第 36 卷 第 5 期 计 算 机 工 程 2010 年 3 月
Vol.36 No.5 Computer Engineering March 2010
·软件技术与数据库· 文章编号：1000—3428(2010)05—0040—03 文献标识码：A 中图分类号：TP311.13

XML 文档到关系数据库的映射策略

李思莉[1]，李 娟[2]
(1. 成都理工大学工程技术学院计算机科学系，乐山 614007；2. 遵义医学院医学工程系，遵义 563000)

摘 要：在分析已有数据映射策略的基础上，提出新的 XML 文档到关系数据库存储的映射策略和算法，以避免在数据转换过程中丢失信息，解决在查询和重构时因转换结果中关系过多或过少带来的问题。结合实例给出转换后的关系表，从而证明该映射策略能有效处理元素的循环引用，并保留元素之间的结构关系。

关键词：XML 文档；映射；关系数据库

Strategy of Mapping XML Document to Relational Database

LI Si-li[1], LI Juan[2]
(1. Department of Computer Science, Engineering & Technical College Chengdu University of Technology, Leshan 614007;
2. Department of Medical Engineering, Zunyi Medical College, Zunyi 563000)

【Abstract】This paper presents an algorithm and a strategy for mapping XML document to relational database based on analysis of some data mapping strategies. It avoids information loss in the process of data transfer and the problems in query and reconstruction caused by too many or too few relations in the transfer results. Example relation tables after transferring are given, which prove that the mapping strategy can solve the problems of element recursion efficiently and maintain the structure relationship between elements.

【Key words】XML document; mapping; relational database

1 概述

XML 是一种标准的信息传输和数据交换格式，它独立于现有的数据库和编程语言，具有异构性、可扩展性以及灵活性等优势。由于 XML 文档必须能被持久保存，因此它涉及到数据库领域。目前，XML 数据的有效存储和管理已成为研究热点。

XML 数据能够存储在关系型数据库、文件系统以及面向对象的数据库中，但目前没有一种数据库类型在处理 XML 文档上占有绝对优势。一般，使用文件系统存储 XML 数据所花费的代价最小，但文件系统不提供任何对 XML 数据进行查询的支持。面向对象的数据库虽然可以群集(cluster)XML 元素和子元素，但是其在处理复杂查询方面并不成熟。由于以 Internet 为平台的应用越来越多，而应用之间交换的大量数据几乎都来自关系数据库，并且最终的结果也要存入关系数据库，因此越来越多的关系数据库管理系统(RDBMS)被用于进行底层的信息存储。如何把来自不同数据源的 XML 数据映射到关系型数据库中、如何把基于 XML 的查询转换成 SQL 查询以及如何把查询结果重新组织成 XML 文档格式都是目前的研究热点。

本文主要研究半结构化数据(通常认为 XML 表达的是半结构化数据)的映射，即设计良好的算法，使不同物理结构的 XML 数据可以有效地存储在关系数据库中。

2 相关工作

由于 XML 文档和关系型数据库具有不同的数据模型，它们之间并不能直接转换，因此需要在 XML 文档和关系型数据库之间进行一种映射，使 XML 文档数据能够永久保存在关系数据库中。目前已提出多种映射策略，这些策略主要分为两大类：(1)面向结构化的算法，其基本思想是把 XML 文档扫描成一棵有方向、有顺序、带标记的树。树的节点表示 XML 元素。子元素和属性的顺序由从节点出来的边的顺序表示。属性、子元素、元素引用都用边来表示，这导致了部份信息损失。(2)带 Schema 或 DTD 的映射算法。这类算法主要集中在文档的结构以及约束的映射上，而忽略元素之间的出现顺序。这同样导致了信息的损失。由于 Schema, DTD 并不是必需的，加上通用性的考虑，因此本文设计的映射策略并不需要 Schema 或 DTD，而是使用关系模型保留元素之间的结构关系，从而弥补了上述算法的不足。

3 Edge映射算法

目前很多映射策略都是建立在 Edge 映射算法[1]之上的。该算法最主要的思想是把 XML 文档扫描成一棵带标记的节点树，把每一条边都放入一个关系表中。

为了显示 XML 数据如何映射到关系数据库中，使用以下 XML 文档作为例子：

```
<family>
<person id=”1” age=”55”>
    <name>Peter</name>
    <address>4711 Fruitdale Ave. </address>
    <child>
      <person id=”3” age=”22”>
        <name>John</name>
        <address>5361 Columbia Ave. </address>
        <hobby>swimming</hobby>
        <hobby>cycling</hobby>
      </person>
```

作者简介：李思莉(1974－)，女，讲师，主研方向：数据库管理，计算机安全；李 娟，讲师

收稿日期：2009-07-10 **E-mail：**serena_0916@126.com

```
    </child>
    <child>
      <person id="4" age="7">
        <name>David</name>
        <address>4711 Fruitdale Ave. </address>
      </person>
    </child>
  <person id="2" age="38" child="4">
    <name>Mary</name>
    <address>4711 Fruitdale Ave. </address>
    <hobby>painting</hobby>
  </person>
  </family>
```

由此可以看出，XML 文档中每一个元素都有名称，每个元素可以包含一个或多个子元素。元素也可以有一个或多个属性，每个属性都具有名称和值。XML 文档结构可以是对称也可以是非对称的。目前已有很多 XML 数据转换的技术存在，其中最主要的思想就是以一棵有序、有向的标记树表示 XML 数据模型。图 1 为上述文档结构的有向标记树。

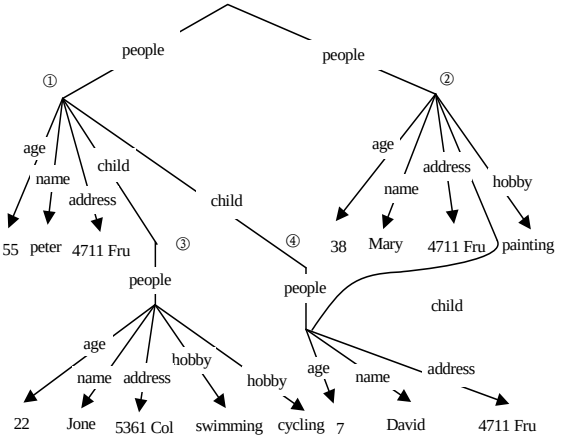

图 1　有向标记树

由于需要把结构和数据分开存放，因此图中所有的边(Edge)放在一个表(Edge table)中，数据表的结构用一个五元组表示: Edge (source, ordinal, name, flag, target)，其中，source 代表节点对象标识符; ordinal 代表属性序号; name 代表属性名称; flag 代表该属性是一个内部对象引用还是指向一个确定值类型; target 指明了属性的父节点。数据按类型分别放在不同的表中。因此，根据 Edge 算法定义，可以得到图 2 所示的映射关系表。

source	ordinal	name	flag	target
1	1	age	int	v1
1	2	name	string	v2
1	3	address	string	v3
1	4	child	ref	3
1	5	child	ref	4
2	1	age	int	v4
...	...	...	...	...

V_{int}

vid	value
v1	55
v4	38
v8	22
v13	7

V_{string}

vid	value
v2	Peter
v3	4711 Fru
v5	Mary
v6	4711 Fru
v7	painting
...	...

图 2　映射关系表

Edge 算法主要有以下缺陷: (1)忽略了元素之间的顺序关系。(2)没有区分属性和子元素。(3)把元素之间的引用也作为普通的属性来处理，从而带来查询及重构时的麻烦。为此提出了很多改进的方法[2-3]，但这些映射算法打乱了 XML 节点原有的顺序，直接导致的问题是数据修改以及查询效率的降低[4]。特别是在 XML 文档数据重构时，需要执行大量的关系表连接操作，其时间代价往往是用户无法接受的。

4　映射策略

针对 Edge[1], Attribute[2], Universal[2]等映射算法的种种问题，本文提出一种新的数据转换映射策略，不仅可以保留任意节点之间的结构关系，而且能够有效地处理 XML 结构树中存在的循环引用，从而减少关系表的连接数目，提高查询和重构效率。

4.1　相关概念

本文依然把 XML 文档抽象成有向、有序的标记树，但是将属性作为其关联元素的子元素。

定义 1 依附于某节点 n 的边数为度(degree)，记为 N(n)。以 n 为终点的边的数目为入度，记为 IN(n)，相应的边为 n 的入边，记为 IE(n)，n 的入边的集合记为 I(n)；以 n 为始点的边的数目称为出度，记为 ON(n)，相应的边为 n 的出边，记为 OE(n)，n 的出边的集合记为 O(n)；节点 n 的度、入度和出度满足如下关系: N(n)=IN(n)+ON(n)。

定义 2 如果 IN(n)=0，并且通过 n 可以到达树中的每一个节点，则 n 为根节点；如果 ON(n)=0，并且 IN(n)=1，则 n 为值节点; 如果 IN(n)=1，并且 ON(n)>0，则统称为元素节点，$\prod$(n)表示节点 n 的父节点。E(n, $\prod$(n))表示父节点到子节点的边。

定义 3 关系模式为 RelationSch(id, parent_id, nodename)。其中，属性分别用于记录节点的 id、父节点的 id 和节点名。

4.2　算法描述

为了更清楚地描述本文算法，对示例文档进行修改。如图 3 所示，该图是对 XML 文档进行深度优先扫描后得到的结构图。该图的标记与图 1 类似，只是增加了节点 id。本策略为每个节点(包括根节点、元素节点和值节点)赋一个唯一的标识，称为节点 id。对于任意 n∈N，其节点 id 记为 id(n)。节点集合 N 可以表示为节点 id 的集合。本算法引入节点 id 是为了保存 XML 文档中任意 2 个节点之间的结构关系，并采用文献[5]的方法作为节点 id 的编码方案。

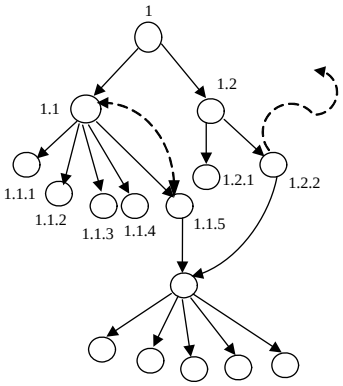

图 3　深度优先扫描后的结构图

图 3 把示例文档中存在的循环引用用带箭头的虚线表示出来，同时为了简单起见，省略了每条边的标记。为了描述方便，从图 3 中抽象出图 4 所示的结构图，把出现次数多于 1 次的元素加上了“*”号。

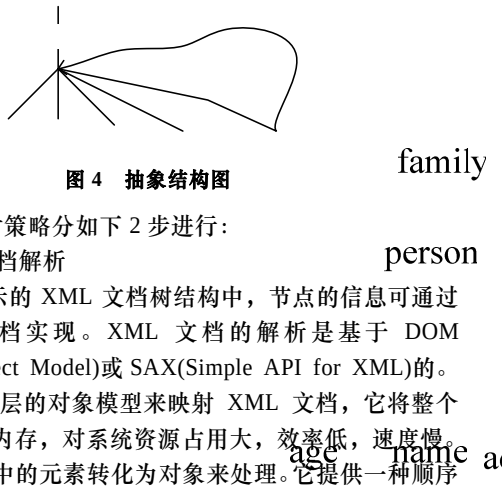

图 4　抽象结构图

本文的映射策略分如下 2 步进行:

(1)XML 文档解析

在图 3 所示的 XML 文档树结构中，节点的信息可通过解析 XML 文档实现。XML 文档的解析是基于 DOM (Document Object Model)或 SAX(Simple API for XML)的。DOM 以一个分层的对象模型来映射 XML 文档，它将整个 XML 文档载入内存，对系统资源占用大，效率低，速度慢。而 SAX 将文档中的元素转化为对象来处理。它提供一种顺序访问机制，可以检测即将到来的 XML 流，因此，不需要所有的 XML 代码同时载入内存中。本文利用 SAX 设计了如下元素循环引用检测算法，为元素的后续映射提供了基础。

算法 元素循环引用检测算法

输入 XML 文档 D

输出 如果存在元素迭代，输出为 true，否则，为 false

```
Queue nodes=getRoot(D)   //用根节点来初始化队列
Queue  nodeslist=[]        //得到一个空队列，放置节点列表，里
//面存放 unended 节点
Set traversed(e),usable(3),visited(n)和 ended(n)为 flase，对任意的
e∈E, 任意的 n∈N
While !empty(nodes) do
    n=first(nodes)
    if !ended(n) then
labelList=I(n)       //traversed(e)为 flase
for each OE(n) in O(n) do
if not traversed(e) then
  if I(n)∩I(n')  ≠Φ  then
  usable(e)=false
  else
   usable(e)=true
endif
end for
endif
for each E(v) and not traversed(e) do
    if (usable(e)||ended(n)) then
     traversed(e)=true
    if traversed(e)=true    //e∈I(n')
then
    nodes.Append(n')
    ended(n')=true
    else
    if !visited(n') then
    nodeslist.Append(n')
    visited(n')=true
  end if
end if
end if
end for
if empty(nodes) and ! empty(nodeslist) then
 n=first(nodeslist)
 nodes.Append(n)
 visited(n)=false
end if
end while
for each node n in D do
if !ended(n)
  return false
end for
return true
```

算法先对 XML 文档进行深度优先遍历，根据定义 2，当第一次到达某元素节点时，标上“visited”，一旦它下面的所有子节点被遍历完，则取消“visited”标记。这一部分用 visited(n)函数完成；可遍历的边用 usable(e)函数完成，边的遍历用函数 traversed(e)完成。根据定义 1，当 n 的所有入边都被遍历完后，ended(n)=true。如果在检测结束后仍然存在没有 ended 的节点，则一定存在元素循环引用的情况，同时可以知道哪些节点存在迭代。

(2)映射转换

通过循环检测算法可以知道哪些节点上存在循环引用。之后需要为这些节点新建关系表以处理元素循环引用问题。除此之外，根据图 4，所有带“*”标记的节点都映射为单独的关系表。最后，对于普通的元素节点 n，如果以 n 为始点，把它所能到达的、同时 2 个节点之间的路径上不存在其他节点的节点放入节点集合 N 中，则任意的 n'∈N 都把它作为 n 关系的属性放入一个关系表中。

通过定义 3 建立节点之间的关系模型，用来保存节点之间的关系，目的是重构 XML 数据。

4.3　映射转换结果

依据上述理论，使用 VS2008 在.NET Framework2.0 上以 C#语言为工具，实现了图 3 对应的 XML 文档到 SQL Server2005 数据库的转换，转换结果如表 1~表 5 所示。

表 1　CircleRelation

circleid	pid	cid
1	1.1	1.1.5

表 2　RelationSch

id	parentid	nodename
1.1	1	person
1.2	1	person
1.1.1	1.1	age
1.1.2	1.1	address
…	…	…

表 3　person

name	age	address	childid	hobbyid
peter	55	4711 Fru	1	null
mary	38	4711 Fru	2	2

表 4　child

name	age	address	circleid	hobbyid
John	22	5361 Col	null	1
David	7	4711 Fru	1	2

表 5　hobby

hobbyid	hobby
1	swimming
2	cycling
3	painting

5　结束语

本文提出一种 XML 文档到关系数据库的映射策略，其中的元素迭代检测算法及映射转换算法有效克服了传统映射算法中 XML 文档中因存在元素迭代引用而导致的转换失败或效率低下问题，同时，新的映射策略克服了关系表过多或

（下转第 45 页）