改进XISS索引技术的仿真研究

王 锦 何先波 贺春林
(西华师范大学计算机学院 南充 637002)

摘 要 研究了数据库查询优化问题,而XISS索引是XML数据库索引中支持正则路径表达式的典型代表。XISS索引对于长查询路径表达式,要产生大量中间结果,连接操作代价十分高,加大了查询的时间和空间复杂度,导致查询的效率和准确率低。为了提高XML数据库查询效率和准确率,提出一种改进的XISS索引技术。首先引入DTD模式信息,简化编码方式;然后对节点索引结构进行改进,减少中间结果的连接次数,使得查询时间与路径长度无关,提高了查询效率和准确率。最后通过实验对改进前后的XISS索引进行仿真。结果表明,改进的XISS索引缩短了建立索引的时间,加快了查询响应的速度,提高了XML数据库查询的效率和准确率。

关键词 索引结构,查询处理,分解路径表达式
中图法分类号 TP311 **文献标识码** A

Simulation Study on Improved Index Technology for XML Data

WANG Jin HE Xian-bo HE Chun-lin
(College of Computer, China West Normal University, Nanchong 637002, China)

Abstract XISS index is currently typical delegate supporting regular path expression in XML data index. XISS index produces large path expression and the intermediate results for long inquires, so join operation cost is very high, which increases the complexity of query, affects the query efficiency. In order to improve the XML data query efficiency, the article put forward an improved XISS index. Improved XISS index at first introduced DTD schema information to improve the coding method, and then the node index structure was improved, to decrease the intermediate links, make query time not related with path length and improve query efficiency. The contrast experiment on the index and the improved XISS was made, and the results show that the improved XISS index decreases indexed time, accelerated inquire response speed and improves the XML data query result.

Keywords Index structure, Query processing, Decomposition of path expressions

1 引言

随着互联网的发展,网络信息不断积累,形成一个前所未有的超大型数据库。可扩展标记语言(Extensible Markup Language,XML)已经成为互联网上数据交换和集成的标准,网络上大量的数据都保存在XML文档中,从海量信息中快速查询到用户所需信息,是一个最基本的问题,因此基于XML文档索引查询优化技术成为研究热点[1]。

国内外学者针对XML文档查询优化进行了大量、广泛而深入的研究。XML文档索引查询包括路径索引、串的索引和节点索引3种类型。路径索引非常适合简单路径表达式的查询,其对于正则路径表达式,查询效果不理想[2]。串的索引不需要进行连接操作,支持各种查询,灵活性相当强,但处理含有通配符的查询时,需要将其转化成简单路径表达式,这样查询的执行效率就受到影响[3]。节点索引能够支持正则路径表达式,执行效率相当高,其中最具代表性的方法为索引存储(XISS)索引。XISS索引能够快速地确定元素和属性间祖先后代关系,其索引结构由元素索引、属性索引和结构索引3部分组成[4]。XISS索引的基本思想是首先将复杂路径分解为简单路径,然后连接各简单路径的处理结果。但XISS目前存在一些不足,如查询时会产生许多中间结果,节点的连接操作代价将会相当高;同时XISS索引没有考虑到XML文档模式信息,会产生冗余的结构连接操作,导致查询效率低下、查询准确率不高[5,6]。

针对当前XISS索引存在的一些不足,提出一种改进的XISS索引方法。即将文档类型定义(Document Type Definition,DTD)信息引入到XML编码方式中,使文档中的每个元素或属性的编码均携带相应的DTD结构信息;然后合并元素索引和属性索引,并改进查询处理方法,从而提高XML查询效率和准确率;最后对改进的XISS索引查询效率和准确率进行测试。仿真结果表明,改进的XISS索引方法提高了查询效率和准确率。

2 XML文档模型

XML是一种结构化标记语言,XML文档包括了数据和

到稿日期:2011-02-20 返修日期:2011-06-11 本文受四川省科技厅基础应用项目(010JY0151)资助。
王 锦(1963-),男,副教授,主要研究方向为计算机应用;何先波(1971-),男,博士生,教授,主要研究方向为计算机网络与嵌入式系统;贺春林(1971-),男,硕士生,教授,主要研究方向为计算机应用。

标记，XML 文档元素包括属性、子元素和文本内容。子元素可以逐层嵌套，因此一个 XML 文档模型通常被看作是一棵有根、有序、带标记的树，树中的节点表示文档中的元素、属性和文本节点，每一个节点均有一个唯一标签和 ID 号，节点之间的嵌套关系通过树中的边表示[7]。下面是一个 XML 文档片段。

```
〈book〉
〈book〉
〈author. id="001"〉
〈name〉John〈/name〉
〈author〉
〈title〉SQL. Server. 〈title〉
〈section〉
〈section〉
〈title〉Table. and. SQL. Server〈title〉
〈/section〉
〈section〉
〈title〉Table. and. View〈/title〉
〈/section〉
〈/book〉
〈book〉
…
〈/book〉
…
〈/books〉
```

上述 XML 文档片段的树模型如图 1 所示。图中省略了文本节点和节点 ID 号。XML 文档树中节点路径是从文档根节点到该节点所经历的标签序列，标签之间采用"/" 分隔。

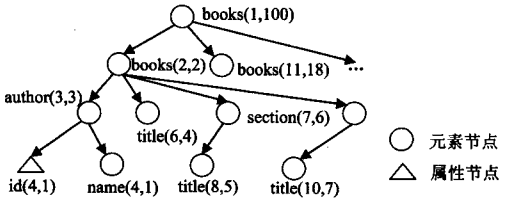

图 1　XML 模型图

3　传统 XISS 索引方法

XISS 采用 B+树作索引，根据相应机制将 XML 文档中名称相同的节点存储在一起，然后通过索引快速查询到全部名称相同的节点。XISS 索引将 XML 数据库的原始查询表达式分解成为一些简单的子表达式，然后分别查询这些子表达式，产生一些中间查询结果；最后将这些结果连接起来，获得查询结果。

3.1　XISS 编码

XISS 采用区间编码技术，每一个节点被赋予一对〈*order*，*size*〉，其中 *order* 表示扩展的前序编码，*size* 表示节点的子孙范围。一对节点的祖先-子孙关系可以通过检验它们的 *order* 和值 *size* 来确定。在图 2 中，每一个节点都由一个〈*order*，*size*〉对标记，这样就确定了一个区间。一个节点的区间可能包含在它的父节点区间中，如节点(25,5)包含在(10,30)和(1,100)中，那么序号为 25 的节点是序号为 10 和 1 的节点的子孙。通过观察这些数据可知，对 XML 文档树中的任意两个节点 x 和 y，如果 $order(x)<order(y)\leqslant order(x)+size(x)$，那么 x 是 y 的一个祖先。

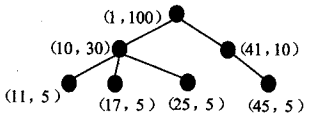

图 2　XISS 的编码方式

3.2　XISS 名字索引和值表

在 XISS 系统中，采用元素或者属性的名字进行搜索，为了达到这种要求，XISS 提供了一系列有效的处理机制。

(1)对于一个固定元素名的字符串，XISS 索引首先找到全部名字相同的元素，然后对文档按各自所属进行分组。

(2)对于一个固定属性名的字符串，XISS 索引首先找到全部名字相同的属性，然后对文档按各自所属进行分组。

(3)对于一个给定元素，可以找到其父元素和子元素；对于一个给定属性，找到其父元素。

由于 XML 数据库中所有值的实体都可以看成是一个可变长的字符串，因此所有不同名字的字符串均被收集到 B+树实现的名字索引中，不同名字的字符串均有一个唯一的名字标识符。这样，在进行名字索引时，就可以避免字符串之间的重复比较，减少了存储和计算方法的开支。采用相同方式将所有的字符串值收集在值表中，为每个 XML 文档分配唯一的文档标识符，该标识符表示查询文档名时的索引关键词。在整个索引系统中，可以根据文档标识及其序号唯一确定一个元素(属性节点)。

3.3　索引结构

XISS 索引包括主结构、属性和结构索引 3 种。元素和属性索引可以快速找到所有相同名称的元素或属性，是最常用的处理路径表达式的操作。每一个元素(属性记录)包括〈*order*，*size*〉和节点的其它信息(节点深度及其父节点的标识符)，按其序号值对元素记录进行排序。在属性索引中引入属性值的标识。结构索引是一个线性表，每个索引记录包含 XML 文档中全部元素和属性固定长度的记录，并按序号值排序。每个文档记录保存着元素名(属性名)的标识以及下一个兄弟、第一个孩子和属性的序号值。

3.4　路径表达式的处理

在 XISS 系统中，采用由 Li Moon 等提出的路径加入算法来处理路径表达式。对于任何复杂的路径表达式，都可以将其分解为简单的子路径表达式，然后分别处理子路径表达式，产生中间结果，最后将中间结果连接起来，组成最终的查询结果。

图 3 具体描述了一个复杂路径表达式 $(E_1/E_2)*/E_3/((E_4[@A=v]|(E_5/-*/E_6))$。图中顶层的叶节点表示只有一个元素或者属性的子表达式，只有一个元素(属性)的子表达式可以通过访问 XISS 系统中的元素(属性)索引而得到；两个子表达式的联合可以通过合并两个中间结果并按文档进行分组而获得解决。

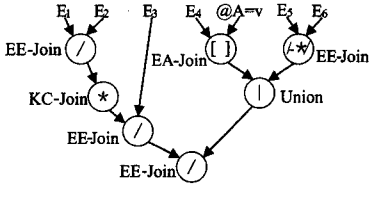

图 3　XISS 的路径表达式分解

4 改进的 XISS

XISS 支持正则路径表达式，但需要对中间结果进行连接操作。这样，对于长的路径表达式，中间结果会很多，连接操作代价将会很高。例如，books//title 只要 1 次连接操作就可以了，但是 books/book/section/title 却需要进行 3 次连接操作。路径索引处理简单路径表达式性能优异，但不能很好地处理正则路径表达式。XML 是对某一类 XML 文档的描述，是一种保证 XML 文档格式正确、有效的方法，但是在传统 XISS 结构中却没有利用 XML 模式这一有效资源，导致查询效率和准确率有时比较低。鉴于此，对 XISS 索引进行如下改进和完善。

(1)分别对 DTD 和 ML 文档进行编码，其中 DTD 用 Dietz 方式编码，XML 文档本身用 Li Moon 方式编码。

(2)索引结构将 XISS 的元素和属性索引进行合并，并带有了 DTD 的信息。

(3)ISS 查询处理过程中，首先查询 DTD 索引，再查询处理 XML 文档。由于 XML 节点索引中带有 DTD 的信息，限制了参加连接算法的 XML 节点，因此查询效果和准确率相应提高。

4.1 编码规则

改进的 XISS 索引编码是分别对 DTD 和 XML 进行编码，其中 XML 和传统 XISS 方法相同，如图 3 所示。DTD 编码将 DTD 树中的节点 n 按照($preorder(n)$, $postorder(n)$, $level(n)$, $element/attribute$)标识，$preorder(n)$表示节点 n 在树中的前序遍历值；$postorder(n)$表示节点 n 在树中的后序遍历值；$level(n)$是节点 n 在树中的层数；$element/attribute$ 记录 n 为元素节点还是属性节点，分别用 0 和 1 表示。

4.2 索引结构

改进的 XISS 索引不仅继承了 XISS 中的名字索引和值表功能，而且对元素、属性和结构索引的不足之处进行了改进。

(1)合并 XISS 的元素和属性索引。路径是从文档根节点到当前节点所经历的标签序列，节点索引按标识符从大到小排序，因此所有最后一个节点同名的路径全部存储在一起。

(2)把路径相同的节点序号值存储在一个列表中，并根据文档号进行分类，通过该列表可以实现结构索引。

(3)结构索引与 XISS 相同，只是增加了一个节点类型的字段，用来对节点按元素或属性进行标识。

4.3 查询处理

4.3.1 路径表达式分解

路径表达式一般可以分解成 3 种子路径表达式。

(1)一个不带谓词的路径表达式，包括简单和正则路径表达式。

(2)最后一个节点带谓词的路径表达式。

(3)两个子路径表达式的连接。

4.3.2 路径查询实现

对 3 种子路径表达式采用下列方式处理。

(1)对不带谓词的路径表达式，仅需在节点索引查找路径中最后一个节点，然后判断节点索引中对应的路径信息是否与查询路径相匹配。若匹配，就沿着该路径所对应的指针在结构索引中找到符合路径条件的节点。

(2)对带谓词的路径表达式，分为去掉谓词和有谓词两部分。采用不带谓词路径表达方法得到一个中间结果，但对于谓词部分产生的结果必须经过谓词判断条件的筛选，然后对两个中间结果集进行连接操作，得到最终结果。

(3)首先分别对两个子表达式进行处理，获得两个中间结果集，然后对两个中间结果集进行连接操作，得到查询结果。

4.3.3 查询处理分析

改进的 XISS 索引充分利用了 DTD 信息，使得 XML 文档中所有节点都携带了 DTD 的相应结构信息。这样对于查询路径，就可以在规模比 XML 文档要小的 DTD 上进行预处理。例如，路径查询/book[@year>="2000"]//name 的查询处理过程如下。

(1)查询路径在 DTD 上进行结构匹配，该路径分解为不带谓词的路径/book//name 和带谓词的路径/book[@year]。

(2)对不带谓词路径调用连接算法，求出满足条件的叶子节点 name 的先序遍历值为 5；求出带谓词路径的目标节点 book 和叶子节点 year 的先序遍历值分别为 1 和 2。

(3)根据 DTD 索引中求出不带谓词路径叶子节点 name 的先序遍历序号值 5，从 XML 节点索引中找出满足条件的所有节点的集合 T。这里有两个节点{name(1,21,10,3,5), name(1,51,10,3,5}。

(4)据 DTD 索引中求出带谓词路径的目标节点 book 和叶子节点 year 的先序遍历值分别为 1 和 2，从 XML 节点索引中找出满足条件的所有节点的集合，A{year(1,3,10,2,2)}和 B{book(1,1,100,1,1))；这里只有一个 XML 文档，因此不需要分组(下同)，并且该节点也满足谓词约束。因此，集合 C 中只含有{year(1,3,10,2,2)}。调用 BC_join 算法，求出节点集 B{book(1,1,100,1)}。

(5)由于只有一个 XML 文档，因此可以直接对集合 B 和集合 T 进行处理。对于节点集合 B{book(1,7,1,0)}和节点集合 T{name(1,16,4,5), name(1,24,4,5)}，求出集合 T 中是节点集 B 中某节点的子孙节点的节点集 P，这个步骤可以通过 BC_join 算法求得。在这个例子中，T 中两个节点都是节点集 B 中 book(1,1,100,1,1)的子孙节点，因此节点集 P 为{name(1,21,10,3,5), name(1,51,10,3,5)}，即路径查询的结果节点集。

5 仿真实验

5.1 实验环境

为了让实验结果具有说服力，采用传统 XISS 作为对比实验。实验所用的测试数据来自 Shakespeare's Play 和 SigmodRecord。实验环境：Intel 3.0G P4 处理器，2G 硬盘，800G 硬盘；操作系统为 Windows XP，程序使用 VC++开发。从文件大小、索引建立时间和查询效率 3 方面对改进 XISS 索引和传统 XISS 索引的性能进行比较。

5.2 系统架构

改进的 XISS 索引是在传统 XISS 索引的基础上进行扩展，并加入了 DTD 索引和相关操作，因此改进的 XISS 索引系统包括装载解析、存储管理、索引管理和查询处理等 4 大模块。所有程序的入口均是一组 shell 脚本，负责装载 XML、查询 XML 路径。改进 XISS 的系统架构如图 4 所示。

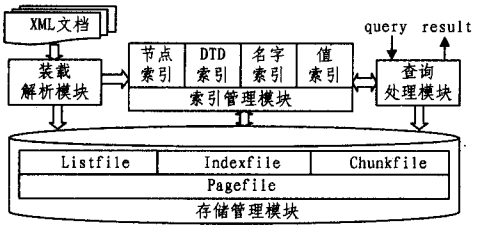

图 4 改进的 XISS 系统结构

5.3 查询测试数据

仿真实验用到的实验数据和查询语句见表 1 和表 2。

表 1 XML 数据集

XML Data	Size (Byte)	Files	Elements	Attribute	DTD File (Byte)
Shakespeare'S play	7.9M	37	327k(22)	0	2k
SigmodRecord	3.5M	989	839k(47)	4775(3)	1k

表 2 查询语句

XML Data	Path Query	Query
Shakespeare'S play	Play/title	Q1
	Play/person group/grades;	Q2
	//act//speech	Q3
	play/act[title="ACTI"]	Q4
	play/scene	Q5
SigmodRecord	sigmodrecord/number	Q6
	//issue/articles/article/title	Q7
	//article/title[articlecode="0001"]/authors	Q8
	/authors[authorposition="01"]	Q9

5.4 结果与分析

5.4.1 建立索引比较

表 3 为两种索引仿真结果。从表 3 对比结果可知，传统 XISS 和改进的 XISS 索引的建立时间和索引大小均相差不大。主要是因为改进的 XISS 索引对 XML 文档和 DTD 同时建立了索引。但由于 DTD 比 XML 文档要小得多，因此它对建立索引总时间影响相当小。

表 3 两种索引建立索引比较

XML Data	XISS		改进的 XISS	
	建立索引时间 (S)	索引大小 (MB)	建立索引时间 (S)	索引大小 (MB)
Shakespeare'S Play	45	24.5	39	21.7
SigmodRecord	38	22.8	37	19.8

5.4.2 查询响应时间

对于两种索引技术的查询响应时间，数据集 Shakespeare'S Play 和 SigmodRecord 的查询结果分别如图 5 和图 6 所示。从仿真结果可知，对于 3 个带有谓词的查询 Q4，Q8，Q9，其查询效率明显提高，简单查询效率也有不同程度提高。因此，改进的 XISS 索引提高了 XISS 索引查询效率。

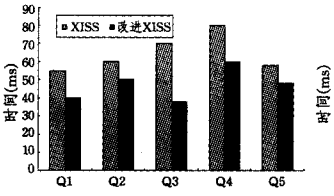

图 5 Shakespeare's Play 查询结果

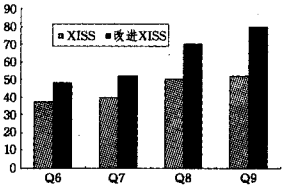

图 6 SigmodRecord 查询结果

结束语 随着 XML 应用的日益广泛，快速准确地查询 XML 文档中的数据已经越来越受到重视。XISS 索引是一种典型的支持正则路径表达式的索引结构，采用查询子路径表达式和中间结果间的连接来对输入进行查询，存在对复杂路径表达式查询效率低的缺陷。本文的 XISS 索引，将模式信息加入到 XML 文档的索引结构中，对 XISS 的索引结构和分解子路径表达式分别进行了改进，最后对算法进行了仿真实验。实验结果表明，改进的 XISS 索引缩短了建立索引的时间，加快了查询速度，从而提高了查询效率。

参 考 文 献

[1] 春平，王超，张鹏. XML 编程从入门到精通[M]. 北京：北京希望电子出版社，2002

[2] 王静，孟小峰，王宇，等. 以目标节点为导向的 XML 路径查询处理[J]. 软件学报，2005，16(5)：827-837

[3] 张鹏，冯建华，房志峰. 一种基于二叉树的 NativeXML 数据库文档编码机制[J]. 计算机应用，2008，28(9)：2331-2334

[4] 张博，耿志华，周傲英. 一种支持高效 XML 路径查询的自适应结构索引[J]. 软件学报，2009，20(7)：1812-1824

[5] 颖捷. XML 索引与查询的若干关键技术研究[D]. 上海：复旦大学，2008

[6] 秦玉杰，李革，黄柯棣. 基于 XML 技术的三维仿真模型的存储[J]. 计算机仿真，2005，22(12)：4-7

[7] 刘振中，董道国，薛向阳. 对 XML 数据索引的回顾[J]. 计算机科学，2004，31(4)：78-83

[8] 陈荣鑫. 基于函数式中间语言的 XML 查询并行化[J]. 重庆理工大学学报：自然科学版，2011，25(7)：81-86

(上接第 137 页)

MPI 和 Hadoop 两种编程模式来进行新的算法搭建，不仅充分利用了这两种技术的优势，同时将多种设计中的优化思想吸收入到该算法中，提高了已有的 K-Means 聚类算法的效率。实验结果证明，所采用的思想和具体实现是有效的，希望对后续研究开发者有启发和应用价值。

参 考 文 献

[1] Wikipedia. K-Means clustering[EB/01]. http://en.wikipedia.org/wiki/K-Means

[2] 崔建，李强，杨龙坡. 基于垂直数据分布的大型稠密数据库快速关联规则挖掘算法[J]. 计算机科学，2011，38(4)：216-220

[3] 郑湃，崔立真，王海洋，等. 云计算环境下面向数据密集型应用的数据布局策略与方法[J]. 计算机学报，2010，33(8)：1472-1480

[4] 王鹏，孟丹，詹剑锋，等. 数据密集型计算编程模型研究进展[J]. 计算机研究与发展，2010，47(11)：1993-2002

[5] 冯丽娜. 并行 K-Means 聚类方法在简历数据中的应用研究[D]. 云南：云南大学，2010

[6] 杨宸铸. 基于 HADOOP 的数据挖掘研究[D]. 重庆：重庆大学，2010

[7] Kantabutra S, Couch A L. Parallel K-Means Clustering Algorithm on NOWS[J]. Technical Journal, 2000, 6(1): 243-247

[8] Forman G, Zhang B. Distributed Data Clustering can be Efficient and Exact[J]. SIGKDD Explorations, 2000, 2(2): 34-38

[9] Boutsinas B, Gnardellis T. On Distributing the Clustering Process[J]. Patter Recognition Letters, 2002, 23(4): 999-1008

[10] 梁红，李伟生. XML 文档的并行聚类算法[J]. 计算机科学，2004，31(10)：243-245

[11] Quinn M J. ParM: Pallel Programming in C with MPI and OpenMP[S]. Beijing: Tsinghua University Press, 2005