XML 文档到关系数据库映射方法的研究 *

耿 飙, 宋余庆, 梁成全, 陈健美
(江苏大学 计算机科学与通信工程学院, 江苏 镇江 212013)

摘 要: 针对现有映射方法对 XML 文档格式要求过严等不足,在模型映射方法基础上提出一种 XML 文档映射关系数据库的新方法。通过给 XML 文档树做标志,将映射算法转换后的数据放到两张预先定义结构的表进行存储。给出了逻辑数据模型、详细设计、映射算法和实验。实验结果表明,该方法能有效地保持 XML 文档的结构,能够对存储的数据进行语义检索,适用于任何树型数据结构。

关键词: 可扩展标记语言文档; 模型映射; 标志; 保持结构; 树型数据结构
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2010)03-0951-04
doi:10.3969/j.issn.1001-3695.2010.03.039

Research on mapping method from XML document to relational database

GENG Biao, SONG Yu-qing, LIANG Cheng-quan, CHEN Jian-mei
(*School of Computer Science & Communication Engineering, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*)

Abstract: To overcome the issues of the existing mapping methods being too strict to XML format requirements, this paper proposed a new method of mapping XML documents to relational database based on the model-mapping method. By making tokens for XML documents tree, data of coversion from mapping method would be stored into the two pre-defined structure tables. Discussed four aspects of logical data model, detailed design, mapping algorithm and experiment. The results of experiment show its ability of maintaining document structure effectively, performing first level sematic search is achievable, being a general solution for any tree data structure can be applied.

Key words: XML document; model mapping; token; maintain structure; tree data structure

0 引言

XML 具有平台无关性、自描述性、可扩展性、简单易于处理等优点,其相关技术的成熟使之成为 Internet 数据表示和交换的标准[1]。随着 XML 应用的不断深入,在 Web 服务、电子商务和数据集成等诸多领域产生了大量的 XML 文档。同时,各种类型的存储方法也不断涌现,比较有代表性的有半结构化数据仓库、文件系统、Native 系统和关系数据库[2]。其中,基于关系数据库技术的成熟及其应用的广泛性,XML 数据存储到关系数据库仍是当前乃至今后很长一段时间内的常用有效手段。

目前,基于关系的 XML 存储的研究受到国内外研究者的重视,发表了一些重要的研究成果,但是总的来说根据存储时是否使用 XML 模式(DTD 或 XML schema)可以分为结构映射方法[3,4]和模型映射方法[5,6]两类。由于与结构映射方法相比,模型映射方法具有以下三个优点:a)支持任何静态(XML 模式不变)或动态(XML 模式不断变化)的 XML 数据存储;b)支持任何格式良好的而没有 XML 模式的 XML 数据存储;c)不需要对数据库模型进行任何扩展就可以支持 XML 的存储。因此,本文就是针对模型映射方法而设计的 XML 文档到关系数据库映射的新方法。

1 相关工作

结构映射方法是在进行关系数据库的 XML 存储时,先根据 XML 模式(或挖掘出 XML 文档中固有的模式信息)生成相应的关系模式,然后再根据生成的关系模式对 XML 文档进行解析分解并将它存放于相应的数据表中。文献[3]的 STORED 方法结合关系数据库技术和半结构化数据处理技术来实现对半结构化数据的管理。文献[7]的 DTD 方法根据 XML 模式信息(DTD)来生成相应的关系模式。文献[8]的 CPI 方法采用混合内联法,充分考虑了 XML 文档的语义信息,完成了从 DTD 到关系模式的映射。文献[4]提出了 P_Schema 思想可以直接映射为关系模式,从而实现了 XML 文档到关系数据库的映射。

模型映射方法是将任何 XML 数据都存放在有固定关系模式的数据库中,而不考虑 XML 文档的模式,其本质就是存储 XML 文档的结构信息。文献[5]提出了做全局(每个节点被分配一个数字代表节点在文档中的绝对位置)、局部(每个节点被分配一个数字代表它在其兄弟中的相对位置)和 Dewey(每个节点给一个标志作为其父标志和私有整型数字的结合)标志。文献[9]提出了 Xrel 方法,用四个固定的表来存储 XML 文档: Path (PathID, Pathexp), Element (DocID, PathID, Start,

收稿日期: 2009-07-01; **修回日期**: 2010-01-29 **基金项目**: 国家自然科学基金资助项目(60841003)

作者简介:耿飙(1983-),男,安徽六安人,硕士研究生,主要研究方向为 XML、数据库、嵌入式系统;宋余庆(1959-),男,江苏镇江人,教授,博导,博士,主要研究方向为数据库技术、医院信息系统、医学图像处理、嵌入式系统(yuqingsong0327@yahoo.cn);梁成全(1982-),男,河南信阳人,硕士研究生,主要研究方向为电子病例系统、嵌入式系统;陈健美(1962-),女,江苏镇江人,副教授,博士,主要研究方向为数据挖掘、医学图像.

End, Ordinal), Text (DocID, PathID, Start, End, Value) 和 Attribute (DocID, PathID, Start, End, Value)。其中的属性 DocID、PathID、Start、End、Value 分别代表 XML 文档的标志、路径表达式、区域起点、终点和字符类型的值。节点区域指的是该节点在文档中某条路径上的起始位置和终止位置,其中暗含了它们之间的包含关系。文献[6]提出了基于群模式的标志 XML 树方法。在该模式中,对一组元素而不是单个元素做标志。元素被分进各个组中,把所有的兄弟节点元素放进一个组,分配一个标志给这个组,代替每个元素给一个标志,然后存储它们到一个关系记录里。

以上关于 XML 存储的研究所提出的方法多种多样,但都不同程度地存在着一些问题。第一种结构映射方法对 XML 文档的格式要求过于严格,耗费大量的数据空间,没有考虑数据库存储及查询方面的性能等因素,而第二种模型映射方法中插入节点后需要重新标志,动态更新非常困难,提取父—子和祖先—后代节点关系很麻烦,不能进行语义检索。为了克服以上问题,鉴于模型映射方法具有的优势,本文从逻辑数据模型、详细设计、映射算法三个方面提出了基于模型映射的 XML 文档到关系数据库映射的新方法。该方法相对于已有的模型方法来说,其优点如下:a)不仅适用于 XML 文档数据,而且还适用于任何树状数据结构,具有通用性;b)增加子树节点(元素和属性),XML 文档原有的标志不需要更改,动态更新非常容易;c)轻易低代价地维持了 XML 文档的原先结构;d)能够对存储的数据进行语义检索。

2　XML 文档到关系数据库映射方法

2.1　XML 逻辑数据模型

定义 1　有序标志树。一篇 XML 文档 D 可以表示为一棵有序标志树 $T=(V,v_0,\Sigma,\text{type},\text{tag},\text{val},\leqslant)$。其中:$V$ 是 XML 节点的集合;$v_0\in V$ 是树的根节点;有穷字母表 Σ 是文档 D 的元素和属性名称组成的集合;函数 $\text{type}:V\rightarrow\{\text{elem},\text{attr},\text{text}\}$ 确定节点类型,若 v 为元素 $\text{type}(v)=\text{elem}$,若 v 为属性 $\text{type}(v)=\text{attr}$,若 v 为文本 $\text{type}(v)=\text{text}$;$V_e=\{v\mid v\in V\wedge\text{type}(v)=\text{elem}\}$ 表示元素节点集合,$V_a=\{v\mid v\in V\wedge\text{type}(v)=\text{attr}\}$ 表示属性节点集合,$V_t=\{v\mid v\in V\wedge\text{type}(v)=\text{text}\}$ 表示文本节点集合;函数 $\text{tag}:V_e\cup V_a\rightarrow\Sigma$ 返回元素或属性节点的名称;函数 $\text{val}:V_a\cup V_t\rightarrow\text{str}$ 返回属性或文本节点的值,str 是 XML 文档中所有合法字符串的集合;二元关系 $\leqslant\in V^2$ 定义 XML 文档顺序,在文档 D 中如果节点 u 出现在 v 之前或 $u=v$,则 $(u,v)\in\leqslant$ 或记为 $u\leqslant v$。

此逻辑数据模型只定义了构成 XML 文档的主要数据,即元素、属性和文本,而忽略处理指令、注释等次要数据,因此 $V=V_e\cup V_a\cup V_t$。元素节点可以有 0、1 或多个子节点,子节点的类型可以是元素、属性或文本;属性和文本节点没有子节点。T 中每个元素和属性节点都被赋予惟一的标志,称为节点 id。对于任意 $v\in V$,其节点 id 记做 $\text{id}(v)$。节点集合可以表示为节点 id 的集合。

例 1,对于如下 XML 文档,图 1 给出了相应的树型结构。

```
〈pub〉
〈book year = "2008"〉
    〈title〉Database System〈/title〉
    〈price〉25.50〈/price〉
    〈author id = "101" sex = "m"〉Kailly Jone〈/author〉
〈/book〉
〈book year = "2009"〉
    〈title〉Introduction to XML〈/title〉
    〈price〉19.80〈/price〉
    〈author id = "102" sex = "f"〉Dan Oja〈/author〉
〈/book〉
〈/pub〉
```

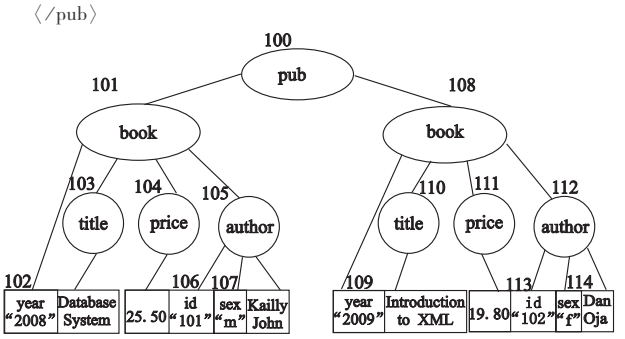

图1　XML文档的树型表示

根据定义 1 可以写出上述 XML 文档的逻辑数据模型 $T=(V,v_0,\Sigma,\text{type},\text{tag},\text{val},\leqslant)$。其中:

$V=\{100,101,\cdots,114\}$;$v_0=100$;

$\Sigma=\{\text{pub},\text{book},\text{year},\text{title},\text{price},\text{author},\text{id},\text{sex}\}$;

$\text{type}(100)=\text{elem},\text{type}(102)=\text{attr},\text{type}(104)=\text{text},\cdots$;

$\text{tag}(100)=\text{pub},\text{tag}(103)=\text{title},\cdots$;

$\text{val}(102)=\text{“2008”},\text{val}(104)=25.50,\cdots$;

$\leqslant=\{(100,101),(101,102),\cdots\}$。

2.2　详细设计

XML 文档映射关系数据库方法基于有序标志树定义的数据模型构建,详细设计如下:将整个 XML 文档看做一个树结构,树中的节点即为 XML 元素、属性和文本等,给 XML 文档节点(元素和属性)赋予惟一的标志。可以对节点进行如先序、中序或后序遍历。如果增加子树节点(元素和属性),XML 文档原有的标志不需要更改。关系模式设置两个表,基于文档本身结构信息的存储放在 documents 主表,基于文档所有内容的存储放在关系表 tags 从表。

1)主表 documents(doc_id, doc_structure)　可以追加新字段到表结构来保存文档本身的所有结构信息。其中:

a)doc_id 是惟一标志的主键字段,由每个文档本身产生来标记该文档;

b)doc_structure 是包含编码字符串的大文本字段,用来描述文档结构,文档结构的任何变化都应该映射在这个字段中。

2)从表 tags(doc_id, tag_id, tag_name, tag_value) 其中:

a)doc_id 是连接 documents 表的外键,反映主从表间的关系;

b)tag_id 是每个标记产生的主键标志字段;

c)tag_name 是标记名或属性名字段;

d)tag_value 是标记属性的文本字段。

构建 doc_structure 字段的规则如下:

a)doc_structure 字段由一系列长序列相关键组成。

b)每个键以字母表中字母为开始,元素用“T”,属性用“A”,必须依照顺序来定界键。字母后面紧跟着一个数值型数

字代表 tag_id。

c)如果元素中有一些属性,那么这个键在 doc_structure 中的标志将由下面的一些键定义这些属性。

d)如果元素中有一些子元素,那么这些子元素将由尖括号括起来的键字符串表示。

2.3　映射算法和示例

本映射算法模型采用 W3C 的文档对象模型(document object model,DOM)来表示内存中 XML 文档,用一个栈来遍历 XML 文档,以相反的顺序把每个元素的子元素压入栈,以便保留它们在 doc_structure 字段里的顺序。映射算法描述如下所示:

```
输入:根据 XML 文档生成的 DOM 树,doc_ id
输出:XML 所有标记插入关系数据库表
初始化栈,根元素入栈;
Do  loop
//构造 doc_structure 字段,将元素和属性标志加入 tags 表
    弹出栈顶元素;
if 栈顶元素为“ > ”then
//父辈元素的所有子元素被写入数据库
    将“ > ”加入 str 字符串中;
else
//栈顶元素是元素
    将元素名和元素值写入数据库表;
    元素标志加入 str 字符串中;
    if 栈顶元素有属性  then
        for  属性集中的每个属性  do
            将属性名和属性值写入数据库表;
            属性标志加入 str 字符串中;
        end  for
    end  if
    if  栈顶元素有子元素  then
        将“ < ”加入 str 字符串中;
        将“ > ”入栈;
        将所有子元素以相反顺序入栈;
    end  if
    if  栈为空  then
        exit  loop
  end  if
end  loop
将 str 字符串写入数据库(documents 表);
```

通过上述算法,所有元素的子元素都被尖括号包围,嵌套的尖括号以区分文档的层次。

为了更详细地阐述上面介绍的映射算法,给出算法示例(XML 文档代码和图 1)。在本文所述的方法里,树里的每个节点(元素和属性)标志假定按照先序遍历的。经过转换之后,这个文档将在 documents 表里有一条记录显示,如表 1 中所示的 doc_id 为 50,表 2 中 tags 表将包含文档内容的所有记录。此时文档对应的 doc _ structure 字段为 T100〈T101A102〈T103T104T105A106A107〉T108A109〈T110T111T112A113A114〉〉。

表 1　Documents 表

doc_id	doc_structure
50	T100〈T101A102〈T103T104T105A106A107〉T108A109〈T110T111T112A113A114〉〉

根据上述方法能很容易地保持文档的结构,如要删除 id 为 101 的作者的 sex 属性,这个属性是 A107,所需做的仅是简单地从 doc_structure 字符串中删除子串 A107 操作。假设需要在已经存在的标记之间新增一个 book 标记,仅仅是在上面的字符串的合适位置做插入操作即可,如新增一本书,其结构如图下所示:

```
〈book year = "2009"〉
    〈title〉Data Structure〈/title〉
    〈price〉22.10〈/price〉
〈/book〉
```

表 2　Tags 表

doc_id	tag_id	tag_name	tag_value
50	100	pub	null
50	101	book	null
50	102	year	2008
50	103	title	Database System
50	104	price	25.50
50	105	author	Kailly Jone
50	106	id	101
50	107	sex	m
50	108	book	null
50	109	year	2009
50	110	title	Introduction to XML
50	111	price	19.80
50	112	author	Dan Oja
50	113	id	102
50	114	sex	f

相应的 tags 表的记录片段如表 3 所示,其等价的键字符串是 T150A151〈T152T153〉。这个新的子串将在合适的位置被插入到 doc_structure 来保留文档的原先次序。此时 doc_structure 字段将变成:

T100〈T101A102〈T103T104T105A106A107〉T150 A151〈T152T153〉T108A109〈T110T111T112A113A114〉〉

表 3　等价的 tags 表

doc_id	tag_id	tag_name	tag_value
50	150	book	null
50	151	year	2009
50	152	title	Data Structure
50	153	price	22.10

3　实验及结果分析

根据上述分析,在微机配置为 CPU Intel Pentium Dual-Core E2180@2.0 GHz,内存 1 GB,硬盘容量 160 GB 下,基于操作系统 Windows XP SP3,采用 Borland C ++Builder 6.0 开发了一个数据转换系统。测试的 XML 文档来自 XML benchmark Xmark[10],通过调节文档生成器 XMLGEN 的比例因子,生成五篇不同大小的 XML 文档。选择的关系数据库类型分别为 Oracle 9i, SQL Server 2005, MySQL 5.0 和 Access 2003 四种。实验测算的是 XML 文档映射关系数据库所需要的时间,时间以 s 为单位。对于每个 XML 文档转换数据库重复五次,记录这些时间的平均值,以便能获得真实、准确的结果。转换性能测试如表 4 所示。

表 4　转换性能测试

文档大小	Access 2003	MySQL 5.0	SQL Server 2005	Oracle 9i
13 KB	0.079	0.068	0.061	0.083
29 KB	0.167	0.145	0.141	0.172
64 KB	0.376	0.305	0.311	0.352
580 KB	2.354	2.842	2.279	3.114
1.3 MB	4.943	5.968	4.786	6.539

测试结果显示：当 XML 文档大小为 13 KB 时，Access 2003、MySQL 5.0、SQL Server 2005 和 Oracle 9i 四种数据库转换消耗的时间分别为 0.079、0.068、0.061 及 0.083 s；依此类推，当文档大小从 13 KB 逐渐增大到 1.3 MB 时，每种数据库映射所消耗的时间亦递增。因此可以得到以下结论：系统能够实现 XML 文档到关系数据库表的映射，而且能轻易地、低代价地保持文档的结构，映射时间是可以接受的，文档大小和映射所需时间呈线性关系。同时也可以看出，上述映射算法是可行的。由于系统是在上述硬件环境下进行的转换，XML 数据转换占用内存比较大，影响了性能，当硬件如内存增大时，系统转换的消耗时间会大大减少。

4　结束语

随着 Web 的广泛应用，XML 正发挥着越来越重要的作用[11]，如何在数据库中有效地存储 XML 文档已经成为人们研究的热点。但由于本身的结构和目前广泛使用的关系数据库不匹配，XML 和关系数据库的结合一直没有很好的解决方案。文中在模型映射法的基础上提出了一种简单的 XML 文档映射为关系数据库的方法。通过此方法，不需要 DTD 或 XML Schema 就能简化映射过程，很容易地以低代价保持文档的结构，能够对存储的数据进行语义检索，可以用于任何树型数据结构，不仅仅是 XML 数据，但它对复杂的混合型 XML 文档还不能完全进行语义检索。以后的工作是改善本方法以完善对复杂的混合型 XML 文档的语义检索的支持，同时还要与其他模型映射方法进行比较来观察其查询性能。

参考文献：

[1] 郑荣，马世龙. 网格环境下基于 XML 的异构数据集成系统[J]. 计算机工程，2008，34(22)：52-54.

[2] 吴爱华，刘小玲，王洪，等. 基于混合映射的 XML 数据的关系存储和查询[J]. 郑州大学学报：理学版，2007，39(2)：157-160.

[3] DEUTSCH A，FERNANDEZ M，SUCIU D. Storing semistructured data with STORED[C]// Proc of ACM SIGMOD Conference. 1999：431-442.

[4] BOHANNON P，FREIRE J，ROY P，*et al*. From XML schema to relations：a cost-based approach to XML storage[C]// Proc of the 18th International Conference on Data Engineering. 2002：64-75.

[5] TATARINOV I，VIGLAS S，BEYER K，*et al*. Storing and querying ordered XML using a relational database system[C]//Proc of SIGMOD. 2002：204-215.

[6] SOLTAN S，RAHGOZAR M. A clustering-based scheme for labeling XML trees [J]. International Journal of Computer Science and Network Security，2006，6(9A)：84-89.

[7] SHANMUGASUNDARAM J，TUFTE K，ZHANG Chun，*et al*. Relational database for querying XML documents：limitations and opportunities [C]//Proc of the 25th International Conference on Very Large Data Bases. 1999：302-314.

[8] LEE D，CHU W W. CPI：constraint-preserving inlining algorithm for mapping XML to relational schema[J]. Data and Knowledge Engineering，2001，39(1)：3-25.

[9] YOSHIKAWA M，AMAGASA T，SHIMURA T，*et al*. XRel：a path-based approach to storage and retrieval of XML documents using relational database[J]. ACM TOIT，2001，1(1)：110-141.

[10] XMark：an XML benchmark project[EB/OL]. http://monetdb.cwi.nl/xml/.

[11] 刘健，马宗民，严丽. 含有效时间时态关系数据库到 XML 映射方法的研究[J]. 计算机科学，2008，35(6)：240-250.

(上接第 950 页)模块远程传递到目的节点。目的节点接收完模块所有信息后，就开始模块恢复工作。它将接收到的模块装载到内核，并向管理软件人注册。至此，模块迁移完毕。模块软件人迁移过程如图 3 所示。

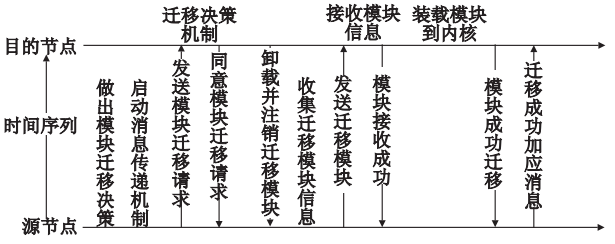

图3　模块软件人迁移过程图

4　结束语

本文提出了基于 Linux LKM 机制的软件人管理系统这个概念。该系统将功能软件人视为一个单独的 Linux 模块，可以动态装载卸载，也可以迁移，实现软件人的高效率调度。功能软件人迁移过程中，系统将部分功能交由内核直接完成，从而降低了系统开销，提高了系统可移植性。

目前，以上工作都已经取得了阶段性的成果。基于 Linux LKM 机制的软件人管理系统还是一个新的概念，对于它的研究尚且存在一些未解问题，比如，如何体现模块软件人的智能性以及迁移完成后软件重构如何实现。笔者在下一步的工作中对这些问题作更深入的研究。

参考文献：

[1] 曾广平，涂序彦. 软件人[C]// 中国人工智能学会第 10 届全国学术年会论文集. 北京：北京邮电大学出版社，2003.

[2] 曾广平，涂序彦，王洪泊. “软件人”研究及应用[M]. 北京：科学出版社，2007.

[3] SHOHAM Y. Agent-oriented programming [J]. Artificial Intelligence，1993，60(1)：51-92.

[4] JENNINGS N R，SYCARA K，WOOLDRIDGE M. A roadmap of agent research and development [J]. Autonomous Agents and Multi-Agent System，1998，1(1)：7-38.

[5] JENNINGS N. On agent-oriented software engineering[J]. Artificial Intelligence，2000，117(2)：277-296.

[6] 焦文品，史忠植. 构造 MAS 的动态体系结构的模型[J]. 计算机学报，2000，23(7)：732-737.

[7] 毛新军. 面向主体的软件开发[M]. 北京：清华大学出版社，2005.

[8] SALZMAN P，BURIAN M，POMERANTZ O. The Linux kernel module programming guide [EB/OL]. (2007-05-18) [2009-07-15]. http://tldp -org/LDP/lkmp.

[9] BOVET D，CESATI M. 深入理解 Linux 内核[M]. 陈莉君，张琼生，张宏伟，译. 3 版. 北京：中国电力出版社，2007.

[10] 毛德操，胡希明. Linux 内核源代码情景分析[M]. 杭州：浙江大学出版社，2001.

[11] HENDERSON B. Linux loadable kernel module HOWTO [EB/OL]. (2006-09-24) [2009-07-15]. http://tldp. org/HOWTO/Module-HOWTO.

[12] CORBET J，RUBINI A，HARTMAN G. Linux device drivers[M]. 3rd ed. [S. l.]：O’Reilly Press，2005.