第 33 卷 第 2 期 计 算 机 工 程 2007 年 1 月
Vol.33 No.2 Computer Engineering January 2007
·软件技术与数据库· 文章编号：1000—3428(2007)02—0062—03 文献标识码：A 中图分类号：TP311

基于 LDAP 的 XML 数据管理研究

饶 辉，贺贵明
(武汉大学计算机学院，武汉 430072)

摘 要：提出一种将 XML 数据解析为 DOM 节点后，把 DOM 节点映射为 LDAP 目录项，使用 LDAP 服务器储存 XML 数据的方案；在 LDAP 目录服务上实现了对 XML 数据的 XPath 查询。实验数据说明了方案的可行性。对比实验表明方案除了具有较快的储存速度，在 LDAP 上实现的 XPath 查询效率高于传统的基于 DOM 的 XPath 查询效率。

关键词：LDAP；XML；DOM；XPath

Research on Management of XML Data Based on LDAP

RAO Hui, HE Guiming
(School of Computer, Wuhan University, Wuhan 430072)

【Abstract】This paper proposes an approach to store XML data in LDAP server through mapping DOM nodes to LDAP entries. It also implements XPath queries in LDAP directory service. The feasibility of the approach, as well as its comparatively high store rate is backed up by experimental data. The more important thing is that the efficiency of evaluation XPath queries in LDAP system is better than traditional DOM-based implementations, which is also supported by the comparative experiments.

【Key words】LDAP; XML; DOM; XPath

XML已成为Internet上数据描述和信息交换事实上的标准，在Internet上得到了广泛的应用，产生了大量的XML数据。如何有效地管理(存储、检索)这些XML数据，现在提出的主要方法是借助于成熟的数据库技术，如关系型数据库、面向对象数据库和XML数据库等，但都处在研究和探索阶段，都有各自的缺点和不足[7]。另一方面，随着LDAP V3 (Lightweight Directory Access Protocol V3)的发布，基于LDAP的网络目录服务在Internet上应用的越来越普遍，LDAP已经成为系统集成和网络管理的一个强有力的工具。但是由于历史的原因，LDAP使用专用的LDIF（LDAP Data Interchange Format）来进行目录信息的交换[4]，并没有使用XML作为数据交换的格式。因此基于XML的应用并不能直接访问LDAP目录信息。现在比较成熟的方法是借助于目录服务标记语言(Directory Services Markup Language, DSML）作为中介来沟通LDAP和XML[1]。

考虑到 XML 和 LDAP 在结构上的极大的相似性（本质上都是层次型的树状结构），可以将这两种技术结合起来，发挥各自的优势来解决 XML 数据的管理问题和 XML 应用直接访问 LDAP 目录服务的问题。本文对使用 LDAP 来管理 XML 数据进行了研究，改进了文献[2,3]的方法，建立了 DOM 节点到 LDAP 目录项（Entry）的映射和转换模型（存储 XML 数据）和基于 LDAP 的 XPath（XML Path Language)查询模型（查询）。通过实验验证了该方案的可行性。需要特别指出的是，由于 LDAP 在查询上的高效性，对比实验还表明，LDAP 上 XPath 查询效率明显高于传统的基于 DOM 的 XPath 查询效率。

1 XML 数据在 LDAP 上的存储

鉴于 XML 数据和 LDAP 结构和逻辑上的极大相似性，可以使用 DOM 解析器将 XML 数据解析为 DOM 树，然后将 DOM 节点映射为相应的 LDAP 目录项保存在 LDAP 数据库中来储存 XML 数据。

这样我们必须在 LDAP 模式中定义新的描述 XML 数据的对象类。幸运的是 LDAP 模式具有良好的可扩展性。我们可以在 LDAP 模式中定义少量的（13 个）新的对象类，在基本不改变 LDAP 模式的情况下，在 LDAP 服务器中储存 XML 数据。系统结构如图 1 所示。

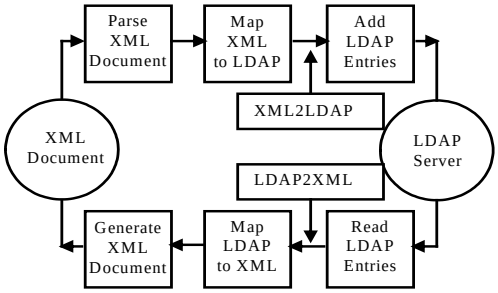

图 1 XML 文档存储在 LDAP 中示意图

XML文档一般由元素、CDATA、注释、处理指令等组成。DOM将XML文档解析成 12 种节点类型，分别为Element、Attr、Text、CDATASection、Entity、EntityReference、ProcessingInstruction、Comment、Document、DocumentType、DocumentFragment和Notation[5]。我们对这 12 种节点类型定义了 12 种LDAP对象类，分别为XMLElement、XMLAttr、XMLText、XMLCDATASection、XMLEntity、XMLEntity Reference、XMLProcInstruction、XMLComment、XML Document、XMLDocumentType、XMLDocFragment 和

作者简介：饶 辉(1978－)，男，硕士生，主研方向：多媒体网络通信；贺贵明，教授、博导

收稿日期：2006-02-28 **E-mail：**rao_hui@163.com

XMLNotation。

定义 XMLNode OBJECT-CLASS :: = {
SUBCLASS OF { top }
MUST CONTAIN {oc, oid, name}
TYPE oc OBJECT-CLASS
TYPE oid DN
TYPE name STRING }

XMLNode 作为这 12 种节点对象类的“基类”，所有的节点类型对象类都定义在 XMLNode 对象类上，是 XMLNode 的“子类”。限于篇幅这里只给出最常用的 XMLElement（Element 节点）和 XMLAttribute（Attr 节点）的对象类定义。

定义 XMLElement OBJECT-CLASS :: = {
SUBCLASS OF{XMLNode}
MUST CONTAIN {order}
// order 是按 XML 文档的顺序解析出的节点次序。
MAY CONTAIN {value}
TYEP order INTEGER
TYPE value STRING }

定义 XMLAttr OBJECT-CLASS :: = {
SUBCLASS OF {XMLNode}
MUST CONTAIN {value}
TYPE value DN, STRING }

这样 DOM 节点可以方便地映射为 LDAP 目录项，从而将 XML 数据存储在 LDAP 数据库中。我们用 C 语言编写了名为 XML2LDAP 的程序实现 DOM 节点到 LDAP 目录项的映射，其逆映射程序为 LDAP2XML。

2 XPath 查询在 LDAP 上的实现

XPath是对XML文档进行定位和检索的基础语言[6]。现在XML主要的查询技术如XQuery，XPoint等都是以XPath查询作为基础，因此主要研究在LDAP上对存储在LDAP数据库中的XML数据进行的XPath查询的问题。

2.1 LDAP 查询模型

LDAP中主要定义了的查询、更新、认证 3 类操作[4]。其中查询操作是最常用的操作。这里重点介绍LDAP的查询模型，后面的分析将要用到这个模型。

定义LDAP查询（LDAP Query）Q_L是一个四元组：$Q_L = (b_L, s_L, f_L, p_L)$，其中$b_L$是搜索基（Search Base），表示搜索开始的位置，一般是一个目录项DN。s_L（Search Scope）是搜索范围，指定了搜索DIT的深度。有 3 种选择：base ，只查询搜索基；onelever，只搜索基的直接子节点；subtree，搜索基的所有子节点。f_L是搜索过滤条件(Search filter)指定目录对象必须满足的条件，搜索过滤器是属性值断言(Attribute Values Assertion)的布尔组合。属性值断言是型如（a op t）的逻辑判断，其中a是属性名，op是关系运算符（≠，=，<，>，≤，≥），t是属性值。p_L是返回属性（可选），指定满足搜索条件的目录对象应该返回哪些属性。

2.2 XPath 查询模型

XPath 是基于路径（Path）的查询，它从称为轴（axis）和节点测试产生的初始节点集合中，选择由判定词过滤后的节点集合作为查询结果。文献[6]对 XPath 进行了详细的介绍。

定义 1 XPath查询：一个XPath查询$Q_X = q_0 / q_1/q_2/\ldots/q_n$，其中$q_i$是一个XPath查询的子查询。

定义 2 XPath子查询：$q_i = (C_i, w_i, C_{i+1})$，其中$C_i$是一组XML节点（集）确定查询的上下文。$w_i$是XPath 表达式，$C_{i+1}$是一组在$C_i$上应用$w_i$产生的结果节点，也叫输出上下文。

定义 3 XPath 表达式：$w_i = a_i :: e_i \ [c_i]$。a_i是轴，它指定了路径选择的节点与上下文节点之间树状关系。XPath定义了 12 种轴关系，如表 1 所示。e_i是节点测试，它指定路径选择节点的节点类型或扩展名称。c_i是过滤节点的布尔表达式断言（可选），它使用专有的表达式进一步细化路径选择的节点集合。

表 1 XPath定义的 12 种轴[6]在LDAP中的实现

n	轴（axis）	b_L	s_L	搜索范围
n	child	n	onelevel	$\{b_L, s_L\}$
n	attribute	n	onelevel	$\{b_L, s_L\}$
n	descendant	n	substree	$\{b_L, s_L\}$
n	descendant –or -self	n	substree	$\{b_L, s_L\} \cup \{n\}$
n	self	n	base	$\{n\}$
n	parent	root	substree	$\{X\}$: 在$\{b_L, s_L\}$中搜索 X(n∈{X, onelevel})
n	following	root	substree	$\{b_L, s_L\} \cap \{order>order(n)\}$且不包含 n 的 descendent 节点
n	following-sibling	root	substree	$\{X,onelevel\} \cap \{order>order(n)\}$
n	ancestor	root	substree	$\{root,X_1,X_{2,\ldots}X_i\}$: $\{b_L, s_L\}$中搜索X_i (n∈{X_i, substree})
n	ancestor -or-self	root	substree	$\{root,X_1,X_{2,\ldots}X_i,n\}$
n	preceding	root	substree	$\{bL, sL\} \cap \{order<order(n)\}$且不包含 n 的 ancestor 节点
n	preceding-sibling	root	substree	$\{X,onelevel\} \cap \{order<order(n)\}$

2.3 基于 LDAP 的 XPath 查询模型

将一个 XPath 查询整体转换为 LDAP 查询，实现起来比较复杂。可以将 XPath 的子查询单独转换为 LDAP 查询，将查询的结果使用类似映射 DOM 节点的方法映射为 LDAP 目录项。这样在 LDAP 中实现 XPath 查询的过程，就是生成一个 LDAP 查询结果树的过程，这个查询结果树的最下层的叶子节点就是 XPath 的查询结果。

定义 XPath 的子查询对象类如下：

定义 XMLQuery OBJECT- CLASS :: = {
SUBCLASS OF{top}
MUST CONTAIN {oc, hash, context, scope, xpathquery, result }
TYPE hash, scope, xpathquery STRING
TYPE context, result DN}

其中hash是一个哈希函数值，它唯一地标识和确定了一个XPath子查询。这样为不同的XPath查询有相同的子查询时可以共享子查询的结果提供了可能。context是查询的输入上下文，也就是前一个子查询的结果节点集（相当于C_i）。scope是查询的搜索范围。xpathquery保存子查询表达式。result是整个查询结果。即在context输入上下文内，运用xptahquery属性定义的查询，在scope范围内查询，结果保存在result内。

在转换过程中，遇到的一个主要问题是 LDAP 只支持向下的搜索，而 XPath 查询能支持向上和向下两种查询。为了更容易地在现有的 LDAP 服务中部署和实施我们的方案，我们改进了文献[2]提出的扩展 LDAP 中搜索范围（Search Scope）的方法。对 XPath 查询中的向上搜索轴（ancestor,parent 等），采用通过改变搜索基（base）和搜索范围（scope）来实现，具体方法见表 1。实验表明这种方法并没有降低搜索效率，因为没有扩展 LDAP 搜索范围的定义，所以更容易在现

有的基于 LDAP V3 的系统上实施。

对于一个XPath子查询q_i，在LDAP中将它转换一个主LDAP查询(main query，M)和若干个精LDAP查询(refinement query，R) 来完成q_i的求值。其中精查询的个数与q_i中的XPath表达式中的判定词（c_i）个数相等。这样一个XPath查询在LDAP中的实现如图 2 所示。

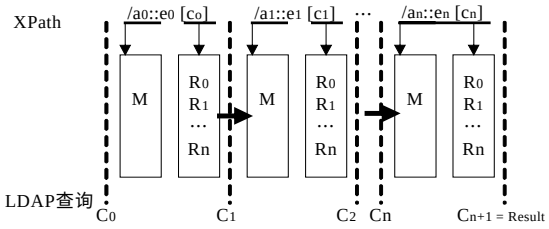

图 2 XPath 查询转换为 LDAP 查询的过程

2.4 算法

算法 XPath2LDAP，实现在 LDAP 上进行 XPath 查询。它通过将 XPath 子查询转换为一个 LDAP 主查询和若干个精 LDAP 查询，来实现整个 XPath 查询。其主要过程如下：

(1)在DIT中搜索到C_0。

(2)将子查询$q_i = (C_i, w_i, C_{i+1}) \in Q_X$ 映射为XMLquery目录项。

1)在DIT中创建XMLQuery节点。其中XMLQuery. context = C_i，XMLQuery.xpathquery = w_i，XMLQuery.hash = hash(w_i)。

2)计算在C_i中的每个节点n的XPath子查询结果，即以节点n为基对XPath表达式$w_i = a_i :: e_i[c_i]$。

在LDAP上求值。这是算法的核心，主要过程是先以n为基求“ $a_i :: e_i$ ”得到LDAP的主查询结果，然后对结果集中的每一个节点，将c_i转换成LDAP属性值断言式（a op t）的形式求LDAP精查询的结果，最后将得到的结果集求并就是节点n的XPath子查询结果。

3)对上一步求出的所有结果求并集，就是q_i的结果C_{i+1}，并赋给XMLQuery.result。

3 结论

我们的方案具有如下的特点：

(1)在 LDAP 上能对任意的 XML 文档进行存储。现在 DOM 在将 XML 文档解析成 DOM 树方面已经非常成熟，可以将任意 XML 文档解析为 DOM 树。这样利用我们的方案可以存储任意的 XML 文档。

(2)在 LDAP 上实现的 XPath 子查询对分布式查询和并行处理能提供很好的支持。我们的方案能记录整个查询路径和过程，能实现对 XPath 查询的应答缓存 (Cache answerbility)，可以有效地提高 XPath 的查询效率。

(3)因为现在 XPath 查询的优化并没有得到很好的解决，XPath 查询的效率仍然较低。而 LDAP 的查询机制非常成熟和完善，它是专门针对查询来设计的，因此其查询效率很高。在 LDAP 上实现 XPath 查询能提高 XPath 的查询效率，我们进行的对比实验也证明了这一点。

(4)我们在 LDAP 上实现了对 XML 数据的存储和访问，因此利用我们的方案基于 XML 应用程序可以直接访问 LDAP 目录服务上的 XML 数据。

为了验证方案的可行性，我们在一台安装 Red Hat Enterprise Linux 3 的 Pentium 4 1.8GHz 的 PC 上进行了相应的实验，采用 OpenLDAP2.3.4（http://www.openldap.org ）作为 LDAP 目录服务软件。实验使用的 XML 数据有：Swiss Prot.xml (http://www.cs.washington.edu/research/ xmldatasets/data/SwissProt.xml),NASA.xml(http://www.cs.washington.edu/research/xmldatasets/data/nasa/nasa.xml) 和 TreeBank_E.xml (http://www.cs.washington.edu/research/xmldatasets/data/treebank/treebak_e.xml)。

首先对存储 XML 数据进行了实验，主要测试用 XML2LDAP 算法将 DOM 树映射为 DIT 的平均存储时间和使用 LDAP2XML 算法将 DIT 还原为 DOM 树的时间。使用开源的 Apache community 的 DOM 解析器 Xerces C++ Parser (Version 2.7.0)(http://xml.apache.org/xerces-c/download. cgi)。这是一个用 C++编写的基于 DOM(Level 2)的解析器。对不同深度的 DOM 树进行了实验，实验结果如表 2，返回时间是指从 LDAP 目录项映射为 DOM Node 的时间。实验结果显示存储速率达到了 3 200 节点/s，从 LDAP 中恢复 DOM 节点的速率达到了 680 节点/s。实验结果相当理想，验证了我们存储方案的可行性。

表 2 存储实验结果

序号	XML 文档	节点数	存储时间(s)	返回时间(s)	存储速率(nodes/s)	返回速率(nodes/s)
1	SwissProt.xml (部分)	78 547	25.8	126.2	3 044.1	622.4
2	SwissProt.xml (部分)	12 845	4.5	19.5	2 854.4	658.7
3	treebak_e.xml (部分)	26 574	8.4	37.6	3 163.6	706.8
4	nasa.xml (部分)	8 940	2.4	12.1	3 725.0	738.9
	平均值				3 196.8	681.7

我们还进行了 LDAP 上的 XPath 查询和传统的基于 DOM 的 XPath 查询效率的对比实验，使用基于 C 语言的开源的 Libxml2（http://www.xmlsoft.org）作为对比实验的 XPath 查询引擎。对 Swissprot.xml 的同一 XPath 查询，测试了它们各自的查询时间。实验结果如图 3。

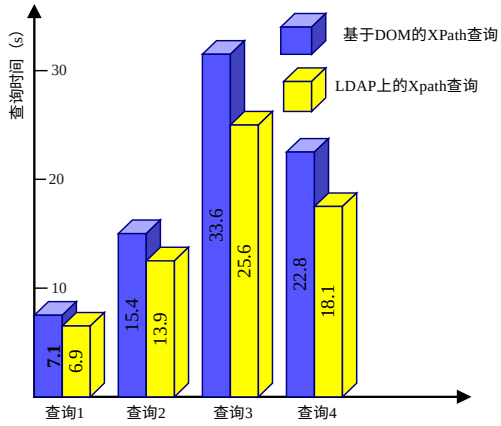

图 3 XPath 查询对比实验结果

从图 3 中可以明显看到基于 LDAP 的 XPath 查询效率在各种条件下都优于传统的 DOM 树的 XPath 查询。其主要原因在于 LDAP 目录服务查询效率非常高，用 LDAP 来进行 XPath 查询，有效地改善了 XPath 的查询效率。

参考文献

1 Koutsonikola V, Vakali A. LDAP: Framework, Practices and Trends[J]. IEEE Internet Computing, 2004, 8(5): 66-72.

2 Marron P J, Lausen G. On Processing XML in LDAP[C]. Proceedings of the 27th VLDB Conference. Roma, Italy: ACM Press, 2001: 601-610.

(下转第 67 页)