第 33 卷 第 7 期 计 算 机 工 程 2007 年 4 月
Vol.33 No.7 Computer Engineering April 2007
·软件技术与数据库· 文章编号：1000—3428(2007)07—0062—03 文献标识码：A 中图分类号：TP311.52

一种基于 XML 的 CMS 元数据索引算法

肖 晨，武东英，郭绍忠，陈 新
（解放军信息工程大学信息工程学院，郑州 450002）

摘 要：内容管理系统(CMS)的元数据管理是其核心模块，负责记录内容的描述信息和帮助其他模块快速定位相关内容，元数据索引是为检索提速的关键技术。根据 XML 语言的特点，可采用它作为元数据的描述语言，因此元数据索引问题转变为 XML 文档的索引问题。该文在对 XML 数据进行形式化描述的基础上，分析了 XML 索引算法的特点，针对 CMS 元数据的特点，提出了一种支持动态更新的 XML 索引算法。

关键词：内容管理系统；XML；元数据；索引；算法

Index Algorithm of CMS Metadata Based on XML

XIAO Chen, WU Dongying, GUO Shaozhong, CHEN Xin
(Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450002)

【Abstract】 Metadata management is a core module in content management system(CMS), which is responsible for noting description information of content and assisting to find out interrelated content quickly for other module. Metadata index is a kind of key technology to increase search speed. According to the character of XML, it is adopted to become description language of metadata. So the question of metadata index is converted into XML document index. This article describes XML data formally, and analyzes the character of algorithm of XML index and puts forward a kind of algorithm of XML index that supports dynamic update.

【Key words】 Content management system (CMS); XML; Metadata; Index; Algorithm

元数据管理模块是内容管理系统中的核心模块之一。元数据(Metadata)一词最早出现于美国航空与宇航局NASA的《目录交换格式》DIF手册中[1]。国际图联IFLA对元数据的定义是：“元数据就是关于数据的数据，此术语指任何用于帮助网络电子资源的识别、描述和定位的数据”[1]。一个元数据由许多完成不同功能的具体数据描述项构成，具体的数据描述项又称元数据项、元素项或元素，用于说明数据内容、质量、条件、查询和其他有关特征的背景信息。元数据主要有描述、检索、定位、管理、评估和交互等几大功能。

XML是W3C组织的XML工作组于 1998 年定义的一个SGML的一个受限子集。XML具有灵活性、自描述性和可扩展性等特点[2]。用XML描述元数据，具有以下的优点：(1)由于XML是一种可扩展的标记语言，因此可以方便地用来表达元数据；(2)XML的样式语言可实现元数据间的转换和显示；(3)利用XML可方便地实现元数据的查询；(4)有众多大的软件厂商提供对XML的支持[3]。因此采用XML进行元数据的描述就把元数据管理转变为对XML文档的管理。

元数据索引旨在针对大量的元数据描述文档进行索引，以提高元数据检索的效率。在基于内容的海量数据管理系统中，元数据描述文档是海量的，高效的索引算法对基于内容的海量数据管理系统来说，也十分重要。

1 XML 数据和索引模型的定义

采用 XML 来描述元数据，对元数据进行索引，也就是对 XML 描述文档进行索引。下面给出有关 XML 数据模型和数据索引模型的定义。

定义 1 (XML数据模型) XML数据可以用一棵带标记的有向树T_{xml}，$T_{xml}=(V_T,E_T,root,L)$表示，$V_T=V_{leaf}\cup V_{non-leaf}$，其中，$V_{leaf}=V_{A-leaf}\cup V_{C-leaf}$、$V_{A-leaf}$为所有的属性节点集合，$V_{C-leaf}$为除属性之外的所有叶节点集合，$V_{leaf}$为所有的叶节点集合，$V_{non-leaf}$为所有的非叶节点集合，$E_T\subseteq V_{non-leaf}\times L\times V_T$，$L$为所有边上的标记集合，其中所有的连接属性节点的边上的标记以@开头，E_T为树中所有边的集合；$root$为T_{xml}的根节点；T_{xml}中的每个节点都根据其在数据中出现的顺序被赋予一个都有的序号oid。

定义 2 (标记路径)标记路径由一串以“/”相隔的连续的标记组成，形如$l_1/l_2/.../l_k$，其中$l_i\in L,1\le i\le k$。

定义 3 (后向标记路径与前向标记路径)由T_{xml}中某一节点v_i开始向根节点方向伸展的标记路径称为节点v_i的后向标记路径，相反由v_i开始向叶节点方向伸展的标记路径称为节点v_i的前向标记路径。

XML数据$T_{xml}=(V_T,E_T,root,L)$的索引可以用一个图型结构$Index(T_{xml})=(V_G,E_G,root,L)$表示，$Index(T_{xml})$中的每个节点对应于$T_{xml}$中一组满足一定结构约束规则的节点集，$Index(T_{xml})$的根节点与标记集则分别等同于$T_{xml}$中的根节点与标记集。

定义 4 (XML数据索引模型)对于一棵XML树$T_{xml}=(V_T,E_T,root,L)$，其索引可以用有向图

基金项目：军队基金资助重点项目
作者简介：肖 晨(1978－)，女，硕士生，主研方向：内容管理，数据挖掘和网格计算；武东英、郭绍忠，副教授；陈 新，讲师、博士生
收稿日期：2006-10-25 **E-mail：**mars2710@163.com

$Index(T_{xml}) = (V_G, E_G, root, L)$ 表示，其中*root*对应于T_{xml}中的*root*，L等同于T_{xml}中的L，$Index(T_{xml})$ 中的每个节点 $v \in V_G$ 对应于 T_{xml} 中的一个节点集 $content(v) \subseteq V_T$，并满足：(1) $\cup_{v \in V_G} content(v) = V_T$；(2) $content(v)$ 中的所有节点满足相同的结构约束规则 S_RULE；(3) 对于 $u, v \in V_G$，有 $content(u) \cap content(v) = \phi$；(4)对于 T_{xml} 中的任意一条边 $(u_T, v_T) \in E_T$，在 $Index(T_{xml})$ 中必然存在一条边 $(u_G, v_G) \in E_G$ 与之对应，并满足 $u_T \in content(u_G), v_T \in content(v_G)$。

由以上的定义可知，各种索引方法的不同关键在于条件(2)中所采用的结构约束规则的不同。

2 基于 XML 的元数据索引算法

索引系统的主要目的是高效地支持查询操作，由于 XML 文档的信息主要包括值信息和结构信息，对 XML 文档的查询也主要包括两个方面：值查询与结构查询。前者包括文档的名称，元素的名称/值，属性的名称/值匹配；后者主要包括结点之间的祖孙父子关系匹配等。对于 XML 索引系统来说，它需要承担的两个主要作用就是高效地支持上述两类查询。

Lore[5]代表了较早期的XML索引方案及查询评估方法的研究，其主要的查询评估对象为规则路径表达式。Lore索引系统的主要问题在于索引结构会随着XML文档规模的增大而急剧膨胀。

另一类早期索引方案是一种精确记录XML文档中出现所有路径模式的索引方案，其中最具代表性的是DATAGUIDE[6]。DATAGUIDE是一种对XML文档实例出现的全部路径模式进行精确的记录的索引方法，假定$p_1,p_2,p_3,...$是XML文档D的全部路径模式，那么D的DATAGUIDE索引就是对$p_1,p_2,p_3,...$的一个整合，DATAGUIDE同时也是一种树型结构，在该树型结构中，每一个节点对应于XML文档中出现的一种路径模式，而两个节点之间以 1 标识的连接表示两个节点的路径模式p，q满足p=q.1(假设p的嵌套深度比q大 1)。该类型索引的一个缺陷是当文档树缺乏规律并且深度嵌套时可能会造成索引规模过大。图 1 是针对图 4 XML文档建立的DATAGUIDE。

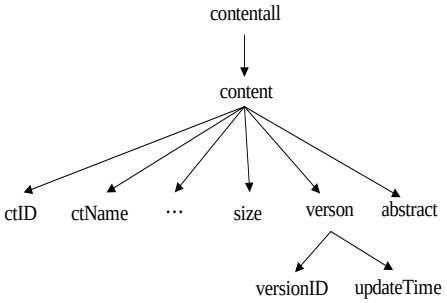

图 1 针对图 4 XML 文档建立的 DATAGUIDE

近年来对XML文档索引结构及规则路径表达式和树型模式表达式评估方法的研究主要包括一种被称为节点数字标识的索引方案及相关结构化连接方法[7,8]，其核心部分使用包含文档节点位置信息的数字标识对XML文档中的节点进行数字标识命名，数字标识采用的基本形式为(DocID，L：R，Z)，这里，DocID表示该节点所在的文档标识，L或R表示该文档节点的先序遍历序号与后序遍历序号，Z表示该节点在XML文档中的嵌套深度。通过一个定理来进行节点祖孙关系的判定：即节点A是节点B的祖先节点当且仅当A的先序遍历序号小于B的先序遍历序号且A的后序遍历序号大于B的后序遍历序号。

文献[9]针对前面结构化连接中采用的索引结构及查询评估方法中的问题提出了一种结合 DATAGUIDE 的数字标识方案，该数字标识方案使用一种被称为 PID 的数字标识方法对文档节点进行数字命名，PID 为 XML 文档中的每一个节点记录从根节点到该节点的文档路径上每一元素节点的同元素节点位置，为了节省 PID 所占用的存储空间，每一层次所使用 bit 数分别进行控制，图 2 是 PID 方案的简单示意。

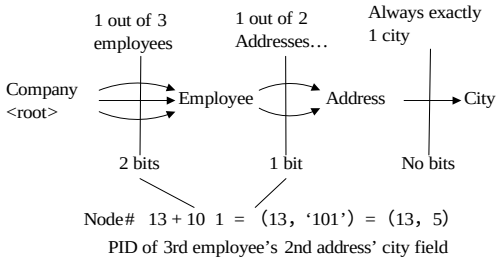

图 2 PID 方案的简单示意图

从图 2 中可以看出，一个公司的第 3 位职员的第 2 个地址所在的城市的节点数字标识为‘101’，由于职员数为 3 个而每个职员最多有两个地址，所以 Employee 的同元素位置使用两个 bit 而 Address 的同元素位置使用一个 bit，而每一层次元素所使用的 bit 数量记录在 DATAGUIDE 的相应层次的元素类型节点上。

但该索引也存在其自身的问题：

(1)文档结构更新效率：当子孙节点数量极为庞大时，效率将急剧降低，这是因为该方法在降低了不同元素类型兄弟节点之间的数字标识的耦合性的同时又大大增强了父子祖孙节点之间的耦合性。

(2)文档加载效率：加载 XML 文档时，全部节点 PID 的建立需要对 XML 文档进行两次遍历，当 XML 文档规模很大时会大幅增加文档加载所需要的时间。

(3)父子祖孙关系的判定效率:相比上文先序及后序节点数字标识方法，该数字标识方法中父子祖孙关系的判定时间与两个节点中层次较低节点的层次成正比关系，而先序及后序节点数字标识方法为常数时间。

3 CMS 中 XML 元数据索引

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation>
  <xsd:documentation xml:lang="zh-cn">
  This is description document shema of content metadata.
  </xsd:documentation>
</xsd:annotation>
 <xsd:element name="content" type="ContentType" />
 <xsd:element name="comment" type="xsd:string" />
 <xsd:complexType name="ContentType">
   <xsd:sequence>
    <xsd:element name="ctID" type="unsignedLong" />
     <xsd:element name="ctName" type="string" />
     <xsd:element name="ctTypeName" type="ContentType" />
     <xsd:element name="localPath" type="string" />
     <xsd:element name="url" type="string" />
     <xsd:element name="size" type="long" />
     <xsd:element name="version" type="versionIdType" />
     <xsd:element name="abstract" type="string" />
     <xsd:element ref="comment" minOccurs="0" />
   </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="versionType">
   <xsd:sequence>
     <xsd:element name="versionID"type="xsd:ID" />
     <xsd:element name="updateTime"type="xsd:dateTime" />
   </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

图 3 内容元数据的XML Schema

对于内容管理系统中内容的XML描述数据，其结构特征是相对固定的，不需采用DifX[4]的动态XML数据索引算法。内容管理系统对于内容的元数据描述，一般包括：内容标识，内容名，内容类型，文件名，本地路径，外部路径，版本号和更新日期，内容摘要等。用XML Schema为内容的元数据描述建模如图 3。

图 3 是针对一个内容的 XML 描述文档模式，由于对 XML 数据进行存储管理时，通常将 XML 数据存放在 XML 文件中，为了减少文件操作的次数，可以将多个内容的 XML 描述数据合并存放在一个 XML 文件中(这里略去了合并后文档的 XML 模式)。图 4 列出了存放多个内容的 XML 描述数据的 XML 文档实例。

```
<contentall>
<content>
 <ctID>01</ctID>
 <ctName>苏州风景图片1</ctName>
 <ctTypeName>.bmp</ctTypeName>
 <localPath>file:///e:/wwwroot/images/1.bmp</localPath>
 <url>http://www.xxgc.mtn/images/1.bmp</url>
 <size>2859210</size>
 <version>
  <versionID>1.0.0</versionID>
  <updateTime>2006-06-14T13:20:00.000+08:00</updateTime>
 </version>
 <abstract>这是肖晨2006年5月在苏州拍的照片</abstract>
</content>
<!--下面为省略的其他内容的元数据描述-->
<!--每个XML文档约有1 000个内容描述信息-->
…
```

图 4 一个 CMS 元数据 XML 文档

图 5 给出了图 4 XML 元数据文档的简化树结构。

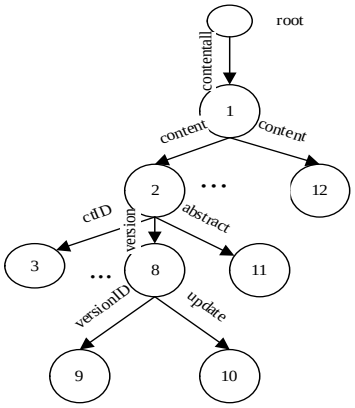

图 5 XML 元数据文档的简化树结构

根据定义 1~定义 4 可很容易地得到图 5 XML 源数据的标记分离索引和完备索引。标记索引的优点是占用空间少，容易建立，但只能有效地支持很少的一部分查询。而完备索引的优点是可以支持任意路径表达查询，其缺点是难以建立，而且会占用很大的存储空间。针对 CMS 中的 XML 元数据的特征，对标记分离索引算法进行了相应的改进。XML 源数据树层次不高，同时结构固定，采用动态的统计聚合标记值的办法，就能取得好的索引和检索的效果。

在内容管理系统中，用户将动态地增加、删除、更新内容数据，同时描述该内容的元数据也将被及时修改。在用户添加新的内容时，如何将新的 XML 元数据索引合并到总的索引中，以及用户要删除和更新内容时，如何快速定位内容的 XML 元数据，用户更新并提交内容后，如何重新为修改后的 XML 元数据建立索引，都是内容管理系统中 XML 元数据索引与其他 XML 数据索引所不同的。

为此，我们采用一种支持动态更新的 XML 索引算法，常规的索引粒度为节点级的算法需要给文档中的节点编号，但虚拟节点编号方式受限于基的取值，一旦文档更新频繁，索引的更新代价则很大。采用基于 LNON 型的倒排索引文件，当文档中的某部分更新时，仅更新索引中相关的部分而不是整个文档。基本思想是在创建索引的时候，解析文档抽取结构和内容信息，将结构信息和内容词条信息都以倒排列表形式出现并且保存到关系数据库中。为了支持文档的更新，对于内容索引表，为词条内容和元素 GID 值分别建立索引，分别作为水平方向上的索引和垂直方向上的索引(B 树实现)。另外，引入元素索引表，使索引结构支持非绝对路径的查询和检索。索引的建立过程主要分为 3 部分：(1)原文档的预处理；(2)以倒排列表形式输出内容信息和结构信息；(3)将表结构信息存入到关系数据库中。主要的数据库表有：UID 索引表，内容索引表，结构索引表，元素索引表，路径索引表和属性索引表。本算法的特点是支持文档的动态更新，采用新的编码方案和索引建立方法，减少结构文档的频繁更新所导致的索引操作的代价。支持的主要更新操作有插入元素、删除元素和内容改变。查询过程与 BUS 索引的 bottom_up 过程相似。用户查询可以被分为基于内容的和基于结构的查询处理。基于内容的查询使用关键词，对布尔型和等级型检索都支持。基于结构的查询使用逻辑上的树型结构，支持对命名路径的查询。

4 实验数据

我们部分实现了该算法的原型，可以存储、索引和查询 XML 文档集合。该原型系统是在 P4 2.8GHz，512MB 内存，Windows XP 操作系统的机器上实现的。使用 SQL Server 2000 作为存储的数据库，用 Java 编程实现。实验数据为基于内容的海量数据管理系统的 XML 内容元数据。实验数据如表 1。

表 1 实验结果数据

所用时间(s)	contentall/content [苏州]	contentall/content/abstract[苏州]	contentall/content/ctName [苏州]
1 个文档	19.562	14.234	10.453
5 个文档	22.826	17.836	14.947
10 个文档	29.462	22.943	17.439
15 个文档	36.037	25.736	22.937
20 个文档	46.025	29.495	24.847
27.532	27.532	27.532	27.532

由实验数据分析可知，对于同一文档，含有内容部分的结点与结构部分目标结点相差越远，查询处理所需要的时间就越多；由于采用关系数据库实现索引结构的存储，因此查询处理的时间与数据量成正比。

5 结束语

针对内容管理系统中内容元数据的特点，本文设计了一种支持动态更新的 XML 索引算法，并实现了原型系统。根据实验结果数据的分析，对基于内容的海量数据管理系统来说，查询代价还太大。下一步将优化关系数据库关于索引结构的设计，实现混合存储结构，降低海量数据的查询成本。

参考文献

1 杨艳丽. 元数据与网络信息资源的管理[D]. 太原: 太原理工大学, 2003-05.

2 鱼　滨. 基于 XML 的集成中间件技术研究[D]. 西安: 西北大学, 2003-05.

（下转第 67 页）