FC-Index：一种压缩XML数据的索引结构

任永功 武佳林 范 丹
（辽宁师范大学计算机与信息技术学院 大连 116029）

摘 要 如何迅速、有效地进行XML数据查找，关键问题是避免对无关元素进行查询。通过合并元素，可以减少文档中元素的数目，同时还能够避免查找冗余结点，有效提高查询效率。提出一种基于Ctree的新索引结构FC-Index，它通过合并结构中“相同”元素压缩结构，从而在查询过程中过滤掉与查找无关的元素。基于FC-Index索引结构提出一种新的查询方法，能够有效地针对FC-Index进行快速查找。

关键词 XML，查询优化，结构优化

FC-Index：A Compact Structure for Indexing XML Data

REN Yong-gong WU Jia-lin FAN Dan
(School of Computer and Information Technology, Liaoning Normal University, Dalian 116029, China)

Abstract How to process XML query quickly and correctly, the key of the question is avoiding to query the irrespective elements. By combining the elements, the number of the elements can be reduced, and we can avoid to query the redundant elements for querying quickly and correctly. A new efficient index structure—FC-Index based on ctree was proposed to leach the irrespective elements in the querying. At the same time, we also proposed a new query algorithm based on FC-Index which can improve the query's performance significantly for FC-Index.

Keywords XML, Query optimization, Structure optimization

XML作为数据描述和查询标准已经被应用得越来越广泛，对XML查询优化问题的研究也成为热点问题。XML文档包含层次嵌套关系，所以XML文档可以看作是一棵文档树。但由于XML文档数据繁多，直接查找会影响查询效率，例如，Xpath通过查找root/book/author来查询满足title为sport的author，在XML文档中可能存在许多相似的路径，这会导致对一些无关的结点进行查找。

为了能够快速有效地找到用户提交的查询，可先对XML文档索引结构进行压缩处理。通过对“相同”结点的压缩，既能够减少文档中结点的个数，又能够减少查询结点的个数，从而提高索引性能。由于XML文档中的结点保持PD，AD关系，因此通过这种关系可以判断出结点与结点、路径与路径之间的“相同”关系。通过利用这一特点可以避免许多冗余查询。本文提出了一种新的索引结构FC-Index，通过处理“相同”结点来优化结构，同时还提出一种新的查询方法，这种方法可以对FC-Index结构进行高效查找。

1 相关工作

通过遍历XML数据树，压缩等价结点，可以有效地减少被查找的结点数目，同时也减少了查找的路径数。现在已经有许多优秀的查询算法，例如：DataGuide[8]，1-index[9]，F&B-index[3]，Ctree[2]等方法。这些方法的相同之处都是通过处理相同路径结点来优化相同路径，使查找效率提高。其中F&B-index和Ctree索引是其中比较典型的代表。F&B-index是根据树从叶子结点向上遍历，通过判断两个结点的后代结点是否相同，来决定是否对结点进行合并。Ctree结构是根据结点的等价来进行合并操作，合并的结点以组的形式来存储，在这种组的形式中能够保存结点原有信息，同时Ctree还保存结点的父子信息，可以迅速查找结点。但Ctree在进行合并时，合并结点的路径并不都是完全相同，例如图1中表示出的结点$c1$和$c2$，$c2$后代比$c1$后代多一个结点$h1$，如果要查找路径为a/c2/h1[f2,g1]，利用Ctree索引就会对a/c1/f1[g1]也查询一遍，影响查询性能。本文提出的结构FC-Index是以F&B-index索引为基础对XML数据树进行合并，能够有效地解决这类问题。为了方便本文后续部分，以“/”或“PC”代表结点之间的父亲关系，以“//”或“AD”表示祖先后代关系。

2 基于合并结点的查询处理

构建用于压缩结点的索引结构步骤可分为两步：第一步遍历结点，查找符合条件的结点，第二步合并结点。

例：对于查询a/c/f[g]而言，为了得到满意的结果，可以对XML文档处理为Ctree索引结构，如图1所示。这样通过遍历Ctree结构，可以得到满意的结果。但是，由于图1(a)中

到稿日期：2008-11-28 本文受国家自然科学基金项目(60603047)，辽宁省教育厅高等学校科研基金(2008341)，大连市优秀青年科技人才基金(2008J23JH026)，辽宁省科技计划项目(2008216014)资助。

任永功(1972－)，男，教授，博士，研究方向为数据挖掘技术等，E-mail：renyg@dl.cn；武佳林(1985－)，男，硕士研究生，研究方向为XML数据库、Web挖掘；范 丹(1985－)，女，硕士研究生，研究方向为Web挖掘、XML数据库、信息检索。

所示的第二层的两个 c 结点是被放到同一组中，如图 1(b)中所示的 c 结点，这就需要在遍历 Ctree 结构查找 c 路径的过程中，也要查找 a/c/f 这条不符合查询路径的路径，影响查询效率。

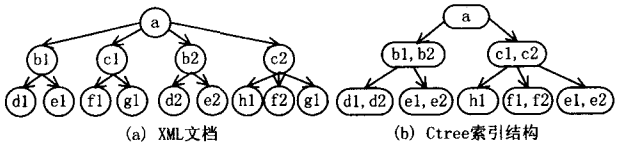


图 1　索引结构

针对这一类问题，本文提出一种新的索引结构，其基本思想是基于 F&B-index 和 Ctree 索引结构为基础，通过判断结点路径的“相同”与否，考虑结点是否进行合并，从而解决在查询过程中遇到影响查询效率的问题，并且经过处理之后得到规范的索引结构，能够更加优化查询效率。

3　FC-Index 的组织结构

图 2 所示的是一个 XML 文档，用{ID, Level, Parent, Label}对 XML 文档进行编码。通过判断结点路径是否“相同”，考虑是否合并结点，则若是把合并的结点以 package 的形式储存起来，称其为 FC-Index。

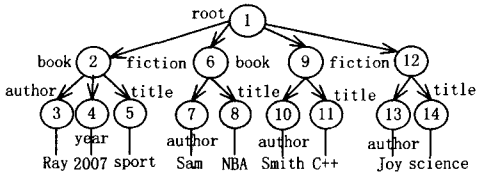


图 2　XML 文档

定义 1　如果任意两个同层的结点 A 和 B，它们的 Label 值相同，并且 Parent 也相同，那么结点 A 和 B 就是等价结点，记为 $A=B$。

例如：如图 2 所示的结构中结点 2 和 9，它们的 Label 都为 book，并且父亲结点都为 root，所以为等价结点。

定义 2　如果索引结点 A 和 B，有 $A=B$，并且 A 和 B 的后代结点也都等价，那么结点 A 和 B 就是路径相同。

对于任意两个路径 $A/B//C$ 和 $D/E//F$，当判断出 A 和 B 路径相同，即是两个路径中的结点相同，那么这两个路径可以合并为一条路径。

定义 3　在 FC-Index 结构中，每个结点叫做 package，在 package 中包含一系列元素。图 3 是 package 的结构图。具体内容如下：

(1)每个 package 中包含对应 XML 文档中结点的 ID, Level, Parent, Label。分别用 package. ID, package. Level, packeage. Parent, package. Label 表示。

(2)package 中存在一个可变的存储空间 group，group 中存储结点的 ID 值。用 package. group[]表示。

(3)package. ID 存储的值是选择 package. group[]中最小的。

(4)对任意 FC-Index 中的结点 node 及其后代结点 cnode, node. package. group[]与 cnode. package. group[]是完全相同的，这样能够保持结点之间的 PC 关系。

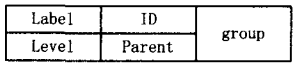


图 3　链表元素结构

在 FC-Index 中，当 XML 文档中的结点符合路径相同的条件时，才能对它们进行合并[4-6]。在对 XML 文档进行 FC-Index 转换处理的过程中，结点结构转变成如图 3 所示的结构，根据相同结点定义，合并之后结点的 parent, label 和 level 不变，ID 是合并结点之间最小的 ID，同时系统会自动分配一个可变的存储空间，存储所有相应合并的结点的 ID，通过 group，结点之间的 PC 关系和兄弟关系不会被破坏[7]。group 是可变的存储空间，如果还存在与合并的结点的相同结点，group 会再次自动分配一个存储单元来存储这个结点的 ID 值。

用一个链表表示一条简单路径，如图 4 就是一个链表元素图。在图 4 中，node6 与 node12 是等价结点，并且遍历结点 node6 和 node12 的后代，能够判断出它们的后代也都为等价结点，可以判断 node6 和 node12 路径相同，所以这两条路径能够合并为图 4 所示的右侧的链表。图 3 是合并之后结点的 package 结构图，{ID, Level, Parent, Label}的值与要合并的结点中 ID 最小的结点相同，以 node6 和 node12 为例，合并之后的结点{ID, Level, Parent, Label}值与 node6 相同，同时创建存储空间 group 来存储合并的结点 ID，即 group 保存的是 node6 和 node12 的 ID。图 5 为对图 1(a)的 FC-Index 变换。

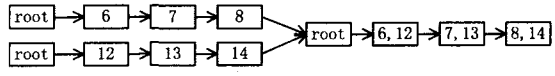


图 4　辅助查询结构

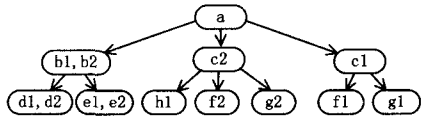


图 5　FC-Index

在遍历 XML 文档树过程中，F&B-index 遍历结点是从叶子结点逐层向上，通过判断孩子结点是否相同，对父亲结点是否进行合并；而 FC-Index 的遍历是对 F&B-index 遍历方法的改进，是从根结点向下遍历，在判断父亲结点是否相同的同时判断后代结点是否相同，经过这样的判断，能够得出两个结点是否为相同路径结点，再根据判断的结果来决定是否合并结点。这样遍历的优点在于它能够在判断结点相同的同时也判断结点路径是否相同，也就避免了对结点路径的判断。

4　基于 FC-Index 的压缩算法

性质 1　对于 XML 文档中任意两个结点 A 和 B，如果 A, B 的路径相同，那么 A 及其后代结点的个数等于结点 B 及其后代结点的个数。

对 FC-Index 进行层次遍历，当两个结点的 Label 相同并且为同一父亲时，先判断这两个结点的后代结点的个数是否相等。如果不等则遍历下一结点，因为这两个结点不是相同路径结点。当后代结点个数相同时，则继续判断后代结点的 Label 是否相同，若相同，则执行合并程序，对结点合并。

算法 1

输入：XML 文档

输出：FC-Index 结构文档

1. if(datafile 不为空)

2.　for each node in datafile

```
    /* 双层循环遍历 datafile 中的结点 */
3.   if(node1==node2)
    /* 判断两个结点是否相同及父亲结点是否相同 */
4.     if(accountnumber(node1,node2))
    /* 计算 node1 和 node2 的后代结点个数是否相同 */
5.       if(matchnode(node1,node2))
    /* 判断结点是否是相同路径结点 */
6.         combinenode(node1,node2);
    /* 合并结点 */
7.       else continue;
8.     else continue;
9.   else continue traversing the nodes;
10. else break;
11. return;
```

首先 XML 文档进行双层遍历，并对遍历的两个同一父亲的结点判断是否等价，若是则调用函数 *accountnumber*()计算两个结点的后代结点的个数，根据性质 1，程序调用函数 *accountnumber*()，能够预先对结点进行判断，排除符合等价结点、但不符合相同路径的结点，通过这一函数，能够在不调用 *matchnode*()函数的情况下，对结点进行预判断，排除结点，从而提高了时间效率；如果符合第 4 行的条件，执行第 5 行，*matchnode*()函数是判断相同路径的函数，最后调用函数 *combinenode*()合并结点。

5 基于 FC-Index 的查询算法

FC-Index 查询算法是专门针对 FC-Index 索引结构进行查询的算法，该算法利用 FC-Index 特有的结构，根据用户输入的路径，快速、准确地查找出结果。

性质 2 在 XML 文档中，任意非叶子结点 A，如果 A 任意两个儿子结点 B，C 是路径相同结点，B，C 后代的叶子结点分别为 $D=\{D_1, D_2, \cdots\}$ 和 $E=\{E_1, E_2, \cdots\}$，那么 D 包含的 Label 个数总和等于 E 包含的 Label 个数总和。

证明：设性质不成立，由于叶子结点下包含的 Label 数目不等，因此，叶子结点个数也不相等，那么叶子结点的父亲路径不相同，就不能进行合并操作。所以性质成立。

算法 2

```
输入：查询路径
输出：查询结果
1. if(XML 文件不空)
2.   for each node in FC-Index
3.     if(matchrout(node1,node2))
    /* 根据输入路径对 FC-Index 进行匹配，查找满足输入路径的结点 */
4.       if(node.group==1)
    /* 判断其中非根结点的 group 的长度是否为 1 */
5.         findnode(node);
6.         return node;
7.       else getnode(node);
8.         return node;
9.     else continue;
10. else can't query the node;
```

输入查找路径，根据输入路径遍历 FC-Index，查找相匹配的路径，由于 FC-Index 是规范结构树，因此，如果没有找到相匹配的路径，则在 FC-Index 中没有要查找的结果。当找到相匹配的结点后，判断其中非根结点的 group 长度是否为 1，如果为 1，引用函数 *findnode*()直接找到结果，因为在进行 FC-Index 变换的过程中，只有在进行合并操作的情况下 group 才会改变存储的值，当 group 长度为 1 时，证明没有进行过合并操作，所以引用函数 *findnode*()可以直接得到结果，否则引用第 7 行函数 *getnode*()得到结果，*getnode*()函数是专门针对合并结点查询的函数，根据性质 2 判断所给路径最后一层的结点数目来查询符合输入路径的结点。

6 试验结果与分析

6.1 实验环境和数据集

本文的实验在一台基本配置为 P4 2.8GHz，1G RAM，80G 硬盘的 PC 上运行，操作系统是 Windows XP，在 VC++6.0 环境下实现了本文提出的查询算法、Ctree 算法以及 XISS 算法。

6.2 对比实验

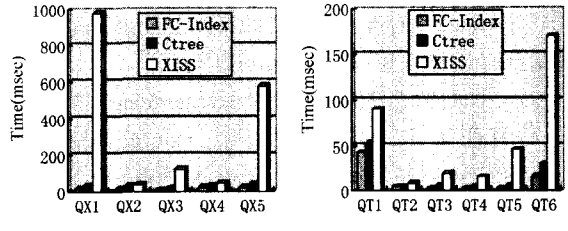
选择 Ctree 和 XISS[10]，以及我们的算法 FC-Index，进行实验对比。所选择的数据集为 XMARK[11] 和 DBLP[11]。XMARK 是综合的、深层的数据集，包含超过 2 百万的元素。DBLP 则是比较流行的以计算机科学参考书为元素的、多达 3 百多万、6 层深度的数据集。

针对数据集 DBLP 的查询为 QT1－QT6，如表 1 所列。

表 1 实验所用查询

QueryNO.	XPath expression
QT1	/inproceedings/title
QT2	/book/author[contains(.,"David")]
QT3	/*/author[contains(.,"David")]
QT4	//author[contains(.,"David")]
QT5	/article[contains(./author,"David")and./year=1995]
QT6	/article[contains(./author,"David")and./year≥1995]

从图 6(a)和图 6(b)来看，FC-Index 所用时间都少于 Ctree 和 XISS。主要原因是 Ctree 进行结点合并根据结点等价来处理，在处理后的结构中包括不完全相同的结点路径，这样在查找相应元素时，会对不相关的路径进行查找，从而影响查询效率。而 FC-Index 则是根据结点路径相同对元素进行处理，构建完整的、规范的树形结构，通过这种规范化的结构，在对路径元素查找时，能够迅速地查找到结果，减少查询时间。对结构简单的文档进行索引，由于其文档结构的嵌套性较低，FC-Index 能够直接查找到结果，在性能方面与 Ctree 相差不多。当结构索引变得很大且复杂时，通过实验结果来看，FC-Index 能够很好地适应复杂的结构文档，在查询结果时，利用我们提出的针对 FC-Index 的查询算法，能够迅速提取出满足条件的结果，很好地处理性能要求。



(a)XMARK (b)DBLP

图 6 过滤后的查询时间对比

(下转第 190 页)

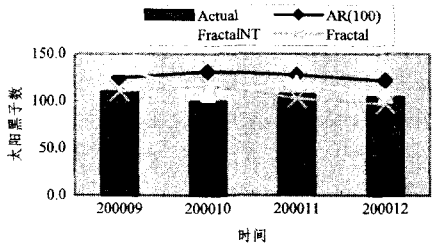


图 6　2000 年 9—12 月月均太阳黑子数的预测结果图

结束语　本文提出了改进嵌入维数和时间延迟计算的 GP 预测算法，既有比较严密的数学基础，又在实际应用中取得了良好的效果：分析结果稳定而准确、预测精度高、运行时间比较短，而且也不需人工干预。因此，它在时间序列研究的领域中具有重要的理论意义，进一步在与时间序列有关的众多应用领域中具有广阔的应用前景。

因为分形理论和近邻预测方法都是新兴的理论，还远远没有达到成熟的阶段，结合分形理论和近邻预测方法的时间序列研究方法还会有进一步的发展。具体来说，如嵌入维数和时间延迟的计算、轨迹的拟合、近似样例的选择、查询样例小邻域内函数的拟合等方面，都还存在改进的空间，值得作进一步的研究，以提出更好的理论和算法。

参 考 文 献

[1] 吕金虎，陆君安，陈士华. 混沌时间序列分析及其应用[M]. 武汉：武汉大学出版社，2002

[2] Grassberger P, Procaccia I. Dimension and Entropy of Strange Attractors from a Fluctuating Dynamic Approach[J]. Physica, 1984, 13D: 34-54

[3] 李信富，李小凡，武晔. 分形理论在地震学中的应用研究[J]. 地球物理学进展，2007(2)

[4] 王德明，曾丹苓，刘娟芳. 利用分形理论研究气液界面特性[J]. 工程热物理学报，2004(5)

[5] 杨新宇，曾明，赵瑞，等. 分形理论在网络流量分析中的应用综述[J]. 计算机工程，2004(23)

[6] Mansurov K. Forecasting currency crises by fractal analysis techniques[J]. Studies on Russian Economic Development, 2008, 19(1): 96-103

[7] Schertzera D, Lovejoy S. Space-time complexity and multifractal predictability[J]. Physica A: Statistical Mechanics and its Applications, 2004, 338: 173-186

[8] 候越先，可丕廉，王雷. 适用于高必要嵌入维的混沌时间序列预测算法[J]. 天津大学学报，1999, 32(5): 594-598

[9] 郑会永，刘华强，戴冠中. 时间序列分维的改进 GP 算法[J]. 西北工业大学学报. 1998, 16(1): 28-32

[10] Albano A M, Muench J, Schwartz C, et al. Singular-value decomposition and the Grassberger-Procaccia algorithm[J]. Phys. Rev. A, 1993, 38: 3017-3026

[11] Kennel, Mathew B, Brown R, et al. Determining embedding dimension for phase-space reconstruction using a geometrical construction [J]. Phy Rev A, 1992, 45: 3403-3411

[12] Broomhead D, King G. Extracting qualitative dynamics from experimental data [J]. Physica D, 1986, 20: 217-236

[13] Fraser A M, Swinney H I. Independent coordinates for strange attractors from mutual information [J]. Phys. Rev. A, 1986, 33: 1134-1140

[14] Rosenstein M T, Collins J J, Carlo D L J. Reconstruction expansion as a geometry-based framework for choosing proper delay times[J]. Physica D, 1994, 73: 82-98

(上接第 144 页)

结束语　本文在分析国内外的工作基础之上，针对已有的 Ctree 进行改进，提出了一种以结点路径相同为原则的索引结构 FC-Index，以对 XML 文档进行规范的结构处理，从而提高查询效率，针对合并后的结构，提出一种有效的查询算法，以对合并后的元素进行有效的查询。基于不同数据集的试验结果表明，本文提出的基于 FC-Index 的查询处理方法可以有效提高查询效率。

参 考 文 献

[1] XMARK(TheXML-benchmarkproject)[OL]. http://monetdb.cwi.nl/ xml

[2] Zou Qinghua, Liu Shaorong, Chu W W[C]// Proc. Ctree: A Compact Tree for indexing XML Data. Washington, DC, USA, 2004: 12-13

[3] Kaushik R, Bohannon P, Naughton J F, et al. Covering indexes for branching path queries[C]// SIGMOD. 2002

[4] Kaushik R, Bohannon P, Naughton J F, et al. Updates for structure indexes[C]// VLDB. 2002

[5] Kaushik R, Krishnamurthy R, Naughton J F, et al. On the integration of structure indexes and inverted lists[C]// SIGMOD. 2004

[6] Kaushik R, Shenoy P, Bohannon P, et al. Exploiting local similarity for efficient indexing of paths in graph structured data[C]// ICDE. 2002

[7] Dong Xin, Halevy A. Indexing Dataspaces[C]// SIGMOD. 2007

[8] Goldman R, Widom J. Enabling query formulation and optimization in semistructured databases[C]// On VeryLargeData Bases (VLDB). 1997

[9] Milo T, Suciu D. Index structures for path expressions[C]// ICDE. 1999

[10] Li Q, Moon B. Indexing and querying XML data for regular path expressions[C]// VLDB. 2001

[11] Kaushik R, Krishnamurthy R, Naughton J F, et al. On the Integration of Structure Indexes and Inverted Lists[C]// SIGMOD. 2004