第 24 卷 第 2 期 华侨大学学报(自然科学版) Vol. 24 No. 2
2003 年 4 月 Journal of Huaqiao University (Natural Science) Apr. 2003

文章编号 1000-5013(2003)02-0201-07

一种 XML 数据到结构化数据的转换方法

陈维斌 喻小光

(华侨大学信息科学与工程学院, 福建 泉州 362011)

摘要 分析半结构化树状层次结构的 XML 文档的结构特征, 描述 XML 结构与关系数据库的对应关系. 给出 XML 文档的 DTD 中各主要元素与关系数据模型之间的映射规则, 设计转换规则脚本的自动生成算法和数据转换算法.

关键词 XML, 关系数据模型, DTD, 转换规则

中图分类号 TP 311.132.3 文献标识码 A

XML 是一种可扩展的标记语言, 具有可扩展性、自描述性, 以及在应用间交换数据的灵活性等特性. 虽然 XML 对半结构化的混合类型数据具有强大的表示能力, 并能以文本格式保存, 达到几乎"永不磨损"的程度[1], 但对数据的处理能力(例如查询、更新等)却极其有限. 目前, 绝大多数的数据处理和数据应用都是基于数据库. 因此, 从 XML 数据源中提取数据保存到数据库. 这是一项有实际意义的工作. 相应地, 数据格式的转换, 模型的映射等都成为当前的研究热点. 其中具有前瞻性的研究莫过于将 XML 文档映射为对象数据库[2]. 由于对象数据库技术尚未成熟, 而关系数据库目前乃至今后相当长的一段时间内仍旧是主流数据库. 因此, 研究将 XML 数据转换成关系模型数据具有现实意义[2,4]. 本文借鉴"模型驱动"和"模板驱动"等数据转换中间件所采用的方法, 提出"规则驱动"的转换技术. 它给 XML 文档中的各种元素制定转换规则, 利用 DTD 文档提供的元素类型信息, 有效地解决转换规则脚本的自动生成等技术问题. 针对不同的 XML 文档灵活地生成规则脚本, 并保证转换的正确性和有效性.

1 XML 格式到关系数据库模式的映射规则

1.1 XML 文档的结构特征分析

虽然 XML 允许用多种方法来组织数据, 但人们普遍采用的还是半结构化的树状层次结构[4]. 此种结构的 XML 文档有且只有一个根元素. 它完全包含文档中可以包含文本(属性或属性值), 或嵌套子元素的其它元素. 下面给出部门和员工信息的 XML 文档为

```
< ? xml version= 1.0"? >
< department id= "01">
  < name> 开发部< /name>
```

收稿日期 2002-11-20 作者简介 陈维斌(1957-), 男, 副教授

基金项目 国家高科技应用部门发展基金资助项目

```
<department id="0101" fatherid="01">
  <name>开发一科</name>
  <person id="0001" groupid="0101">
  <name>张三</name>
  <email>zhangsan@sina.com</email>
</person>
</department>
<person id="1101" groupid="01">
  <name>李四</name>
  <email>lisi@sina.com</email>
</person>
</department>
```

该文档数据若用关系数据库模式描述,则如图1所示.该模式的两个主要关系,如表1,2所示.

表1 Department 关系

DepID	DepName	FatherDepID
01	开发部	0
0101	开发一科	01

表2 Person 关系

PersonID	GroupID	PersonName	Email
0001	0101	张三	zs@sina.com
1101	01	李四	ls@sina.com

1.2 XML 数据格式与关系数据模型之间的映射

由1.1节的分析可以看出,由于XML文档格式与关系数据模型在结构上存在着一定的差异,所以两者之间无法直接进行数据交换.但这两种结构的基本元素之间存在一定对应关系.(1) XML外层元素对应于关系数据模型中的表(关系),其属性和被它包含的内层简单元素对应表的属性;属性和内层简单元素的值,对应属性的值.(2) 内层非简单元素也对应表,但该表将作为该元素的外层元素所对应表的从表,具有主码-外码参照关系.上述这种对应关系是建立结构映射的基础,有其基本映射规则.(1) XML数据中的外层元素以及内层非简单元素,映射为关系数据库的表.(2) XML数据中的简单元素、元素的属性,映射为关系数据库表的字段.(3) XML数据中的元素之间相互位置关系,映射为关系数据库中表与表的关联.该规则具有一定的普遍性,但前提是对应的XML文档必须是格式完善(遵循XML语法规范)并且有效的(遵循XML文档的DTD的规则).除了上述基本规则之外,还必须给出各种元素的转换规则(将在2.1节给出).因此,不同的XML文档其转换规则集是不同的,也即对每个待转换的XML文档必须给出一个转换规则脚本.转换程序按照转换规则脚本所给的规则将XML数据转换成结构化数据,并存储在指定的关系数据库中(图2).由此看出,规则驱动的转换方

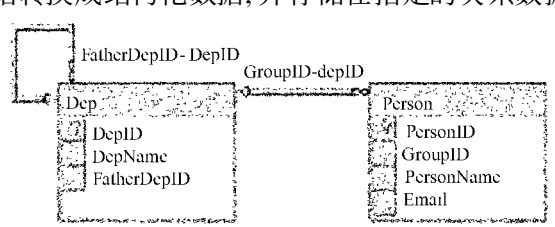

图1 部门和员工数据库模式

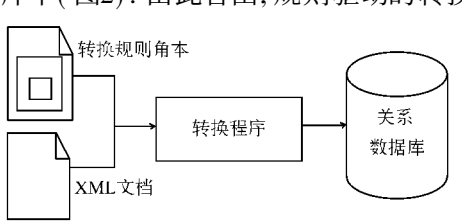

图2 转换方法示意图

法的关键, 在于转换规则的格式规范和转换脚本的自动生成.

2　转换规则脚本的设计

2.1　转换规则脚本规范

我们将转换规则脚本设计成 XML 文档, 其文档类型声明(DTD)如下:

```
<! ELEMENT X2RConversion (Entrance)>
<! ELEMENT Entrance (Record+ )>
<! ELEMENT Record (Default* , Node* ,
Reference* , Record* )>
<! ATTLIST Record
entity CDATA # IMPLIED
id CDATA # IMPLIED
path CDATA # IMPLIED
refid CDATA # IMPLIED>
<! ELEMENT Default EMPTY>
<! ATTLIST Default
field CDATA # REQUIRED
value CDATA # REQUIRED>
<! ELEMENT Node EMPTY>
<! ATTLIST Node
path CDATA # REQUIRED
field CDATA # REQUIRED>
<! ATTLIST Reference
field CDATA # REQUIRED
reftype (entity | record) # REQUIRED
refvalue CDATA # REQUIRED
refpos CDATA # REQUIRED>
<! ELEMENT Reference EMPTY>
```

该DTD 规范了转换脚本的书写格式, 为转换程序的通用性奠定了基础. 下面的转换脚本就是该规范的具体应用. 现结合该脚本对规则的主要部分作进一步说明为

```
< X2RConversion>
  < Entrance>
    < Record entity= "dep" id= "st - dep" path= ". department">
    < Default field= "fatherDepID" value= "0" />
    < Node path= "# id" field= "depID" />
    < Node path= ". name" field= "depName" />
    < Node path= "# fatherid" field= "fatherDepID" />
    < Record entity= "dep" id= "run - dep" path= ". department">
    < Node path= "# id" field= "depID" />
    < Node path= ". name" field= "depName" />
    < Node path= "# fatherid" field= "fatherDepID"/>
    < Reference field= "fatherDepID" reftype= "entity" refvalue= "dep[depID]" refpos
= "- 2" />
    < Record refid= "run - dep" />
    < Record refid= "leaf - person" />
  < /Record>
  < Record entity= "person" id= "leaf - person" path= ". person">
    < Node path= "# id" field= "personid" />
```

```
    < Node path= ". name" field= "personname" />
    < Node path= ". contact. email" field= "email" />
    < Reference field= "groupid" reftype= "entity" refvalue= "dep[ depID]" refpos= "- 1"
/>
    < /Record>
      < /Record>
    < /Entrance>
< /X2RConversion>
```

(1) X2Rconversion 元素是转换规则脚本的根,它的下层只包括一个 Entrance 元素,用来容纳所有的转换规则. (2) Record 元素为待转换 XML 文档中的非简单元素(以下简称表结点),与关系数据模型中的对应表建立对应关系. 它的属性 entity 指明表名、id 指表的别名、path 指表结点在待转换文档中的位置. (3) Entrance 元素直接嵌套的 Record 元素是转换入口,每一个转换入口对应一套转换规则. Record 元素分两种情况. (a) 当表之间不存在关联时,每个表将各自对应一个转换入口,有 N 个表就有 N 个转换入口 Record. (b) 当表之间存在关联时,只有主表对应一个转换入口,而从表的 Record 将嵌套在主表的 Record 中,作为子 Record 出现. 也就是说,从表的转换规则将作为主表转换规则的一部分,它们在主表建立后被驱动. (4) path 是转换规则的一个重要属性,它标定了待转换元素或属性在待转换文档中的位置. 在转换入口定义的 path 是转换入口的转换标识后缀. 除转换入口处的 path 外,下层元素中的 path 都是相对值,添加了上层的完全值后就成为该元素的完全值. 非转换入口的 Record 的 path 一样是要加在上层元素的 path 的完全值之后,构成该 Record 的完全值. 由该值确定转换位置后,转换操作同转换入口的 Record. (5) Default 元素对应一个字段的默认值赋予. (6) Node 元素为代表字段的结点或属性与相应的数据库表字段建立对应关系,它对应一个字段值的转换. 当 path 属性标识了一个元素时,将元素的内容赋给指定的字段. 当 path 属性标识了一个属性时,将属性的值赋给指定的字段. (7) Reference 元素的转换规则. 负责将元素的位置转换为实体的联系,具体由 field, reftype, refvalue 和 refpos 等 4 个属性进行描述.

2.2 转换规则脚本的生成

通过扫描 DTD 文档可以获取对应的 XML 文档所包含的元素信息(元素类型、元素之间的关系),利用这些信息并结合读取 XML 文档来生成各元素的转换规则. 下面给出各步骤的主要算法. (1) 扫描 DTD 文档获取元素信息. DTD 文档有三类元素,即基本元素、包含子元素的元素和不包含子元素的元素. 它们的基本特征是基本元素(根元素)包含文档中声明的所有其他元素;不能包含子元素的元素声明的一般形式为<! ELEMENT 元素名(# PCDATA)>;包含子元素的元素的声明为<! ELEMENT 元素名(子元素列表)>. 我们将 DTD 文档中元素之间的关系及每个元素的信息,保存在文档结构信息文件中. 该文件的逻辑结构为

```
ELEMENT -INFO{
        NAME:           //元素名
        ID:             //唯一编号,用来区别各个元素
        PARENT -ID:     //父元素的编号
        CHILD -ID:      //第一个子元素编号
```

```
        BROTHER-ID:      //下一个兄弟元素编号
        POSITION:        //在父元素的子元素列表中的位置
        SIGN:            //标记是否为不确定元素
    }
```

检索DTD 文档, 获取各个元素信息来建立文档结构信息文件的算法为

```
读取DTD 文档中的基本元素信息, 建立相应的结点 root;
root. name= 基本元素名, 顺序号= 1, root. id= 顺序号, root. parent-id= 0;
if 基本元素有子元素 then
  while 存在子元素 do
    设置其在基本元素中的位置编号及唯一编号, 并判断此元素是否为不确定元素;
    将每个静态子元素的自身编号、父元素编号、位置编号、下一个兄弟编号、标记及名称
    压入栈; 检测是否存在下一子元素; loop
  if 栈不为空 then 取一个元素, 建立新结点; 检索出此元素的声明;
    if 有子元素 then
      按照基本元素有子元素的方法处理; //用递归实现深度优先遍历整个 DTD 文档
        唯一编号不断累加, 位置编号从 1 开始, 每增加一个子元素位置编号加 1;
     end if
  end if
else //基本元素声明为 ANY 类型
  while 存在不为任何元素子元素的元素 do
    取出此元素的声明; 建立新的结点 b, b. name= 该元素名, b. id= 累加得到的顺序号,
b. parent-id= 1; // 直接将这种结点作为根结点的子结点.
    if 已经存在此结点 then
        根据信息找到这些已经排列的元素的最后一个结点 c, c. brother_ id= b. id, b.
        position= c. position + 1; else 不存在这样的结点 then b. position= 1 end if
    if b 有子元素 then 按照基本元素有子元素的方法处理;
        唯一编号递逆增, 位置编号从 1 开始, 每增加一个子元素位置编号加 1; end if
    loop
end if
write 元素信息 to 文档结构信息文件中
```

(2) 根据对 DTD 的分析结果生成转换规则脚本. 该处理的主要算, 可法描述为

```
open 文档结构信息文件;
while 文档结构信息文件指针不为空
    Read 结点信息 from 文档结构信息文件; Push to 元素信息栈;
    文件指针指向下一信息结点 loop
while 栈不为空 do
    取出一个元素;
    if 该元素为字符型元素 then
```

```
        在该元素的父结点对应的 Record 结点中, 建立一个 Node 结点(是一个字段);
    if 该元素具有确定的全部子元素 then
        if 该元素的父结点编号为 1 then 直接建立一个 Record 结点(表);
        else
            在该元素的父结点对应的 Record 结点中, 嵌套一个 Record 结点(子表);
        end if
        将该元素所有子元素压栈;//递归处理
    end if
    if 该元素具有不确定的子元素 then
        if 该元素的父结点编号为 1 then 直接建立一个 Record 结点(表);
    else
        在该元素的父结点对应的 Record 结点中, 嵌套一个 Record 结点(子表);
    end if
    取出它的一个子元素;
    if 该子元素是确定元素 then
        在其父元素对应 Record 结点中创建 Node 结点;
    else 在该子元素的父结点对应的 Record 结点中, 嵌套一个 Record 结点(子表); 将
        该子元素压栈;
    end if
  end if
loop
```

3 数据转换方法简介

3.1 创建数据库、表和关联

(1) 用待转换的 XML 文档的名称建立同名数据库(创建何种数据库可由用户自行选择). (2) 广度优先遍历转换规则脚本, 根据 Record 结点及其嵌套的 Node 结点, 创建与此相对应的数据库表; 根据 Record 结点及其嵌套的 Record 结点建立表与表之间的关联.

3.2 数据转换

所谓的转换算法, 我们可以概述为广度优先遍历待转换的 XML 文档的 DOM 树, 及为当前结点的转换标志. 即根到当前结点的路径(将每个途经的结点的名字, 按从根到当前结点的顺序用“.”连接起来), 在转换规则脚本中寻找匹配. 如果匹配不成功或匹配的不是 Record 结点, 则不做任何操作, 继续遍历下一个结点. 如果匹配的是 Record 结点, 则执行对应的规则以创建对应记录并进行字段值的转换, 甚至还包括从表的记录的创建和字段值的转换. 继续进行广度优先遍历, 直到待转换文档结束. 算法中有几个需要说明的地方. (1) 只有规则中遇到 Record 结点时, 才能创建记录, 并执行相关的数据转换工作. (2) 在同层规则(同一个 Record 中) 执行时, 是按 Default 元素、Node 元素、Reference 元素、Record 元素的次序进行的. 一个次序在后的有效操作, 可覆盖次序在前的有效操作. 下面用图 1 所示 XML 文档的转换过程来说明该算法. (1) 访问待转换文档的 DOM 树的根结点 department, path = . department. (2) 访

问转换规则脚本, path 匹配转换入口, 开始执行转换规则. (3) 为 dep 表建立一个元组, 为 fatherNodeID 填写默认值 0. 将 department 的 id 属性赋给 depID 字段, 将 department. name 的值赋给 depName. 该 department 没有 fatherid 属性, 所以第三个 node 将不执行. 执行结果为 "| 01 | 开发部 | 0 |". (4) 访问第一个 name 结点, path= . department. name 在规则中匹配非 Record 结点, 所以不做任何操作. (5) 访问第二个 department 结点, path= . department. department 匹配 Record 结点, 操作与第一个 department 一样. 执行结果为 "| 0101 | 开发一科 | 01 |". (6) 访问 person 结点, path= . department. person 匹配 Record 结点, 建立 person 表的元组. 将该结点的 id 属性值赋给 personid 字段; person. name 的值赋给 personname 字段, person. email 的值赋给 email 字段. 将上层元素(即第一个 department) 的 id 属性值赋给 groupid 字段, 以建立引用关系. 执行结果为 "| 0001 | 0101 | 张三 | zs@sina. com |". (7) 继续进行广度优先遍历, 直到待转换文档结束. 最后的转换结果, 如表 1, 2 所示.

4　结束语

规则脚本的自动生成, 它是规则驱动的数据转换工具的关键技术之一. 我们采用 Visual Basic6. 0, 结合组件编程技术, 实现了规则脚本生成器和数据转换工具的原型系统. 该工具能对带有 DTD 文档的半结构化 XML 文档进行正确的转换.

今后有两点主要工作. (1) 改进脚本生成器, 使其具有对无 DTD 文档的 XML 文档自动生成转换规则脚本的能力. (2) 研究 XML 格式和关系数据模型的形式化映射规则, 实现两种数据格式的双向、高效、无损转换.

参　考　文　献

1　Harold E R 著. XML 实用大全[M]. 杜大鹏等译. 北京: 中国水利水电出版社, 2000: 48～68, 746～752
2　施伟斌, 孙未未, 施伯乐. XML 数据的结构化处理方法[J]. 计算机研究与发展, 2002, 39(7): 819～824
3　谷长勇, 徐志伟, 褚兴军等. XML 结构和关系数据库的一种形式化映射[J]. 计算机工程, 2001, 27(11): 16～17
4　瞿裕忠. 一个基于 XML 的数据交换原型系统[J]. 计算机工程, 2000, 26(9): 35～37

A Method for Converting XML Data into Structured Data

Chen Weibin　　Yu Xiaoguang
(College of Info. Sci. & Eng., Huaqiao Univ., 362011, Quanzhou, China)

Abstract　The present work begins with an analysis of the structural characteristic of the semi structured, arborescent, hierarchy XML document, and a description of the corresponding relation between XML structure and relational database. On this basis, the authors give the mapping rules between main elements in DTD of XML document and relational data model; and design the algorithm for the autogeneration of conversion rule script and the algorithm for data conversion.

Keywords　XML, relational data model, DTD, conversion rule