关系模式与 XML 模式的相互转换

The exchang between relation schemas and XML schema

蔡小芳　张永胜　（济南 山东师范大学信息管理学院 250014）

摘要:XML 数据具有自描述特性,所以能够从自身得到描述自己的类似于数据库的数据模式,又由于 XML 具有树型结构的特点,由此可以把 XML 模式转化为关系模式,同样关系模式也能够转化为 XML 模式,本文将分别给出如何实现这两个模式的相互转化的算法。

关键词:XML 模式　关系模式

1　引言

XML 实际上是一种定义语言,使用者可自由定义标记,并通过元素之间的嵌套包含来体现层次关系。它主要有 3 个要素:DTD 或 Schema(模式),XSL (eXtensible Stylesheet Language),XLL (eXtensible Link Language)。其中,DTD 或 Schema 规定 XML 文件的逻辑结构,定义 XML 文件中的元素、元素属性以及元素与元素之间的关系,它可帮助 XML 分析程序校正 XML 文件标记的合法性,XSL 用于规定 XML 文档的显示方式,它能够在客户端使 Web 浏览器改变文档的表示形式,而不需再与服务器进行通讯交互。XLL 主要用于进一步扩展目前 Web 上已有的简单链接[3]。XML 文档的结构树是由元素、子元素、属性三者的连接构成。

由于 XML 具有结构化(它的逻辑结构表现为一个图/树结构)、保值性(XML 数据能够自描述)和数据独立性等特点,在数据呈现和数据交换领域,XML 得到了越来越多的应用,然而,已有的大部分数据都存储在关系数据中。因此,如何将这些数据转换为 XML 格式的数据就显得很重要。

关系数据库用二维表来存储数据,其中每行为一条记录,记录是对等的,没有先后次序。由于关系数据库成熟的管理、存取、更新和检索技术,我们有必要将 XML 模式转化为关系模式。

关系数据库包含两种信息,分别表示对象和关系。大部分关系数据库至少满足 3NF 或 BCNF 范式,其主关键字也往往被设计成单字段。从这两个特点出发我们设计一个实现和结果都比较简单的算法。一般而言,表示关系的表具有外关键字,且该外关键字是表示对象的表的关键字。通过外关键字产生的约束是 IND (Inclusion Dependency) 中最常见的一种约束,可以通过 ODBC/JDBC 接口来获取。关系数据库中的约束还有字段约束,是否可空约束、检查子句设置的约束和触发器产生的约束[1]。本文主要考虑外关键字约束和字段值是否可空约束两种语义约束。

现有的算法大多都用文档类型定义 DTD 来实现,但 DTD 将逐步被 XML 模式代替。

2　转化算法

在进行转换之前,首先判断原数据库的类别,根据类别进行转换,具体的算法思想如下:

```
if (数据类型是 XML 文档)
  then
    调用模块 2.1 将 XML 文档转换为关系模型
  else
    调用模块 2.2 将关系模型转换为 XML 文档
```

2.1 XML 模式转化为关系模式的算法

我们可以采用下面的方法来实现把 XML 模式转化为关系模式。

2.1.1 对每个含有子元素或混合内容的元素生成一个表和一个主键列;

2.1.2 对每个含有混合内容的元素生成一个单独表,该元素保存的 PCDATA 数据,通过父表主键连接;

2.1.3 对每个单值属性和每个只出现一次的,且只含 PCDATA 类型数据在父表中形成一列,如果子元素或属

性可选,则此列为 NULL;

2.1.4 对每个多值属性和每个多次出现的,只含 PCDATA 类型的子元素,生成一个单独表保存值,并通过父表的主键连接;

2.1.5 对每个有子元素或含混和内容的子元素,通过父表的主键连接父表到子元素的表。

虽然上面的算法可以实现转化,但不够鲜明,而且容易出现错误,所以我们可以采用另外一种方法,从而以一种直观的方法实现把 XML 模式转化为关系模式。

(1) XML Schema 文件析取为 XML 模式图

① 读入 XML Schema 文件,并且初始化为 XML 模式图(记为 G);

② 如果已经读到文件尾,就结束算法,否则读入下一个元素;

③ 如果这个元素在模式 G 中已经存在,就结束算法,否则在模式图 G 中创建这个元素(记为 new);

④ 如果新读入的元素 new 没有子元素就结束算法,否则读取 new 的下一个子元素(记为 sub);

⑤ 如果这个子元素 sub 在模式图 G 中已经存在,就在模式图中连接 sub 和 new,然后转入第 4)步。

(2) 由 XML 模式图生成一个包含所有将要转化为关系的元素节点表。

① 读入 XML 模式图 G,并且初始化元素节点表(记为 T);

② 对于 XML 模式图 G 中的任何一个节点元素 i,如果满足以下两种情形则把 i 加入到元素节点表中;

情形 1:节点 i 的入度为 0

情形 2:节点 i 的入度大于 0,并且存在子元素 j,有(i,j)的弧的个数没有上限;

③ 利用深度优先算法遍理历 i,并且把它们链接到 i 上,如果 i(记为 new)有子元素,读入记为 sub,并且作出情形判断:

情形 1:如果新读入的元素 sub 是"*"或者在节点 new 和 sub 之间有环路,那么设置 node. ParentID 为 sub 的 ID,读取其父元素,继续遍历;

情形 2:如果新读入的元素 sub 是"?",那么读入 sub 的子元素,并且设置为 sub。

(3) 由元素节点生成关系模式的算法

① 对元素节点表 T 中的每一个节点,创建一个关系,关系名为节点名字,把关系的标志(ID)作为其属性,如果附加属性里有父节点 ID,也将其添加进去;

② 对每一个作为关系名的关系表中的节点,遍历其所有子节点,并将它们的名称作为关系属性。

2.2 关系模式转化为 XML 模式的算法

XML 数据的典型特征是具有树型结构。与关系数据库相比,它能表示更多的信息。下面以数据库为例来分析关系模式中数据表和两种约束的转换。

数据库包含多个表,表中包含多条记录,记录又有多个字段,正好用 XML 的树型结构来表示。表对应三层结构:上层是表,中层是记录,下层是字段,故可以将表转换为一个复合类型的元素,该元素的子元素对应记录,这些子元素的子元素对应记录中的字段。使用 XML 中的 ID 属性与 IDREF 属性之间的关系可以很好的表示主关键字和外关键字之间的关系。使用 XML 中的 nillable 属性可以很好的表示字段值是否可空约束。

下面以表商品 good (good – id, good – name, good – city, good – price) 为例来说明其中字段的转换。一般来说,字段被直接转化为一个元素,同时定义相应的类型属性,并指出其值是否可空。如果 good – city 是外关键字,那么使用属性 IDREF 表示如下:

```
<xs:element name = "good – city" >
    <xs:ComplexType >
        <xs: attribute name = " idref" type = " xs:
IDREF" / >
    </xs:ComplexType >
</xs:element >
```

如果 good – id 是字符类型,且是被引用的主关键字,那么使用属性 ID 表示如下::

```
<xs:element name = "good – id" >
<xs:ComplexType >
  <xs:SimpleContent >
    <xs:extension base = "xs:sring" >
      <xs: attribute name = " id" type = " xs:ID" /
>
    </xs:extension >
  </xs:SimpleContent >
</xs:ComplexType >
</xs:element >
```

如果 good – price 是字符类型,且可以为空,那么使用 nillable 表示如下:

```
<xs:element name = "good – price" type = "xs:
string" nillable = "true"/ >
```

根据以上的分析，可以抽象出下面几种模式。其中 N 表示 XML 模式中的一个合法名字，可以表示数据库名、表明或列名等。这几种模式只考虑了简单的属性，如果存在其他属性，那么可以直接修改对应的模式。

模式一（M1）用于映射关系模式的列：

```
<xs:element name = "N" type = "N" nillable = "
N"/ >;
```

模式二（M2）用与映射关系模式的主关键字列：

```
<xs:element name = "N" >
  <xs:ComplexType >
    <xs:SimpleContent >
<xs:extension base = "N" >
 <xs:attribute name = "id" type = "xs:ID"/ >
  < /xs:extension >
 </xs:SimpleContent >
 </xs:ComplexType >
 </xs:element >
```

模式三（M3）用于映射关系模式中的外关键字列：

```
 <xs:element name = "N" >
    <xs:ComplexType >
     < xs: attribute name = " idref" type = " xs:
IDREF"/ >
      </xs:ComplexType >
      </xs:element >
```

模式四（M4）用于映射关系模式中的表：

```
  <xs:element name = "N" >
    <xs:ComplexType >
      <xs:sequence >
       <xs:element name = "row"
        minoccurs = "0"
       maxoccurs = "unbound" >
       <xs:ComplexType >
        <xs:sequence >
        ( M1|M2|M3 ) +
        </xs:sequence >
       </xs:ComplexType >
       </xs:element >
      </xs:sequence >
     </xs:ComplexType >
 </xs:element >
```

模式五（M5）用于映射关系模式中数据库：

```
<? xml version = "1.0" encoding = "UTF – 8"? >
  < xs: schema xmlns: xs = " http://www. w3. org/
2001/X MLSchema"
   targetNamespace = " http://www. db. com/db"
>
   <xs:element name  = "N" >
   <xs:ComplexType >
     <xs:sequence >
      M4 +
     </xs:sequence >
    </xs:ComplexType >
   </xs:element >
</xs:schema >
```

利用上面的转换模式我们可得到关系模式转化为 XML 模式的算法如：

（1）将数据库中的每个关系表转换为 XML 文档中的元素；

（2）利用上边的几种模式转换将关系表中的元组转换为 XML 文档中的子元素，并将关系表中个元素的属性赋给相应的子元素；

（3）输入新的关系表，递归调用（1）（2），直至输入完毕；

（4）通过关系表的主键，外键将各元素、子元素串接成树状结构的 XML 文档，合并属性；

（5）删除关系数据库中因规范化而导致的冗余，优化 XML 文档。

3　应用举例

下面我们结合一个具体的文件来说明如何将 XML 模式转化为关系模式。下面给出一个 DTD 文件及其对应的 XML Schema 文件的部分代码：

```
< ! ELEMENT department( deptName,operator) >
< ! ELEMENT material( desc,operator * ,assistant) >
< ! ELEMENT assistant EMPTY >
< ! ELEMENT assistant operatorID IDREF IMPLIED >
< ! ELEMENT room ( desc, operator,keeper) >
```

```
<! ELEMENT operator( name,contact) >
<! ELEMENT operatorID ID# REQUIRED >
<! ELEMENT name( firstname? lastname) >
<! ELEMENT firstname( #PCDATA) >
<! ELEMENT lastname( #PCDATA) >
<! ELEMENT contact( #PCDATA) >
```

其对应的 XML Schema 文件的部分代码：

```
<xsd:element name = "department" >
    <xs:ComplexType >
    <xs:sequence >
    <xsd: element name = " deptName" type = "
xsd:string"/ >
    <xsd:element ref = "operator"/ >
    </xs:sequence >
    </xs:ComplexType >
    </xsd:element >
      .
      .
      .
     <xsd: element name = " lastname" type = "
xsd:string"/ >
     <xsd:element name = " contact" type = "xsd:
string"/ >
```

利用上面的转换算法，其对应的模式图为图 1 所示。

它对应的关系模式为：

department(departmentID: integer, department
deptName: string, operator. name. firstname:
string, operator. name. lastname: string, operator.
contact: string)

material(materialID: integer, material. assistant. operatorID: integer, material. desc: string)

room(roomID: integer, room. parentID: integer,
room. desc: string, room. keeper. parentID:
integer, operator. name. firstname: string, operator. name. lastname: string, operator.
contact: string)

operator(operatorID: integer, operator. parentID: integer, operator. name. firstname: string,
operator. name. lastname: string, operator. contact: string)

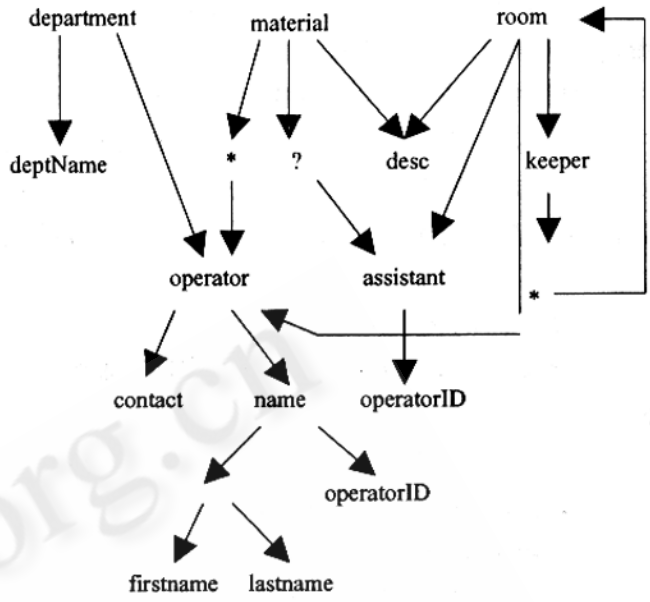

图 1

4 结束语

由于 XML 良好的结构和带有自描述的 XML 模式特性，XML 可以作为各种异种数据源之间进行通信的标准。XML 文档的存储转换技术一直是实际应用中的重要问题。基于关系模式和 XML 模式的相互转换可以把 XML 文档处理要求转化为相应的数据库操作，因而能够利用现有的数据库成熟技术进行数据处理，再将处理结果转变为 XML 数据，就可以实现在异类应用应用程序间交换数据[4]。当前主流关系数据库系统如：Oracle9i、Microsoft SQL Server 等都在加入 XML 存储和数据转换的解决方案。

参考文献

1 Abraham Silberschatz, Hery F Korth, S Sudarshan, 杨冬青、唐世渭等译, 数据库系统概念[M], 北京 机械工业出版社, 2000。

2 The XML Handbook[M]. Prentic Hall PTR, 1998.

3 曾春平、王超、张鹏, XML 编程从入门到精通[M], 北京希望电子出版社, 2002 -01。

4 The World Wide Web Consortium (W3C). Xquery 1.0: An XML Query Language[EB\OL]. http://www.w3.org/TR/xquery/.