基于索引的 XML 查询技术研究*)

肖 袁 吉根林

(南京师范大学计算机系 南京 210097) (苏州大学江苏省计算机信息处理重点实验室 苏州 215006)

摘 要 介绍了目前 XML 数据查询技术的研究现状,对主要的 XML 索引查询技术作了较深入的探讨,其中包括:基于路径索引的 XML 查询方法,如 DataGuide、1-index、A(k)索引等;基于编码的 XML 索引查询方法,如 Anc_Desc_B^+、XR 树+XR-Stack 算法等。文中对相关 XML 索引查询方法的优点和不足进行了分析。

关键词 XML 查询,XML 索引,路径索引

Study on XML Indexing Query

XIAO Yuan JI Gen-Lin

(Department of Computer Science, Nanjing Normal University, Nanjing 210097)

(Jiangsu Province Key Laboratory of Computer Information Processing, Suzhou University, Suzhou 215006)

Abstract This paper introduces the research situations on XML querying techniques and discusses deeply the main techniques of XML indexing query, which includes the querying techniques based on path index, such as DataGuide, 1-index, A(k) and indexing query based on coding, such as Anc_Desc_B^+、XR +XR-Stack arithmetic. This paper also analyzes the advantage and disadvantage of the methods of XML indexing query.

Keywords XML query, XML index, Path index

1 引言

XML(可扩展标记语言)已成为 Web 应用中数据表示和数据交换的标准,随着 Internet 的快速发展,尤其是电子商务、Web 服务等应用的广泛使用,XML 类型的数据成为当前主流的数据形式。因此 XML 数据的管理技术尤其是 XML 数据查询技术成为当前的研究热点。

XML 查询主要有以下两种:(1)值查询:通过限定在元素内容或属性上的取值而进行的选择查询。(2)结构查询:通过路径表达式,对文件中标记的元素之间的结构关系进行查询。元素之间的结构关系包括:祖先/后裔(ancestor/descendant)关系、双亲/孩子(parent/child)关系、之前/之后(preceding/following)关系、左兄弟/右兄弟(preceding-sibling/following-sibling)关系等。对上述查询,一种方法是建立 XML 文档树的路径索引,并通过路径索引加速 XML 查询的计算;另一种方法是对 XML 文档树中的结点(或边)进行编码,通过编码直接判断结点之间的结构关系或元素内容。

对 XML 数据建立有效的索引是影响 XML 数据查询性能的重要因素,本文将对主要的 XML 索引查询技术做较深入的探讨,其中包括基于路径索引的查询方法和基于编码的索引查询方法。

2 基于路径索引的 XML 查询技术

路径索引的基本思路是:将 XML 文档转换成 XML 数据图,通过扫描 XML 数据图得到路径索引图,其中索引图是用较少的边来存储 XML 数据图中的边。目前主要的 XML 路径索引有:DataGuide[1]、1-index[2]、A(k)[3]、D(k)[4]、M(k)[5]、APEX[6]、Fabric[7]索引。下面分别对这些索引进行介绍。

2.1 DataGuide 索引

DataGuide[1]是斯坦福大学提出的 Lore 系统中所用的一种索引结构,它是对半结构化数据的一个简洁而精确的概括。它将 XML 数据图中由根结点开始经过相同标签值的所有路径存储在 DataGuide 中,在 DataGuide 中以一个结点集表示 XML 数据图中的多条边。一个 XML 数据图可能对应多个 DataGuide 索引图,为此 Lore 系统定义了一个"Strong DataGuide"模型,Strong DataGuide 的优点是如果在 XML 数据图中的简单路径到达的结点相同,那么这些简单路径存储在"Strong DataGuide"中的相应的结点集中。例如,图 1(a)中 XML 数据图它所对应的"Strong DataGuide "索引如图 1(b)所示。Lore 系统支持文档的动态更新,当文档更新时,"Strong DataGuide"也被更新。

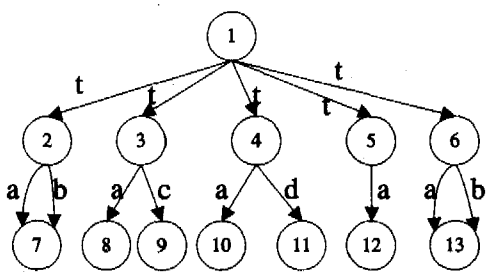
2.2 1-index 索引

DataGuide 存在下列不足:(1)DataGuide 是对 XML 数据图精确的概括,若 XML 数据图是图结构,那么建立 DataGuide 的时间和所需的空间可能是 XML 数据图大小的指数倍。(2) DataGuide 中各个结点的扩展集可能相交。为了解决 DataGuide 的上述两个问题,1-index[2]提出两结点"相似"概念,其含义是:若结点 u,v 具有相同的标签且结点 u' 是 u 的父结点,结点 v' 是 v 的父结点,则结点 u' 与 v' 相似。

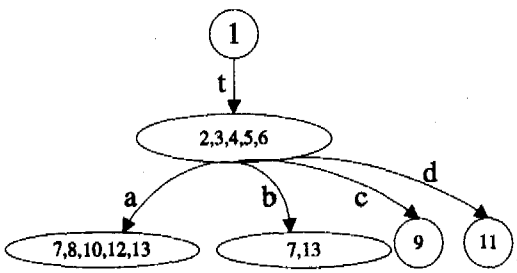
1-index 索引中将"相似"的结点存放在一个扩展集中,例如,图 1(a)中表示的是 XML 数据图,图 1(c)是图 1(a)XML 数据图的 1-index 索引。图 1(c)1-index 所表示的索引的扩展集为{7,13},{8,10,12},{9},{11},各个扩展集之间并没有相

*)本文得到江苏省高校自然科学基金(04KJB520075)的资助。肖 袁 硕士,主要研究方向为 XML 技术。吉根林 教授,博士,主要研究方向为数据库和数据挖掘技术等。

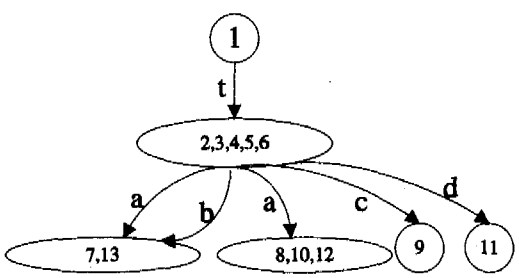
交，图1(b)DataGuide所表示的索引的扩展集为{7,8,10,12,13},{7,13},{9},{11}，很明显扩展集之间相交。这可能造成DataGuide中所有扩展集的结点总数是XML数据图中结点总数的指数倍。



(a)XML数据图



(b)Strong DataGuide索引



(c)1-index索引

图1 1-index索引和DataGuide索引比较

利用两结点"相似"概念使得1-index具有如下两个优点：(1)索引大小和XML数据图大小成线性关系。(2)索引的扩展集之间不相交，所有扩展集的结点总数和XML数据图中结点总数相等。

2.3 A(k)索引

由于DataGuide和1-index保存XML数据图中所有边的信息，因此使用这两种索引进行查询代价均很高。为解决该问题从而提出A(k)[3]索引。

A(k)索引提出结点之间具有"k相似度"的概念，所谓k相似度是指：(1)结点u,v若具有相同的标签，则结点u,v具有0相似度。(2)若结点u,v具有$k-1$相似度；结点u'是u的父结点，结点v'是v的父结点且u'与v'具有$k-1$相似度，则结点u,v具有k相似度。A(k)索引的基本思想是将XML数据图中结点的相似度为k的结点存储在索引图的同一个结点集中，这就意味着所有路径长度为k的路径全部存储在索引图中。

A(k)索引的查询策略分为两种情况：(1)查询语句的路径长度$\leqslant k$时，在A(k)索引中采用自底向上或自顶向下的查询策略都可以得到精确的查询结果。(2)查询语句的路径长度$>k$时，在A(k)索引中无论采用自底向上或自顶向下的查询策略所得到的查询结果都有可能包含错误的查询结果。所以在这种情况下还要将所得到的查询结果在XML数据图上进行验证，以确保所得到的查询结果是正确的。

2.4 D(k)索引

A(k)索引中每个索引结点的相似度都为k；D(k)索引[4]是对频繁使用的路径进行索引，而频繁使用的路径长度往往是不相同的，因此D(k)索引中的每个结点的相似度k也不相同。

D(k)索引有如下两个性质：(1)每个索引结点与一个相似度k有关。(2)给定一个索引结点的相似度k，它的父结点的相似度至少为$k-1$。

D(k)索引主要由两个算法组成：(1)索引创建算法：首先根据频繁使用的路径决定每一个索引结点所需的最大相似度k，然后由A(0)索引开始迭代，分裂每个索引结点，直到它们的相似度达到最大相似度k。(2)更新算法：对频繁使用的路径进行更新后，现有索引中的结点的最大相似度也要进行相应的更新。

2.5 M(k)索引

D(k)索引主要存在以下两个问题：(1)在更新算法中，对不相关结点的扩展集也进行分裂操作。分裂后的索引大小比满足查询需要的最小索引的大小要大得多。(2)若结点v的相似度为k，v的父结点的相似度满足$>k-1$，D(k)索引更新算法对结点v的扩展集进行分裂操作，而实际上对于满足这类条件的结点v，不要对其结点的扩展集进行再分裂操作。引起上述两个问题的原因是D(k)索引的k值只能取一个值。

M(k)[5]索引的特点：(1)每个索引结点的相似度k不相同。(2)为了避免对不相关结点集进行分裂，M(k)索引的更新算法除了使用路径表达式之外还使用了频繁使用路径的查询结果。M(k)索引是多个类似M(k)索引的集合，表示为I_0，$I_1,\cdots,I_k$，分别存储结点相似度从0分裂到k的索引。M*(k)索引中各个索引之间结点分裂前后的结点由指针连接。

M*(k)索引有如下性质：(1)每个索引I_i具有M(k)索引的性质。(2)索引I_{i+1}是由索引I_i的分裂得到。由I_i索引结点分裂得到I_{i+1}的索引结点，I_{i+1}索引结点的相似度至多增加1。(3)若I_i索引分裂得到I_{i+1}索引，结点的相似度不增加，则结点相似度在以后的索引分裂中也不会增加。

M*(k)索引优点：由于k有不同取值，因此它可以高效查询路径长度不同的频繁使用路径，并且避免了D(k)索引的两个缺陷。

2.6 APEX索引

DataGuide索引和1-index索引保存从根结点开始的所有简单路径。若用户给定的查询路径不是从根结点开始，则使用上述两个路径索引需要对索引进行遍历式导航，查询效率降低。

APEX索引[6]是DataGuide索引的一种近似，能够有效地处理某些相对路径查询，它创建的索引是基于查询频度，对频繁使用路径创建索引。当查询频度有变化时，APEX索引可以进行有效的更新以适应查询频度的变化。

APEX索引的优点：(1)索引可以随着查询频度的改变而更新。(2)能够有效处理查询语句的开始标签不是从根结点开始的简单路径查询。

APEX索引的缺点：(1)频繁使用路径如何判定。(2)对非频繁使用路径的查询效率较低。(3)查询从根结点开始的

路径效率较低。

2.7 Fabric 索引

Fabric 索引[7]的基本思想是将半结构化数据之间的关系表示成路径，将路径编码成字符串，然后在字符串上面建立索引。

Fabric 索引的优点：(1)能够管理大量长而复杂的字符串。(2)其结构是平衡的，因而不同的访问所需的查询代价几乎相同。(3)既可以作为路径索引，也可以作为值索引。

Fabric 索引是从 Patric Trie(PT)树结构发展而来的，Patric Trie 是一种字符串的编码方式，是一种多路树，采用树形结构将字符串之间的差异表示出来。正是因为 PT 树只描述串之间的差别，所以它增长得很慢，同时索引项的长度对索引大小影响很小。Fabric 索引支持文档的更新操作。

利用 Fabric 索引管理 XML 路径文档路径分为两类：简单路径和精确路径。简单路径将 XML 文档从根节点到叶子结点的路径按照顺序编成字符串。精确路径不仅可以实现简单路径的编码功能也可以实现对其他信息进行编码，从而可以支持复杂查询。

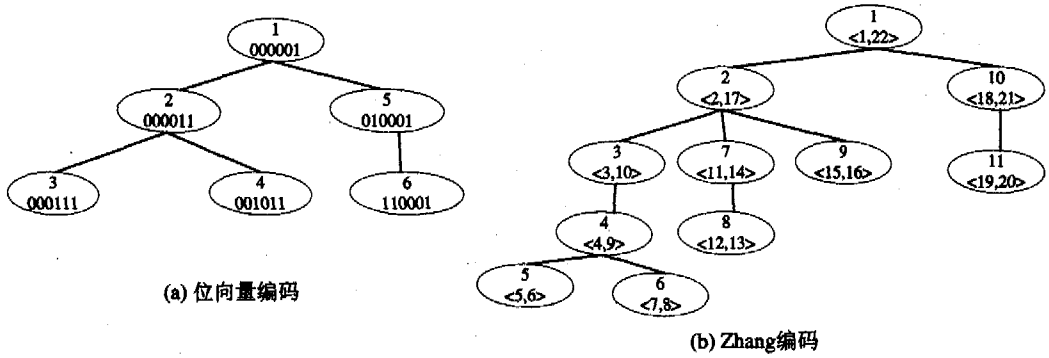
3 基于编码的 XML 索引查询技术

基于路径索引的 XML 查询技术能够有效地解决单路径查询问题，但是不能很好地解决分支路径查询；编码索引查询技术不仅可以有效地查询单路径，也解决了分支路径查询问题。基于编码的 XML 索引查询技术主要有：XR+XRstack 算法[8]、Anc_Desc_B$^+$ 算法[9]等。

3.1 XML 数据的编码方案

目前常用的 XML 数据的编码方案主要有：位向量编码[12]、区间编码[13]等。

位向量编码：树 T 中的每个结点被编码为一个 n 位向量，n 是树 T 中的结点数量，在某个位置 i 上的一个“1”惟一标识第 i 个结点；并且在一个自顶向下(或自底向上)的编码方案中，每一个结点继承它祖先(或后裔)结点的所有位上的“1”，如图 2(a)所示。



(a) 位向量编码　　(b) Zhang编码

图 2　编码方法

区间编码：树 T 中的每一个结点被赋予一个区间编码 $[begin,end]$，并且满足：一个结点的区间编码包含它的后裔结点的区间编码，即树 T 中的结点 u 是结点 v 的祖先，当且仅当 $begin(u)<begin(v) \wedge end(v)<end(u)$。基于区间编码的方案目前有三种：(1)Dietz 编码[13]：树 T 中每个结点被赋予一个先序遍历序号和后序遍历序号的二元组$\langle pre, post\rangle$。(2)Li-Moon 编码[10]：树 T 中的每一个结点被赋予一个二元组$\langle order, size\rangle$，$order$ 位结点的扩展先序遍历序号，它的取值是非连续的，为结点的插入预留序号空间，$size$ 为结点的后裔范围。(3)Zhang 编码[9]：XML 文档树中的每一个结点被赋予一个二元组$\langle begin, end\rangle$。对树 T 的所有结点进行先序遍历，每一个结点在遍历树时分别被访问两次并产生两个序号。一次是在遍历该结点的所有后裔结点之前访问该结点，并产生该结点的序号 $begin$；另一次是在遍历完该结点的所有后裔结点后再一次访问该结点，并产生该结点的另一个序号 end，如图 2(b)所示。

3.2 Anc_Desc_B$^+$ 算法

一般而言，基于编码的查询算法都是基于结构连接算法的，所谓结构连接算法是指利用结点的编码来判定结点之间是否具有祖先/后裔关系。Anc_Desc_B$^+$ 算法[9]的基本思想是：对于给定的查询语句，首先要生成它的祖先列表和后裔列表，然后利用 B$^+$ 树索引跳过不参加连接的祖先列表中的结点和后裔列表中的结点，避免对祖先列表和后裔列表中的所有结点进行扫描。

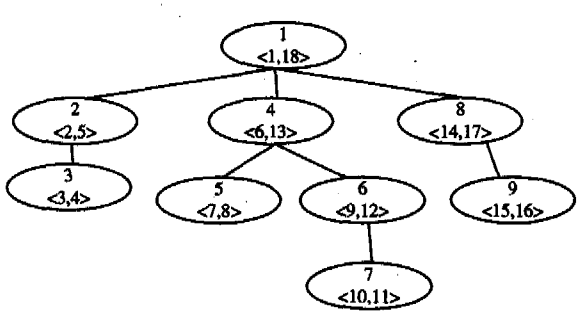


图 3　跳过后裔结点情况

例如，对于图 2(b)XML 数据图，假设某个查询语句所对应的祖先列表中的结点为{1,2,3,4,7,8,9,10}，后裔列表中的结点为{5,6,11}。现要查找结点 10，则通过扫描祖先列表和后裔列表来判断结点之间的关系，在判定完结点 5,6 的父结点为结点 4 之后，判断结点 11，由于结点 11 的父结点是 10，且结点 10 是祖先列表中满足 $begin>end(2)$ 的结点，因此跳过祖先列表中结点{7,8,9}，而直接定位到结点 10。

在图 3 所示的 XML 文档中，假设某个查询语句所对应的祖先列表中的结点为{1,2,8}，后裔列表中的结点为{3,4,5,6,7,9}。在判定出结点 2,3 具有祖先后裔关系后，利用 B$^+$ 树索引直接定位到结点 9，结点 9 是后裔列表中满足 $begin>begin(2.$ 条件的第一个结点，避免了后裔列表中结点{4,5,6,7}的扫描。

(下转第 96 页)

图 12 XML 的文档编码后的性能比较

总结 本文在分析研究了 DMB EPG 相关协议原理的基础上，设计实现了 DMB EPG 的数据存储方案，分析了 XML 传输性能问题，并采用二元 XML 设计，实现了 DMB EPG 的编码器，最后分析比较了三种编码的性能。

进一步的工作有以下几个方面值得研究，如数据经过编码后的后续处理过程，即文档封装成 MOT 对象、分段、打包和传输技术以及终端如何进一步改善解码技术等。

参 考 文 献

1 ETSI TS 102 818 (2005-01), XML Specification for DAB EPG. http://www.worlddab.org/irc.aspx? sub=10

2 ETSI TS 102 371 (2005-01). Transportation and Binary encoding for DAB EPG. http://www.worlddab.org/irc.aspx? sub=10

3 ETSI EN 301 234(1999-02). Multimedia Objects Transportation Protocol. http://webapp.etsi.org/action/OP/OP20060519/en_301234v020101o.pdf

4 ETSI TS 300 401(2001-05). Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers. http://www.worlddab.org/irc.aspx? sub=10

5 万常选. XML 数据库技术. 清华大学出版社，2005

6 Sosnoski E. 改善 XML 的传输性能. http://www.ibm.com/developerworks/cn/xml/

7 Bayardo R J, Gruhl D, et al. An Evaluation of Binary XML Encoding Optimizations for Fast Stream Based XML Processing. http://www2004.org/proceedings/docs/1p345.pdf

8 XML Binary Characterization Working Group. http://www.w3.org/XML/Binary/

9 李莉明. 二进制 XML 浅析. 计算机与数字工程，2005，33(22)：1～5

(上接第 80 页)

3.3 XR 树＋XR-Stack 算法

在 Anc_Desc_B^+ 算法中，利用 B^+ 树索引能够有效地跳过后裔列表中所有不参加连接的结点，但是对于祖先列表中所有不参加的结点，并不能利用 B^+ 树索引来有效地跳过，这是因为：在祖先列表中无法利用 B^+ 树索引直接定位结点的第一个可能的祖先结点，需要多次定位才能跳到结点的第一个可能的祖先结点。为了解决这个问题，XR 树索引被提出[11]，基于 XR 树索引不仅能够有效地跳过后裔列表中所有不参加连接的结点，而且能够跳过祖先列表中所有不参加连接的结点。例如，图 3 中若要找到后裔结点 9 的父结点 8，在 Anc_Desc_B^+ 算法中不能够一次定位到结点 9 的父结点，需要多次在祖先列表中进行定位才能够找到结点 9 的父结点 8，而 XR 树＋XR-Stack 算法能够在祖先列表中一次定位到结点 9 的父结点 8。

3.4 XISS 系统

XISS 系统[11]的主要思想是：将一个正则路径表达式分解成子路径表达式，将子路径表达式产生的中间结果进行结构连接操作得到最终的查询结果，并且该系统支持文档更新操作。

XISS 系统主要由五个索引组成：元素索引、属性索引、结构索引、名称索引、值索引。这五个索引采用 B^+ 树作为索引结构。

XISS 系统将正则路径表达式分解成五个基本的子表达式：(1)单元素单属性子表达式：通过查找元素索引、属性索引得到查找结果。(2)一个元素和一个属性子表达式：EA-join。(3)具有两个元素的子表达式：EE-join。(4) Kleene cloure (＋，＊)：KC-join。(5)子表达式之间的联合操作。将分解的子表达式所得的中间结果进行连接得到最终的正则路径表达式的查询结果。

XISS 缺点：只能解决单路径查询，对于分支路径查询效率较低。

结束语 本文主要研究了 XML 查询技术中的两种主要方法，其中基于路径索引的 XML 查询方法只能够解决单路径查询，但是路径索引的创建不受 XML 文档结构的约束，即 XML 文档可以是树结构，也可以是图结构。基于编码方式下的 XML 索引查询方法能够有效地解决分支路径查询，但是这种方法都对相应的 XML 文档有要求，其所要查询的 XML 文档为树形结构。怎样很好地将上述两种 XML 查询方法中的优点结合起来，是将来研究的热点之一。

参 考 文 献

1 Goldman R, Widom J. DataGuide: enabling query formulation and optimization in semistructured databases. In: 23th International Conference on Very Large Data Bases, pages, Athens, Greece, 1997. 436～445

2 Milo T, Suciu D. Index structures for path expressions. In: 7th International Conference on Database Theory, Jerusalem, Israel, 1999. 277～255

3 Kaushik R, Shenoy P, Bohannon P, et al. Exploiting Local Similarity for Efficient Indexing of Paths in Graph Structured Data. In: 10th International Conference on Database Theory, San Jose, California, USA, 2002. 129～140

4 Chen Q, Lim A, Ong K W. D(k)-index: An adaptive structural summary for graph-structured data. In: Proc. of the 2003 ACM SIGMOD Intl. Conf. on Management of Data, San Diego, California, USA, 2003. 134～144

5 He Hao, Yang Jun. Multiresolution Indexing of XML for Frequent Queries. In: Proceeding of the 20th International Conference on Data Engineering. Boston, USA, 2004. 683～694

6 Chung C, Min J, Shim K. APEX: An adaptive path index for XML data. In: Proc. of the 2002 ACM SIGMOD Intl. Conf. on Management of Data, Madison, Wisconsin, 2002. 121～132

7 Cooper B F, Sample N, Franklin M J, et al. A Fast Index for Semistructured Data. In: Apers P M G, eds. Proceedings of the 27th VLDB International Conference on Very Large Database, Rome, Italy, SanFrancisco: MorganKaufmannPublishers, 2001. 341～350

8 Jiang Haifeng, Lu Hongjun, Wang Wei, et al. XR-Tree: Indexing XML Data for Efficient Stuctural Joins. In: Casati F, et al. eds. Proceedings of 19th IEEE ICDE International Conference on Data Engineering, Bangalore, India, 2003. 253～263

9 Chien S Y, Vagena Z, Zhang Donghui, et al. Efficient Structural Joins on Indexed XML Documents. In: Papadias D, et al. eds. Proceedings of the 28th VLDB International Conference on Very Large Database, Hong Kong, China, 2002. 263～274

10 Li Quanzhong, Moon B. Indexing and Querying XML Data for Regular Path Expressions. In: Proceedings of the 27th VLDB Conference, Roma, Italy, 2001. 361～370

11 Wirth N. Type Extensions. ACM Transactions on Programming Language and Systems, 1988, 10(2): 204～214

12 Dietz P F. Maintaining Order in a Linked List. In: Lewis H R ed. Prcoeedings of the 14th Annual ACM Symposium on Theory of Computing. San Francisco, California, USA, 1982. 122～127

13 Bruno N, Koudas N, Srivastava D. Holistic Twig Joins: Optimal XML Pattern Matching. In: Franklin M J, ed. Proceedings of the 21th ACM SIGMOD International Conference on Management of Data. Madison, Wisconsin, USA, 2003. 310～321