计算机科学2004Vol.31№.9A

# XML 文档结构索引的更新维护

郭启晶　洪晓光

（山东大学计算机科学与技术学院　济南250061）

**摘　要**　XML 作为一种数据表示方式，越来越为人们所接受。最近，基于 XML/半结构化数据的索引的查询引起了人们的广泛关注。有人提出来用结构索引来支持基于 XML/半结构化数据的查询。由于 XML/半结构化数据的结构不严格、不规则，结构索引会随着数据的增加或删除而变化，维护结构索引就成了首要问题。在本文中，我们研究了在 XML 文档集合上增加多个文档和删除一个文档时结构索引的更新，提出了在这两种情况下的索引维护算法，这两种算法比现在已知的其他算法的性能要更优秀。

**关键词**　结构索引，Bisimularity 关系，划分加细，增加多个文档，删除文档

## Updates for Structure Indexes of XML Files

GUO Qi-Jing　HONG Xiao-Guang

(School of Compute Science and Technology, Shandong University, Jinan 250061)

**Abstract**　With the rapidly increasing popularity of XML for data representation, the problem of queries in XML databases has received considerable attention recently. Structure indexes have been proposed as supporting data structure for this problem. Because the structure of XML data is irregular, and will be changed with the addition or deletion of data, updates for structure indexes are the first problem to solve. In this paper, we study two kinds of updates of XML data: addition of multiple files and deletion of a file in XML database. We propose algotithms to update structure indexes for these two kinds of updates, which are better than other algorithms we have known.

**Keywords**　Structure index, Bisimularity, Refine, Add file, Delete file

## 1　背景

XML 作为一种数据存储方式，正越来越为人们所接受。而如何高效地查询 XML 文档也就引起了人们的广泛关注。为了支持对 XML 文档的路径查询，有人提出了一种新的结构索引[3]。由于 XML 所表示的这种半结构化数据的结构不严格、不规则，结构索引可能会随着数据的增加和删除而不断地变化，如何维护结构索引就成了首要问题。

当在一个 XML 文档集合上增加或者删除文档时，如果删除掉原有索引再重新建立索引的话，由于文档集合的数据量非常庞大，其代价是非常高的。在文[4]中，给出了一种在 XML 文档集合上增加一个文档时，根据原索引和建立在新增文档上的索引来计算更新后的结构索引的方法，该方法要比上面提到的方法快得多。

对实际的 XML 文档集合，可能会不断地增加文档，如果每增加一个文档就对索引进行一次更新的话，代价令人难以接受，而且对一些应用来说也是不必要的。例如一个远程监控系统，被监控系统会不断地发送文档给监控系统，监控系统在被监控系统发生某事件或者某个时间才需要查询。在本文中，我们提出了方法，在增加多个文档时，对索引进行集中维护；同时也提出了删除一个文档时索引的维护算法，该算法要比删除再重建索引要快得多。

本文的第2部分，引入了在文[3,4]中提到的 bisimularity 关系和结构索引的概念；第3部分先引入文[4]中的增加一个文档时的索引维护算法，然后提出了增加多个文档的索引维护算法；删除文档时的索引更新维护方法将在第4部分给出；最后进行了结论。

## 2　基础工作

在这里，不考虑图中树边与非树边的区别，把 XML 文档用一个有向、有标记的图来表示 G＝(VG, EG, ΣG, root, label, oid, value)。首先定义唯一的根元素 root，由 label 函数定义其标记为 ROOT，为每个 XML 文档生成一棵文档树，作为子树加到 root 节点下，然后由 oid 函数为每个节点赋予唯一的 oid 值，ΣG 为所有节点的标记的集合。文档树的生成方法如下：

·对每个 XML 元素，定义一个节点，由 label 函数定义其标记为该元素的标记；

·对每个字符数据，定义一个节点，由 label 函数

郭启晶　研究生，研究方向：数据库应用。

定义其标记为唯一的 VALUE，定义其 value 值为相应的字符数据值(只有文档树中的叶子节点有 value 值)；

·在元素与子元素、元素与字符数据节点之间添加边表示其关系。

下面是一个图书馆的例子。图1表示了一个数据图，节点左侧的数字为 oid，右侧的字符串为 label，叶节点下面的字符数据是它的 value 值(图1只表示了该数据图的一部分)。

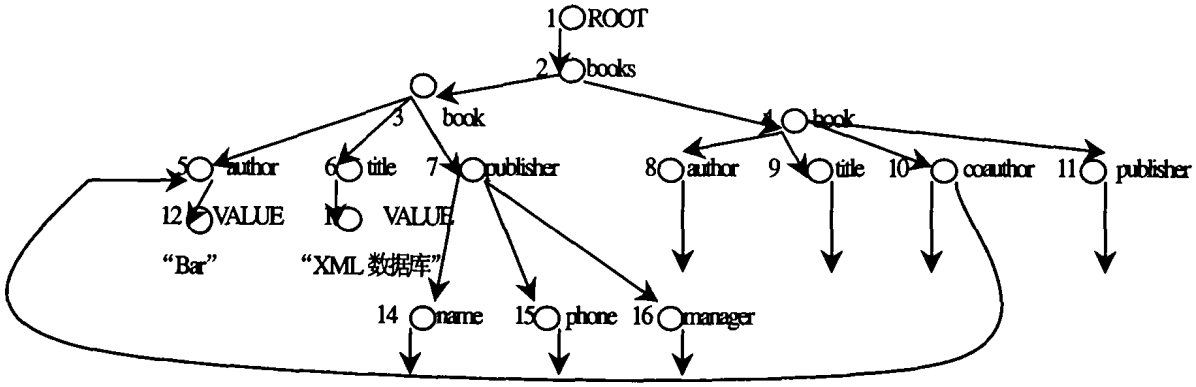


图1 一个图书馆对应的数据图的一部分

结构索引也采用了有向标记树的形式。在结构索引中，节点和边要比数据图中的节点和边少得多。它为每个节点 A 定义了 extent 来辅助查询应答，记为 ext(A)。如果 I(G)为对应于数据图的索引图，A 为 I(G)中的一个节点，则 A 的 extent，ext(A)，是 VG 的一个子集。

在本文中提到的由数据图的一个划分得到的索引图按下面的三步来构造：

1. 为每个等价类关联一个索引节点；

2. 对每个索引节点，定义其 extent 为形成该索引节点的等价类中的所有节点的集合；

3. 对任意两个索引节点 A、B，如果从 ext(A)中的某个节点到 ext(B)中的某个节点有边，那么，增加一条从 A 到 B 的边。

如果数据图节点的两个划分 P1、P2满足：任意两个数据节点在 P1中属于同一个等价类，在 P2中它们也属于同一个等价类，那么 P1叫做 P2的一个划分加细，也叫 P2的一个细分。当我们讲到一个索引图是另一个索引图的细分时，指的是生成它们的划分来说的。

下面引入图上的 bisimularity 的概念。在本文中提到的结构索引都是基于这一概念。

**定义1** 图 G 的顶点集 VG 上的二元关系 bisimularity，记为≈，对任意两个节点 u，v∈VG，如果 u=v，则 u≈v；否则 u≈v，当且仅当下面的条件成立：

1 u，v 有相同的标记；

2 对 u 的每一个父节点 u′，都存在 v 的一个父节点 v′，使得 u′≈v′，反之也成立。

图1中的节点6和9就满足这一关系，而节点5和8就不满足这一关系，因为对于节点5的父节点10，节点8不存在父节点与之满足关系。有这样一个简单的推论：如果两个节点满足 bisimularity 关系，那么这两个节点的流入路径上的节点也满足这一关系。

由上面的定义可以看出：bisimularity 关系是 VG 上的等价关系，该关系的等价类是节点集 VG 的一个划分，由前面的构造方法得到的索引图就是结构索引，称为 Bisim(G)。一般情况下，索引图的大小要小于数据图，即使在最坏的情况下，其大小也仅仅是等于数据图。

在文[3]中，Paige 和 Tarjan 提出了一种计算由图上的 bisimularity 关系定义的划分的算法(在本文中，我们称为 PT 算法)，进而得到索引图。采用适当的数据结构，该算法可以在 O(mlogn)时间内实现(这里，m 是图 G 的边数，n 是图 G 的节点数)。

## 3 增加多文档的索引维护

在一个 XML 文档集合上可能要不断地增加多个文档，下面来分析其索引的维护。

### 3.1 增加一个文档的索引维护

文[4]中提出了一种增加一个文档时的索引维护方法。对于一个 XML 文档集合，它上面已建立了结构索引。现在向文档集增加一个新文档(简单起见，假设无文档间关联)。在下面的方法中，利用原有索引和新加入文档的索引来计算修改后的结构索引，而不用删除原有索引，然后再重建索引。假设增加新文档前的数据图为 G，索引为 I(G)，新加入的文档的数据图为 H，计算得到 H 的索引图 I(H)。在原有文档集中增加一个新文档，对应于数据图上就是在原数据图的 root 节点下增加一个子图 H。把 I(H)作为子图加到 I(G)的 root 下，得到 I′，那么，I′是实际的索引 Inew 的一个细分。把 I′看作是数据图，计算它的索引，这就是 Inew。在由 I′计算 Inew 时，Inew 中节点的 extent 是 I′中节点的集合，为了

得到由数据节点组成的 extent，需要进行进一步的操作：定义 extent 为形成它的 I′中节点的 extent 的并集。本文称该方法为 add-file 算法。下面的理证明了该方法的正确性。文[4]给出了定理的证明。

**定理1** G 是一个数据图，Bisim(G)是 G 的由 bisimularity 关系定义的索引，Bisimref(G)是由 Bisim(G)的一个细分得到的索引，则 Bisim(Bisimref(G))= Bisim(G)(图相等是指同构)。

该方法的时间复杂度为 O(mHlog(nH)+(mIH+mI)log(nIH+nI))，其中 mH，mI，mIH 分别表示图 H，I，IH 的边数，nH，nI，nIH 分别表示 H，I，IH 的节点数，该时间复杂度与原数据图 G 无关。

### 3.2 增加多文档的维护方法

现在来分析在一个 XML 文档集合上不断增加文档时结构索引的维护。

由上面的索引维护方法，很自然地想到的一种方法就是每增加一个文档就对索引进行一次更新。假设 XML 文档集合对应的数据图为 G，索引图为 IG，依次增加 n 个文档，对应的数据图分别为 H1，H2，…，Hn，计算它们的索引图 IH1，IH2，…，IHn，然后由 IG 和 IH1调用 add-file 算法得到 I1′，由 I1′和 IH2计算得到 I2′，…，最后由 In-1′和 IHn 计算得到 In′。

用这种方法计算索引，要比每次都删除索引再重建索引要快得多。然而，每增加一个文档就要访问原索引图一次，虽然该索引图要比文档集合的数据图小得多，但当文档集合很大时，其索引图的大小也不得不考虑。而且对于一个实际的应用来说，每增加一个文档就对索引进行一次维护，可能是不允许的。我们提出了这样的一种思想：对一个 XML 文档集合，并不是每增加一个文档就立即对索引进行更新，而是等增加了多个文档后，选择一个无查询需求或者查询需求较少时，对文档集合的索引进行集中更新。

假设 XML 文档集合对应的数据图为 G，索引图为 IG，依次增加 n 个文档，数据图分别为 H1，H2，…，Hn，把这 n 个文档看作一个新的文档集合，求出该文档集合对应的索引图，然后把该索引图增添到原文档集合的索引中，并调用 PT 算法计算出更新后的索引。

**算法1**

```
procedure add-multifile(G,H1,H2,…,Hn)
begin
    step 1 令 G 的索引图为 IG
    step 2 把 H1,H2,…,Hn 看成是一个新的文档集合，计
           算出其索引图，令其为 IH
    step 3 把 IH 的 root 节点与 IG 的 root 节点合并，得到新
           图，令其为 I′
    step 4 把 I′看作数据图，计算其索引图，令为 Inew
    step 5 对 Inew 中的每个索引节点 A，令 ext(A)=B，则
           ext(A)=ext(B1)∪ext(B2)∪…∪ext(Bn)其中，
           B1,B2,…,Bn∈B
    step 6 return Inew
end
```

在算法1的第二步中，可以采用下面的三种方法求新增加的 n 个文档的索引图：方法一：把 H1，H2，…，Hn 看成子树，添加到新的 root 节点下，得到数据图 Hn′，调用 PT 算法计算出其索引图 IH；方法二：先求出 H1，H2，…，Hn 的索引图 H1′，H2，′…，Hn′，然后把 H1′，H2，′…，Hn′作为子树，添加到新的 root 节点下，得到最终索引图 IH 的一个细分，再调用 PT 算法计算出索引图 IH；方法三：先计算出 H1的索引图 IH1，然后调用 PT 算法计算 H2的索引图 IH2，调用 add-file 算法把 IH2加上，得到 IH2′，重复该操作，依次增加数据图 H2，H3，…，Hn，得到最终的索引图 IH。

下面来分析这三种方法的时间和空间复杂性。方法一只需调用一次 PT 算法，但却要保存所有的 n 个数据图，需要很大的存储空间；方法二需要调用 n+1次 PT 算法，但只需保存 n 个索引图，所需空间大为减少；方法三要调用2n+1次 PT 算法，但其所需保存的索引图最多时只有两个(不考虑结果中的索引图)，其存储空间的消耗比方法二要少得多。

从 I′的构造过程可以看出，对任意两个数据节点 u，v∈VG，u，v 在 I′中属于同一个等价类，那么，它们在 Inew 中也属于同一个等价类。根据定理1，算法1得到的索引图 Inew 就是我们所要求的更新后的索引图。

对于更新时间的选择，可具体问题具体分析：对一个远程监控系统来说，可以每天更新一次，或者当需要维护远程系统时来进行更新；对一个 Web 数据服务器来说，可以选择深夜访问量较少时来进行索引更新。总之，有这样一个原则，那就是把索引更新放到无查询需求或者查询需求尽可能少的时候来进行，以把对用户查询的影响减小到最低限度。

由于在前一次索引更新后、后一次索引更新前，索引会与数据产生不一致，在对 XML 数据进行访问时要进行特殊处理。可以采用两种方法来解决这一问题：一是对已经建立了索引的数据，根据其索引来访问，而对于还未建索引的数据，直接访问它们，由于这部分数据的量不会太大，该方法是可行的；二是维护一个小的、只基于新增加文档的索引(维护这一索引的代价很小)，对访问操作分别在这两个索引上查找。当然对我们的远程监控系统来说，这一问题并不存在，因为平时并不需要查询，只在维护远程系统时才需要查询文档。

## 4 删除文件

对在 XML 文档集合中删除一个文档时的索引维护，首先来分析要删除的文档所对应的数据图中的节点在文档集合的索引图中的存在方式。假设原

XML 文档集合对应的数据图为 G，索引图为 IG，要删除的文档对应的数据图为 H，索引图为 IH，更新后的数据图为 G′，索引图为 IG′。对 IH 中的每一个索引节点 A，ext(A)中的每一个数据节点，在索引图 IG 中仍然属于同一个等价类，也就是说在 IG 中存在某个节点 B，使得：(1)ext(A)真包含于 ext(B)，即在 G 中存在某个不属于 H 的节点与 ext(A)中的节点属于同一个等价类，或者(2)ext(A)=ext(B)，即 ext(B)中的节点都是 H 中的节点。

当从一个 XML 文档集合上删除一个文档时，我们从上面分析中提到的两种情况来说明索引的维护操作：对(1)ext(A)真包含于 ext(B)，则 ext(B)=ext(B)-ext(A)；(2)ext(A)=ext(B)，则可以删除节点 B 及其所有的子孙节点，因为其子孙节点的 extent 也只由 H 中的数据节点组成(如果两个节点不满足 bisimularity 关系，那么它们的子孙节点也不会满足这一关系)。

**算法2**

```
procedure del-file(G,H)
begin
step1 计算数据图 H 的索引图，令为 IH，为 IH 的根节点(即文档的根元素对应的节点)增加一个父节点，作为新的根节点
step2 IH 的根节点入队列1
step3 令数据图 G 的索引图为 IG，IG 的 root 节点入队列2
      whlie(队列1不空)
step4 取队列1的队头节点 A，取队列2的队头节点 B
step5 求出 A 的所有子节点，求出 B 的所有子节点
step6 对 A 的每个子节点 Im，在 B 的所有子节点中找出节点 In，使得(ext(Im)真包含于 ext(In))
            则 ext(In):=ext(In)-ext(Im);Im 入队列1;
              In 入队列2;
          或者 (ext(Im)=ext(In))
            则 在索引图 IG 中删除 In 及其所有子树;
        end of while
    end//end of procedure
```

该算法的主要流程是遍历索引图 IG 和 IH，其时间复杂度为 O(nIH * log(nIG))，这 nIG、nIH 为索引图 IG 和 IH 的节点数。而如果重建索引的话，时间复杂度将会是 O(mG′log(nG′))，mG′为删除文档后数据图 G′的边数，nG′为其节点数。算法2要比重建索引快得多。

下面来说明该算法的正确性。假设原 XML 文档集合的数据图为 G，索引图为 IG，要删除的文档的数据图为 H，索引图为 IH，更新后的数据图为 G′，索引图为 IG′，算法2得到的图为 I。我们来说明 I=IG′(这里，图的相等是指同构)。对任意两个数据节点 u，v∈G，而不属于 H，u，v 在 IG 中属于同一个等价类，算法2并没有破坏这一条件，u，v 在 I 中也属于同一个等价类，而 u，v 在 IG′中当然也在同一个等价类中，所以，I 是 IG′的一个细分；反之也成立，因此有 I=IG′。这就证明了算法2得到的图就是我们所要求的更新后的索引图。

**总结** 在本文中，我们讨论了在一个 XML 文档集合上增加多个文档和删除一个文档时的结构索引的维护操作，提出了用于这两种情况下的更新索引的算法，它们比我们已知的其他方法要更好。这些算法不只适用于远程监控系统，对于其他的类似的应用也是适用的。其实，这些算法也不只适用于以文档集合形式存放的 XML 数据，对于其他形式的 XML 数据，比如存放在关系数据库中的 XML 数据也是适用的。

## 参 考 文 献


1 Graves (美)，尹志军等译. XML 数据库设计. 机械工业出版社，2002，8

2 Connoly(美)，宁洪等译. 数据库系统-设计、实现与管理. 电子工业出版社，2004，1

3 Paige R，Tarjan R E. Three partition refinement algorithms. SIAM Journal on Computing，Dec. 1987

4 Kaushik R，Bohannon P，Nauhgonand J F，Shenoy P. Updateforstructure indexes. In：Proc. of the 28th VLDB Conf. 2002


(上接第67页)

将结果返回给用户。达到合理的分配任务，提高了资源的利用率。

**结论** 通过网格能创建虚拟组织或虚拟企业，组织、商业机构和个人通过共享一部分或者所有的资源参与虚拟组织。本文提出了网格环境下基于虚拟组织的调度模型，建立模型的目的是合理分配、调度资源来解决科学、工业、商业广泛存在大规模的计算问题，此模型主要由资源登记服务器和任务调度服务器构成。登记服务器记录虚拟组织节点所提供服务的属性。任务调度服务器接受用户应用并分配调度工作任务。它根据资源提供者的属性和用户任务的性质通过调度分配算法来分解任务，分配给各节点处理，使虚拟组织各节点协同工作，共同完成复杂计算任务。

## 参 考 文 献


1 Foster I. What is the Grid? A Three Point Checklist. July 20，2002. http://www-fp. mcs. anl. gov/~foster/Articles/WhatIsTheGrid. pdf

2 Foster I，Iamnitchi A. On Death，Taxes，and the Convergence of Peer-to-Peer and Grid Computing. In：2nd International Workshop on Peer-to Peer Systems

3 都志辉，陈渝，刘鹏. 网格计算. 北京：清华大学出版社，2002

4 Sullivan W T，Werthimer D，Bowyer S，Cobb J，Gedye D，Anderson D. A new major SETI project based on Project Serendip data and 100，000 personal computers. In：Proc. of the Fifth Intl. Conf. on Bioastronomy，Italy，1997

5 Chapin S，Karpovich J，Grimshaw A. The Legion Resource Management System. In：Proc. of the 5th Workshop on Job Scheduling Strategies for Parallel Processing，April 1999. (http://legion. virginia. edu)

6 Yu Jia，Venugopal S，Buyya R. Grid Market Directory：A Web Services based Grid Service Publication Directory. Jan 2003. http://www. gridbus. org/papers/gmd. pdf

7 Globus Web Site. http://www. globus. org/