022, China)

Abstract Along with the development of XML database technology, the research on XML query optimization is increasing, but it is still the weak link of XML database. In this paper, we found out the atomic operation in the traditional query estimation model, and used the method based on statistical learning to find out the operating costs and the functional relationship between these factors, so as to establish the operation model based on cost.

Keywords XML query optimization, Cost-based, Operation model, Algorithm analysis

XML 已成为互联网中数据存储和共享的事实标准。利用数据库技术对 XML 数据进行管理主要有两种方法：1）支持 XML 的数据库，此种类型的数据库通过扩展传统关系数据库或对象数据库，使之能够支持 XML 数据的处理，其优点在于可以充分利用已有的关系数据库技术，但不能利用 XML 数据自身的特点来对 XML 数据进行处理，且查询过程的多重转换可能会导致效率低下；2）为 XML 数据量身定做的原生 XML 数据库，它能充分利用 XML 数据自身的特性，克服支持 XML 的数据库的效率低下和信息丢失等问题。由于 XML 数据的层次结构等特点，在关系数据库领域已相当成熟的查询优化、存储、事务管理等技术无法直接移植到 XML 数据库当中。因此，原生 XML 数据库成为数据库领域的研究热点[1]。查询处理是数据库系统最重要的功能之一，而查询优化技术对查询处理的效率有着至关重要的影响。

1 基于代价查询优化的相关模型

一个 XML 文档含有元素（属性）标记（tag）、元素（属性）值，但是元素之间、值之间以及元素与值之间往往都有关联，那么，维护哪些数据的统计信息呢？若单纯依赖统计信息往往不可能给出精确的估算结果，因此需要在统计信息的基础上，根据某种估算法则估算出最终的选择度，那么估算法则如何确定，其误差又是怎样的呢？

在 XML 查询中，一个给定的查询或逻辑查询计划在转换为物理查询计划时往往有多种选择，通常要对多个物理查询计划进行评价，选择最好的物理查询计划，这种评价往往都是基于代价的。每个物理查询计划的真实代价在执行之前是不可能精确知道的，因此需要对这些代价进行估算。在估算物理查询计划的代价时，主要考虑如下因素[2]：1）每个物理操作的代价，即每个物理操作本身的开销；2）操作结果的大小，如操作返回的元组或者元素数目等；3）连接顺序，对于多个连接操作而言，不同的连接顺序对执行性能有很大的影响；4）其他，如结果是否有序、是否采用流水线等。

在这几个问题中，物理操作的代价与相应的算法有很大关系，也与缓冲区大小、操作结果大小等因素有关，其的估算与具体操作有关，其本身会影响操作代价，而且直接决定了下一步操作的输入，因此又影响了下一步操作的开销。另外，操作结果的大小也对其他优化技术（如是否采用流水线技术）有影响。对一个网络环境或交互环境下的查询来说，操作结果大小的估算可以及时给出一个近似的响应，这种响应可以缩短用户等待的时间，而且可以帮助用户进行进一步优化查询。下面给出传统估算模型。

1.1 简单路径表达式的选择度估算模型

路径表达式选择度估算一般是指路径表达式求解的结果大小估算。此类模型共有 10 种选择度估算方法：statiX，Lore，CST，Path Tree，Markov Table，Markov Histogram，XSketch，XSketches，Bloom，XSeed。这些选择度估算方法都只能估算单个路径表达式或不含分支的路径表达式，也可以允许路径表达式中出现简单分支或值谓词。

1.2 小枝查询的估算模型

小枝查询往往是由多条路径构成的，而且小枝查询的返回结点可能有多个，因而小枝查询选择度的估算更复杂。常用的有Twig-XSketch 概要图估算[3]、TreeSKetch 概要结构

本文受海南省教育厅项目：面向多媒体的高速率无线传输技术研究（Hnky2015-55）资助。
黄寿孟（1975－），男，硕士，副教授，主要研究方向为现代教育技术、计算机基础教学，E-mail：huang123888@126.com。

估算[4]、TreeLattice 分解的估算模型[5,6]、XCluster 概要结构模型[7]、SLT 树概要结构模型[8]。

1.3 结构连接结果大小估算模型

XML 的特殊结构(如嵌套性、相关性等)决定了查询新方法:位置直方图[9]、区间模型和位置模型[9]、值-位置直方图[10]。在复杂路径表达式中,经常存在含值的谓词,值-位置直方图通过维持值的分布与结点位置分布之间的关系,能够用于这种估算,再配合位置直方图或区间模型等方案,能够对大多数路径表达式进行估算。

2 操作代价模型

以上估算模型研究主要是对查询结果大小进行估算,如路径表达式的执行结果、小枝查询的结果等。传统数据库广泛采用了基于代价的查询优化,其中,物理查询计划的选择依赖于每个查询操作的代价。传统的查询往往分解为一系列的原子操作,每种原子操作的代价模型都是事先已知的,因而在物理优化时可以通过枚举或者启发式的方法找出最佳的原子操作序列,也即查询计划。在 XML 数据库中,基于代价的查询优化比关系数据库要难得多,主要是因为原子操作的复杂性使得为这些操作构造代价模型相当困难。

在关系数据库中,代价模型主要是在分析的基础上建立的,如通过对算法或者原理的分析而建立起模型。但对 XML 数据库而言,通过分析来建立模型并不容易。因此代价建模采用了基于统计学习的方法,其基本思想是:首先找出影响操作代价的一些查询和数据特征,然后通过对训练数据的学习,找出操作代价和这些影响因素之间的函数关系,从而建立起操作的代价模型[11]。具体步骤如下:

(1)找出决定操作代价的算法、查询以及数据方面的特征,这一步需要对操作进行分析。

(2)可以想象,这些影响因素大都是"后验"因素,即查询执行后才能确定其值,比如返回结果大小或者扫描元素数目等,而代价模型需要在事先确定这些值才能计算代价,因此需要事先估算这些特征值,这期间需要用到一些统计和分析方法。

(3)利用某种统计或机器学习的方法来找出特征值和代价之间的函数关系,从而建立起代价模型。

(4)在查询优化中应用该代价模型,并通过一些自调节过程来改进。

现在通过一个实例介绍上述过程。其中,考察的对象是一种称为 XN_{AV} 的操作,而代价则主要考虑 CPU 代价。XN_{AV} 操作是一种导航式的操作,是由 TURBOXPath[12] 算法改编而来,给定一个 XPath 表达式,返回与之匹配的 XML 树结点的引用。其算法的操作描述如下。

算法 XN_{AV}(P:解析树,X:XML 文档)

```
//P 是与 XPath 表达式对应的解析树
1. match_buf←{root of P};
2. while (not end-of-document) do
3.    x←next event from traversal of X;
4.    if (x is a startElement event for XML node y)
5.       if (y matches some r∈match_buf;
6. set r's status to true;
7. if (r is a non-leaf)
8.             set r children's status to false;
9.             add r's children to match_buf;
10.         if (r is an output node)
11.            add r to out_buf;
12.         if (r is a predicate tree node)
13.            add r to pred_buf;
14.   else if (no r∈match_buf is connected by //-axis)
15.   skip through X to y's following sibling;
16. else if (x is endElement event for XML node y)
17.   if (y matches r∈match_buf)
18.         remove r from match_buf;
19.   if (y is in pred_buf)
20.         set y's status to the result of evaluating the predicate;
21.      if (the status of y or one of its children is false)
22.         remove y from out_buf;
```

3 代价模型估算分析

按照操作代价建模方法的 4 个步骤以 XN_{AV} 算法操作符为例分析全过程。

3.1 特征识别

XN_{AV} 算法用到了 3 个缓冲区:输出缓冲区 output_buf、谓词缓冲区 pred_buf 和正在匹配的缓冲区 match_buf。这些缓冲区中的元素数目很明显会影响操作代价,将这 3 个特征变量记为 # out_bufs, # pred_bufs 和 # match_bufs。该操作访问过的结点数目也是一个特征,记为 # visits;此外还有操作返回的元素数目,记为 # results。每当操作发出一个页面请求时,CPU 就会查找页面缓冲区,引起相应代价,因而页面请求数目也是一个特征,记为 # p_requests。最后还有一部分代价是在第 18,20 和 22 行处的动作所带来的代价,这些语句因为遇到 endElement 事件而引起,所以激发这些动作的 endElement 事件的数目也是一个特征,记为 # post_process。

3.2 特征值估算

通过一种比较简单的方法来维护统计信息:维护所有简单路径的统计信息。简单路径是只带有孩子轴或者是孩子轴和结点测试构成的路径,实际就是标记路径,其为每一个简单路径 p 维护如下统计信息。

(1)基数:即文档树中与 p 匹配的结点数目,记为 $|p|$。

(2)孩子数目:即 $|p/*|$。

(3)后裔数:即 $|p//*|$。

(4)页基数:为了响应 p 所需的页面数,记为 $\|p\|$。

(5)后裔页基数:为了响应 $p//*$ 所需的页面数,记为 $\|p//*\|$。

即每个简单路径 p 的统计信息可以用一个五元组表示,所有简单路径可以通过一个路径树组、TreeSketch 或者表组织在一起。

这些简单路径的统计信息旨在估算各个特征变量,下面列出了估算各个特征变量的算法,这些算法都是根据特征变量的含义而设计,大多比较直观。

估算算法:

Visits(proot:解析树根结点)

```
1. v←0;
2. for (each non-leaf node n in depth-first order)
3.    do p←the path from proot to n;
4. if (p is a simple path(i.e.,no //-axis))
5. if (one of n's children is connected by //-axis)
6.          v←v+|p//*|;
7. skip n's descendants in the traversal;
```

```
8. else v←v+|p/*|;
9. return v;
```

pages(proot:解析树根结点)

```
1. p←0;R←Ø;
2.   L←list of all root-to-leaf paths in depth-first order;
3.   for (every pair of consecutive paths l_i,l_{i+1}∈L)
4. do add common subpath between l_i and l_{i+1} to R;
5. for (each l∈L)
6. do p←p+‖l‖;
7. for (each r∈R)
8.   do p←p-‖r‖;
9. return p;
```

Buf-Inserts(p:线性路径)

```
1. If (p is not recursive)
2. return |p|;
3.   else m←0;
4. for (each recursive node u such that p=l//u)
5. do m←m+∑_{i=1}^{d}|{l//u}*i|
6. return m;
```

Results(proot:解析树根结点)

```
1. t←the trunk in proot
2. Return |t|;
```

Match-Buffers(proot:解析树根结点)

```
1. m←0;
2. for (each non-leaf node n)
3.   do p←the path from proot to n;
4.     m←m+Buf-Inserts(p)×fanout(n);
5. return m;
```

Pred-Buffers(proot:解析树根结点)

```
1. r←0;
2. for (each predicate-tree node n)
3. do p←the path from proot to n;
4. r←r+buf-Inserts(p);
5. return r;
```

Out-Buffers(proot:解析树根结点)

```
1. t←the trunk in proot
2. return Buf-Inserts(t);
```

Post-Precess(proot:解析树根结点)

```
1. L←all possible paths in the parse tree rooted at proot;
2. n←0;
3. for (each l∈L)
4. do n←n+Buf-Inserts(l);
5. return n;
```

在上述算法中，Results 函数返回的是 trunk 的基数，即去掉解析树中的分支后得到的树。在 Pages 函数中隐含了如下假设：访问结点 x 时的页面保留在缓冲区中直到 x 的所有后裔访问完毕为止。在 Buf-Inserts 函数中，$l\{//u\}*i$ 表示 l 后面跟上 i 个 $//u$ 的串接，如 $i=2$ 时，就表示 $l//u//u$，这里 u 是对应的一个递归结点。因为每当解析树中的一个结点 r 得到匹配时，如果 r 不是叶结点，就把 r 的孩子结点加入 match_buf 中(见 XN_{AV} 算法第 5 和 7－9 行)，因此在 Match-Buffers 函数中，用到了 fanout 函数，fanout(u)表示 u 的孩子结点数目。

3.3 统计学习

这一步骤的任务是对查询 q 确定一个函数 f，使得 $cost(q)=f(v_1,v_2,\cdots,v_d)$，其中$(v_1,v_2,\cdots,v_d)$表示 d 个特征值。采取有指导的训练，即训练数据由 n 个点 $x_1,x_2,\cdots,x_d,\cdots$构成，每个点 $x_i=(v_{1,i},v_{2,i},\cdots,v_{d,i},c_i)$，$l\leqslant i\leqslant n$，其中 $v_{j,i}$ 表示第 i 份测试数据的第 j 个特征值，在训练时，这些特征值可以是估算值也可以是观测值。C_i 表示观察到的第 i 个查询代价。

结束语　XML 查询优化技术有其自身的复杂性，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[13]，这使得 XML 查询优化尤其困难。在很多应用中，经常需要在查询之前对查询结果进行估算，这种估算一般要求尽可能及时，估算值可以用于查询优化和初步响应等。此操作代价估算模型基于 TURBOXPath 算法，改编并优化操作代价，从而演化成估算模型，是一种导航式的操作模型，估算与之匹配的 XML 树结点的引用统计，列出估算各个特征变量的代价模型，从而提高查询优化的效率。

参 考 文 献

[1] 梁晓翀. 基于代价估算的 XPath 查询优化[D]. 广州：华南理工大学，2012

[2] 万常选，刘喜平. XML 数据库技术[M]. 北京：清华大学出版社，2008

[3] Polyzotis N，Garofalakis M N，Ioannidis Y E. Selectivity Estimation for XML Twigs[C]// Proceedings of the 20th International Conference on Data Engineering. Boston，MA，USA. IEEE Computer Society，2004：264-275

[4] Polyzotis N，Garofalakis M N，Ioannidis Y E. Approximate XML Query Answers[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. Paris，France. ACM Press，2004：263-274

[5] Wang C，Parthasarathy S，Jin R. A Decomposition-Based Probabilistic Framework for Estimating the Selectivity of XML Twig Queries[C]// Proceedings of the 10th International Conference on Extending Database Technology. Munich，Germany. Lecture Notes in Computer Science vol. 3896，Springer，2006：533-551

[6] Wang C，Parthasarathy S，Jin R. A Decomposition-Based Probabilistic Framework for Estimating the Selectivity of XML Twig Queries[R]. Technical Report，2005

[7] Polyzotis N，Garofalakis M N. XCluster Synopses for Structured XML Content[C]// Proceedings of the 22nd International Conference on Data Engineering. Atlanta，GA，USA. IEEE Computer Society，2006，63

[8] Fisher D K，Maneth S. Structural Selectivity Estimation for XML Documents[C]// Proceedings of the 23nd International Conference on Data Engineering. Istanbul，Turkey. IEEE Computer Society，2007：626-635

[9] Wang W，Jiang H，Lu H，et al. Containment Join Size Estimation：Models and Methods[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. San Diego，California. ACM Press，2003：145-156

[10] 王宇，孟小峰，王珊. OrientX 中的统计信息收集方法研究[C]// 第 20 届全国数据库学术会议论文集. 长沙，2003：94-98

[11] Zhang N，Haas P J，Josifovski V，et al. Statistical Learning Techniques for Costing XML Queries[C]// Proceedings of the 31st International Conference on Very Large Data Bases. Trondheim，Norway. ACM Press，2005：289-300

[12] Josifovski V，Fontoura M，Barta A. Querying XML Streams[J]. VLDB Journal，2005，14(2)：197-210

[13] 孟小峰，王宇，王小锋. XML 查询优化研究[J]. 软件学报，2006，17(10)：2069-208